UNIVERSIDAD DE CASTILLA - LA MANCHA

Escuela Superior de Informática

Departamento de Tecnologías y Sistemas de Información

FRAMEWORK PARA LA MONITORIZACIÓN MÓVIL DE PACIENTES

AUTOR

D. Vladimir Villarreal Contreras

DIRECTORES

Dr. José Bravo Rodríguez

Dr. Ramón Hervás Lucas

Memoria presentada para optar al título de Doctor en Informática

Junio 2012

Autor:

Vladimir Villarreal Contreras
Universidad Tecnológica de Panamá y Modelling Ambient Intelligence (MAmI)
vladimir.villarreal@utp.ac.pa

RESUMEN

En el transcurso de estos últimos años se ha producido un cambio global en la manera de tratar las enfermedades. Se han hecho muchos estudios para facilitarle al paciente una vida más llevadera, que le permita desarrollar sus actividades diarias. Esto ha dejado de ser una actividad exclusiva del ámbito médico y asistencial y se ha convertido en un campo de investigación multidisciplinar. Se han tratado de adaptar nuevas propuestas cambiando la manera en que los servicios sanitarios son proporcionados a los pacientes.

La evolución de los diferentes dispositivos tecnológicos ha permitido integrar nuevas tecnologías para el tratamiento y seguimiento de enfermedades en las cuales los pacientes encuentran un apoyo a través de su uso. La amplia integración de dispositivos móviles con altas características técnicas, y con grandes capacidades de comunicación, ha emergido permitiendo realizar múltiples actividades, que no sólo incluyen las actividades para las cuales fueron desarrollados inicialmente.

Esta tesis se ha desarrollado con el propósito de ofrecer una solución tecnológica que permita a los pacientes un mejor y más oportuno seguimiento y control de sus enfermedades, ofreciéndole respuestas constantes en el tiempo adecuado. Para ello se ha desarrollado un marco de trabajo (*framework)* conceptual llamado "**MoMo"** (**Mo**nitorización **Mó**vil) que permite el desarrollo de aplicaciones móviles parametrizadas, el seguimiento médico a través de dispositivos móviles (que son de uso cotidiano por el paciente) y los dispositivos biométricos (que actualmente ofrecen altas prestaciones tecnológicas).

La generación de aplicaciones basada en el *framework* es *genérica*, *adaptable*, *remota* y *móvil*. ***Genérica,*** porque permite el desarrollo de aplicaciones para múltiples tipos de enfermedades, siguiendo el diseño conceptual del *framework* desarrollado. ***Adaptable,*** porque ofrece servicios ajustados a cada tipo de enfermedad y particularizada a las características de cada usuario, haciendo la interacción lo más transparente posible. ***Remota,*** porque el personal médico será capaz de conocer todos los datos obtenidos por los dispositivos biométricos de los paciente a través del dispositivo móvil de manera no intrusiva. ***Móvil,*** ya que el desarrollo está basado en la integración de dispositivos de pequeño tamaño, portátiles e inalámbricos. Esto le da mayor autonomía al paciente.

El *framework* propone la integración de un conjunto de patrones llamados "*MobiPattern*" para el diseño de interfaces de usuario, especificación de funcionalidades y desarrollo estandarizados de módulos, lo que permite la reutilización y rediseño de toda la arquitectura. De igual manera se ha definido una clasificación ontológica "*MoMOntology*" que permite la formalización de datos dentro de la arquitectura. Para facilitar el desarrollo y en un futuro el mantenimiento, se ha definido e implementado una estructura en capas "*MoMoLayer*" de todos los elementos que intervienen en el diseño y funcionamiento del *framework*.

A partir del *framework* se ha implementado una arquitectura *software* que da lugar a algunos prototipos para ser evaluados en pacientes con alguna enfermedad a monitorizar (diabetes y tensión arterial específicamente), de los cuales se han obtenido algunas

valoraciones a través de un cuestionario para evaluar aspectos de diseño, contenido y utilidad de los prototipos finales. Además de las valoraciones por parte de los pacientes, se ha realizado una evaluación sistemática basada en métodos que permiten evaluar arquitecturas *software*. Con estos métodos se han valorado aspectos de calidad como son capacidad de modificación (modificabilidad) de la arquitectura, facilidad de uso (usabilidad) y grado de rendimiento de todos los elementos en conjunto.

Estas evaluaciones nos ha dando como resultado aspectos que validan el trabajo desarrollado en esta tesis, demostrando que la arquitectura desarrollada ofrece gran facilidad de modificación, basado en un desarrollo generalizado de todos sus elementos y a través de la reutilización de sus componentes en el desarrollo y funcionalidad de otros. Además hemos obtenido un alto grado de rendimiento, ofreciendo respuestas rápidas ante las solicitudes de servicios por parte del paciente.

No pretendemos reemplazar las actividades médicas, ni mucho menos ofrecer una solución que cure enfermedades; lo que si hemos pretendido ha sido ofrecer una solución tecnológica que ayude a los pacientes a darle seguimiento a su enfermedad, con una fácil interacción y sin afectarles a las actividades diarias que desarrolla. En pocas palabras facilitarle el día a día.

Este trabajo no finaliza aquí, existen muchas otras líneas que se pueden seguir para darle mayor robustez a esa "*multi-monitorización móvil*" que ha sido nuestro objetivo inicial.

ABSTRACT

Global change as a means of treating diseases has been a widely discussed topic in recent years. Many studies have been conducted in the hope of providing the patient a more manageable life and enabling the ill to continue carrying out their daily activities. This is no longer an exclusive activity of the medical and healthcare field and has become a multidisciplinary research field. New proposals have been initiated to change the way in which health services are provided to patients.

The evolution of different technological devices has become integrated in the treatment and follow-up of diseases. The extensive integration of mobile devices allowing for high technical characteristics and wide communication has emerged permitting a wide range of activities, which include greater functionality than the purposes for which they were initially developed.

This thesis has been developed with the purpose of offering technological solutions that allow patients improved and timelier follow-up and control of their disease, offering constant and timely responses. A conceptual *framework* called "**MoMo**" (**Mo**bile **Mo**nitoring) allows for the development of parameterized mobile applications. This is conducive to medical tracking through mobile devices, which are in daily use by the patient, and biometric devices that currently offer high technological performance.

The applications based on the *framework* are generic, adaptable, remote and mobile. ***Generic***, because they allow for the development of applications for multiple diseases following the conceptual design of the developed *framework*. ***Adaptable***, because they offer services tailored to each type of disease and personalize the user's characteristics, making the interaction more transparent. ***Remote***, because the medical staff can be aware of data obtained by the patient biometric devices through the mobile device in a non-intrusive manner. ***Mobile***, because development is based on the integration of small size, portable and wireless devices. These applications provide greater autonomy for the patient.

The *framework* proposes the integration of a set of patterns called "**MobiPattern**" for the design of user interfaces, specification of standardized modules functionalities and development, thus enabling reuse and redesign of the entire architecture. Similarly an ontological classification "**MoMOntology**" allows for the formalization of data. "**MoMoLayer**", which defines and implements a structure in layers, has been designated to facilitate the development and maintenance of all the elements involved in the design and functioning of the *framework*.

From the *framework* implemented gives rise to prototypes to be evaluated in patients with illnesses that can be monitored, specifically diabetes and arterial tension. Assessments have been obtained through questionnaires that evaluate aspects of design, content and usefulness of the end prototypes. In addition to the patients' evaluations, assessment methods have evaluated *software* architecture. These methods have been valuable in assessing qualities,

such as the ability of modification, architecture, usability and efficiency of all the combined elements.

These assessments have resulted in aspects that validate the work developed in this thesis, showing that the developed architecture provides great ease of modification, based on a widespread development of all its elements and through the reuse of components in the development and functionality of others. We have also obtained a high degree of performance, offering rapid responses to patients' requests for services.

It is not the intent of this thesis to replace medical activities, much less offer a solution to cure diseases. What has been attempted in this study is to offer a technological solution that helps patients to follow their illness with easy interaction and without affecting their daily activities. This work does not end here; many other lines of study can be followed to give greater strength to "mobile multi-monitoring."

AGRADECIMIENTOS

En primera instancia quiero agradecerles a mis directores de tesis, *Pepe* y *Ramón*, por su apoyo constante. Sus conocimientos, su orientación, su ardua manera de trabajar, su persistencia, su paciencia y su motivación han fundamentado mis bases como investigador. Siempre estaré en deuda con ambos por todo lo recibido durante el periodo de tiempo que ha durado esta Tesis Doctoral.

En segunda instancia, quiero agradecer a *Brigitte* y *Gael*, mi esposa e hijo, por la comprensión y paciencia que ha mostrado durante todos estos años. Años en los que a pesar de mis responsabilidades en el desarrollo de la tesis, he tenido siempre tiempo para dedicárselos a ellos.

En tercera instancia a mi *madre* y *hermana* por siempre estar preocupados por mí, y estar pendientes de mis progresos constantemente.

A la *Secretaría Nacional de Ciencia y Tecnología* y al *Instituto para la Formación y Aprovechamiento de los Recursos Humanos* de mi país, por haberme elegido como becario para desarrollar estudios de doctorado en esta Universidad. De igual manera a la *Universidad Tecnológica de Panamá*, por confiar en mí como docente, ofreciéndome el tiempo necesario para desarrollar mi investigación doctoral. Tengan por seguro que revertiré todo mi conocimiento adquirido en beneficio de mi Universidad y mi país.

Quiero agradecerle a mi compañero y amigo Jesús Fontecha por ser un excelente amigo y compañero de trabajo. El ambiente de trabajo creado ha sido excelente, su apoyo en todo momento, me han sido de gran ayuda. Siempre contará con mi amistad.

Quiero culminar diciendo que soy un hombre afortunado de contar con personas como las que he mencionado. Siempre lo tendré presente.

A todos, muchas gracias, estaré eternamente agradecido.

DEDICATORIA

Dedico este trabajo a las dos personas más importantes en mi vida y quienes han estado apoyándome.

Para mi esposa *Brigitte*, a ella especialmente le dedico esta tesis. Por su paciencia, por su comprensión, por su empeño, por su fuerza, por su amor, porque la quiero. Es la persona que más directamente ha sufrido las consecuencias del trabajo realizado. Realmente ella me ha dado la fuerza y soporte que me ha permito dar el máximo de mí. Siempre le estaré agradecido.

Para mi hijo, *Gael Enrique*, su nacimiento ha coincidido con el desarrollo de la tesis. Él es lo mejor que me ha pasado, y ha venido a este mundo para darme el último empujón para terminar el trabajo de esta tesis. Es sin duda mi más grande inspiración.

Mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

"Si deseas hacer algo en la vida, no dudes que encontrarás miles de motivos para no hacerlo, es mejor dejarte llevar por el deseo que experimenta tu mente."

El autor

CAPÍTULO PRIMERO

"Liz, es una señora a la que le acaban de diagnosticar "*diabetes*". Utiliza un medidor de glucosa para controlar el nivel de azúcar en su sangre. Liz tiene que realizar varias mediciones en el día para mantener controlado el nivel deseado de glucosa. De igual manera tiene que hacer todas las anotaciones en su cuaderno, cada vez que se realiza una medición, anotando las irregularidades y cambios presentados en el día. Si tiene algún problema tiene que llamar al médico para consultarle lo sucedido. A Liz le gustaría tener un seguimiento constante de sus medidas, obteniendo recomendaciones y mensajes cada vez que varía sus niveles de glucosa en sangre. De igual manera le gustaría tener a mano, sin necesidad de registrar en su cuaderno, todas las incidencias que ha presentado en los últimos días. Quisiera también que su doctor estuviera informado sin que ella le esté llamando. Esto le daría mejor calidad de vida y un seguimiento más constante de su enfermedad"

1 INTRODUCCIÓN.

Actualmente son muchos los estudios que se han desarrollado para atenuar el fuerte impacto que ha ocasionado el gran número de enfermedades crónicas sobre los seres humanos. Esta no es solamente una tarea de los especialistas en medicina, sino de todas aquellas disciplinas que, de una u otra manera, ejercen influencia sobre los avances tecnológicos de la era en que vivimos. El desarrollo de tecnologías que permiten la monitorización, el control, el adiestramiento y la posibilidad de pronósticos son algunos de los campos en los que se ha investigado. La computación ha demostrado ser una de esas disciplinas, desarrollando conceptos aplicables a situaciones en donde los que padecen enfermedades encuentran un alivio. El uso de dispositivos como el teléfono móvil, PDAs, etc.,

ha permitido la clasificación, procesamiento y recuperación de datos generados de pacientes frente a las fluctuaciones de los valores normales de señales vitales acordes a la enfermedad que padecen.

La comunicación entre estos dispositivos móviles y los dispositivos biométricos de medición, permite la captura, procesamiento y almacenamiento de datos, que son utilizados por otras aplicaciones para la generación de material educativo, de prevención y de seguimiento a pacientes. Con el uso de estas tecnologías los médicos podrían examinar a los pacientes tan solo consultando el perfil de cada uno de ellos, incluyendo el historial de mediciones, estadísticas, gráficos que contengan todos sus datos y, tras el chequeo, enviar los resultados al dispositivo móvil y actualizar el tratamiento.

La salud pública es fundamental para la competitividad de la Unión Europea (UE). Según la UE gasta un 8% de su PIB en salud, destinando alrededor de 100.000 millones de euros a las enfermedades respiratorias, 135.00 millones de euros a enfermedades cardiovasculares, y en torno a un 3% del PIB y 500.000 millones de euros en días de trabajo perdidos por problemas de salud y accidentes laborales. Los sistemas de asistencia sanitaria han tenido que arreglárselas con un número creciente de demandas de nuevas intervenciones en salud, que son cada vez más avanzadas tecnológicamente y más caras. Debemos tener en cuenta que los recursos disponibles son limitados, por lo tanto, debemos buscar un equilibrio en las necesidades individuales de cada paciente (Braun, 2010).

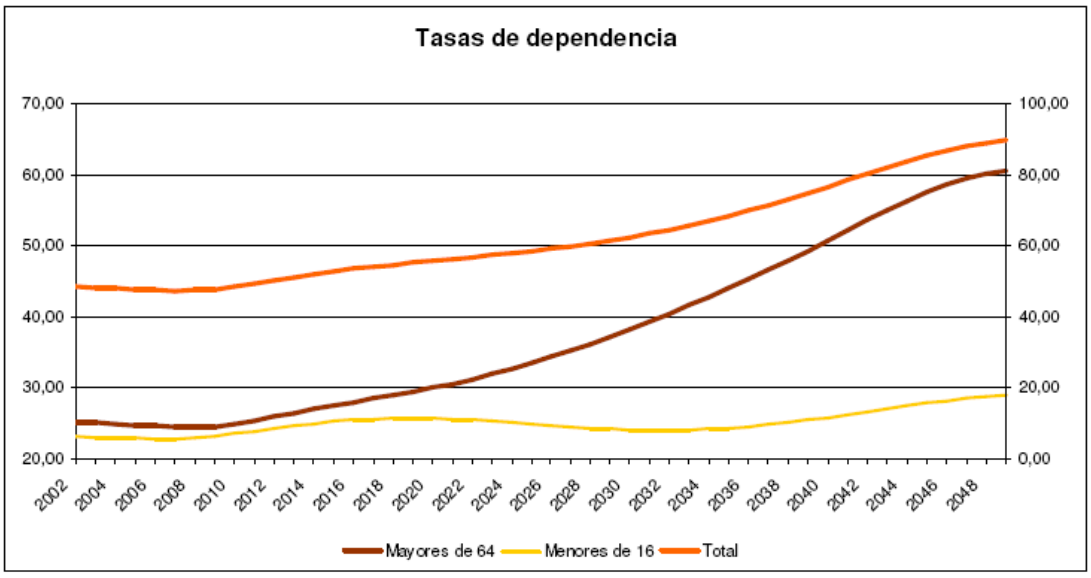
Por un lado, el cuidado médico y asistencial de pacientes representa una gran inversión para los gobiernos de cada país, y por el otro, existen actualmente muchas tecnologías que pueden facilitar esas actividades propias del cuidado y con las cuales podríamos obtener soluciones oportunas y en el menor tiempo posible.

1.1 DESCRIPCIÓN DEL PROBLEMA

El envejecimiento de la población española es un factor que incrementará en el futuro el porcentaje de población dependiente, ya que existe una estrecha relación entre dependencia y edad. Como se muestra la figura 1-1, según las proyecciones de población del INE (INE, 2011), el número de personas mayores de 65 años supondrá en veinte años más del 25% de la población total española, y dentro de ese grupo de población se producirá un fuerte aumento de las personas mayores de 80 años, edad a partir de la cual la tasa de prevalencia de dependencia crece notablemente.

El cuidado de las personas dependientes hoy en día está a cargo de sus propios familiares. Sin embargo, según los cambios que han presentado la sociedad, horarios de trabajo, menos días en el hogar y falta de organización en las familias hacen que este modelo de poco formal sea insostenible a medio plazo.

Cuando los pacientes están en un entorno asistencial, se encuentra en muchas ocasiones ailsado de su entorno social; una situación que complica enormemente la monitorización, o bien complica la intimidad del paciente. La monitorización de las personas que se encuentran en situaciones de riesgo, se realiza a través de equipos que no son móviles, o bien su transporte se hace complicado.



Fuente: Proyección de Población a Largo Plazo

Figura 1-1. *Evolución de la población española*

Otro problema es la falta de personal en los complejos residenciales para atender a todos los pacientes de una forma activa, ya que habitualmente, con el objeto de reducir costes, la relación entre el número de pacientes y personal médico es muy alta, y esto penaliza la atención a los pacientes. Las soluciones de control médico a través de tecnologías, facilita la monitorización remota y móvil de cada uno de los pacientes, consiguiendo que el paciente gane en calidad de vida (Gonzalez, 2009).

La generación de aplicaciones para la monitorización de pacientes debe ser *genérica*, *adaptable*, *remota* y *móvil*. ***Genérica,*** porque permite el desarrollo de aplicaciones para múltiples tipos de enfermedades, siguiendo el diseño conceptual del *framework* desarrollado. ***Adaptable,*** porque ofrece servicios ajustados a cada tipo de enfermedad y particularizada a las características de cada usuario, haciendo la interacción lo más transparente posible. ***Remota,*** porque el personal médico será capaz de conocer todos los datos obtenidos por los dispositivos biométricos de los paciente a través del dispositivo móvil de manera no intrusiva. ***Móvil,*** ya que el desarrollo está basado en la integración de dispositivos de pequeño tamaño, portátiles e inalámbricos. Esto le da mayor autonomía al paciente.

El desarrollo de esta tesis aportará, no sólo información de los valores biométricos, sino que se podrán configurar actividades de control para cada uno de los pacientes. De este modo, no sólo se le harán llegar los valores biométricos del paciente al personal médico, sino que se le harán llegar notificaciones sobre las situaciones de riesgo que se den al paciente.

1.2 METAS MARCADAS

Esta tesis busca ofrecer una solución a la monitorización de pacientes a través de un conjunto de sensores, que comuniquen los valores biométricos del paciente a un dispositivo móvil que acompañará a este.

Por otra parte, el proyecto pretende dotar de inteligencia al dispositivo móvil encargado de realizar la tarea de captura de datos desde el dispositivo biométrico, de tal forma, que a través de reglas de comparación de datos, el dispositivo pueda determinar el estado del paciente. En el caso de que se llegue a la conclusión de que el paciente se encuentre en una situación de riesgo, el dispositivo deberá informar al personal médico o algún familiar, a través de un mecanismo de alarma, para que este actúe si lo considera necesario permitiendo que el tiempo de intervención en una situación de riesgo de un paciente se reduzca al mínimo. De esta forma, si está ocurriendo una situación de riesgo, el personal médico queda avisado de inmediato. De la misma forma, al ser avisado, se eliminan los tiempos y personal empleado en hacer rondas para comprobar el estado de los pacientes.

Con ello se facilita la independencia de los pacientes mejorando su calidad de vida. Ambos aspectos marcan un requisito muy importante en la definición de la tesis, y es que la monitorización debe permitir la movilidad del paciente permitiendo una monitorización continua. Los datos y las notificaciones sobre situaciones de riesgo deben, en la medida de lo posible, seguir llegando al personal médico.

Facilitar la movilidad del paciente no sólo debe ser un aspecto a tener en cuenta al momento del desarrollo de la arquitectura *software*, sino también en la selección del *hardware* a utilizar y sobre todo de los sensores.

La psicología del paciente, es un factor que influye enormemente en la calidad de vida del paciente, y tener un cuerpo lleno de cables no ayuda que el paciente se sienta libre. Por tanto se ha definido como requisito adicional, que para que la monitorización permita una completa movilidad, los sensores deberán transmitir de forma inalámbrica los valores biométricos recogidos del paciente.

Estos dispositivos se encargarán de acumular todos los datos de los sensores, para que en cada cierto intervalo de tiempo puedan enviarlos a un servidor central donde sean procesados y almacenados, de tal forma que estén a disposición del personal médico.

Se trata de una monitorización "***continua***" del paciente, de tal forma que el personal médico, desde su puesto de trabajo o bien desde cualquier punto con acceso a los datos recogidos, pueda conocer en cada momento los valores biométricos del paciente. El conocimiento en tiempo real de la evolución del paciente, permite detectar cuadros de riesgo sobre sus datos y se puede realizar un diagnostico preventivo de la situación de este. Esto permite que la medicina pase a una fase preventiva, conociendo con anticipación lo que le puede ocurrir a cada paciente.

La solución consiste en el desarrollo de un *Framework* conceptual (llamado **MoMo**, por sus siglas en ingles ***Mo****bile* ***Mo****nitoring*) que permite el desarrollo de arquitecturas *software* para dispositivos móviles, el cual se encargará de recoger la información de los sensores configurados por el personal médico para el paciente. Estos datos serán almacenados y evaluados en busca de situaciones de riesgo en el paciente. Finalmente, estos datos y las notificaciones de situaciones de riesgo serán enviados al servidor central para que estén disponibles al personal médico.

Como hemos mencionado anteriormente, se pretende ofrecer una solución tecnológica, no solamente mediante un control de la enfermedad en su día a día, sino apoyándose en el uso de dispositivos móviles (por ejemplo, el teléfono móvil) que representa el avance tecnológico con el que mayormente interactúa una persona de manera constante y transparente. Esto permite la generación de aplicaciones parametrizadas para uso en dispositivos móviles, basada en una serie de requerimientos para su implementación.

El uso de tecnología móvil representa una gran oportunidad para los cuidados médicos, ya que ofrece la posibilidad de mejorar la calidad de vida de los pacientes, que podrían ser atendidos de forma remota. Además, este es considerado el dispositivo electrónico más extendido en el mundo y del cual se utiliza un bajo porcentaje de su capacidad de procesamiento.

Con esto se busca la creación y desarrollo de un *framework* para la generación de aplicaciones, que permita la monitorización móvil de pacientes, para diferentes tipos de enfermedades, facilitando la comunicación entre el paciente y el doctor (Villarreal, Bravo et al., 2010). Además, este *framework* provee una monitorización constante, soportada en una arquitectura automática basada en su perfil individual, autocontrol y módulos educativos para cada enfermedad.

El *framework* está compuesto por cuatro elementos importantes: la estructura del *Framework* para la monitorización móvil, una estructura por capas, servicios de formalización y procesamiento de datos y un sistema de multi-monitorización móvil.

1.3 HIPÓTESIS DE TRABAJO

La hipótesis de partida que definen el marco en el que se ha desarrollado la investigación de esta tesis doctoral es la siguiente:

Es posible desarrollar una solución tecnológica para la generación de módulos y aplicaciones parametrizadas que faciliten la monitorización móvil de pacientes permitiendo el autocontrol sanitario a través de dispositivos móviles y biométricos; mediante guías un *framework conceptual* que guie el desarrollo de dicha solución.

1.4 OBJETIVOS

La hipótesis antes enunciada implica la definición de una serie de objetivos y sub-objetivos a cumplir en el contexto de la presente tesis:

1.4.1 Objetivo General

Diseñar y desarrollar un *framework conceptual* para la generación de módulos y aplicaciones parametrizadas que permita la monitorización móvil de pacientes, en la que intervienen diferentes dispositivos y servicios, los cuales actúan dependiendo del tipo de enfermedad, el perfil y las señales vitales de cada paciente.

1.4.2 Objetivo Específico 1

Diseñar y desarrollar un modelo ontológico, orientado a la definición del dominio en estudio y a la descripción de cada uno de sus elementos. Además, las ontologías son interpretadas por los módulos para permitir la interoperabilidad entre cada uno de ellos y la aplicación final.

Meta 1.1: Realizar una revisión sobre la existencia de ontologías desarrolladas relacionadas a la monitorización de pacientes.

Meta 1.2: Diseñar un modelo ontológico con la propuesta de nuevas ontologías, la utilización de algunas existentes o la modificación de una de ellas.

1.4.3 Objetivo Específico 2

Diseñar y desarrollar un *framework conceptual* de referencia para la implementación de módulos que permita la monitorización móvil de pacientes, independiente de la tecnología, así como la integración dinámica de nuevos elementos del entorno.

Meta 2.1: Evaluar cada una de las propuestas que se han presentado referente a monitorización móvil de pacientes.

Meta 2.2: Evaluar cada una de las tecnologías de comunicación y procesamiento con que cuentan los diferentes dispositivos móviles existentes, para seleccionar los más adecuados para el desarrollo de nuestro *framework*.

1.4.4 Objetivo Específico 3

Definir y desarrollar una distribución en capas que demuestre la eficiencia en cuanto a comunicación entre cada uno de los elementos del *framework*.

Meta 3.1: Realizar una revisión sobre la utilización de la distribución en n-capas de los elementos de un sistema.

Meta 3.2: Definir una arquitectura distribuida en capas, que facilite el mantenimiento y actualización del *framework* propuesto.

1.4.5 Objetivo Específico 4

Definir unidades de diseño o patrones propios para la generación de módulos asociados a cada enfermedad y al perfil individual del paciente. Además las unidades de diseño deben permitir definir la ubicación de cada uno de los elementos de la arquitectura *software* generada.

Meta 4.1: Realizar una revisión sobre la utilización patrones para la generación de aplicaciones.

Meta 4.2: Evaluar esos patrones de aplicaciones fijas para aplicaciones móviles.

Meta 4.3: Reajustar y definir patrones para la generación de aplicaciones móviles en entornos médicos.

1.4.6 Objetivo Específico 5

Definir una estructura de relación entre cada uno de los módulos generados por el *framework*.

Meta 5.1: Relacionar cada uno de los elementos del *framework* propuesto, para las respectivas pruebas en tiempo de ejecución.

1.4.7 Objetivo Específico 6

Definir y desarrollar un modelo de comunicación entre dispositivos móviles y biométricos.

Meta 6.1: Probar la arquitectura *software* generada a través de conexiones remotas, ya sea a través de la red de telefonía móvil o red inalámbrica.

Meta 6.2: Realizar pruebas de comunicación entre los dispositivos móviles elegidos y los dispositivos biométricos existentes.

1.4.8 Objetivo Específico 7

Validar la propuesta para la generación de módulos accesibles desde dispositivos móviles.

Meta 7.1: Realizar validaciones en diferentes entornos, asociados cada uno de ellos al uso de la arquitectura *software* generada mediante su uso por parte de pacientes y médicos.

1.5 MATERIALES Y DISPOSITIVOS

Presentamos una clasificación de tecnologías que permitan la adquisición, procesamientos, interpretación y comunicación de información vital, facilitando la monitorización móvil de pacientes a través la arquitectura *software* generada a partir del *framework*.

Para el desarrollo de nuestra arquitectura, hemos clasificado los materiales que utilizaremos en tres grupos: dispositivos móviles y biométricos, tecnologías de desarrollo y tecnología de comunicación. Hemos evaluado cada una de las prestaciones tecnológicas que ofrecen haciendo las oportunas pruebas, para su implementación.

a. Dispositivos Móviles:

Dado el variado número de niveles de funcionalidad asociado con dispositivos móviles, era necesario hacer una clasificación de los mismos, por ello en el 2005, T38 y DuPont Global Mobility Innovation Team[1] propusieron los siguientes estándares para la definición de dispositivos móviles.

- ***Dispositivo Móvil de Datos Limitados (Limited Data Mobile Device)***: teléfonos móviles clásicos. Se caracterizan por tener una pantalla pequeña de tipo texto. Ofrecen servicios de datos generalmente limitados a SMS y acceso WAP.

[1] Fuente: http://www.nationmaster.com/encyclopedia/Mobile-device

- ***Dispositivo Móvil de Datos Básicos (Basic Data Mobile Device)***: se caracterizan por tener una pantalla de mediano tamaño, menú o navegación basada en iconos, y ofrecer acceso a emails, lista de direcciones, SMS, y, en algunos casos, un navegador web básico. Un típico ejemplo de este tipo de dispositivos son los teléfonos inteligentes ("smartphones").

- ***Dispositivo Móvil de Datos Mejorados (Enhanced Data Mobile Device)***: se caracterizan por tener pantallas de medianas a grandes (por encima de los 240 x 120 pixeles), navegación de tipo *stylus*, y que ofrecen las mismas características que el "Dispositivo Móvil de Datos Básicos" (Basic Data Mobile Devices) más aplicaciones nativas como aplicaciones de Microsoft Office Mobile (Word, Excel, PowerPoint) y aplicaciones corporativas usuales, en versión móvil, como *Sap*, portales intranet, etc.

El teléfono móvil es un dispositivo inalámbrico electrónico basado en la tecnología de ondas de radio, que tiene la misma funcionalidad que cualquier teléfono de línea fija (Baz Alonso, Ferreira Artime et al., 2008). Su principal característica es su portabilidad, ya que la realización de llamadas no es dependiente de ningún terminal fijo y no requiere ningún tipo de cableado para llevar cabo la conexión a la red telefónica. Aunque su principal función es la comunicación de voz, como el teléfono convencional, su rápido desarrollo ha incorporado funciones adicionales como mensajería instantánea (SMS), agenda, juegos, cámara fotográfica, agenda, acceso a Internet, reproducción de video e incluso GPS y reproductor mp3.

Por otro lado los *Smartphone*s o teléfonos inteligentes son dispositivos electrónicos que funcionan como un teléfono móvil con características similares a las de un ordenador personal. Es un elemento a medio camino entre un teléfono móvil clásico y una PDA ya que permite hacer llamadas y enviar mensajes de texto como un móvil convencional pero además incluye características cercanas a las de un ordenador personal. Una característica importante de casi todos los teléfonos inteligentes es que permiten la instalación de programas para incrementar su funcionalidad. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero (Baz Alonso, Ferreira Artime et al., 2008).

Los teléfonos inteligentes se distinguen por muchas características, entre las que destacan las pantallas táctiles, un sistema operativo, así como la conectividad a Internet y el acceso al correo electrónico. El completo soporte al correo electrónico parece ser una característica indispensable encontrada en todos los modelos existentes.

Actualmente existen en el mercado diversidad de móviles con características de procesamiento cada vez más sorprendentes. Esto facilita el desarrollo de aplicaciones, aunque en su mayoría los móviles restringen su acceso para desarrollar aplicaciones propias según las necesidades del usuario.

Por ello, y aprovechando la variedad de móviles que existen en el mercado, hemos optado por desarrollar nuestras aplicaciones para móviles basados en **Android**. En la Tabla 1-1 se ha hecho una pequeña clasificación de las principales marcas con sus respectivos modelos[2].

[2] Fuente http://www.android.es/movilesandroid.html#ixzz1eBWTIPlc

Decimos pequeña porque listar todos los dispositivos móviles con SO **Android**, llevaría un capítulo entero.

Marca	Modelos
Acer	Acer A1
Dell	Dell Ophone mini3
GeeksPhone	GeeksPhone ONE
General Mobile	DST L1
Haier	Haier H7
Huawei	Huawei U8230
HTC	HTC Click – Fiesta – HTC Tattoo, HTC Dream – G1 – DevPhone1, HTC Hero – G2 Touch, HTC Lancaster, HTC Magic – Sapphire – myTouch 3G – Ion
Lenovo	Lenovo O1 Ophone
LG	LG GW620 Eve
Motorola	Motorola Heron, Motorola Morrison – Cliq, Motorola Sholes, Motorola Sawgrass
Philips	Philips V808
Samsung	Samsung Bigfoot, Samsung Galaxy
Sony	Sony Ericsson XPERIA X3 Rachael
Otros	HighScreen PP5420, Innocomm Skate, Sciphone N12, ZTE Android

Tabla 1-1. *Clasificación resumida de modelos de móviles con Android SO.*

b. Servidor Central

En el servidor central es donde se alojarán los elementos de la arquitectura *software* y que se comunicará a través de red con los dispositivos móviles del médico y del paciente, facilitando la transmisión y recepción de los datos involucrados en el proceso de monitorización.

c. Dispositivos Biométricos:

Para el desarrollo de esta tesis doctoral, contamos con dispositivos biométricos que permite la lectura de las mediadas vitales del paciente, lo que facilita la comunicación con el dispositivo móvil y su futura interpretación. Algunos de estos dispositivos son glucómetro, tensiómetro y acelerómetro explicados a continuación:

- **Glucotel**: glucómetro que cuenta con tecnologías Bluetooth que permite la trasmisión de medidas de glucosa en sangre, facilitando la transmisión en tiempo real de los resultados. Este dispositivo proporciona disponibilidad inmediata de los

resultados en línea desde una base de datos. Permite la comunicación con el teléfono móvil y ordenadores.

- **PressureTel**: tensiómetro con tecnología Bluetooth, para medir la tensión arterial de pacientes. Cuenta con una aplicación en Java para integrar en teléfonos móviles.

- **Omron 705IT**: monitor que mide la presión sanguínea. Permite la comunicación con el teléfono móvil, a través de tecnología inalámbrica.

- **WiTilt v3**: este es uno de de los acelerómetros inalámbricos de SparkFun. El WiTilt v3 incorpora muchos nuevos rasgos incluyendo la batería LiPo, el cargador empotrado, el recinto, el giroscopio de eje y muchos más. Cuenta con lectura inalámbrica con tecnología Bluetooth. Permite la comunicación con los teléfonos móviles.

d. Tecnologías de desarrollo:

Como grupo de investigación se han hecho aplicaciones para dispositivos móviles basados en JME, NFC y RFID. Existen en el mercado diferentes sistemas operativos para dispositivos móviles dentro de los que podemos encontrar Symbian OS, Windows Mobile, IPhone OS, Palm OS, Android and Blackberry OS. En la tabla 1-2, mostramos las características técnicas de cada uno de ellos basados en aspectos de desarrollo.

Para el desarrollo de esta tesis, hemos utilizado ***Android***[3] como sistema operativo, debido a su flexibilidad, estandarización y la presencia de variedades de dispositivos móviles en el mercado. ***Android***, es un sistema operativo orientado a dispositivos móviles basado en una versión modificada del núcleo Linux. Inicialmente fue desarrollado por ***Android Inc***., compañía que fue comprada después por Google, y en la actualidad lo desarrollan los miembros de la Open Handset Alliance (liderada por Google).

Esta plataforma permite el desarrollo de aplicaciones por terceros a través del SDK, proporcionado por el mismo Google, y mediante el lenguaje de programación Java. Una alternativa es el uso del NDK (Native Development Kit) de Google para emplear el lenguaje de programación C. Las ventajas más apremiantes encontradas en Android son:

- Android es una plataforma para dispositivos móviles que contiene una pila de *software* donde se incluye un sistema operativo, middleware y aplicaciones básicas para el usuario.
- Busca el desarrollo rápido de aplicaciones, que sean reutilizables y verdaderamente portables entre diferentes dispositivos.
- Los componentes básicos de las aplicaciones se pueden sustituir fácilmente por otros.
- Cuenta con su propia máquina virtual, **Dalvik**, que interpreta y ejecuta código escrito en Java.

[3] Fuente: http://developer.android.com/index.html

- Permite la representación de gráficos 2D y 3D.
- Posibilita el uso de bases de datos.
- Soporta un elevado número de formatos multimedia.
- Servicio de localización GSM.
- Controla los diferentes elementos *hardware*: Bluetooth, Wi-Fi, cámara fotográfica o de vídeo, GPS, acelerómetro, infrarrojos, etc., siempre y cuando el dispositivo móvil lo contemple.

e. Tecnologías de Comunicación:

Esta corresponde a la forma en que se comunicarán tanto la arquitectura *software* generada con los dispositivos móviles, así como los dispositivos móviles con sus respectivos dispositivos biométricos. Para ello, y apoyándonos en las tecnologías y prestaciones que nos ofrecen estos dispositivos (los soportados por los móviles y dispositivos biométricos) implementaremos nuestra arquitectura basados en las siguientes tecnologías de comunicación:

- **Bluetooth:** es una especificación industrial para redes inalámbricas de área personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,5 GHz.
- **GPRS:** General Packet Radio Service (GPRS) o servicio general de paquetes vía radio es una extensión del Sistema Global para Comunicaciones Móviles (Global System for Mobile Communications o GSM) para la transmisión de datos no conmutada (o por paquetes).
- **LAN:** una red de área local, red local o LAN (del inglés local área network) es la interconexión de varias computadoras y periféricos.
- **WiFi** es un sistema de envío de datos sobre redes computacionales que utiliza ondas de radio en lugar de cables, además es una marca de la Wi-Fi Alliance (anteriormente la WECA: Wireless Ethernet Compatibility Alliance), la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11.

Además existen otras tecnologías de comunicación que pueden ser utilizadas en desarrollos posteriores que hagamos a la arquitectura propuesta. En esta tesis no se han utilizado. Estas tecnologías de comunicación son:

- **ZigBee** es el estándar de opción entre otras tecnologías inalámbricas debido a su conectividad de bajo poder, eficiente y capacidad de conectar un número grande de dispositivos en una sola red. ZigBee usa la frecuencia 2.4GHz. Esto permite la utilización de aplicaciones inalámbricas, usar un grupo de estándares de protocolos de comunicación de alto nivel, basadas en el estándar IEEE 802.15.4 para redes de área inalámbricas personales.
- **Wibree** es una tecnología de radiofrecuencia compatible con Bluetooh pero, a diferencia de ésta, ahorra hasta un diez por ciento de energía al estar en los dispositivos electrónicos que cuentan con conexión inalámbrica.

SO	Año	Tipo de dispositivo	SDK	Interfaz	Tipo de Aplicación	Firma	Dispositivos
Palm OS	1996	PDA's	Gratuito	Apuntador y teclado	Apps. Nativas y JME	Opcional	Familia Palm
Symbian OS	1998	Smartphones	Gratuito	Apuntador y Teclado	Apps. Nativas y JME	Obligatoria	Nokia, Sony Ericsson, Samsung, Siemens…
Balckberry OS	1999	Smartphones	Gratuito	Teclado	JME	Opcional	Familia Blackberry
Windows Mobile	2000	Smartphones y PDA's	Gratuito	Apuntador y teclado	Apps. Nativas y Compact *Framework*	Opcional	HP, HTC, Samsung, Dell
IPhone OS	2008	Smartphones y Tab	Gratuito	Touch	Apps. Nativas y JME	Obligatoria	Familia Apple
Android	2008	Smartphones y Tab	Gratuito	Touch y apuntador	Apps. Nativas y JME	Opcional	HTC, LG, Samsung…

Tabla 1-2. *Clasificación de los sistemas operativos para dispositivos móviles conocidos.*

1.6 METODOLOGÍA DE TRABAJO

El desarrollo de esta tesis se ha llevado a cabo con la implementación de diferentes metodologías que han permitido identificar hallazgos importantes para esta disciplina de estudio, por medio de la aplicación de diferentes técnicas y métodos de investigación y desarrollo. Respecto al cumplimiento de los objetivos específicos de la investigación, mencionados con antelación para el desarrollo de la tesis, se ha utilizado principalmente metodologías de ingeniería de *software* que permite la optimización y eficiencia en cada fase del desarrollo. Estas etapas de las metodologías de ingeniería de *software*, se corresponden a las etapas del método del desarrollo de *framework*, que son:

- Analizar el dominio,
- Presentación de la propuesta,
- Desarrollar la propuesta,
- Implementar la propuesta desarrollada,
- Evaluar la propuesta desarrollada.

Ambos métodos se complementarán al momento de diseñar, desarrollar e implementar el *framework* propuesto.

Primera fase. Revisión del estado del arte

Para la elaboración de la tesis, se ha llevado a cabo un estudio exhaustivo y profundo de la literatura sobre monitorización de pacientes, basándonos en aspectos como desarrollo de aplicaciones, área de estudio, uso de dispositivos, entre otros; a través de la selección de artículos publicados con el objetivo de resumir el estado actual de la investigación.

Esta primera fase contempla dos aspectos, el primero una evaluación conceptual sobre teorías, técnicas y material informativo sobre conceptos asociados a cuidado de pacientes, clasificación de enfermedades, uso de dispositivos; y por otro lado, la evaluación de los trabajos desarrollados en estos campos. La elaboración del estado del arte nos ha permitido identificar: *los principales investigadores que trabajan en este campo*, *los últimos grandes avances y descubrimientos*, *importantes lagunas en la investigación*, y *finalmente identificar las limitaciones de las investigaciones realizadas*, *proponiendo nuevos diseños y mejorando aspectos encontrados*.

Segunda fase. Presentación de propuesta del Marco de Trabajo

En la segunda fase se ha presentado una propuesta de una solución que supere los problemas que han sido detectados en la etapa anterior. Para ello se ha propuesto el desarrollo de la arquitectura de monitorización de pacientes a través de dispositivos móviles, en donde se ofrece una solución a los que necesiten cuidados asistenciales oportunos y puntuales para facilitarles el día a día con el uso de nuestro *framework* y de las aplicaciones que este genere. Además se ha propuesto un modelo en capas que facilita la comunicación entre cada uno de los elementos del *framework*, permitiendo así un desarrollo más organizado y en situaciones futuras una actualización acorde a la estructura desarrollada.

Con ello se ha clasificado mediante ontologías las principales domínios de aplicación que se contemplan al momento de desarrollar nuestro *framework*, distribuyéndolas en: *ontologías de perfil de pacientes*, *clasificación de enfermedades*, *clasificación y generación de módulos de control*, *clasificación de dispositivos de medición* y por último *generación de aplicaciones*. Esta generación de aplicaciones y la interrelación entre cada uno de sus elementos es posible mediante la definición y uso de guías de diseño (patrones), las cuales permiten, por un lado, la generación de módulos y aplicaciones acorde a un perfil y unidades de medidas de un paciente y, por el otro, la correcta ubicación de cada uno de los elementos generados (módulos y aplicaciones) en los distintos tipos de dispositivos móviles (teléfono móvil, PDA, etc.).

Podemos decir con ello que nuestra arquitecturas *software* generada no será estática con respecto a las primeras aplicaciones generadas por el *framework*.

Tercera fase. Desarrollo de la arquitectura software

La tercera fase corresponde al desarrollo de la arquitectura en donde se utilizarán las herramientas descritas anteriormente. En esta etapa se implementan todas las propuestas hechas en la fase anterior para el desarrollo del *framework* según el diseño conceptual propuesto.

Cuarta fase. Implementación y evaluación

Corresponderá a la cuarta fase, la implementación y evaluación de la arquitectura *software* generada a partir del *framework*, en donde se implementarán casos de estudio puntuales con la participación de especialistas médicos, personal dedicado a cuidado asistencial, desarrolladores de *software* y pacientes con enfermedades específicas, evaluando la efectividad, pronta respuesta y funcionalidad de la arquitectura *software*.

1.7 ESTRUCTURA GENERAL DE LA PROPUESTA DESARROLLADA

Para tener una primera aproximación introductoria al contenido de la tesis, se presenta la estructura general de propuesta desarrollada y enmarcada en los aspectos más relevantes para su posterior descripción en cada uno de los capítulos:

A. Estructura de *Framework* para la monitorización móvil: En la figura 1-2, se muestra la arquitectura desarrollada, en primer lugar, en el perfil (*PatientProfile*) se definen las características de cada paciente. El *framework* define una estructura común para los datos de cada paciente (*ID, nombre, dirección, número de teléfono, y otros*) basada en la información facilitada por el médico (Villarreal, Bravo et al., 2010). Luego, los datos son personalizados a cada paciente, a través de un perfil individual y en el cual se almacenarán los datos posteriores, permitiendo el control médico del usuario.

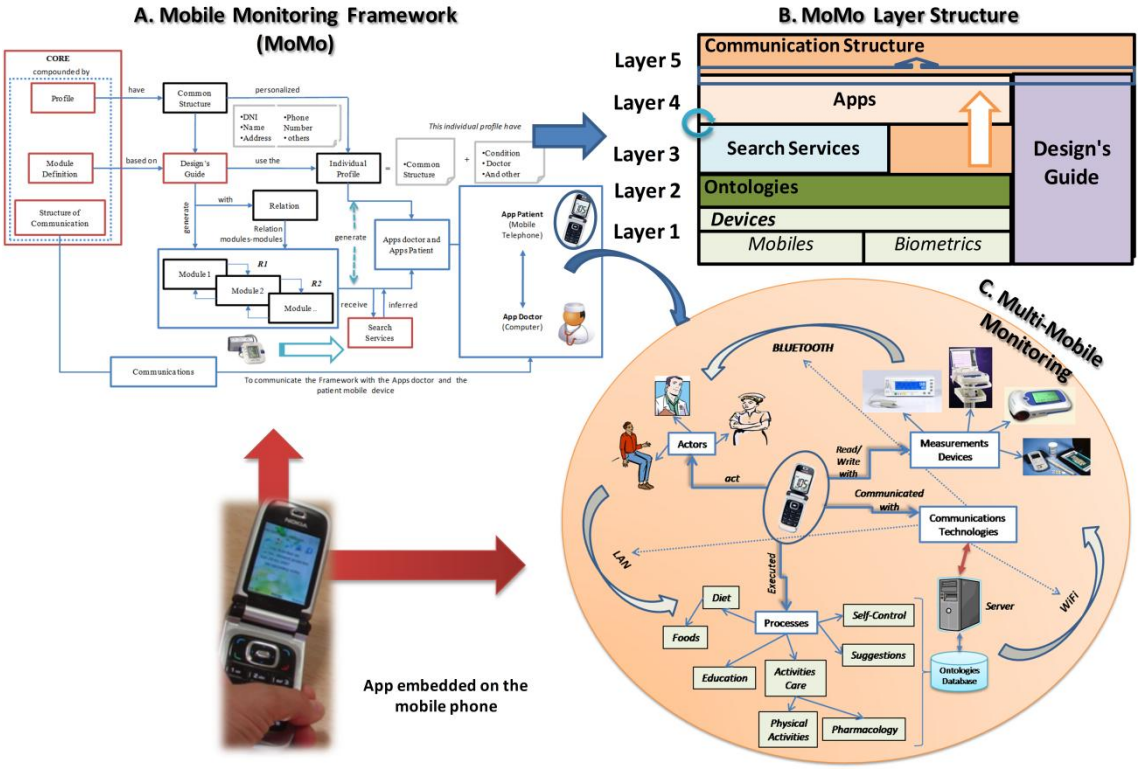


Figura 1-2. *Elementos que componen el Framework propuesta*

En segundo lugar, el *framework*, permite la definición de los módulos (*ModuleDefinition*) que serán diseñados e implementados, a través de **patrones**. Estos patrones establecerán las relaciones entre módulos dependiendo del perfil individual

del paciente. Con la definición de módulos, las relaciones entre todos ellos, y el perfil individual, es posible generar las correspondientes aplicaciones móviles.

En tercer lugar, la estructura de comunicación (*CommunicationStructure*) define el protocolo transmisión/recepción de datos para los dispositivos, así como el posterior comportamiento de las aplicaciones frente a cambios o variaciones de señales vitales de cada paciente. De un punto de vista físico, el dispositivo enviará, de manera inalámbrica los datos obtenidos del biosensor. Se incluye en este apartado la transmisión de móvil-dispositivo biométrico, móvil-servidor, entre otros.

B. Sistema de Multi-Monitorizacion Móvil: se busca con ello desarrollar una arquitectura de multi-monitorización en donde intervienen múltiples actores, procesos, dispositivos y tecnologías de comunicación.

Esto facilita el día a día de los pacientes, mejorando la comunicación con los especialistas médicos, aumentando la calidad asistencial, reduciendo costes, aumentando la eficacia de esta provisión de asistencia sanitaria y educándolos en cada actividad que realicen sobre el estado y auto-control de su enfermedad. Con ello el paciente se siente más independiente y más integrado a la sociedad, proporcionándole nuevas herramientas para la prevención, monitorización, diagnóstico, tratamiento y seguimiento de las personas.

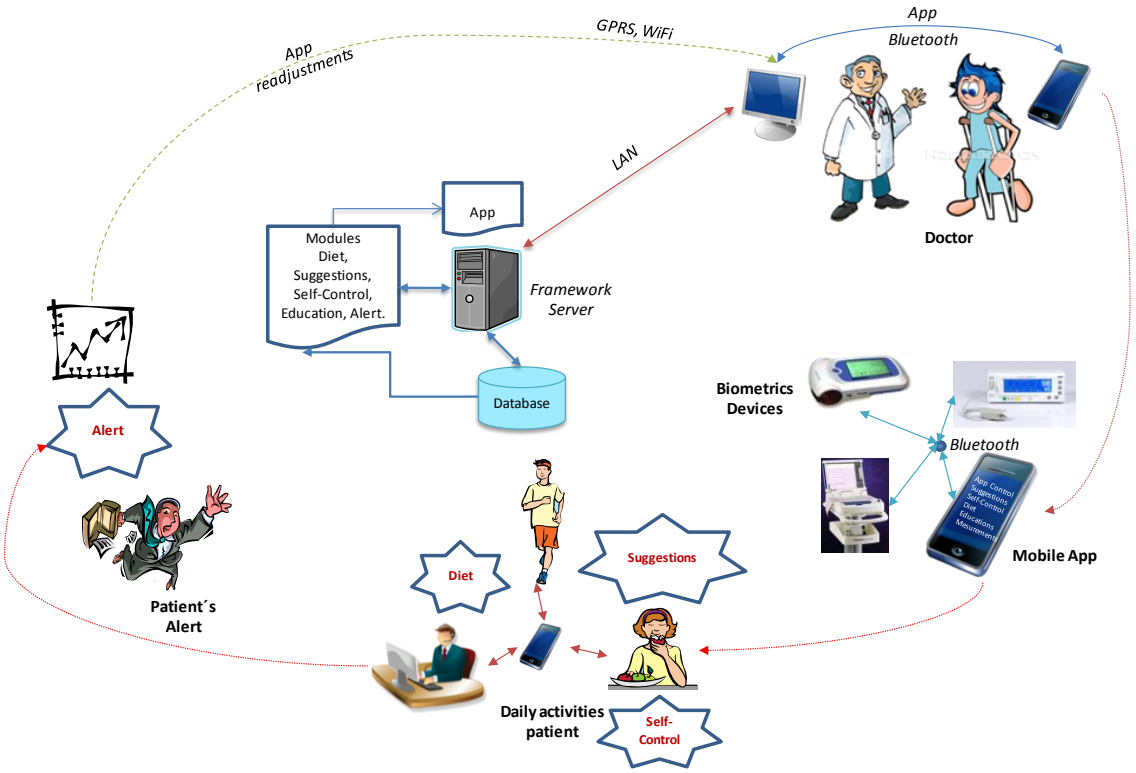


Figura 1-3. *Un caso de nuestra arquitectura*

En la figura 1-3, presentamos un esquema general, que nos da una panorámica del funcionamiento de la arquitectura software deseada, basados en los esquemas de la

figura 1-2, pero ajustados a un caso diario de un paciente que utiliza la arquitectura que pretendemos desarrollar. En un servidor central contamos con la información de cada paciente lo que permite el funcionamiento de las aplicaciones para monitorización móvil de pacientes, a través de dispositivos móvil y mediante la lectura de señales desde dispositivos biométricos (ver funcionamiento en figura 1-4). Un paciente visita a su médico (que siempre tomará las decisiones oportunas para cada tipo de enfermedad), el cual cuenta con una aplicación generada por el *framework* para el control de pacientes, en donde actualiza los datos del perfil, medidas y tipo de enfermedad que tiene el usuario. Se le hace una revisión del estado de su enfermedad, actualizando su perfil según cada visita. Una vez finalizada la actualización, la aplicación configura los correspondientes módulos de control dieta, recomendaciones, actividades, alertas, educación, entre otros, que serán organizados por la aplicación final que se embebe el dispositivo móvil del paciente a través de la tecnología de comunicación con la que cuenten en su momento (Bluetooth, WiFi, GPS, LAN, etc.).

El paciente regresa a sus actividades diarias mientras el dispositivo móvil está en comunicación con el o los dispositivos biométricos, el cual emite señales cuando el paciente efectúe la medición. Al detectar variaciones en los niveles deseados de sus signos vitales acordes a la enfermedad previamente analizada, ejecuta los módulos correspondientes que han sido embebidos en el dispositivo móvil. En caso que se detecte muy marcadas variaciones, por ejemplo variaciones del nivel de glucosa en pacientes diabéticos, este puede activar señales de alerta comunicándoles tanto al paciente como al médico de estas variaciones, lo cual permite un reajuste inmediato de los módulos previamente instalados. Los datos pueden ser almacenados localmente y transmitido solamente cuando se tenga una cantidad determinada almacenada. Para minimizar el tamaño de almacenamiento se definirá un método de compresión para ocupar el menor uso de banda de red con que se cuente.

Esta actualización puede hacerse ya bien sea a través de la red (WiFi, Ethernet, etc.) o visitando al médico en caso de que sean muy urgente estos cambios. Con ello la aplicación almacena la periodicidad de estas fluctuaciones a través de un servicio de búsqueda almacenará el funcionamiento de los módulos para control de la enfermedad.

En la figura 1-4 se muestra el flujo funcional del *framework*, generado a partir de la aplicación con que cuenta el médico. Se define la relación entre cada uno de los *patrones* y las *ontologías*, a través del núcleo del *framework*.

Una vez generado cada módulo, se crea una aplicación final que es instalada en el dispositivo móvil del paciente, el cual es sincronizado con el dispositivo biométrico con que cuente. De esta manera se inicia la funcionalidad de la aplicación de monitorización con el dispositivo biométrico y móvil.

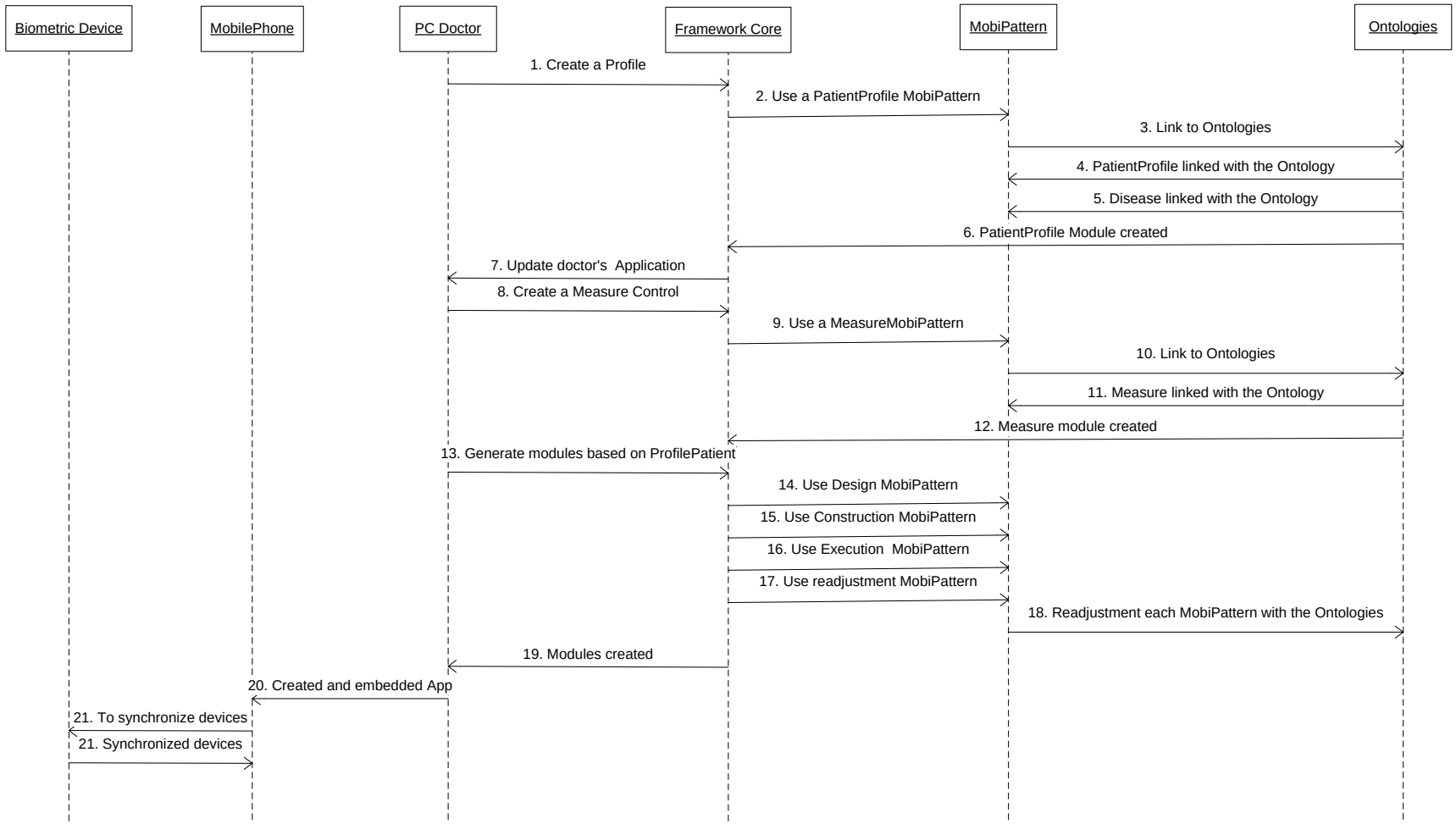


Figura 1-4. *Diagrama funcional del framework (MoMo)*

1.8 ESTRUCTURA DE LA MEMORIA

La estructura de la memoria está compuesta de la siguiente manera:

- **Capítulo primero. Introducción**: Este capítulo, especifica los aspectos introductorios de la memoria. Se especifican la motivación que nos llevo a desarrollarla, los objetivos que nos hemos propuestos para su desarrollo, así como una panorámica a la situación actual ante la problemática abordada en la tesis. Además, se define la metodología de trabajo usada para la obtención de los objetivos previamente especificados.

- **Capítulo segundo. Fundamentos teóricos. Visión tecnológica**: Este capítulo es acerca del marco teórico sobre el cual ha sido desarrollada la tesis. Se describen cada una de las tecnologías existentes para desarrollar aplicaciones, adaptándolas al tema de esta memoria. Se hace énfasis en la definición conceptual de temas relacionados al desarrollo de aplicaciones, definición del modelo de n-capas, especificación de las estructuras de patrones, y por último, la definición de metodologías y lenguajes para la definición de ontologías. Además se da una panorámica de Android como sistema operativo escalable en el desarrollo de aplicaciones móviles.

- **Capítulo tercero. Trabajos relacionados. Monitorización de pacientes:** En el tercer capítulo de esta memoria, se describen y analizan de forma cualitativa y cuantitativa, las aportaciones relevantes en las áreas de investigación relacionadas con esta tesis: aspectos de desarrollo de aplicaciones móviles para la monitorización de pacientes, investigaciones que aportan aspectos tecnológicos que no necesariamente sean para la definición de una aplicación final y propuestas de algunos investigadores en el campo de monitorización de pacientes.

- **Capítulo cuarto. MoMo: diseño conceptual del *framework*:** En este capítulo, se describen los elementos que forman el *framework* diseñado. Se define una distribución de los elementos en capas, llamada "*MoMoLayers"*, que describe la distribución uniforme del *framework*, facilitando su implementación y mantenimiento posterior. En segundo lugar, se describe las diversas ontologías que hemos implementado, facilitando el manejo y tratamiento de la información y que además sirven de base para la definición de la lógica descriptiva de la arquitectura *software*. Por último, se describen los patrones diseñados para la generación de aplicaciones parametrizadas, llamados "*MobiPattern*". Estos patrones facilitan la correcta definición estructural y funcional de cada uno de los módulos que componen una aplicación final de *MoMo*.

- **Capítulo quinto**. **Implementación de la Arquitectura *Software* a partir del *Framework***: En este capítulo se explica al detalle la implementación de todos los elementos definidos en el diseño conceptual. Se identifican cada uno de los módulos o elementos que intervienen en el desarrollo, así como la estructura y flujo de datos a utilizar. Además de la relación patrón-interfaz, se definen las zonas contenedoras que se utilizarán para definir cada uno de los elementos y su relación con el patrón definido. La implementación de todos estos elementos permite la generación de prototipos que serán evaluados en el capítulo posterior.

- **Capítulo sexto. Evaluación: Aplicando métodos y casos de estudio:** Luego de definida la arquitectura *software* y creados los prototipos que se pueden generar, presentamos el capítulo de evaluación. Para este capítulo se han realizado dos tipos de evaluaciones, la primera, basada en métodos de evaluación de arquitectura *software* propuestos por algunos autores, en donde se evalúan tres criterios de calidad: facilidad de modificación, facilidad de uso y grado de desempeño de la arquitectura. Este tipo de evaluación, se ha realizado entre los usuarios finales y programador, considerando la opinión de ambos tipos de usuarios. La segunda evaluación se ha hecho a usuarios finales con necesidad de monitorización, en donde se evalúan aspectos de calidad referentes a la satisfacción, eficiencia, entre otros. Ambos tipos de evaluaciones nos dan una panorámica de la funcionalidad y calidad de la arquitectura software generada, así como la identificación de aspectos que pueden ser corregidos en trabajos futuros.

- **Capítulo séptimo. Conclusiones y líneas futuras:** En este capítulo se resumen las lecciones aprendidas en el desarrollo de esta tesis. Además se exponen los objetivos alcanzados y su grado de consecución. Por último, se exponen las líneas futuras y posibles ampliaciones que complementen y den continuidad a este trabajo.

- **Apéndices.** Existen varios apéndices que contemplan la información aportada en esta memoria, organizados de la manera siguiente:

    - **Anexo 1. Criptografía. Cifrado de cadenas con DSE**: Se especifica el tipo de cifrado implementado en la arquitectura desarrollada para la seguridad de los datos trasmitidos.

    - **Anexo 2. Codificación para la obtención de de medidas de un paciente**: Se especifica los elementos (clases, propiedades, etc.) que permiten la obtención de medidas ya sea de forma manual como automática.

    - **Anexo 3**. **Codificación para la selección de un tipo de ejercicio desarrollar por un paciente**: Se especifica los elementos que permiten al usuario seleccionar algún tipo de ejercicio antes o después de tomada la medida de su enfermedad. Esto le permite al paciente controlar su enfermedad según las actividades que realiza en el día a día.

    - **Anexo 4**. **Definición de los elementos de la cuadrícula del menú de usuario**: Se define la estructura que forma el menú principal del usuario, lo que le da mayor facilidad de uso al momento de interactuar con la arquitectura.

    - **Anexo 5**. **Modelo de evaluación de la facilidad de uso, según el método SALUTA**: Se presenta un modelo de evaluación realizada a los usuarios, basada en el método SALUTA. El método evalúa la facilidad de uso de la arquitectura, según unas tareas desarrolladas en un contexto determinado.

El gráfico de la figura 1-5 muestra la distribución de todos los puntos mencionados previamente y que dan una mayor panorámica al contenido de esta tesis doctoral.

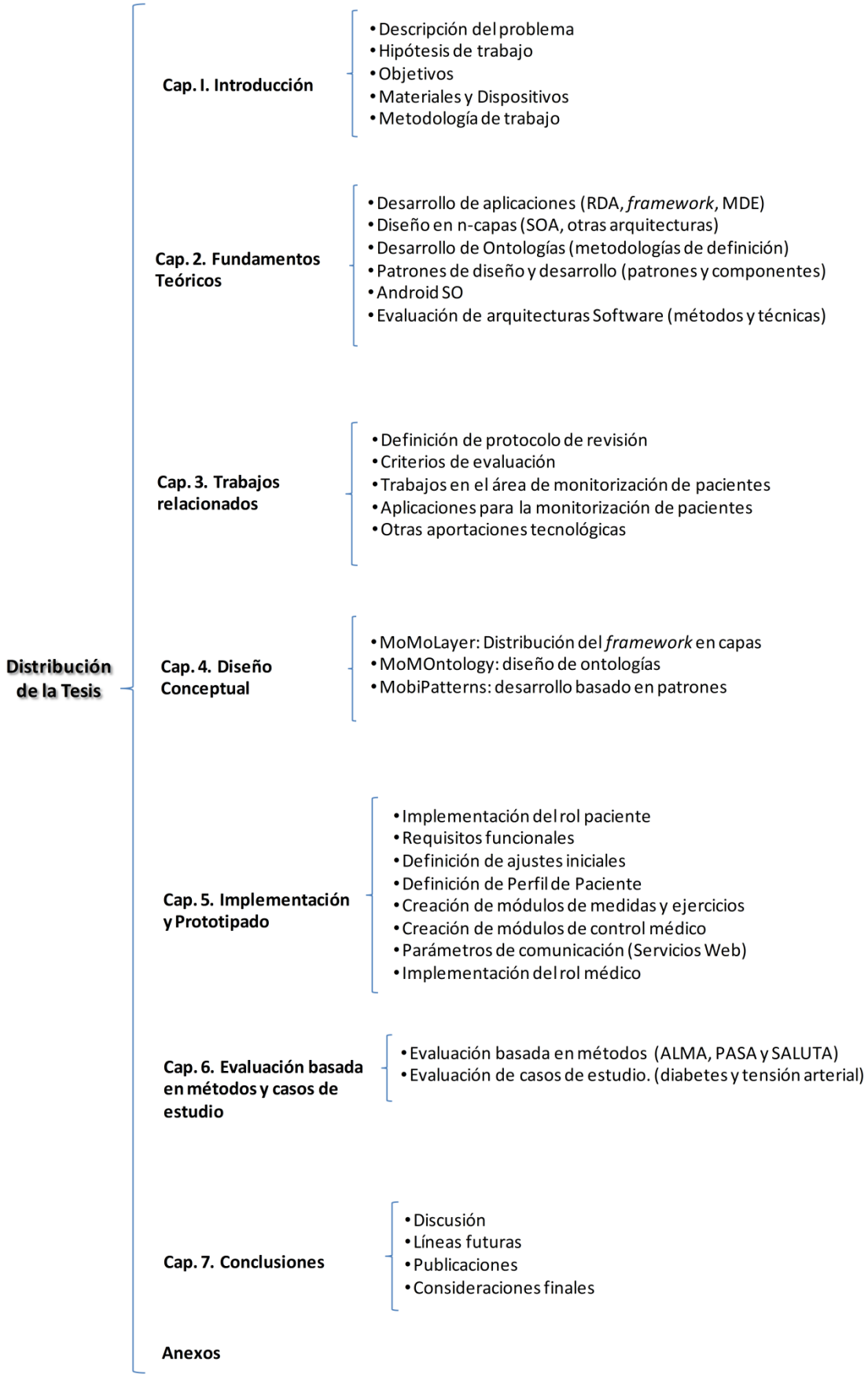


Figura 1-5. *Distribución de la memoria de tesis*

2

CAPÍTULO SEGUNDO

2 FUNDAMENTOS TEÓRICOS: VISIÓN TECNOLÓGICA.

2.1 INTRODUCCIÓN

El desarrollo de aplicaciones y tecnologías para la monitorización de pacientes a través de tecnología móvil ha sido estudiado desde diferentes campos. Cada campo utiliza la tecnología acorde a los requerimientos de los usuarios finales. Investigadores de diferentes áreas han hecho esfuerzos para entender y mejorar conceptos, tecnologías y aplicaciones en el campo de monitorización móvil de pacientes. Con el objeto de aclarar los conceptos, técnicas y tecnologías utilizadas en esta tesis, se presenta este capítulo de conceptos.

Se explican las tecnologías de desarrollo de aplicaciones, según el tiempo de desarrollo, tecnologías utilizadas y tipo de desarrollo. Se introduce el concepto de desarrollo basado en n-capas, para la distribución de los elementos en toda la arquitectura desarrollada. Por otra parte, se explica qué es una ontología, las principales metodologías y los lenguajes de representación de ontologías. Se hace una explicación sobre el uso de patrones en la generación de aplicaciones y además se hace énfasis en las características funcionales de **Android** OS, el cual ha sido seleccionado para desarrollar los prototipos de las aplicaciones móviles.

2.2 DESARROLLO DE APLICACIONES

Desarrollar una arquitectura *software*, así como los elementos que la definen, es el área central de esta investigación. Para ello analizaremos las diversas posibilidades que existen para desarrollar *software*, según las nuevas tecnologías con que se cuenta.

2.2.1 Rapid Application Development (RAD)

Desarrollo rápido de aplicaciones (**RAD**) es una metodología de desarrollo de *software* introducida en la década de los 90 y presentada en forma de libro por el tecnólogo de la información James Martin (Martin, 1991). El desarrollo rápido de aplicaciones fomenta la creación de un prototipo de *software* al estilo rápido y sucio que cumpla con la mayoría de las necesidades del usuario, pero no necesariamente todas. El desarrollo se llevará a cabo en una serie de ciclos cortos, llamados *cuadros de tiempo*. Cada uno de ellos profundizaría la funcionalidad de la aplicación un poco más. Las características que se aplicarán en cada casilla se acordaron de antemano y este es plan de juego rígidamente respetado. El fuerte énfasis en este punto vino de la mala experiencia con las prácticas de desarrollo en que los nuevos requisitos que tienden a ser agregados al proyecto fueron evolucionando, causó un caos masivo interrumpiendo los planes ya cuidadosamente preparados y los calendarios de desarrollo.

La metodología de desarrollo rápido de aplicaciones defiende el desarrollo a cargo de equipos pequeños, personas experimentadas, utilizando herramientas **CASE** (*Computer Aided Software Engineering*) para mejorar su productividad.

Los defensores de RAD creen que el desarrollo de prototipos rápidos era una buena manera para eliminar requisitos innecesarios o mal planteados del cliente mediante la obtención de una respuesta inmediata por ellos. Por lo general, esto sería visto como un desarrollo no deseado que podría causar estragos en los horarios acordados. La metodología de desarrollo rápido de aplicaciones, sin embargo, se convirtió en un estándar y aceptada en el proceso de desarrollo.

Con su énfasis en equipos pequeños y ciclos cortos de desarrollo, no es de extrañar que, en la doctrina de desarrollo rápido de aplicaciones, la reutilización de código fuera premiada también como un medio de ayudar a hacer el trabajo. Esto hizo que se adoptaran principios de desarrollo rápido de aplicaciones para abrazar lenguajes orientados a objetos (Martin, 1991).

Hoy en día, el desarrollo rápido de aplicaciones como una metodología formal, ya no es una práctica generalizada. Algunos sostienen, sin embargo, que es un caso de evolución revolucionario en los estadistas. Es un apoyo al paradigma orientado a objetos, y al uso de herramientas de ingeniería de *software* para mejorar la productividad del programador, que fue sin duda adelantada a su tiempo marcando su énfasis en equipos pequeños, ciclos cortos, el desarrollo iterativo y evitando los requisitos prolongados de una reunión previa, que comparte muchas similitudes con la programación extrema o metodologías ágiles de desarrollo.

Además de que denota una metodología formal de desarrollo *software*, el desarrollo rápido de aplicaciones se convirtió en una palabra de moda de comercialización y se aplicó por

casualidad a una gran variedad de productos de desarrollo de *software*. Para facilitar el rápido desarrollo, se insistió en la idea de reutilizar *software*. La noción de componentes de *software* comenzó a ser alimentada.

En el uso común hoy en día, la frase desarrollo rápido de aplicaciones ha perdido la mayor parte de su significado original, e incluso en las filas de los profesionales de TI, donde muchos no saben que se refiere a una metodología de desarrollo de *software* formal. Casi cualquier herramienta de *software* que se utiliza en la creación de otros programas se describe en la literatura de marketing como algo que facilita el desarrollo rápido de aplicaciones. Cuando se utiliza de manera informal en este sentido, la frase desarrollo rápido de aplicaciones por lo general indica que la herramienta en cuestión genera automáticamente parte del código del programa. Hoy en día las herramientas de *software* utilizadas por la mayoría de los programadores para desarrollar un nuevo *software* se llaman entornos de desarrollo integrados *(***IDEs***).* Casi todos ellos incluyen algunas características de desarrollo rápido de aplicaciones. Al crear un nuevo programa el ingeniero de *software* puede indicar qué tipo de aplicación desarrollar, por ejemplo, una aplicación de consola, un programa con una interfaz gráfica de usuario o una relacionada a una base de datos. A continuación el **IDE** generará una plantilla base de código que el programador toma como punto de partida para su trabajo propio.

2.2.2 Desarrollo basado en *frameworks*

En la actualidad se han llevado a cabo grandes esfuerzos para explotar las mejores prácticas del desarrollo de *software*, el diseño de patrones, el diseño basado en componentes así como de los marcos de trabajo de arquitecturas de productos *software* para llevar a cabo un diseño coherente y su respectiva implementación (Kaisler, 2005). Son muchas las posibilidades de desarrollo de *software*; todas ellas tienen en común el hecho de que son soluciones preestablecidas para problemas específicos en distintos niveles de granularidad y abstracción. Los programadores no construyen programas desde cero, sino que suelen emplear patrones apropiados, dentro del contexto de una problemática en particular. Sin embargo los patrones de diseño sirven para abordar problemas pequeños. Por ello, los patrones de diseño se combinan dentro de componentes *software* para resolver problemas más grandes. Los componentes son artefactos más complejos, que pueden estar formados por varios patrones de diseño. Los componentes, a su vez, están frecuentemente integrados en *frameworks*. Los *frameworks*, sin embargo tienen una característica especial: que son específicos de un dominio. Es decir, se desarrolla un *framework* con el propósito de implementar una estructura de solución para una clase particular de problema dentro de un dominio.

Los *frameworks* no son solamente los trozos de código reutilizable, también pueden ser especificaciones y diseños que pueden ser separados en partes y combinados posteriormente en diversas configuraciones. Estos son conocidos como los *framework* de modelos. Un *framework* de modelos es un conjunto de relaciones, restricciones o transformaciones aplicadas en diferentes estados del diseño que se obtienen a partir de soluciones probadas dentro de un mismo contexto y que resuelven una problemática similar (D'Souza and Wills, 1998). El desarrollo basado en *frameworks* ha sido una importante herramienta para el desarrollo de soluciones a problemas con características específicas, y se dice que es la

segunda generación de metodologías para el desarrollo de sistemas informáticos (Ambler, Nalbone et al., 2005).

Un *framework* es un término utilizado en la computación en general, para referirse a un conjunto de bibliotecas, utilizadas para implementar la estructura estándar de una aplicación. Todo esto se realiza con el propósito de promover la reutilización de código, con el fin de ahorrarle trabajo al desarrollador al no tener que rescribir ese código para cada nueva aplicación que desee crear.

Según Gamma, el *framework* determina la arquitectura de una aplicación (Gamma, Johnson et al., 1994). Este es un buen enfoque, ya que el *framework* se encarga de definir la estructura general, sus particiones en clases y objetos, las responsabilidades clave, así como la colaboración entre dichas clases y objetos. Todos estos parámetros son definidos por el *framework*, evitando que el usuario tenga que definirlos y se pueda enfocar en cosas específicas de la aplicación. "El *framework* captura las decisiones de diseño que son comunes a su dominio de aplicación" (Gamma, Johnson et al., 1994). Un *framework* no sólo promueve la reutilización de código sino también la reutilización de diseño.

Además un *framework* ayuda a que se desarrolle una aplicación de manera más rápida, ya que no pierde tiempo en algunos detalles de diseño que muchas veces quitan más tiempo del que tomó construir la aplicación. Además las aplicaciones que se construyen tienen estructuras similares, son más fáciles de mantener y consistentes para los usuarios.

En los últimos años, la aplicabilidad del uso de *frameworks* en situaciones de cuidados de salud, basados en el uso de dispositivos móviles ha adquirido gran interés por parte de los investigadores. Y es que con la existencia de un dispositivo de uso cotidiano, se pueden lograr grandes avances que facilitan las tareas de cuidado en diferentes áreas. Para ello hemos realizado una investigación sobre la existencia de *frameworks*, pero especialmente en el área de cuidados médicos y con el uso de dispositivos móviles.

Para el desarrollo de la esta tesis, utilizaremos el desarrollo basado en *framework,* combinando las características de desarrollo que ofrece para la obtención de un producto final más estable y eficiente.

2.2.3 Model-Driven Engineering (MDE)

2.2.3.1 MDA

La arquitectura dirigida por modelos (*Model Driven Architecture, MDA*) es un acercamiento al diseño de *software* propuesto por el OMG (*Object Management Group*). (OMG, 1999) Es una visión particular del OMG sobre MDE. Aunque MDE ofrece una aproximación más amplia al desarrollo basado en modelos, MDA propone muchos conceptos usados en MDE.

Según este estándar los modelos pueden ser clasificados en función de su nivel de abstracción definiendo tres tipos de modelos:

- **Computation Independent Model (CIM).** Un modelo CIM describe una aplicación en función de conceptos independientes de la computación, especificando los requisitos

del sistema. Estos modelos son independientes de la forma en la que el sistema posteriormente será implementado. De hecho los modelos CIM son principalmente usados por los expertos del dominio o consultores de negocio que no tienen por qué tener conocimiento de cómo los modelos y otros artefactos del proceso de desarrollo se usan.

- **Platform Independent Model (PIM).** Estos modelos se centran en describir el funcionamiento de una aplicación mediante un enfoque computacional pero sin emplear conceptos relacionados con ninguna plataforma de ejecución concreta (según MDA una plataforma es un conjunto de subsistemas y tecnologías que ofrecen un conjunto coherente de funcionalidad a través de interfaces y patrones de uso). Por lo tanto, una especificación es válida para más de una plataforma. Para describir un modelo PIM se puede usar un lenguaje de modelado de propósito general o un lenguaje específico del área en la que el sistema se usará.
- **Platform Specific Model (PSM).** Un modelo PSM describe un sistema desde el punto de vista de una plataforma concreta. Este tipo de modelos combina la especificación de la aplicación realizada en los modelos PIM con detalles relativos a cómo ésta se implementa en una plataforma concreta. Son por tanto válidos para una única plataforma, con la cual deben tener una relación directa que permita la generación de código desde ellos.

El proceso que propone MDA para el desarrollo de aplicaciones, para una plataforma concreta típicamente, consiste en la definición de un modelo PIM, independiente de la plataforma de ejecución, para ser transformado a un modelo PSM de la plataforma en cuestión, y, por último, ser transformado a código (o a otras estructuras de datos).

Las transformaciones que se ejecutan idealmente deben estar automatizadas y ser ejecutadas por herramientas, aunque en muchos casos debe haber cierta intervención humana en el proceso. El uso de los modelos CIM y la automatización de sus transformaciones apenas están descrito por la especificación de MDA, no quedando del todo claro qué se entiende como modelo CIM. En general se considera que estos modelos se usan sólo para especificar requisitos y obtener, de forma manual, los modelos PIM, aunque no se rechaza que esta transformación pueda ser automatizada.

Pueden notarse en la figura 2-1 los elementos que forman MDA y las funcionalidades o campos en los que se ha utilizado. Seleccionamos dentro de los campos de estudio la facilidad que ofrece en entornos de cuidados en el hogar.

Como se mencionó previamente, MDA es un enfoque para el desarrollo, integración e interoperabilidad de sistemas de información. De acuerdo a esto, ofrece un enfoque abierto y de tecnología neutral ante el reto del constante cambio de los negocios y de la tecnología (Miller and Mukerji, 2003). Los conceptos claves en MDA son *modelos*, *meta modelos* (que definen lenguajes abstractos por medio de los cuales se representan los lenguajes), y las *transformaciones* (que toman uno o más modelos y producen uno o más modelos a partir de estos) (ORMSCWhitePaper, 2001).

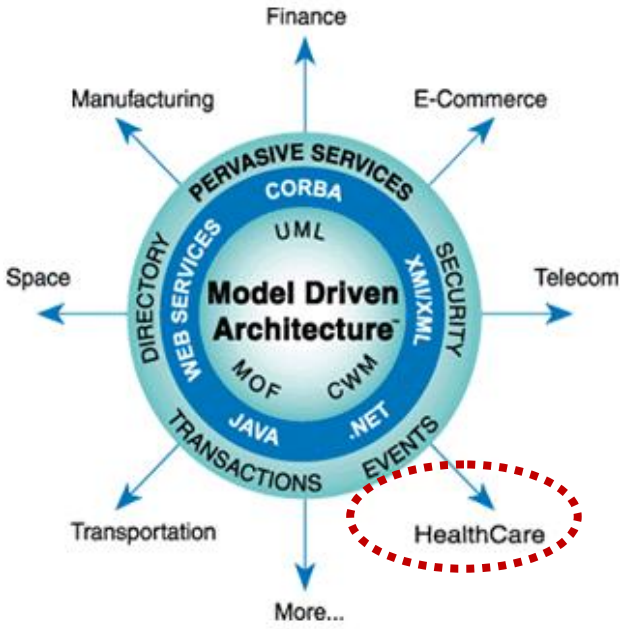


Figura 2-1. *Arquitectura de MDA.*

MDA ha creado gran interés, dado que asegura incrementar la productividad, la flexibilidad y la portabilidad de modelos de sistemas *software* (Miller and Mukerji, 2003). MDA define un enfoque que separa la especificación de la funcionalidad de un sistema de la especificación de la implementación de esa funcionalidad en una plataforma tecnológica específica. A su vez, define un enfoque para el desarrollo de *software* basado en modelado y *mapping* automático, a partir de los modelos hasta sus implementaciones (Bodart and Vanderdonckt, 1995).

La esencia de los fundamentos de MDA consiste en tres ideas, complementarias entre sí:

- *Representación directa*, desplaza el foco del desarrollo de *software* lejos del dominio de la tecnología para dirigir las ideas y conceptos al dominio del problema;
- *Automatización*, propone usar herramientas computacionales para mecanizar el proceso del desarrollo del *software*;
- *Estándares Abiertos,* los estándares han promovido el progreso de la tecnología, permitiendo la generación de un ecosistema de herramientas para propósitos generales y todo tipo de nichos especializados.

2.2.3.2 Aproximaciones a MDE

Como hemos visto en *Model-driven Architecture* (MDA) conceptos básicos que se centran en la arquitectura, en objetos y en modelos. El OMG ha incluido una visión de alto nivel en su documento de MDA frente a la utilización de estos artefactos en los modelos. Aunque MDA reconoce las cosas como un espacio de modelado más rico que la simple dicotomía entre PIM y PSM y la automatización de las transformaciones entre modelos, se queda lejos de la definición de un enfoque de ingeniería real.

Igual que MDA, el principio básico de *Model-driven Engineering* (MDE) es que todo es un *modelo*, en comparación con el principio básico de orientación a objeto (OO) que dice que todo es un *objeto (Bézivin, 2004)*. MDE, sin embargo tiene un alcance mayor que MDA, que

combina procesos y análisis de la arquitectura (Kent, 2002). Según MDA puede ser vista como una encarnación específica del enfoque MDE. MDE es un enfoque abierto e integrador que abarca muchos otros espacios tecnológicos de manera uniforme.

En un enfoque de desarrollo de *software* MDE (*Model-driven Engineering*) (Kent, 2002) los modelos constituyen el principal artefacto en el proceso de desarrollo. Actualmente se utiliza en el desarrollo de *software* (Kent, 2002).

El uso de MDE pretende facilitar el desarrollo de *software*, simplificando la especificación de aplicaciones gracias a la posibilidad de abstracción que ofrecen los modelos y generando una implementación ejecutable a partir de esta especificación. Además, MDE pretende gestionar el desarrollo de aplicaciones complejas permitiendo la especificación por separado de modelos correspondientes a las distintas vistas de un mismo sistema, que después serán integradas y transformadas.

Según (Atkinson and Kuhne, 2003) un propósito de MDE es la mejora de la productividad, dotando de más funcionalidad a los artefactos desarrollados, de un mayor tiempo de vida y una mejor adaptación ante cambios.

MDE ofrece una visión más amplia ya que combina proceso y análisis con arquitectura, considerando que MDA se centra excesivamente en las categorías PIM y PSM y sus transformaciones y olvida muchos otros aspectos relevantes. MDE tiene en cuenta la existencia de más de un criterio para la clasificación de los modelos frente al único criterio de abstracción tecnológica de MDA (MDA admite la posibilidad de que existan más criterios pero se centra en el criterio de abstracción) (Kent, 2002).

Kent (Kent, 2002) considera una serie de criterios adicionales para clasificar los modelos que dan lugar a un mayor número de perspectivas que CIM, PIM y PSM. Según esto se puede considerar como que cada uno de estos criterios formaría una dimensión en un sistema n-dimensional y las perspectivas corresponderían con cada una de las intersecciones entre las distintas dimensiones. Además, considera que los criterios de clasificación de modelos se pueden agrupar a su vez en tres grupos principales. El primero estaría formado por un único criterio que, al igual que ocurre con MDA, correspondería con la abstracción respecto de la tecnología de implementación. El segundo grupo se compondría de criterios existentes en el área de desarrollo de *software* orientado a aspectos. En este grupo se pueden encontrar criterios como el tema, que puede cubrir áreas desde el trato con los clientes hasta el procesado de órdenes o criterios como el aspecto, encargado del control de la concurrencia y la distribución. Por último, en un tercer grupo, se clasifican criterios relacionados con aspectos de gestión como autoría, versión, localización y participantes en los procesos modelados.

Definiremos la arquitectura final siguiendo los tres modelos especificados en MDA clasificados en función de su nivel de abstracción. Estos modelos nos permitirán obtener una panorámica de requerimientos y de implementación previas al desarrollo.

2.3 EL DISEÑO DE N-CAPAS

Las capas dentro de una arquitectura son un conjunto de servicios especializados que pueden ser accesibles por múltiples clientes y que deben ser fácilmente reutilizables. Las capas además, según el escenario y tipo de aplicación, están separadas físicamente.

La organización más conocida y aceptada es la distribución en tres capas, estas son: *capa de presentación, capa de negocio* y *capa de persistencia*.

La capa de *presentación* es la que ve el usuario, le muestra la información y captura la información del usuario. A la capa de presentación también se le conoce como interfaz de usuario. Esta capa se comunica únicamente con la capa de negocio. La capa de *negocio* es donde residen los programas , se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos (pesistencia), para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación. Por último, la capa de *persistencia*, es donde residen los datos y es la encargada de acceder a los mismos. Está formada gestores de bases de datos que almacenan los datos, reciben solicitudes de almacenamiento de información desde la capa de negocio.

Por su parte (Barbarán, Díaz et al., 2007), ha definido una arquitectura en capas, centrada en la vigilancia ambiental de las centrales nucleares donde los sensores son delegados en posición fija, equipado con dispositivos de medición de la radiación, por dentro y por fuera de la planta. Para ello define tres capas: capa de *comunicación*, capa de *interacción* y la capa de *alto nivel*.

La capa de *comunicación* establece las primitivas de comunicación y acceso a dispositivos físicos a través de PDAs y redes de sensores inalámbricos. La capa de *interacción*, es el puente entre la capa de comunicación, la de alto nivel y las capas de aplicación. Por último, la capa de *alto nivel*, está formada por dos componentes: el componente de *configuración* y el componente de *monitorización*. Estos componentes permiten a los programadores de aplicaciones realizar fácilmente acciones complejas tales como la monitorización y configuración de redes de sensores y MANETs.

Además, un grupo de la Universidad de UAE propuso el método "*Control de Acceso basado en Rol*", con sus siglas en inglés **RBAC** para redes *ad-hoc* en cuidados médicos. Este método puede darle a un usuario miembro un rol de *grant* basado en sus responsabilidades en la organización. Los modelos **RBAC** están recibiendo una atención cada vez mayor como un planteamiento general de control de acceso, pero el método **RBAC** no se ocupa de autenticación o seguridad de los datos (Memon, 2009).

El modelo n−capas de informática distribuida ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma en la mayor parte de las empresas pertenecientes a Fortune 1000[4].

[4] http://money.cnn.com/magazines/fortune/

Este cambio radical en los modelos de computación, desde los sistemas monolíticos basados en mainframe y los tradicionales sistemas cliente−servidor, hacia sistemas distribuidos multiplataforma altamente modulables, representa simplemente la punta del *iceberg* de lo que está por llegar en el mundo del desarrollo de aplicaciones, tal y como se pone de manifiesto en las últimas tendencias de las grandes empresas de tecnología, como *Sun* con su estrategia *Sun Tone*, o *Microsoft* con *.Net* (Patel, McRoberts et al., 2009) .

Dentro de las ventajas que se pueden obtener del desarrollo arquitecturas en n-capas están:

- Desarrollo de aplicaciones más robustas;
- Mantenimiento y soporte más sencillo de un componente de la arquitectura
- Mayor y mejor flexibilidad al añadir nuevos módulos para agregarle nuevas funcionalidades a la arquitectura
- Alta escalabilidad, permitiendo añadir nuevos *hardware* (dispositivos biométricos y móviles). No es necesario añadir más código ya que la arquitectura diseñada facilita este proceso.

Como tecnología, las arquitecturas de n−capas proporcionan una gran cantidad de beneficios para el desarrollo de aplicaciones que necesitan soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes.

A diferencia de lo que se pudiera pensar, el desarrollo en n−capas no es un producto o un estándar, es un concepto estratégico que ayuda a la construcción y despliegue lógico de un sistema distribuido. Los sistemas de n−capas subdivididos ayudan a facilitar el desarrollo rápido de aplicaciones y su posterior despliegue, con beneficios incrementales fruto de los esfuerzos del desarrollo en paralelo coordinado y del *outsourcing* inteligente, resultando un enorme decremento del tiempo de desarrollo y de sus costes.

El diseño para clientes ligeros minimiza los problemas de despliegue de las aplicaciones, mientras que maximiza la accesibilidad a la misma desde una amplia variedad de plataformas heterogéneas. Los *frameworks* basados en n−capas se crean para obtener las ventajas de los estándares abiertos de la industria que permiten a las aplicaciones resultantes operar en entornos distribuidos multiplataforma. Utilizando estos potentes estándares abiertos se permite a los integradores de sistemas (Patel, McRoberts et al., 2009) desarrollar soluciones.

El diseño de aplicaciones basado en n−capas considera a la red como un *pool* de servicios distribuidos, un concepto mucho más ambicioso que el simple acceso de un cliente a un servidor. La separación de la presentación, lógica de negocio y datos es realizada en un número indefinido de capas lógicas, permitiendo a cada capa ser desarrollada, mejorada, gestionada y desplegada de forma independiente. Esta es precisamente la base para el modelo de informática de red en n−capas. Las plataformas multicapa funcionan consistentemente a lo largo de un variado conjunto de *hardware*, permitiendo escalar las operaciones del negocio desde un simple portátil, hasta un dispositivo móvil, desde el dispositivo más simple hasta el más complejo de los servidores. Todas las aplicaciones basadas en n-capas permiten trabajar con clientes ligeros, tal como navegadores de internet, *WebTV*, teléfonos inteligentes, *PDAs* y muchos otros dispositivos

Los sistemas de n−capas utilizan técnicas de desarrollo basadas en componentes combinados con los estándares abiertos de Internet, para crear aplicaciones multiplataforma muy potentes con bajos costes, fáciles de mantener y con gran efectividad. Lo que realmente es útil en el modelo de n−capas es la posibilidad de distribuir objetos independientes sobre el número de capas que sean necesarias y enlazarlas dinámicamente, cuando sea necesario, para proporcionar una flexibilidad ilimitada a la aplicación (Chu, Juez-Nan et al., 2001).

El desarrollo de aplicaciones en n−capas es un proceso iterativo de división del problema en piezas manejables denominadas componentes. Estos componentes, o "*Componentes de Negocio − Business Objects*" son "modelos *software*" basados típicamente en la "vista" de un objeto real, evento o proceso de negocio. Los componentes *software* individuales pueden formar parte y adaptarse tanto de estructuras independientes como de sistemas colaborativos.

El diseño de aplicaciones en n−capas es ideal para la creación de sistemas adaptables, donde cada componente puede ser utilizado y reutilizado en nuevas combinaciones para satisfacer requisitos de negocio dinámicos. Esto permite a los desarrolladores y a las nuevas aplicaciones reutilizar componentes existentes que modelan lógica de negocio sobradamente probada. En un entorno tremendamente cambiante como el actual, utilizar aplicaciones basadas en diseños de n−capas posibilita a las empresas ser más ágiles y adaptables en proporcionar valor a sus clientes. Los sistemas basados en n−capas tienen el potencial de reducir drásticamente tanto el *time−to−market* (tiempo de promoción) para las nuevas aplicaciones de negocio, como el coste total de mantenimiento, adaptando estos complejos y caros sistemas a las siempre cambiantes necesidades empresariales (Yoneki and Bacon, 2003).

2.3.1 Arquitecturas de desarrollo basadas en capas

Para el diseño de *software*, existen algunas aproximaciones que facilitan la distribución en capas de todos los elementos que componen el desarrollo. Estudiaremos algunas de ellas, obteniendo los aspectos más importantes que faciliten una definición posterior dentro de la arquitectura que desarrollamos.

2.3.1.1 Service Oriented Architecture (SOA)

La arquitectura orientada a servicios de cliente (en inglés *Service Oriented Architecture*), es un concepto de arquitectura *software* que define la utilización de servicios para dar soporte a los requisitos del negocio (Erl, 2005).

Permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros. SOA[5] define las siguientes capas de *software*:

- **Aplicaciones básicas** - Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad;

[5] http://www.ibm.com/developerworks/library/ar-archtemp/

- **De exposición de funcionalidades** - Donde las funcionalidades de la capa aplicativa son expuestas en forma de servicios (generalmente como servicios *web*);
- **De integración de servicios** - Facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración;
- **De composición de procesos** - Que define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio;
- **De entrega** - donde los servicios son desplegados a los usuarios finales.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

2.3.1.2 Recommended Process and Models (RPM)

La metodología *RPM* presentada por C. Larman (Larman, 2004) presupone una estructura de tres capas que es típica de los Sistemas de Información. Estas tres capas son:

- **Capa de la Presentación**: esta capa reúne todos los aspectos del *software* que tiene que ver con las interfaces y la interacción con los diferentes tipos de usuarios humanos. Estos aspectos típicamente incluyen el manejo y aspecto de las ventanas, el formato de los reportes, menús, gráficos y elementos multimedia en general.

- **Capa del Dominio de la Aplicación**: esta capa reúne todos los aspectos del *software* que tienen que automatizan o apoyan los procesos de negocio que llevan a cabo los usuarios. Estos aspectos típicamente incluyen las tareas que forman parte de los procesos, las reglas y restricciones que aplican. Esta capa también recibe el nombre de la capa de la lógica de la aplicación.

- **Capa del Repositorio**: esta capa reúne todos los aspectos del *software* que tienen que ver con el manejo de los datos persistentes, por lo que también se le denomina la capa de las Bases de Datos).

RPM toca muy superficialmente los problemas asociados al desarrollo de una capa de presentación. Menciona que es conveniente usar un enfoque de prototipaje y que pueden ser útiles los casos reales de uso; sin embargo no proporciona métodos, principios o lineamientos metodológicos que apoyen el desarrollo de una *metáfora de diseño para la interfaz,* el *diseño lógico* de la presentación o su *diseño físico* (Larman, 2004).

Para el desarrollo de nuestra arquitectura definiremos nuevas capas basándonos en las conceptualizaciones previamente definidas. Se definirán capas *lógicas*, que son aquellas que pertenecen a la lógica de programación o desarrollo de la aplicación final; y capas *físicas*, que definen la distribución de la arquitectura entre todos sus elementos (Larman, 2004).

2.4 METODOLOGÍAS Y LENGUAJES PARA LA DEFINICIÓN DE ONTOLOGÍAS

El término ontología se utilizaba inicialmente en el campo de la filosofía para referirse a "*un sistema de clasificación particular de una cierta visión del mundo*" (Guarino, 1998). Este sistema es independiente del lenguaje en el que se especifique, por lo que se dice que una

ontología es siempre la misma, independientemente del lenguaje que se use para describirla. En cambio, en los últimos años, se comenzó a usar en el campo de la Inteligencia Artificial, donde una ontología se refiere a un "*mecanismo ingenieril, constituido por un vocabulario específico que se usa para describir una cierta realidad junto con un conjunto de descripciones aceptadas que muestran el significado usado de las palabras del vocabulario*" (Guarino, 1998). Las ontologías son vocabularios comunes para las personas y aplicaciones que trabajan un dominio. Según el Grupo de Trabajo de Ontologías del Consorcio W3C, una ontología define los términos que se usan para describir y representar un cierto dominio.

Según Steve (Steve, Gangemi et al., 1998) es común dividir las ontologías en los siguientes tipos: *ontologías de un dominio* que representa el conocimiento especializado sobre un dominio o subdominio, como la medicina, las aplicaciones militares o la cardiología; *ontologías genéricas* la cual representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o tipos de objetos; *ontologías representacionales* que especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan *meta-ontologías* (*meta-level o top-level ontologies*); *ontologías de tareas o de técnicas básicas* describen una tarea, actividad o artefacto (denominadas *task ontologies*) como por ejemplo la venta de productos o el diagnóstico de una enfermedad y por último, *ontologías de aplicación* describen conceptos que dependen tanto de un dominio específico como de una tarea específica y, generalmente son una especialización de ambas.

Otra clasificación propuesta (Heijst, Schreiber et al., 1997) es la de *ontologías terminológicas* que especifican los términos que son usados para representar el conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado, ontologías de información especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información y ontologías de modelado de conocimiento para la conceptualización del conocimiento.

2.4.1 Aplicaciones y uso de las Ontologías

Algunas de las aplicaciones y usos dentro del desarrollo ontológico, por el que se recomiendan su uso, están:

- Repositorios para la organización del conocimiento.
- Servir de herramientas de referencia en la construcción de sistemas de bases de conocimiento que aporten consistencia, fiabilidad y falta de ambigüedad a la hora de recuperar información.
- Normalizar los atributos de los metadatos aplicables a los documentos.
- Posibilitar el trabajo cooperativo al funcionar como soporte común de conocimiento entre organizaciones, comunidades científicas, etc.
- Permitir la interoperabilidad entre sistemas distintos.
- Servir de base para la construcción de lenguajes de representación del conocimiento.

2.4.2 Principios y Metodologías para el diseño de Ontologías

Según (McGuinness and Harmelen, 2004), se describe un enfoque iterativo en el desarrollo de ontología, inicialmente abordándose de manera frontal. Los pasos a seguir según estos autores son: ***a. determinar el dominio y alcance de la ontología***, para el cual nuestro dominio es el proceso de monitorización específicamente a la monitorización móvil de pacientes, ***b. consideración de la reutilización de las ontologías existentes*** (más adelante veremos la evaluación de algunas ontologías existentes en esta área), ***c. enumerar términos importantes para la ontología***, clasificando términos propios a pacientes, dispositivos, módulos de control, monitorización, etc., ***d. definir las clases y jerarquías de clases***, para cada uno de los elementos que intervienen en nuestra propuesta (monitorización de la cual derivamos la monitorización móvil), ***e. definir las propiedades de las clases***, en la definición del perfil común se establecen propiedades, dispositivos, módulos de control, monitorización, etc., ***f. definir las propiedades de cada clase***, se han definido cardinalidad, características, etc. para la monitorización***,*** ***g. crear instancias***, asociados a un determinado perfil de paciente, dispositivos, módulos de control, monitorización, etc.en nuestro caso.

Para el diseño de una ontología es particular es necesario contar con una metodología, que brinde soporte durante todo el proceso de representación del dominio, existen diferentes metodologías, entre ellas podemos destacar:

a. Metodología **Methontology**
b. Metodología **CyC**
c. Metodología **Uschold** y **King**
d. Metodología **On-To-Knowledge**
e. Metodología **Sensus**

Nombre de la Metodología	Especificaciones de la Metodología				
	Ciclo de Vida	***Herramientas***	***Proceso de Modelado***	***Nivel de Abstracción***	***Ontologías base***
Methontology	*Secuencial con prototipos de desarrollo y actividades de administración y apoyo paralelo al desarrollo.*	*ODE WebODE, OntoEdit, Protegé.*	*Identificación directa de conceptos y organización taxonómica.*	*Elevado*	*No*
CyC	*Secuencial*	*No especificada*	*Extracción manual.*	*Alto*	*No*
Uschold y King	*No especificada.*	*No especificada*	*Identificación directa de conceptos.*	*Normal*	*No*
On-To-Knowledge	*Incremental y cíclica con prototipos de desarrollo.*	*OntoEdit, Corporum.*	*Casos de uso Semilla*	*Bajo*	*No*
Sensus	*No propuesto*	*No especificada*	*Semilla*	*Normal*	*Si*

Tabla 2-1. *Comparación de las características de las metodologías de desarrollo de ontologías evaluadas.*

Cada una de ellas presenta ciertas características comunes con enfoques diferentes que permiten utilizarlas dependiendo de las particularidades del dominio a representar, y de los recursos con que se cuente. En la tabla 2-1, se evalúan cada una de estas características según las especificaciones de las metodologías.

De todas las metodologías evaluadas, las que se utilizan con más frecuencia son ***On-To-Knowledge*** y ***Methontology***, que nos permiten un desarrollo basado en prototipos que pueden cambiar de acuerdo a la evolución y sobre todo permiten obtener un nivel elevado de detalle en su conceptualización, aspecto importante que garantiza una correcta representación del conocimiento y por ende calidad de la ontología.

Cada una de estas metodologías se centra en dos pasos fundamentales, lo que garantiza la validez de las ontologías desarrolladas, estos son:

- Extracción del conocimiento (manual y de diversas formas);
- Utilización de herramientas de procesamiento de lenguaje natural, para la adquisición de nuevos conocimientos.

Analizaremos cada unas de estas metodologías para poder seleccionar la que mejor se adapte a nuestras necesidades de desarrollo.

2.4.2.1 Metodología Methontology

METHONTOLOGY (Gómez-Perez, 1998) y (Fernandez-Lopez, 1999) permite la construcción de ontologías en el nivel del conocimiento. Esta metodología incluye los siguientes pasos:

- la identificación del proceso de desarrollo de la ontología,
- un ciclo de vida propuesto y
- la metodología como tal.

El proceso de desarrollo de la ontología identifica las tareas que deben realizarse cuando se construye una ontología estas son:

- *planificación,*
- *control,*
- *control de calidad,*
- *especificación,*
- *adquisición de conocimiento,*
- *conceptualización,*
- *integración,*
- *formalización,*
- *implementación,*
- *evaluación,*
- *mantenimiento,*
- *documentación y*
- *administración de la configuración*.

El ciclo de vida basado en prototipos evolutivos identifica las etapas que atraviesa la ontología durante su tiempo de vida. Finalmente, la metodología en sí misma, especifica los pasos que se deben seguir para desarrollar cada actividad, las técnicas usadas, los productos que se producirán y como serán estos evaluados. La fase principal en el proceso de desarrollo de una ontología usando el enfoque de **METHONTOLOGY** es la fase de *conceptualización*. Durante ambas: especificación y conceptualización, se completa un proceso de integración usando ontologías de la casa y ontologías externas.

Este entorno de trabajo es parcialmente soportado por un *software* llamado *Ontology Design Environment* (**ODE**) (Fernandez-Lopez and Gomez-Perez, 2002) y (Blázquez, Fernandez-Lopez et al., 1998). Han sido desarrolladas varias ontologías usando esta metodología:

- ***CHEMICALS***, una ontología en el dominio de los elementos químicos;
- ***Ontologías de contaminantes ambientales*** (Gómez-Perez, 1998) que representa los métodos de detección de los diferentes componentes contaminantes de varios medios: agua, aire, tierra, etc. y la concentración máxima permitida de estos componentes, tomando en cuenta toda la legislación (normas de la Unión Europea, normas Españolas, Alemanas, Estadounidenses, etc.);
- ***Ontology Reference*** (Arpírez, Gomez-Perez et al., 1998) es una ontología en el dominio de las ontologías que juega el papel de una especie de páginas amarillas de ontologías;
- la ontología ***KA***(Knowledge Adquisition)(Blázquez, Fernandez-Lopez et al., 1998) que modeliza a la Comunidad de Adquisición de Conocimientos en una ontología llamada *Ontology*, accesible a toda la comunidad a través del *Ontology Server*.

Esta metodología para construir ontologías ha sido propuesta por la **FIPA** (*Foundation for Intelligent Physical Agents*) que promueve la interoperabilidad entre aplicaciones basadas en agentes.

2.4.2.2 Metodología CyC

Nace como un proyecto de Inteligencia Artificial que busca la construcción de una ontología comprensible para habilitar el razonamiento humano (Lenat and Guha, 1990).

Los pasos para la construcción de la ontología usando esta metodología son:

- Extracción manual del conocimiento común (de diversas fuentes);
- Utilización de herramientas de procesamiento de lenguaje natural o aprendizaje natural para la adquisición de nuevo conocimiento en la ontología;
- Delegación de la mayor parte de la codificación en las herramientas.

El proyecto **CyC** surgió en el año de 1984, por parte de la Corporación de Tecnología en Computación y Microelectrónica. La base de conocimiento de **CyC** es propietaria, aunque una pequeña versión fue liberada y está disponible como **OpenCyC**, la misma que busca definir un vocabulario común para el conocimiento automatizado.

Actualmente **CyC** cuenta con más de un millón de aserciones en su base de conocimientos, y que han sido definidas por el humano mediante el lenguaje **CyCL** (un lenguaje de programación parecido a Lisp).

2.4.2.3 Metodología Uschold y King

La metodología de **Uschold y King** (Uschold and King, 1995) se basa en la experiencia de construir el "*Enterprise Ontology",* el cual incluye un conjunto de ontologías para modelado empresarial. Permite la construcción de ontologías en base a otras ya existentes. Ellos proponen los siguientes pasos:

- Identificar el propósito y el alcance de la ontología;
- Construir la ontología capturando conocimiento, codificando conocimiento e integrando tal conocimiento con ontologías existentes;
- Evaluar la ontología;
- Documentación;
- Lineamientos para cada fase.

2.4.2.4 Metodología On-To-Knowledge

Es un proyecto de la **IST** (*Tecnologías de la Sociedad de la Información*) (Staab, Schnurr et al., 2001), mediante este proyecto se han desarrollado herramientas y métodos que soporten la administración de conocimiento, apoyado en una ontología compartible y usable.

Esta metodología aplica ontologías a la información electrónica con la finalidad de mejorar la administración de conocimiento. Esta metodología incluye la identificación de metas que deberían ser conseguidas por herramientas de gestión de conocimiento y está basada en el análisis de escenarios de uso y en los diferentes papeles desempeñados por trabajadores de conocimiento y accionistas en las organizaciones. Cada una de las herramientas de la arquitectura de **OKT** (*On-To-Knowledge*) se centra en el desarrollo de aplicaciones dirigidas por ontologías y, finalmente, describe el uso y la evaluación de la metodología mediante casos de estudio. Incluye los siguientes aspectos mostrados en la figura 2-2:

- Identificación de metas, las cuales deberán ser cumplidas por herramientas de gestión de conocimiento. (Viabilidad)
- Refinamiento
- Evaluación de la ontología a partir de casos de estudio.
- Mantenimiento

2.4.2.5 Metodología Sensus

La metodología basada en **Sensus** (Swartout, Patil et al., 1997) es un enfoque *top-down* para derivar ontologías específicas del dominio a partir de grandes ontologías. Los autores proponen identificar un conjunto de términos *semilla* que son relevantes en un dominio particular. Tales términos se enlazan manualmente a una ontología de amplia cobertura. Los usuarios seleccionan automáticamente los términos relevantes para describir el dominio y acotar la ontología **Sensus**.

Consecuentemente, el algoritmo devuelve el conjunto de términos estructurados jerárquicamente para describir un dominio, que puede ser usado como esqueleto para la base de conocimiento. Esta metodología sirvió para construir la ontología **Sensus** y recomienda los siguientes pasos: (Swartout, Patil et al., 1997)

- Tomar una serie de términos como semillas;
- Enlazarlos manualmente;
- Incluir todos los conceptos en el camino que va de la raíz de **Sensus** a los conceptos semilla;
- Añadir nuevos términos relevantes del dominio;
- Opcionalmente, añadir, para aquellos nodos por los que pasan más caminos, su subárbol inferior.

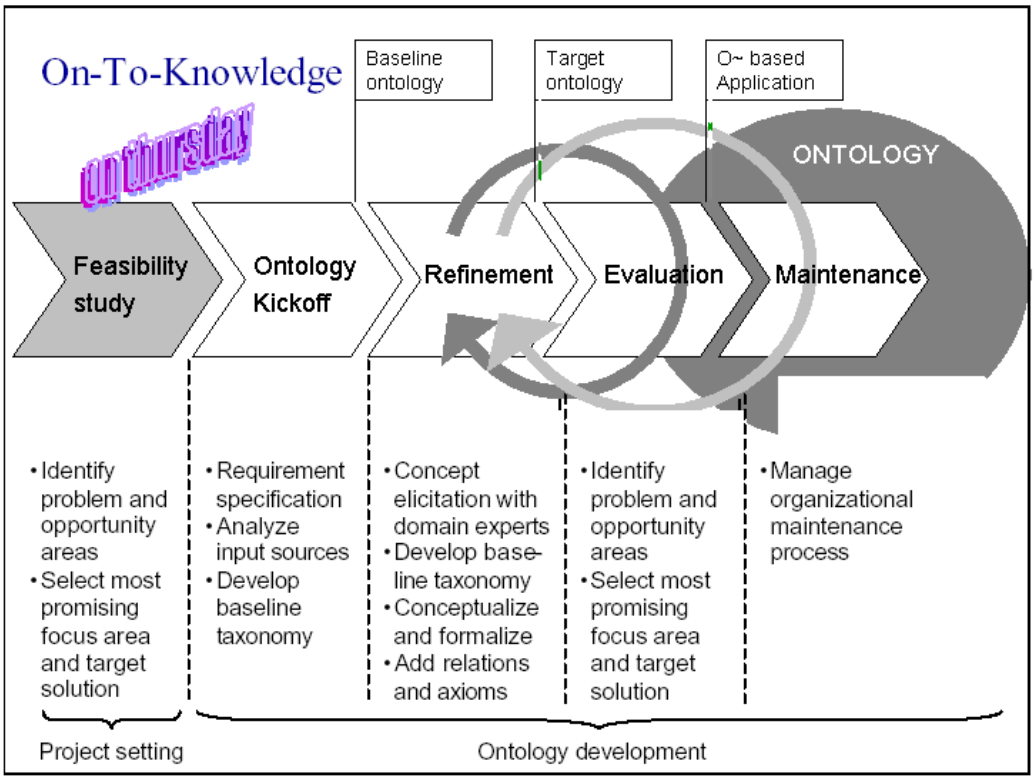


Figura 2-2. *Estructura de la metodología On-To-Knowledge*

La necesidad de evaluar las ontologías se identifica también en las tres metodologías anteriores. La metodología de **Uschold** incluye esta actividad, pero no establece como podría llevarse a cabo. Finalmente, **METHONTOLOGY** propone que deben llevarse a cabo actividades de evaluación a través de todo el tiempo de vida del proceso de desarrollo de la ontología. La mayoría de la evaluación se hace en la fase de conceptualización.

De todas ellas, sólo **METHONTOLOGY** y **On-To-Knowledge** aportan un conjunto de técnicas y métodos detallados para realizar algunas de las actividades más importantes, son las únicas que establecen relaciones (de precedencia, co-ocurrencia, etc.) entre las actividades y especifican las entradas y salidas que se deben obtener en cada una de ellas.

Para nuestra investigación utilizaremos la metodología **METHONTOLOGY**, que nos guiará en el proceso de desarrollo de nuestras ontologías. Para formalizar cada ontología utilizaremos el lenguaje **OWL**, que nos permite definir cada uno de los elementos que la componen.

2.4.3 Lenguajes para formalización de Ontologías

Se definen algunos de los principales lenguajes para formalizar ontologías, así como la importancia y funcionalidad de cada uno de ellos, en el diseño de ontologías.

a. **RDFS:** (por su siglas en inglés, *Resource Description Framework Schema*) ((W3C), 2008). En **RDFS** una clase es cualquier recurso que tenga una propiedad *rdf:type* cuyo valor sea *rdfs:Class*. Un recurso puede ser una instancia de más de una clase y es posible representar una jerarquía de clases a través de *rdfs:subClassOf*, que cumple la propiedad transitiva. Las propiedades en RDFS son instancias de la clase *rdf:Property*, usando *rdfs:range* para indicar que los valores de una propiedad son instancias de una clase (W3C).

b. **OIL:** (*Ontology Inference Layer*) (Fensel, Horrocks et al., 2000) fue el primer lenguaje de representación de ontologías basado en estándares **W3C**. Su sintaxis está basada en la de **XML** y se definió como una extensión de **RDFS**. **OIL** se encuentra estructurado en capas, formando cada una de ellas un *sublenguaje*. La capa base o núcleo coincide totalmente con **RDFS** y cada una de las capas superiores añade funcionalidad y complejidad a la capa anterior. Según (Horrocks, Fensel et al., 2000), el lenguaje **OIL** presenta una serie de limitaciones entre las que se pueden destacar las siguientes: no es posible realizar la sobrescritura de valores heredados de una superclase; presenta falta de expresividad a la hora de declarar reglas o axiomas; y, no soporta dominios concretos (números enteros, cadenas de caracteres, etc.).

c. **DAML+OIL:** Este lenguaje es una propuesta más reciente que las dos anteriores, pues surgió como fruto de la cooperación entre los grupos de trabajo **OIL** y **DARPA** (*US Defense Advanced Research Projects Agency*), creadores del lenguaje **DAML** (*DARPA's Agent Markup Language*). Tenían como finalidad conseguir extender el nivel de expresividad de **RDFS** (Peis-Redondo, Herrera-Viedma et al., 2009). **DAML+OIL** hereda muchas de las características de **OIL**, pero difiere en otras. A nivel práctico, Horrocks, Goble y Bechhofer (Horrocks, Fensel et al., 2000) indican que **DAML+OIL** demuestra ser más útil como soporte para ontologías que *RDF Schema*, sin embargo presenta algunas carencias como formato de intercambio y modelado de ontologías. Este lenguaje es soportado por un gran número de herramientas y aplicaciones, lo que demuestra que **DAML+OIL** no presenta demasiada complejidad técnica, pero su complejidad conceptual (*dificultad de uso y aprendizaje por parte del creador de la ontología)* sí puede considerarse un problema (Peis-Redondo, Herrera-Viedma et al., 2009).

d. **OWL:** (*Ontology Web Language*) surgió en julio del 2002, momento en el cual el grupo de trabajo *WebOnt* del W3C publica el primer borrador ((W3C), 2008). Es un lenguaje que deriva de **DAML+OIL** y que está cimentado sobre **RDFS**. **OWL** tiene una mayor capacidad semántica que **RDF**, ya que permite definir muchas restricciones entre clases y entre propiedades. Actualmente existen tres variantes o sublenguajes (McGuinness and Harmelen, 2004): OWL *Lite, OWL DL* y *OWL Full*.

- **OWL Lite.** Es la variante más sencilla, proporciona lo necesario para crear jerarquías de clases y restricciones simples.
- **OWL DL.** Proporciona la máxima expresividad manteniendo la capacidad de procesamiento computacional completo.
- **OWL Full.** Proporciona mayor expresividad, pero sin garantías computacionales.

2.5 PATRONES EN EL DISEÑO DE APLICACIONES

El concepto original de patrones de modelo fue acuñado por el arquitecto Cristóbal Alexander en los años 70. El modelo que él describió define paseos, jardines, público sitios, y el diseño de edificios solos y su alineación.

La implementación de patrones que permitan el rápido, flexible y ordenado desarrollo de aplicaciones ha tomado gran importancia al momento de desarrollar estructuras de *framework* o de generación de aplicaciones.

Algunas investigaciones han utilizado patrones en diferentes áreas, facilitando el desarrollo de prototipos, aplicaciones e interfaces. En (Daria, John et al., 2000), se ha desarrollado una serie de patrones de itinerario reusables, flexibles que permite el desarrollo rápido de itinerarios de agentes complejos. Usado en conjunto con una librería de tareas, estos patrones de itinerarios han reducido el tiempo de desarrollo de agentes arriba del cuarenta por ciento.

Por su parte (Daniel, 2007), describe el desarrollo de un lenguaje de patrones que permite el diseño de la interacción móvil en un sistema de diálogo multimodal. La realización sobre el dispositivo de móvil muestra la flexibilidad de este acercamiento, aún aunque no se generen disposiciones y elementos gráficos *ad hoc*.

2.5.1 ¿Qué es un patrón?

En programación orientada a objetos se entiende por **patrón** *una solución probada que se puede aplicar con éxito a un determinado tipo de problemas que aparecen repetidamente en el desarrollo de sistemas software*.

Los patrones no son una librería. Van más bien en la línea de un esqueleto básico que cada desarrollador luego adapta a sus necesidades y a las peculiares características de su aplicación. Se describen fundamentalmente en forma textual, acompañada de diagramas y de seudo-código.

Definiciones:

- "Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular". (Gamma, Johnson and et al., 1994)

- "Un patrón es un pedazo de información con nombre, instructivo y significante, que captura la esencia de una familia exitosa y completa de soluciones a un problema recurrente en un contexto dado". Brad Appleton (Appleton, 2000).

- "Cada patrón es una regla de tres partes, la cual expresa una relación entre un contexto dado, un conjunto de fuerzas que ocurren repetitivamente en ese contexto y cierta configuración de *software* que permite a esas fuerzas resolverse por sí mismas". Richard Gabriel (Gabriel, 1998).

- "Cada patrón es una regla de tres partes, la cual expresa una relación entre un cierto contexto, un problema y una solución. El patrón es, resumiendo, al mismo tiempo una cosa que tiene su lugar en el mundo, y la regla que nos dice cómo crear esa cosa y cuándo debemos crearla. Es al mismo tiempo una cosa y un proceso; al mismo tiempo una descripción de una cosa que tiene vida y una descripción del proceso que la generó. Estos patrones en nuestras mentes son, más o menos, imágenes mentales de los patrones en el mundo: son representaciones abstractas de las reglas morfológicas que definen los patrones en el mundo. Sin embargo, son realmente diferentes. Los patrones en el mundo solo existen. Pero esos mismos patrones en nuestras mentes son dinámicos. Tienen fuerza. Son generativos. Nos dicen qué hacer, cómo se pueden generar y, en ciertas circunstancias, que los debemos crear. Cada patrón es una regla que describe qué debemos hacer para generar la entidad que los define". Christopher Alexander (Alexander, 1979).

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas *software*.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable.

Según la escala o nivel de abstracción los patrones pueden ser:

- **Patrones de arquitectura**: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de *software*.
- **Patrones de diseño**: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de *software*.
- **Dialectos**: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Además, también es importante reseñar el concepto de "*anti-patrón de diseño*", que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el *software*. La idea de los anti-patrones es dar a conocer los problemas que acarrean ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida por haber cometido los mismos errores.

2.5.2 Descripción de patrón y plantillas de patrones

Dependiendo del autor, del nivel de abstracción y de la publicación misma se han presentado varios formatos para encapsular la información de un patrón. Los puntos más significativos que debe contener un patrón son:

- **Nombre**: Significativo y corto, fácil de recordar y asociar a la información que sigue.
- **Problema**: Un enunciado que describe las metas y objetivos buscados y el contexto.
- **Contexto**: Define las precondiciones en las cuales ocurren el problema y su solución.
- **Fuerzas**: Descripción de las fuerzas y restricciones relevantes en el problema y cómo interactúan o entran en conflicto.
- **Solución**: Las relaciones estáticas y reglas dinámicas que describen cómo solucionar el problema.
- **Ejemplos**: Uno o más ejemplos que ilustren el contexto, el problema y su solución.
- **Contexto Resultante**: El estado en el cual queda el sistema después de aplicar el patrón y las consecuencias de hacerlo.
- **Racionalidad**: Una explicación justificada de los pasos o reglas en el patrón.
- **Relaciones**: relaciones estáticas y dinámicas del patrón con otros.
- **Usos conocidos**: Describe ocurrencias del patrón conocidas y su aplicación dentro de los sistemas existentes.

La propuesta de Gamma (Gamma, Johnson et al., 1994), por ejemplo, propone que los elementos esenciales de un patrón son los siguientes:

1. Un **nombre del patrón**. Es una forma abreviada que pueda darnos una idea del problema al que se aplica, sus soluciones y consecuencias. Al asignar un nombre, estamos facilitando la tarea de diseño puesto que nos comunicamos a un mayor nivel de abstracción. Es bastante difícil encontrar nombres adecuados que sirvan a este propósito.

2. El **problema** describe cuando aplicar el patrón. Aquí se explica el problema y su contexto. Un añadido útil es el de las condiciones de aplicabilidad del patrón.

3. La **solución** describe los elementos que constituyen el diseño, sus relaciones, responsabilidades y colaboraciones. Se insiste mucho en que esta solución es como una

plantilla que provee una descripción abstracta de un problema de diseño y cómo una disposición general de elementos (en este caso clases y objetos) puede resolverlo.

4. Las **consecuencias** son los resultados y compromisos de aplicar el patrón.

Estos son los elementos esenciales, pero cuando se trata de realizar una descripción concreta de un patrón, la plantilla propuesta estará compuesta por una serie de secciones que permiten una estructura más detallada que la ofrecida por la enumeración de los elementos esenciales.

Siguiendo a (Gamma, Johnson et al., 1994), encontramos la siguiente lista y descripción de secciones dentro de la plantilla que describe cada patrón. Este formato estructurado es útil puesto que permite la separación semántica de lo que podría haber sido un texto completo y permite además su almacenamiento en una base de datos para su posterior acceso si deseamos aumentar su reutilización. Esta lista y descripción de secciones están formadas así: 1) Nombre del Patrón y Clasificación, 2) Intención, también conocido como (Sinónimo), 3) Motivación, 4) Aplicabilidad, 5) Estructura, 6) Participantes, 7) Colaboraciones, 8) Consecuencias, 9) Implementación, 10) Ejemplo de código, 11) Usos conocidos y 12) Patrones relacionados.

Para el desarrollo de cada uno de los patrones en esta tesis, nos basaremos en la definición inicial que han hecho los autores referenciados, ajustándolos a las necesidades de nuestro *framework*. Las capas serán divididas y clasificadas según las necesidades lógicas y físicas del diseño.

2.6 ANDROID: UN SISTEMA OPERATIVO ESCALABLE

Android es un sistema operativo basado en *Java* que se ejecutan en *Linux*. Es un sistema muy ligero y completamente equipado (Rogers, Lombardo et al., 2009). Fue desarrollado inicialmente por *Android Inc*., una firma comprada por Google en el 2005. Es el principal producto de la *Open Handset Alliance* (una alianza comercial de 78 compañías para desarrollar estándares abiertos para dispositivos móviles), un conglomerado de fabricantes y desarrolladores de *hardware*, *software* y operadores de servicio.

La estructura del sistema operativo **Android** se compone de aplicaciones que se ejecutan en un *framework* **Java** orientadas a objetos sobre el núcleo de las *bibliotecas de Java* en una máquina virtual **Dalvik** con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje *C* incluyen un administrador de interfaz gráfica (*surface manager*), un *framework* **OpenCore**, una base de datos relacional **SQLite**, una API gráfica **OpenGL ES 2.0 3D**, un motor de renderizado **WebKit**, un motor gráfico **SGL**, **SSL** y una biblioteca estándar de *C* **Glibc** (Android Developer, 2011).

Dentro de las características que se pueden mencionar cabe destacar (Rogers, Lombardo et al., 2009):

- Aplicación marco que permita la reutilización y sustitución de componentes;
- **Dalvik** máquina virtual optimizada para dispositivos móviles;

- Navegador integrado basado en el motor **WebKit** de código abierto:
- Gráficos optimizado alimentado por una biblioteca de gráficos 2D personalizados, gráficos en 3D basado en OpenGL ES 1.0 (aceleración de *hardware* opcional);
- **SQLite** para almacenamiento de datos estructurados;
- Media de apoyo comunes para audio, vídeo, y aún formatos de imagen (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF);
- Telefonía GSM (dependiente del *hardware*);
- Bluetooth, EDGE, 3G y WiFi (dependiente del *hardware*);
- Cámara, GPS, brújula y acelerómetro (dependiente del *hardware*);
- Rico entorno de desarrollo incluyendo un emulador de dispositivos, herramientas para la depuración, la memoria y de perfiles de rendimiento, y un plugin para el IDE de Eclipse.

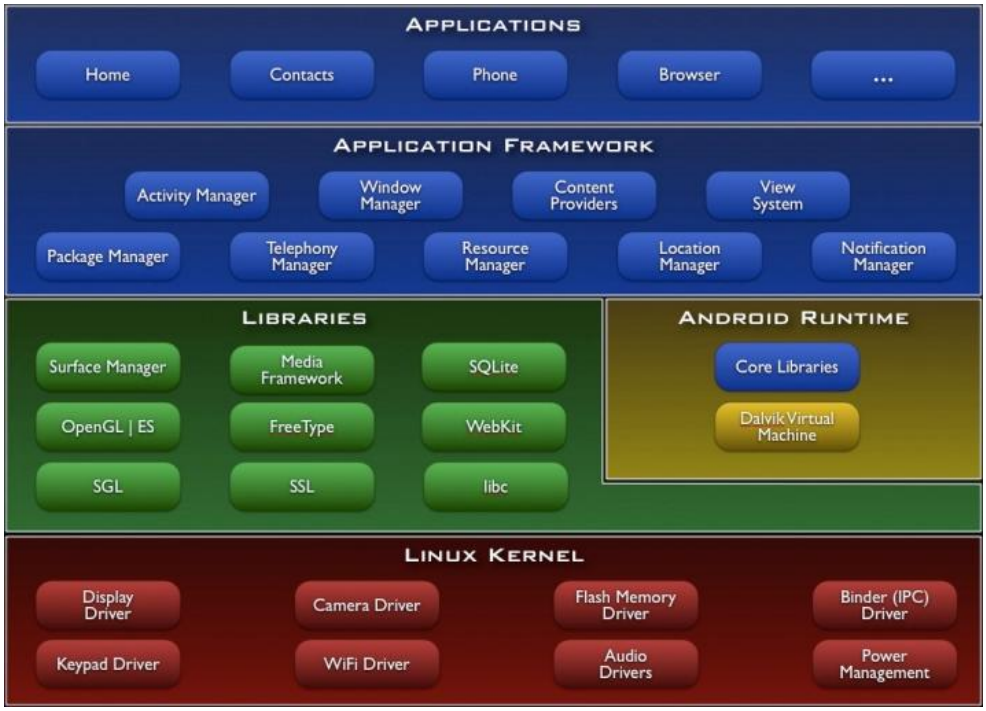


Figura 2-3. *Diagrama de sistema operativo Android*

Los componentes principales del sistema operativo de Android se muestran en la figura 2-3, estos son (Android Developer, 2011):

- **Aplicaciones**: las aplicaciones base incluyen un cliente de correo electrónico, programa de *SMS*, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación *Java*.

- **Marco de trabajo de aplicaciones**: los desarrolladores tienen acceso completo a los mismos APIs del *framework* usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- **Bibliotecas**: Android incluye un conjunto de bibliotecas de *C/C++* usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: *System C library* (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y *SQLite*, entre otras.

- **Runtime de Android**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje *Java*. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual *Dalvik*. *Dalvik* ha sido escrito de forma que un dispositivo puede ejecutar múltiples máquinas virtuales de forma eficiente. *Dalvik* ejecuta archivos en el formato *Dalvik Executable* (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y ejecuta clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida "dx".

- **Núcleo Linux:** Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el *hardware* y el resto de la pila de *software*.

2.7 EVALUACIÓN DE ARQUITECTURAS SOFTWARE

El objetivo de evaluar arquitecturas *software* es conocer si se han cumplido todos los requerimientos funcionales inicialmente planteados, es decir, asegurar que el sistema construido cumple con las necesidades planteadas por los usuarios finales (Brey, Escobar et al., 2005).

Según (Kazman, Clements et al., 2001) el principal paso al momento de evaluar una arquitectura *software* es conocer *qué es lo que se quiere evaluar*. De esta forma podemos seleccionar dentro de un gran número de métodos y técnica de evaluación que han sido desarrolladas y validadas por algunos investigadores.

Las mediciones que se realizan sobre una arquitectura *software* pueden tener distintos objetivos, dependiendo de la situación en la que se encuentre el arquitecto y la aplicabilidad de las técnicas que emplea. Algunos de estos objetivos son: cualitativos, cuantitativos y máximos y mínimos teóricos (Kazman, Clements et al., 2001).

- **Medición cualitativa**: se aplica para la comparación entre arquitecturas candidatas y tiene relación con la intención de conocer la opción que mejor se adapte a ciertos atributos de calidad. Este tipo de medición brinda respuestas afirmativas o negativas, sin mayor nivel de detalle.
- **Medición cuantitativa:** busca la obtención de valores que permiten tomar decisiones en cuanto a los atributos de calidad de una arquitectura de *software*. El esquema general es la comparación con márgenes establecidos, como lo es el caso de los requerimientos de desempeño, para establecer el grado de cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella.

- **Medición de máximos y mínimos teóricos:** contempla los valores teóricos para efectos de la comparación de la medición con los atributos de calidad especificados. El conocimiento de los valores máximos o mínimos permite el establecimiento claro del grado de cumplimiento de los atributos de calidad.

Atributo de Calidad	Descripción
Disponibilidad	Es la medición de disponibilidad del sistema para su uso (Barbacci, 1995).
Confidencialidad	Es la ausencia de acceso no autorizado a la información (Barbacci, 1995).
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman, Clements et al., 2001).
Desempeño	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria (IEEE 610.12).
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci, 1995).
Seguridad Externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci, 1995).
Seguridad Interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos (Kazman, Clements et al., 2001).

Tabla 2-2. *Descripción de atributos de calidad observables vía ejecución* (Bass, Klein et al., 2000).

Atributo de Calidad	Descripción
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema.
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados (Bass, Klein et al., 2000).
Integridad	Es la ausencia de alteraciones inapropiadas de la información (Barbacci, 1995).
Interoperabilidad	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de integrabilidad (Bass, Klein et al., 2000).
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema.
Mantenibilidad	Capacidad de modificar el sistema de manera rápida y a bajo costo.
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser *hardware*, *software* o combinación de los dos (Kazman, Clements et al., 2001).
Reusabilidad	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones.
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental.
Capacidad de prueba	Es la medida de la facilidad con la que el *software*, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el *software* fallará en su próxima ejecución de prueba.

Tabla 2-3. *Descripción de atributos de calidad no observables vía ejecución* (Bass, Klein et al., 2000).

Una pregunta que salta a nuestras mentes es saber cómo identificar qué vamos a evaluar en nuestra arquitectura *software*. A grandes rasgos, (Bass, Klein et al., 2000) establece una clasificación de los atributos de calidad en dos categorías:

- **Observables vía ejecución**: son aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. En la tabla 2-2 se muestra algunos de estos atributos y su descripción breve.

- **No observables vía ejecución**: son aquellos atributos que se establecen durante el desarrollo del sistema. La descripción de algunos de estos atributos se presentan en la tabla 2-3.

2.7.1 Técnicas de evaluación de arquitecturas *software*

Existen un grupo de técnicas para evaluar arquitecturas *software*, dichas técnicas se clasifican en cuantitativas y cualitativas (Brey, Escobar et al., 2005). En la figura 2-4, se muestra la clasificación de estas técnicas en general. Cada técnica debe ser especificada según el tipo de arquitectura a evaluar y el momento en el que se quiere hacer la evaluación (temprana, para aplicaciones aún en desarrollo inicial y tardío, para aplicaciones ya instaladas y desarrolladas).

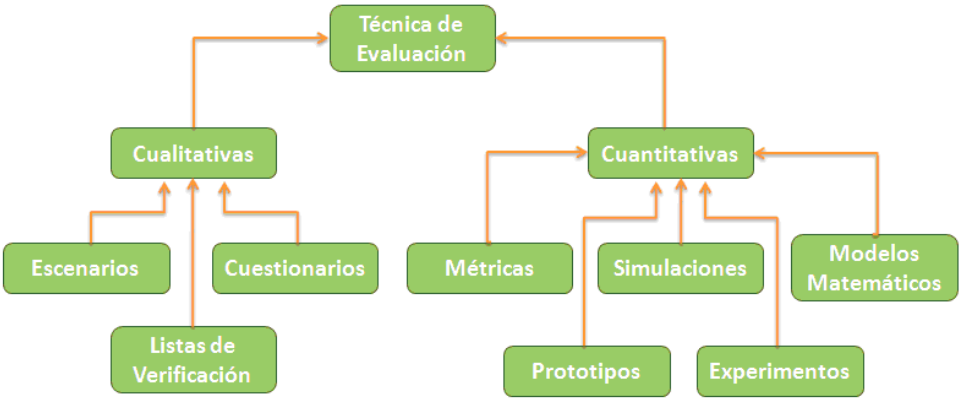


Figura 2-4. *Clasificación de las técnicas de evaluación* (Brey, Escobar et al., 2005).

Además de diferenciar cada una de ellas, también se pueden utilizar en conjunto para obtener con más detalle, resultados más seguros.

Cada una de estas técnicas utiliza instrumentos de evaluación específicos. Dentro de los instrumentos de evaluación asociados a algunas de las técnicas podemos encontrar:

- **Basada en escenarios**: utiliza instrumentos de evaluación como: *perfiles (profile)* y *árboles de utilidad (Utility Tree)*. Un *perfil* es un conjunto de escenarios, generalmente con alguna importancia relativa asociada a cada uno de ellos. Un árbol de utilidad es un esquema en forma de árbol que presenta los atributos de calidad de un sistema de *software*, refinados hasta el establecimiento de escenarios que especifican con suficiente detalle el nivel de prioridad de cada uno (Kazman, Clements et al., 2001).
- **Basada en simulación**: utiliza instrumentos de evaluación como: *lenguajes de descripción arquitectónica (ADL)* y *modelos de colas*.
- **Basada en modelos matemáticos:** utiliza instrumentos de evaluación como: *cadenas de Markov* y *Reliability Block Diagrams.*
- **Basada en experiencia:** utiliza instrumentos de evaluación como: *intuición y experiencia*, *tradición* y *proyectos similares.*

2.7.2 Métodos de evaluación de arquitecturas *software*

(Kazman, Clements et al., 2001) proponen que la existencia de un método de análisis de arquitecturas hace que el proceso sea repetible, y ayude a garantizar que las respuestas correctas con relación a la arquitectura pueden hacerse temprano, durante las fases tempranas de diseño. Es en este punto donde los problemas encontrados pueden ser solucionados de una forma relativamente poco costosa. De manera similar, un método de evaluación sirve de guía a los involucrados en el desarrollo del sistema, en la búsqueda de conflictos que puede presentar una arquitectura, y sus soluciones.

A pesar que la mayoría de los métodos poseen un aspecto *cuantitativo*, su evaluación no deja de ser un aspecto mayormente *cualitativo* a la hora de desarrolla la evaluación en su conjunto. Cada método propone establecer rangos ponderados que establecen las prioridades de funcionalidad y diseño de la arquitectura que se evalúa. Por esta razón, resulta conveniente estudiar los métodos de evaluación de arquitecturas *software* propuestas hasta el momento.

De acuerdo con (Kazman, Clements et al., 2001) hasta hace unos años no existían métodos de utilidad general para evaluar arquitecturas de *software*. Si alguno existía, sus enfoques eran incompletos, *ad hoc*, y no repetibles, lo que no brindaba mucha confianza. En virtud de esto, múltiples métodos de evaluación han sido propuestos. A continuación se explican algunos de los más importantes.

2.7.2.1 Software Architecture Analysis Method (SAAM)

Este método fue presentado por (Kazman, Clements et al., 2001), según su autor, el **Método de Análisis de Arquitecturas de *Software*** (por sus siglas en inglés, **SAAM**), es el primero que fue ampliamente promulgado y documentado. El método fue originalmente creado para el análisis de la ***modificabilidad*** de una arquitectura, pero en la práctica ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad, tales como modificabilidad, portabilidad, escalabilidad e integridad.

El método de evaluación SAAM se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada. De acuerdo con (Kazman, Clements et al., 2001), las salidas de la evaluación del método SAAM son las siguientes:

- Una proyección sobre la arquitectura de los escenarios que representan los cambios posibles ante los que puede estar expuesto el sistema.
- Entendimiento de la funcionalidad del sistema, e incluso una comparación de múltiples arquitecturas con respecto al nivel funcional que cada una soporta sin modificación.

Con la aplicación de este método, si el objetivo de la evaluación es una sola arquitectura, se obtienen los lugares en los que la misma puede fallar, en términos de los requerimientos de modificabilidad. Los pasos que contempla el método de evaluación SAAM son:

1. **Desarrollo de escenarios**

   Un escenario es una breve descripción de usos anticipados o deseados del sistema. De igual forma, estos pueden incluir cambios a los que puede estar expuesto el sistema en el futuro.

2. **Descripción de la arquitectura**

   La arquitectura debe ser descrita haciendo uso de alguna notación arquitectónica que sea común a todas las partes involucradas en el análisis. Deben incluirse los componentes de datos y conexiones relevantes, así como la descripción del comportamiento general del sistema. El desarrollo de escenarios y la descripción de la arquitectura son usualmente llevados a cabo de forma intercalada, o a través de varias iteraciones.

3. **Clasificación y asignación de prioridades de los escenarios**

   La clasificación de escenarios puede hacerse en dos formas: *directos* e *indirectos*. Un escenario directo es el que puede satisfacerse sin la necesidad de modificaciones en la arquitectura. Un escenario indirecto es aquel que requiere modificaciones en la arquitectura para poder satisfacerse. Los escenarios indirectos son de especial interés para SAAM, pues son los que permiten medir el grado en el que una arquitectura puede ajustarse a los cambios de evolución que son importantes para los involucrados en el desarrollo.

4. **Evaluación individual de los escenarios indirectos**

   Para cada escenario indirecto, se listan los cambios necesarios sobre la arquitectura, y se calcula su costo. Una modificación sobre la arquitectura significa que debe introducirse un nuevo componente o conector, o que alguno de los existentes requiere cambios en su especificación.

5. **Evaluación de la interacción entre escenarios**

   Cuando dos o más escenarios indirectos proponen cambios sobre un mismo componente, se dice que interactúan sobre ese componente. Es necesario evaluar este hecho, puesto que la interacción de componentes semánticamente no relacionados revela que los componentes de la arquitectura efectúan funciones semánticamente distintas.

6. **Creación de la evaluación global**

   Debe asignársele un peso a cada escenario, en términos de su importancia relativa al éxito del sistema. Esta asignación de peso suele hacerse con base en la metas del negocio que cada escenario soporta.

2.7.2.2 Architecture Trade-off Analysis Method (ATAM)

Según (Kazman, Clements et al., 2001), el **Método de Acuerdos de Arquitectura** (por sus siglas en inglés, **ATAM**) está inspirado en tres áreas distintas: *los estilos arquitectónicos*, el

análisis de atributos de calidad y el *método de evaluación SAAM*. El nombre del método SAAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de calidad interactúan con otros, estos es, los tipos de acuerdos que se establecen entre ellos.

El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados. (Kazman, Clements et al., 2001) proponen el término enfoque arquitectónico dado que no todos los arquitectos está familiarizados con el lenguaje de estilos arquitectónicos, aún haciendo uso indirecto de estos. De cualquier forma, estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas, entre otros.

El método de evaluación ATAM comprende nueves pasos, agrupados en cuatro fases. Estos pasos son los siguientes:

Fase 1: Presentación

1. **Presentación de ATAM**

   El líder de la evaluación describe el método a los participantes, trata de establecer las expectativas y responde a las preguntas propuestas.

2. **Presentación de las metas del negocio**

   Se realiza la descripción de las metas del negocio que motivan el esfuerzo, y aclara que se persiguen objetivos de tipo arquitectónico.

3. **Presentación de la arquitectura**

   El arquitecto describe la arquitectura, enfocándose en cómo ésta cumple con los objetivos del negocio.

Fase 2: Investigación y Análisis

4. **Identificación de los enfoques arquitectónicos**

   Estos elementos son detectados, pero no analizados en esta etapa.

5. **Generación del *Utility Tree***

   Se elicitan los atributos de calidad que engloban la "utilidad" del sistema (desempeño, disponibilidad, seguridad, modificabilidad, usabilidad, etc.), especificados en forma de escenarios. Se anotan los estímulos y respuestas, además se establece la prioridad entre ellos.

6. **Análisis de los enfoques arquitectónicos**

   Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos del paso 4. En este paso se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.

Fase 3: Pruebas

7. **Lluvia de ideas y establecimiento de prioridades de escenarios**

   Con la colaboración de todos los involucrados, se implementa el conjunto de escenarios.

8. **Análisis de los enfoques arquitectónicos**

   Este paso repite las actividades del paso 6, haciendo uso de los resultados del paso 7. Los escenarios son considerados como casos de prueba para confirmar el análisis realizado hasta el momento.

Fase 4: Reporte

9. **Presentación de los resultados**

   Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

2.7.2.3 Architecture Level Modifiability Analysis (ALMA)

El atributo de calidad que analiza **ALMA** (Bengtsson, Lassing et al., 2004) en una arquitectura de *software* es la *facilidad de modificación*. La facilidad de modificación en un sistema de *software* es la sencillez con la cual este puede ser modificado a cambios en el entorno, cambios en los requerimientos o cambios a la especificación funcional.

ALMA es un método de evaluación orientado a metas: dependiendo de la meta, este método puede ser usado para predecir el costo de mantenimiento en una arquitectura, evaluar los riesgos al haber una modificación en ésta, o comprarar un conjunto de arquitecturas para determinar cuál es la más apropiada en soportar cambios.

ALMA puede ser utilizado una vez que concluye la especificación de la arquitectura (evaluación clásica), no obstante este método puede ser usado si la arquitectura ha sido implementada (evaluación tardía). La técnica de evaluación que utiliza este método es el uso de *escenarios de cambio*. Los escenarios de cambio son usados para capturar eventos futuros en los que se requiere que el sistema sea adaptado. Antes de iniciar la evaluación es necesario que se cuente con la especificación de la arquitectura de *software*, así como los requerimientos no funcionales. Este método está compuesto por cinco pasos, que se describen a continuación:

1. **Definir la meta de evaluación**

   Dependiendo del objetivo de la evaluación se selecciona alguna de las tres metas. Estas metas pueden ser: *predicción del costo de mantenimiento* (estimar el esfuerzo para modificar la arquitectura), *evaluación de riesgos* (tipos de cambios complejos o que sean inflexibles), y *selección de un conjunto de arquitecturas* (compara arquitecturas y selecciona la que más soporta cambios)

2. **Describir la arquitectura de *software***

   En este punto se colecta la información de las partes más relevantes de la arquitectura como son la descomposición de ésta en componentes, las relaciones entre componentes así como las relaciones que existen en el entorno del sistema.

3. **Obtener escenarios**

   Una vez que se cuenta con la información de la arquitectura se procede a encontrar y definir los escenarios de cambio, estos escenarios son agrupados en categorías. Dependiendo de la meta de evaluación se seleccionan los escenarios. Por ejemplo, si la meta es estimar el esfuerzo de mantenimiento, se recomienda seleccionar aquellos escenarios que corresponden a los cambios que tienen una alta probabilidad de que ocurran durante la vida operacional del sistema. Esta actividad finaliza cuando se han identificado los principales escenarios de cambio.

4. **Evaluar escenarios**

   En este punto se realiza un análisis de impacto que consiste en identificar los componentes afectados por los escenarios previamente definidos, determinar el efecto del cambio sobre los componentes así como determinar los efectos del cambio en otros componentes. Los resultados pueden ser expresados de manera *cuantitativa* o *cualitativa.*

5. **Interpretar resultados**

   Finalmente se interpretan los resultados así como las conclusiones del análisis dependiendo de la meta de evaluación seleccionada.

Este método es recomendado para evaluar diferentes dominios como: sistemas de control embebidos, sistemas médicos, telecomunicaciones y sistemas administrativos.

2.7.2.4 Performance Assessment of Software Architecture (PASA)

En el método **PASA** (Williams and Smith, 2002) se utilizan un conjunto de técnicas para *analizar el desempeño* de una arquitectura de *software*. Entre estas técnicas se encuentran el uso de estilos arquitectónicos, anti-patrones, guías de diseño, modelos, así como técnicas empleadas en la ingeniería del desempeño de *software*.

El atributo de calidad que analiza PASA es el *desempeño.* El desempeño se interesa por conocer qué tanto tiempo le toma al sistema de *software* responder cuando uno o varios eventos ocurren, así como determinar el número de eventos procesados en un intervalo de tiempo dado.

PASA puede ser usado una vez que la especificación de la arquitectura ha concluido o ésta se encuentre implementada. La técnica de evaluación principal que utiliza PASA es el uso de *escenarios.* Estos proporcionan una medida de razonamiento con respecto al desempeño del sistema. Si se requieren análisis más detallados, los escenarios sirven como punto de partida

para construir modelos de desempeño. Este método está compuesto de nueve pasos que se describen a continuación:

1. **Presentar el método de evaluación**

   En este paso, se elabora una presentación que contiene el objetivo de realizar la evaluación, en qué consiste el método, qué información de la arquitectura es necesaria para efectuar la evaluación, así como los resultados de la evaluación.

2. **Presentar la arquitectura**

   A continuación el equipo de desarrollo presenta la arquitectura actual de una manera general sin entrar en detalles.

3. **Identificar casos de uso críticos**

   Se seleccionan los casos de uso que son importantes para la operación del sistema como son aquellos que demandan una respuesta de tiempo rápido para el usuario así como los que presentan algún riesgo de desempeño. Es importante aclarar que un caso de uso puede contener uno o varios escenarios, estos describen las secuencias de acciones requeridas para ejecutar el caso de uso. Los escenarios deben ser especificados en UML como diagramas de secuencia.

4. **Seleccionar los escenarios de desempeño principales**

   Por cada caso de uso crítico se deben identificar los escenarios que son importantes con respecto al desempeño. Por ejemplo, seleccionar aquellos escenarios que son ejecutados frecuentemente así como aquellos que son críticos desde la percepción de desempeño del usuario.

5. **Identificar objetivos de desempeño**

   Por cada escenario de desempeño principal, se debe especificar al menos un objetivo de desempeño. Los objetivos de desempeño pueden ser expresados de diferentes maneras. Por ejemplo, expresarlos en tiempo de respuesta, capacidad de respuesta, restricciones o utilización de recursos. Cabe señalar que el objetivo de desempeño debe ser *cuantitativo* para que este pueda ser medido.

6. **Clarificar la arquitectura y discutirla**

   En este paso se estudia la arquitectura más a detalle por lo que se tienen reuniones con el arquitecto y miembros del equipo de desarrollo para aclarar dudas con respecto a las interacciones entre componentes si existe información acerca del desempeño del sistema se colectan métricas.

7. **Analizar la arquitectura**

   En este paso se utilizan diferentes técnicas para analizar el desempeño de la arquitectura. Por ejemplo, se identifican estilos arquitectónicos con la finalidad de detectar efectos negativos en el desempeño, se identifican anti-patrones de

desempeño que documenten problemas comunes de desempeño y se sugieren formas de cómo resolverlos, se elaboran modelos de desempeño con el objetivo de identificar problemas en la arquitectura. También se pueden utilizar técnicas de ingeniería del desempeño de *software* para elaborar modelos más precisos.

8. **Identificar alternativas**

   Si se encuentran problemas de desempeño, se identifican alternativas de solución para satisfacer los objetivos de desempeño. Estas alternativas pueden incluir el cambio de estilos arquitectónicos, modificar la arquitectura para eliminar anti-patrones de desempeño, o alternativas para optimizar la interacción entre componentes.

9. **Presentar resultados**

   Finalmente se elabora un documento con los resultados de la evaluación que incluyen los objetivos de la evaluación, hallazgos encontrados, pasos específicos a seguir y recomendaciones.

2.7.2.5 Scenario based Architecture Level Usability Analysis (SALUTA)

SALUTA (Folmer, Gurp et al., 2003) es el primer método desarrollado para evaluar la *facilidad de uso* de una arquitectura *software*. La facilidad de uso es la facilidad con la cual el usuario puede aprender a operar, preparar entradas e interpretar las salidas de un sistema o componente.

Este método hace uso de un marco de referencia elaborado por los mismos autores en el que se expresan las relaciones que existen entre la facilidad de uso y la arquitectura de *software* (Folmer, Gurp et al., 2003). Este marco de referencia incluye un conjunto integrado de soluciones de diseño tales como patrones de diseño y propiedades que tienen un efecto positivo sobre la facilidad de uso en un sistema de *software*. SALUTA analiza cuatro atributos que están directamente relacionados con la facilidad de uso en un sistema de *software*, estos son: *facilidad de aprendizaje*, *eficiencia de uso, confiabilidad* y *satisfacción.* Se recomienda efectuar la evaluación una vez que la especificación de la arquitectura ha finalizado y ésta no se ha implementado.

La técnica principal de evaluación que utiliza este método es el uso de *escenarios de uso*. Estos se agrupan en uno o varios perfiles de uso. En un perfil de uso se representa la facilidad de uso requerida por el sistema de *software*. Para efectuar la evaluación se requiere que la especificación de la arquitectura se encuentre disponible, así como el documento que incluya los requerimientos no funcionales relacionados con la facilidad de uso. Este método está compuesto por cuatro pasos, descritos a continuación:

1. **Crear perfiles de usuario**

   En este punto se identifican los usuarios y se organizan en categorías. A continuación se identifican las tareas más significativas del sistema, se identifica el contexto donde será usado el sistema, se determinan los valores para los atributos: facilidad de aprendizaje, eficiencia de uso, confiabilidad y satisfacción. Para reflejar la diferencia en

la prioridad de los atributos, se asigna a cada uno un valor numérico no repetido entre 1 y 4. Por último, se seleccionan los escenarios más representativos que tienen mayor prioridad con respecto a sus atributos.

2. **Describir la facilidad de uso proporcionada**

   Se colecta la información de la arquitectura *software* para determinar el nivel de soporte en los escenarios de uso. Esto se realiza efectuando un análisis basado en patrones o basado en propiedades. En ambos análisis se utiliza el marco de referencia para determinar la presencia de patrones o propiedades de facilidad de uso en la arquitectura a evaluar.

3. **Evaluar escenarios**

   En este paso se evalúa cada uno de los escenarios que forman parte del perfil de uso. Por cada escenario se analizan los patrones y propiedades previamente identificados. Se recomienda usar el marco de referencia para analizar cómo un patrón o propiedad particular afecta a un atributo específico en un escenario de uso. Finalmente en este paso se expresan de manera cuantitativa los resultados del análisis, que indican el grado de soporte de facilidad de uso en cada uno de los escenarios.

4. **Interpretar resultados**

   En este paso se obtienen los resultados de la evaluación que contienen el grado de facilidad de uso que soporta la arquitectura evaluada.

SALUTA se encuentra en etapa de maduración ya que solamente se ha probado en algunos casos de estudio, sin embargo los resultados que se han obtenido son satisfactorios.

2.7.2.6 Survivable Network Analysis (SNA)

SNA (Ellison, Linger et al., 1999) es un método desarrollado por CERT (Computer Emergency Response Team) que forma parte del SEI (*Software Engineering Institute*). Este método ayuda a identificar la *capacidad de* supervivencia en un sistema analizando su arquitectura. La supervivencia es la capacidad que tiene un sistema para completar su misión a tiempo ante la presencia de ataques, fallas o accidentes. Un ejemplo de la definición anterior es la siguiente: un cajero automático debe garantizar al usuario los servicios esenciales aun cuando este se encuentre en presencia de algún ataque externo o falla interna. SNA utiliza tres propiedades claves para evaluar la supervivencia en un sistema. Estas son:

1. **Resistencia:** es la capacidad del sistema para repeler ataques, fallas o accidentes.

2. **Reconocimiento:** es la capacidad de detectar ataques, fallas o accidentes y si estos ocurren evaluar los daños.

3. **Recuperación:** Es la capacidad de mantener en operación los servicios esenciales en presencia de ataques, fallas o accidentes.

Este método puede ser utilizado en la construcción de la arquitectura (evaluación temprana), una vez que la construcción de esta se ha terminado (evaluación tardía) o si la arquitectura se encuentra implementada.

La técnica de evaluación que utiliza SNA es el uso de *escenarios*. Este método hace uso de dos tipos de escenarios. El primer tipo son los escenarios normales de uso, estos se componen de una serie de pasos donde los usuarios invocan a servicios y obtienen acceso a activos tales como base de datos. El segundo tipo de escenarios son los de intrusión, en los que se representan diferentes tipos de ataques al sistema. Para llevar a cabo la evaluación, se requiere que se cuente con la especificación de la arquitectura. Este método está compuesto por cuatro pasos que a continuación se describen:

1. **Definición del sistema**

   En este paso se obtiene la misión del sistema, se discute el entorno de uso en términos de capacidades y ubicaciones de los usuarios, se analizan posibles riesgos que el sistema pueda presentar en condiciones adversas, finalmente se analiza la arquitectura *software*.

2. **Definición de las capacidades esenciales**

   A continuación se seleccionan los servicios esenciales y los activos que usan. Se deben seleccionar aquellos servicios y activos que sean críticos para garantizar en condiciones adversas la misión del sistema. Una vez seleccionados, estos se trazan a través de la arquitectura para identificar los componentes que participan en proporcionar los servicios esenciales.

3. **Definición de las capacidades que comprometen al sistema**

   En este paso se selecciona un conjunto representativo de posibles ataques basados en el entorno de operación del sistema. Se definen los escenarios de intrusión y se trazan a través de la arquitectura para identificar componentes que comprometen la supervivencia del sistema logrando así el acceso y daño a este.

4. **Analizar la supervivencia**

   Finalmente se identifican los escenarios de intrusión que contienen los componentes esenciales y que comprometen la supervivencia del sistema. Por cada componente identificado se procede a analizarlo en términos de las capacidades de resistencia, reconocimiento y recuperación. Estos análisis se colocan en una tabla llamada mapa de supervivencia que contiene por cada escenario de intrusión las estrategias de supervivencia actuales y las recomendadas con respecto a cada una de las capacidades.

El documento que se obtiene al final de la evaluación es un documento que incluye modificaciones recomendadas a la arquitectura acompañadas del mapa de supervivencia. Este método se considera maduro ya que se ha utilizado satisfactoriamente en sistemas comerciales y de gobierno.

En la tabla 2-4 se muestra una comparativa entra cada uno de los métodos analizados. Esta comparación nos permite evaluar los beneficios y posibilidades que ofrecen cada uno de ellos, para luego seleccionar el que más se ajusta a nuestras necesidades.

Para evaluar nuestra tesis, se usarán algunos de estos métodos, lo que nos permite comprobar cada uno de los objetivos iniciales planteados desde el inicio del desarrollo de la arquitectura.

	SAAM	ATAM	ALMA	PASA	SALUTA	SNA
Atributo de calidad	Facilidad de modificación, Seguridad, Confiabilidad, Desempeño.	Facilidad de modificación, Funcionalidad.	Facilidad de modificación.	Desempeño	Facilidad de uso.	Supervivencia.
Tipo de evaluación	Temprana/ Clásica	Clásica	Clásica	Clásica	Clásica	Temprana/ Clásica
Técnica de evaluación	Utility Tree, Análisis arquitectónico	Lluvia de ideas, Análisis de escenarios.	Escenarios de cambio.	Escenarios.	Escenarios de uso.	Escenarios normales de uso, Escenarios de intrusión.
No. de pasos	6	9	5	9	4	4
Salidas	Proyección sobre las arquitecturas de los escenarios que representan los cambios, entender la funcionalidad del sistema.	Identificación de los estilos o enfoques arquitectónicos.	Dependiendo de la meta de evaluación se generan los resultados.	Hallazgos encontrados, pasos específicos a seguir y recomendaciones.	Grado de facilidad de uso que soporta la arquitectura evaluada.	Modificaciones recomendadas a la arquitectura y mapa de supervivencia.
Duración	No especificada	No especificada	No especificada	7 días	No especificada	No especificada
Validación del método	Todo tipo de sistemas.	Sistemas Web, sistemas comerciales y financieras.	Sistemas de control embebido, sistemas médicos, sistemas administrativos.	Sistemas basados en Web, aplicaciones financieras y sistemas en tiempo real.	Algunos casos de estudio incluyen principales sistemas Web.	Sistemas comerciales y de gobierno

Tabla 2-4. *Tabla comparativa de los diferentes métodos de evaluación analizados*[6].

2.7.2.7 Otros métodos de evaluación

Existen otros métodos que apoyan la evaluación de arquitectura *software*, por ejemplo, *Active Reviews for Intermediate Designs* (**ARID**) (Kazman, Clements et al., 2001), que se utiliza para realizar evaluaciones tempranas del desarrollo y evalúa aspectos de documentación y la convivencia de los servicios que provee el diseño propuesto. El *Método de negociación WinWin* (In, Kazman et al., 2001), que provee un marco de referencia general para identificar y resolver conflictos de requerimientos, mediante la elicitación y negociación de artefactos en función de las condiciones de ganancia. Otro método es *Cost-Benefit Analysis Method* (**CBAM**)

[6] Fuente: http://www.osgg.net

(In, Kazman et al., 2001), este es un marco de referencia que no toma decisiones por los involucrados en el desarrollo del sistema. Ayuda en la elicitación y documentación de los costos, beneficios e incertidumbre, y provee un sistema de toma de decisiones racional.

Cada método aporta un beneficio según el aspecto de calidad que se quiera evaluar. Estableciendo las técnicas que utilizan cada uno de ellos y el procedimiento más adecuado para llevar a cabo una evaluación completa.

2.8 CONCLUSIONES

En una etapa inicial de esta investigación, se ha explorado el campo teórico y el terreno donde se ubica las propuestas a desarrollar en los próximos capítulos. Con esto se aclaran conceptos que permiten abordar la investigación

El objetivo de este capítulo ha sido el de proporcionar una breve panorámica sobre los temas tratados en nuestra tesis. Hemos explicado cada uno de los diferentes conceptos necesarios para entender el presente trabajo y los paradigmas sobre los cuales nos hemos apoyado, además de las técnicas y tecnologías usadas para su desarrollo.

Las especificaciones en detalle de las tecnologías elegidas para el desarrollo de cada uno de los elementos del *framework*, serán abordados en las diferentes secciones del capítulo sobre el diseño conceptual. Es por ello, que se han definido en cortas líneas, dentro de este capítulo, las que consideramos nos ofrecen procedimientos o metodologías de desarrollo acorde a los objetivos planteados inicialmente.

3

CAPÍTULO TERCERO

3 TRABAJOS RELACIONADOS: MONITORIZACIÓN DE PACIENTES

3.1 INTRODUCCIÓN

La revisión sistemática del estado del arte nos permite identificar, evaluar e interpretar todos los posibles trabajos relevantes asociados a una pregunta en particular o tema de un área. Para ellos utilizaremos algunas técnicas o protocolos (Kitchenham, 2004) que nos permiten la adecuada selección de estudios primarios y secundarios. El principal objetivo de esta revisión es identificar trabajos, investigaciones y publicaciones hechas en el campo de la monitorización móvil de pacientes a través de un *framework* o a través de generadores de aplicaciones.

Como mencionamos previamente esta revisión sistemática se basa en la metodología B. Kitchenham (Kitchenham, 2004). Ella propone directrices específicas para llevar a cabo la revisión sistemática en ingeniería de *software*.

Su propuesta define una revisión sistemática de la siguiente manera: "*Una revisión sistemática es un medio de evaluar e interpretar todas los trabajos disponibles pertinentes a una pregunta de investigación en particular, área temática, o fenómeno de interés. Las revisiones sistemáticas tienen como objetivo presentar una evaluación justa de un tema de*

investigación mediante una fiable, rigurosa y verificable metodología."

La propuesta se basa en directrices similares para los investigadores con el objetivo de reutilizar el conocimiento y la experiencia de una conocida y consolidada área de investigación. Su intención principal con las directrices es introducir el concepto de una revisión rigurosa de la evidencia empírica actual a la comunidad de ingeniería de *software*. Hay varias características que diferencian a una revisión sistemática de una revisión de la literatura convencional:

- **Definición del protocolo de revisión**: es en este apartado donde se especifica la pregunta de investigación que se aborda y los métodos que se utilizarán para llevar a cabo la revisión.
- **Definición de la estrategia de búsqueda:** la estrategia de búsqueda tiene por objeto detectar tanto de la literatura relevante como sea posible.
- **Búsqueda de documentados:** por lo que los lectores puedan evaluar su rigor y su integridad.
- **Criterios de inclusión y exclusión explícitos:** para evaluar cada estudio primario (una revisión sistemática es un estudio secundario basado en estudios primarios).
- **Revisiones sistemáticas**: especifica la información que será obtenida de cada uno de los estudios primarios, incluidos los criterios de calidad para evaluar cada estudio primario. Una revisión sistemática es un requisito previo para el meta-análisis cuantitativo.

3.2 DEFINICIÓN DEL PROTOCOLO DE REVISIÓN

Un protocolo de revisión define los métodos que se utilizarán para llevar a cabo una revisión sistemática específica. Un protocolo predefinido es necesario para reducir el sesgo del investigador (Kitchenham, 2004).

Los siguientes puntos son los indicados en las directrices para el desarrollo de un protocolo de revisión:

- **La pregunta de investigación:** Es la intención de la revisión sistemática para responder a una pregunta que identifica ámbitos de las futuras actividades de investigación. La pregunta de investigación de esta revisión sistemática es la siguiente:

  "Es posible el desarrollo de un framework que permita la generación de aplicaciones parametrizadas que facilite la monitorización a través de la comunicación entre dispositivos móviles y biométricos para el control médico de pacientes"

- **Definición de la estrategia de búsqueda:** nuestra estrategia de búsqueda se basa en dos aspectos evaluados, el referente a la monitorización móvil de pacientes y el desarrollo de aplicaciones para dispositivos móviles.

  - **Búsqueda y extracción de datos:** los pasos a seguir para la búsqueda y extracción de datos se basaron en los siguientes pasos:

- ***Selección del tema*:** se ha estudiado temas referentes al desarrollo de aplicaciones móviles y la monitorización de pacientes.

- ***Búsqueda de artículos con títulos referentes a este tema***: se han incluido los temas referentes a la generación de aplicaciones para dispositivos móviles que faciliten la monitorización y control de pacientes.

- ***Lectura de resúmenes (abstracts) y palabras claves*:** hemos leído los *abstracts* de los posibles estudios primarios y sus palabras claves. Las palabras claves seleccionadas para nuestra búsqueda son: *framework*, *generadores de aplicaciones*, *movilidad*, *patrones de diseño*, *guías de diseño*, *arquitectura*, *monitorización*. Se han incluido estudios escritos en inglés y español.

- ***Selección de estudios primarios*:** los estudios o documentos que se asemejan más al área de estudio fueron leídos en profundidad clasificándolos según las palabras claves encontradas.

- ***Análisis comparativo de los estudios primarios***: se han evaluado cada uno de los artículos comparándolos con la propuesta de desarrollo de los cuales se han extraído conocimientos que diferencian los trabajos desarrollados con la *framework* propuesta.

Para esta investigación, se ha hecho un estudio más profundo. En primer lugar, búsquedas de palabras clave de los papeles más referenciados en:

- IEEE Xplore (http://www.computer.org)
- ACM (http://portal.acm.org)
- Portal de SpringerLink (http://www.scopus.com)
- ELSEVIER (http://elsevier.com/)
- SCOPUS.

Además de evaluación de artículos publicados en congresos y conferencias en temas de AAL (*Ambient Assisted Living*), desarrollo de aplicaciones móviles, comunicación móvil, entre otros.

3.3 CRITERIOS DE EVALUACIÓN

Para el análisis y evaluación de cada uno de los trabajos relacionados hemos definido un conjunto de criterios que nos permitirán evaluar y ubicar aspectos de diseño, adaptabilidad, comunicación, seguridad y costos en cada investigación. Los criterios de evaluación para cada trabajo evaluado son:

- **Diseño**
  - ***Cohesión***: la manera en la cual dividimos físicamente un sistema en piezas (particularmente en relación con la estructura del problema) puede afectar

significativamente la complejidad estructural del sistema resultante, así como el número total de referencias intermodulares.

Podemos decir que un sistema modularmente más efectivo es aquel cuya suma de la relación funcional entre pares de elementos que pertenezcan a diferentes módulos sea mínima. Entre otras cosas, esto tiende a minimizar el número de conexiones intermodulares requeridas y el acoplamiento intermodular. Esta relación funcional intermodular se conoce como cohesión.

Dentro de la evaluación del estado del arte, analizaremos el grado de cohesión que tienen las aplicaciones que se han diseñado para la monitorización y control de pacientes. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-1. *Asignación de peso para el sub-criterio Cohesión*

- ***Acoplamiento:*** módulos altamente "acoplados" estarán unidos por fuertes interconexiones, módulos débilmente acoplados tendrán pocas y débiles interconexiones, en tanto que los módulos "desacoplados" no tendrán interconexiones entre ellos y serán independientes. El *acoplamiento* es un concepto abstracto que nos indica el grado de interdependencia entre módulos.

  En la práctica podemos materializarlo como la probabilidad de que en la codificación, depuración, o modificación de un determinado módulo, el programador necesite tomar conocimiento acerca de partes de otro módulo. Si dos módulos están fuertemente acoplados, existe una alta probabilidad de que el programador necesite conocer uno de ellos en para intentar realizar modificaciones al otro.

  De igual manera que se ha hecho en el subcriterio anterior, se analizará el grado de acoplamiento que tienen las aplicaciones que se han diseñado para la monitorización y control de pacientes. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alto	AT	1
Media	MD	2
Bajo	BJ	3

Tabla 3-2. *Asignación de peso para el sub-criterio Acoplamiento*

- ***Usabilidad***: la utilidad de un sistema tiene una componente de funcionalidad (utilidad funcional) y otra basada en el modo en que los usuarios pueden usar dicha funcionalidad. Es este componente el que nos interesa. Podemos definir la usabilidad como la medida en la cual un producto puede ser usado por usuarios

con el fin de conseguir objetivos específicos con alto grado den efectividad, eficiencia y satisfacción en un contexto determinado.

Para nuestro análisis, se evalúan los aspectos esenciales para conocer qué tan amigable son las aplicaciones propuestas o desarrolladas para la monitorización de pacientes. Además del funcionamiento, se evalúa la forma o diseño para saber si la interacción es sencilla o compleja. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-3. *Asignación de peso para el sub-criterio Usabilidad*

- **Adaptabilidad**
  - ***Capacidad evolutiva:*** nos permite conocer la capacidad que tiene una aplicación de adaptarse a cambios que se producen en el entorno, es decir, la capacidad que tienen las aplicaciones de evolucionar con el tiempo.

    Esto es la capacidad de ajustarse a otras necesidades cambiantes con respecto a la idea inicial de su implementación. Muchas propuestas son desarrolladas como una solución puntual, limitándolas a evolucionar en el futuro. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-4. *Asignación de peso para el sub-criterio Capacidad evolutiva*

  - ***Migración del dominio de aplicación***: además de la capacidad evolutiva encontramos un aspecto importante a evaluar y es la migración del dominio de aplicación. Esta no es más que la facilidad de adaptación que tiene la propuesta inicial de ser usada en otras áreas diferentes a la inicialmente desarrollada. En nuestro caso podría referirse a la facilidad de funcionar tanto en una enfermedad X como en una enfermedad Y.

    Algunos desarrollos no contemplan la posibilidad de que toda o parte de la arquitectura desarrollada pueda ser implementada en otro dominio de aplicación diferente para el que fue desarrollado inicialmente.

    El desarrollo se piensa, desde el principio, para un requerimiento particular de dominio, no tomándose en cuenta que los cambios tecnológicos y sociales pueden hacer que un desarrollo inicial se implemente en otros entornos. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-5. *Asignación de peso para el sub-criterio Migración del dominio de aplicación*

- ***Migración Tecnológica***: el desarrollo de una aplicación dentro de un entorno particular debe poder ajustarse continuamente a los cambios tecnológicos que se presentan contínuamente. Especialmente en el desarrollo de aplicaciones para dispositivos móviles, en donde el cambio tecnológico de *software* y *hardware* van creciendo constantemente a pasos agigantados.

  Desarrollar una aplicación para entornos médicos, sin tomar en cuenta la posibilidad que tiene ésta aplicación de ajustarse a nuevos cambios tecnológicos, dificulta la posterior migración y la correcta adecuación tecnológica.

  En este criterio se evalúa la capacidad que tienen las aplicaciones desarrolladas de migrar fácilmente ante cambios tecnológicos en los dispositivos y sistemas operativos para los que fueron desarrollados inicialmente. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-6. *Asignación de peso para el sub-criterio Migración Tecnológica*

- **Comunicación**
  - ***Comunicación externa o transmisión de datos***: Existe un sinnúmero de tecnologías de comunicación entre dispositivos. Cada una de ellas con las ventajas y características que las definen. Desarrollar aplicaciones móviles es sumamente diferente al desarrollo aplicaciones para entornos fijos. Hay que tomar en cuenta muchos aspectos que en un desarrollo normal no se tomarían.

    La transmisión de datos es uno de los aspectos más importantes ya que con ella se analizan aspectos como *tiempo de respuesta*, *tipo de comunicación*, *tiempo funcional del dispositivo en el cual está instalada la aplicación móvil* y, muy importante, la *seguridad de los datos al ser transmitidos (este criterio será evaluado como un criterio separado)*.

    En este criterio se evalúan las tecnologías de trasmisión de datos sobre las cuales se han desarrollado algunas investigaciones que hacen referencia a la monitorización de pacientes. Se toma en cuenta el tipo de datos que se trasmite y la tecnología sobre la que está soportado el envío y recepción de estos datos. Es decir, si se han utilizado tecnologías de trasmisión/recepción entre todos los elementos que intervienen en la aplicación desarrollada o propuesta. Para este criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-7. *Asignación de peso para el sub-criterio Transmisión de datos*

- **Seguridad**
  - ***En el Tratamiento de los datos:*** La Ley 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal (en adelante "Ley de Protección de Datos"), limita el uso de la informática y otras técnicas y medios de tratamiento automatizado de datos de carácter personal, con el fin de garantizar el honor, la intimidad familiar y personal de las personas físicas y el pleno ejercicio de sus derechos.

    La Ley de Protección de Datos en su artículo 9 establece la obligación del responsable del fichero de adoptar las medidas de índole técnica y organizativa que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado, que se pueda presentar del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que estén expuestos.

    El tratamiento de los datos de pacientes es uno de los principales inconvenientes que se tiene a la hora de desarrollar aplicaciones de este tipo. En este criterio se evalúan los aspectos referentes al tratamiento privado de estos datos. Si se han contemplado directrices para la protección de datos utilizados. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-8. *Asignación de peso para el sub-criterio Tratamiento de los datos*

  - ***En la Transferencia de datos***: La transferencia de datos entre dispositivos es un aspecto que debe evaluarse desde diferentes ámbitos. Uno de ellos es la seguridad con que los datos son trasmitidos, esto es, mantener la integridad y evitar la posibilidad de que estos datos se pierdan en el camino o sean captados por otros. Existen diferentes técnicas informáticas que permiten mantener un alto grado de seguridad en la transferencia de datos.

    En este criterio se evalúan aspecto de seguridad en la transferencia de datos entre cada uno de los dispositivos que intervienen en las investigaciones que se han desarrollado. Este aspecto es de gran importancia, ya que se analizará cada propuesta, para conocer qué soluciones se han utilizado para mantener la seguridad de comunicación entre todos los elementos de cada propuesta. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alta	AT	3
Media	MD	2
Baja	BJ	1

Tabla 3-9. *Asignación de peso para el sub-criterio Transferencia de datos*

- **Costos**
  - ***De Implementación***: Al momento de implementar la arquitectura *software* debe tenerse en cuenta el costo que trae consigo su desarrollo, es decir, conocer de antemano en qué tipo de dispositivo móvil será utilizado y sobre que plataforma se ejecutará.

    Existen en el mercado actualmente diferentes tipos de dispositivos móviles con características propias, asociadas directamente al costo que tiene cada uno de ellos, según las prestaciones tecnológicas con que cuentan.

    En este criterio se evalúan los aspectos relacionados a los costos de desarrollo de la aplicación. Es decir costos en dispositivos *hardware* y *software* necesarios para el desarrollo. Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alto	AT	1
Medio	MD	2
Bajo	BJ	3

Tabla 3-10. *Asignación de peso para el sub-criterio Costo de implementación*

  - ***De Mantenimiento***: además del costo económico evaluado en el criterio previo, está el costo de mantenimiento. Este costo hace referencia al esfuerzo que conlleva actualizar, cambiar o mejorar la propuesta inicial a otros entornos o en el mismo entorno con nuevos requerimientos.

    Si se ha hecho un diseño bien distribuido en cada uno de sus componentes, se hará más fácil el mantenimiento posterior de la aplicación móvil desarrollada. Es importante para ello ubicar de manera clara, no solo para los desarrolladores iniciales, todos los elementos de tal manera que se puedan identificar cada una de sus partes, facilitando el reajuste de la aplicación inicial sin tener que revisar toda su estructura.

    Para este sub-criterio se han utilizado las siguientes siglas y pesos:

Valoración	Siglas	Peso
Alto	AT	1
Medio	MD	2
Bajo	BJ	3

Tabla 3-11. *Asignación de peso para el sub-criterio Costo de mantenimiento*

3.4 TRABAJOS EN EL ÁREA DE MONITORIZACIÓN DE PACIENTES.

Facilitar la comunicación con el paciente y el médico a través de tecnologías que permitan el seguimiento de cada paciente, ha significado siempre un gran avance en el área de ambientes asistenciales. Los cuidados asistenciales (*AL*) y cuidados en salud (*healthcare*) son actividades provistas por profesionales especialistas en estas actividades o personal capacitado en estos entornos.

Numerosas propuestas han sido presentadas recientemente para sistemas que usan la infraestructura fija en donde pacientes y médicos estén juntos en las actividades de diagnóstico y monitorización. Estos sistemas se centran en una aplicación específica, como la transmisión de datos de electrocardiogramas (*ECG*) a través de Internet para que el médico no necesite hacer una llamada a la casa del paciente (Magrabi, Lovell et al., 1999), o el caso general de supervisar los parámetros no especificados en uno (Park, Park et al., 1998) o varios (Lee, Park et al., 1997) hogares.

Se han propuesto también comunicaciones de corto alcance que permitan establecer una relación unilateral entre el monitor *ECG* de un paciente y su ordenador personal, para proporcionar la adquisición más conveniente de datos de *ECG* para su transmisión final (Kong, Ng et al., 2000). El objetivo de estas propuestas es proporcionar ayuda adicional a los pacientes ambulatorios con alto riesgo de complicaciones médicas, mientras se mantiene el costo de operación del médico suficientemente baja.

La llegada de bajos costos de dispositivos inalámbricos también ha generado varias implementaciones para la recogida de datos mediante redes personales (Brhargava and Zoltowiski, 2003) (Gao, Greenspan et al., 2005) (Krco and Delic, 2003) (Pattichis, Kyriacou et al., 2002) y las comunicaciones inalámbricas de largo alcance (Kong, Ng et al., 2000) (Lin, Jan et al., 2004) (Pattichis, Kyriacou et al., 2002). (Bhargava and Zoltowiski, 2003) propone algunas de las consideraciones de seguridad necesarias para un médico basada en redes *ad hoc*, en respuesta a la evolución de las especialidades y las implementaciones específicas del proveedor. Por su parte (Varady, Benyo et al., 2002) propone una arquitectura abierta para las redes de cama que, si bien intrínsecamente utilizan conexiones a través de alambres, fácilmente podría extenderse a la esfera de los dispositivos inalámbricos.

Además, los avances en computación ubicua han alentado a las aplicaciones para la visualización de datos en dispositivos informáticos móviles, como *PDAs* y *Pocket PC* (Hung and Zhang, 2003) (Nelwan, van Dam et al., 2002). Por lo tanto, mucha atención se ha prestado a las áreas de la telemedicina y la adquisición automática de datos. Las soluciones propuestas y aplicadas permiten proporcionar atención sanitaria a distancia en el hogar y en movimiento. Lo que ha estado ausente de todos estos sistemas, sin embargo, es la integración de la adquisición de datos con la recuperación de datos.

Algunos investigadores han tratado de dar su aporte en diferentes áreas tecnológicas mencionadas a continuación:

Roy (Nirmalya, Gautham et al., 2007) propone un *framework* que soporta la fusión eficiente de datos de percepción del contexto para aplicaciones de cuidados médicos que son

asumidas como un contexto ambiguo. Provee una aproximación sistemática para derivar fragmentos del contexto y manejar la probabilidad de que exista ambigüedad en ese contexto. Con esto se desarrolla un *framework* que soporta la fusión de datos del contexto y la predicción de una situación del usuario basada en Redes Bayesianas Dinámicas para aplicaciones *healthcare* de contexto en ambientes pequeños.

Nuestra Propuesta: tiene pocos problemas de ambigüedad en los datos, para lograr esto, hemos definido un perfil individual para cada paciente; la funcionalidad de la arquitectura software está basada en este perfil. Para evitar la ambigüedad, la definición de módulos está directamente asociada a la definición del perfil individual del paciente y a la utilización de dispositivos biométricos puntuales.

Broens (Broens, Halteren et al., 2007) propone el desarrollo de un *framework* que incorpora el uso de información de contexto orientado a pacientes que padecen epilepsia, el sistema envía mensajes a las diferentes dependencias asociadas al *framework*. En caso de que un paciente tenga síntomas de posible ataque epiléptico, mediante un sistema de seguridad de epilepsia (*ESS*) que incluye un monitoreo ambulante del paciente (*BAN*), el sistema le comunica al paciente que tiene variaciones sintomatológicas que le pueden ocasionar una epilepsia. El sistema revisa si existe algún familiar cerca del paciente, que pueda ofrecerle cuidados médicos o primeros auxilios, de no encontrarse se envía un mensaje a algún voluntario médico, enviando los síntomas y la localización del paciente.

Nuestra Propuesta: propone la monitorización móvil de pacientes con el médico, paciente y la comunicación a través del dispositivo móvil. El dispositivo móvil pertenece al paciente y es el elemento principal y clave en la comunicación y el autocontrol, lo que facilita el seguimiento del mismo. Este dispositivo cuenta con cierta autonomía al momento de presentar mensajes de control para cada enfermedad. Decimos cierta, ya que es al final el médico que interviene frente a situaciones de alerta extrema.

Preuveneers (Preuveneers and Berbers, 2008) ha investigado cómo la plataforma de teléfono móvil puede contribuir con los individuos diagnosticados con diabetes, para manejar sus niveles de glucosa en la sangre sin recurrir a ningún sistema adicional (más allá del equipo que ellos usan actualmente) o sin haber aumentado con sensores de actividad adicionales, como podómetros, acelerómetros o monitores de latidos de corazón. Supervisando la ubicación y la actividad de usuario con el teléfono móvil, reconociendo el comportamiento pasado y conociendo los niveles de glucosa de sangre con datos de contexto, consiguen ayudar en la toma de decisiones bien informadas sobre la dosificación de medicina diaria a ingerir para mantener así los niveles de glucosa estables (ver figura 3-1).

Nuestra Propuesta: contempla una monitorización de las medidas de los pacientes. No es necesario conocer la localización del paciente, pero es crucial para conocer las actividades que el paciente realiza. Esto permite a nuestro sistema aprender para situaciones futuras. Nuestros casos de estudios pueden ser cualquier enfermedad que padece una persona y que pueda ser contralada a través de un dispositivo biométrico.

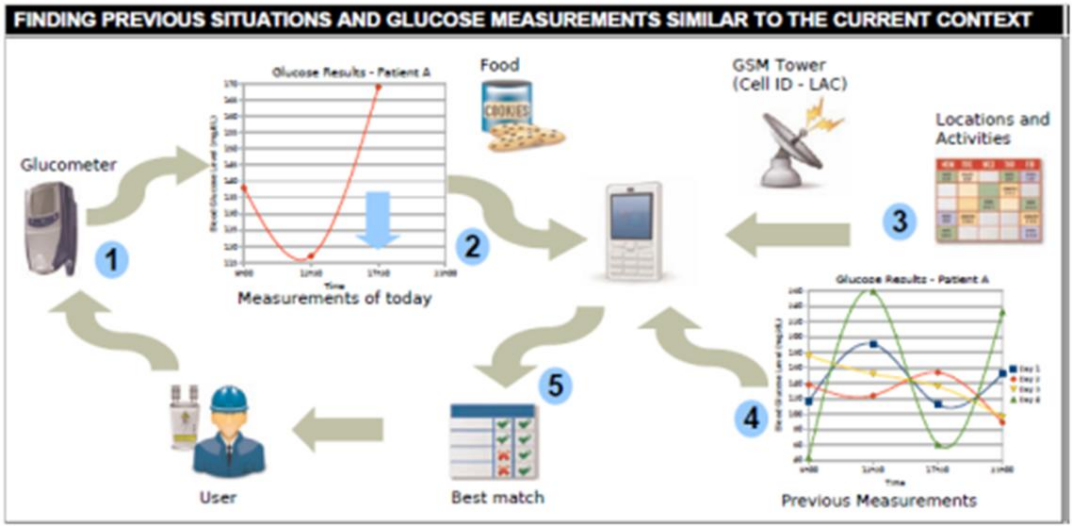


Figura 3-1. *Propuesta de Preuveneers*

Kebler (Kebler, 2007) habla de cómo usar la información de contexto para mejorar el análisis de semejanza. Define tres usos de medida de semejanza en el dominio geoespacial e investiga cuáles aspectos de la definición de Dey y Abowd de contexto (p. ej. la identidad, la actividad, la ubicación y el tiempo) juegan un papel crucial para definir la semejanza en cada uno de estos casos.

Nuestra Propuesta: sigue la ideología planteada por Kebler, definiendo ontologías que nos permitan manejar la información y analizarla de tal manera que ofrezca servicios adecuados para el control de pacientes. Es importante definir aspectos como **perfil del paciente** (*identidad según Kebler*), **actividades** (*actividad según Kebler)* y **medida** (no contemplada por Kebler). La *ubicación* y el *tiempo* según Kebler, son dos aspectos intrínsecos de la **actividad** y la **medida** en nuestra arquitectura software**.**

Universidad de Georgetown, Gentag Inc. y la Corporación Internacional de Aplicaciones de Ciencia (SAIC, NYSE: SAI) (Peeters, Bense et al.) han desarrollado un método para la obtención de medidas de glucosa menos doloroso que lo habitual, con el que se monitoriza y toma medidas al paciente a través de un parche colocado en la piel, con un sensor inalámbrico y un teléfono móvil. En este sistema se pueden obtener beneficios como el control de una bomba de insulina y una geolocalización del paciente (a través de GPS) en caso de emergencia.

Nuestra Propuesta: es aplicada a múltiples enfermedades, logrando así la multi-monitorización buscada inicialmente. La arquitectura software puede ser redefinida para funcionar en un gran número de mediciones, es decir, el comportamiento de la aplicación instalada en el dispositivo móvil tendrá el mismo comportamiento funcional, variando los parámetros iniciales según las medidas de la enfermedad a monitorizar.

Dagtas (Dagtas, Natchetoi et al., 2007) presentó una solución inalámbrica para la monitorización de pacientes con necesidades de asistencia médica. Se utiliza para ello el teléfono móvil y sensores. Las principales funciones de la arquitectura propuesta son: recolección de señales a través de una red de sensores inalámbricos usando

protocolos como *ZigBee* y *Bluetooth*, optimización en el análisis de datos a través de una arquitectura adaptativa que permite el procesamiento eficiente de datos y la notificación y alerta en tiempo real.

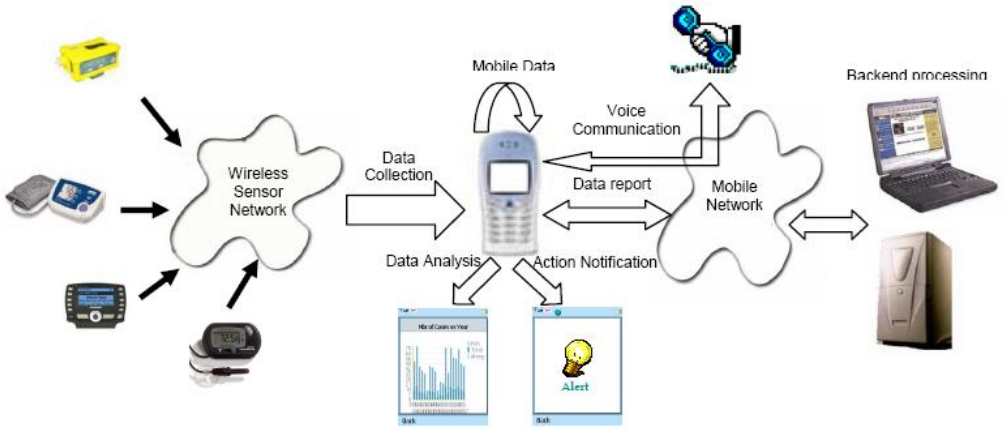


Figura 3-2. *Arquitectura general propuesta por Dagtas.*

En la figura 3-2, se muestra la arquitectura general, que consiste de tres partes. La primera, los datos médicos que son recolectados desde sensores y transmitidos a dispositivos móviles a través de un *WSN* (*Wireless Sensor Network*). La segunda, los datos recolectados, son procesados por una aplicación en *J2ME*[7] que se ejecutan en un dispositivo móvil. Y finalmente, el modelo de recolección de datos, combina los datos desde varias fuentes.

Nuestra Propuesta: a diferencia de la propuesta por Dagtas, hace un análisis de datos y genera alertas asociadas al perfil de cada paciente, considerándose ésta nuestra principal aportación a la monitorización móvil de pacientes. Todas las alertas, recomendaciones, sugerencias, entre otras, están basadas en un motor de búsqueda que está constantemente actualizándose cada vez que se recibe una nueva medición de una enfermedad. Se ha integrado un módulo de envío de SMS para la persona de contacto de un paciente en caso de urgencia.

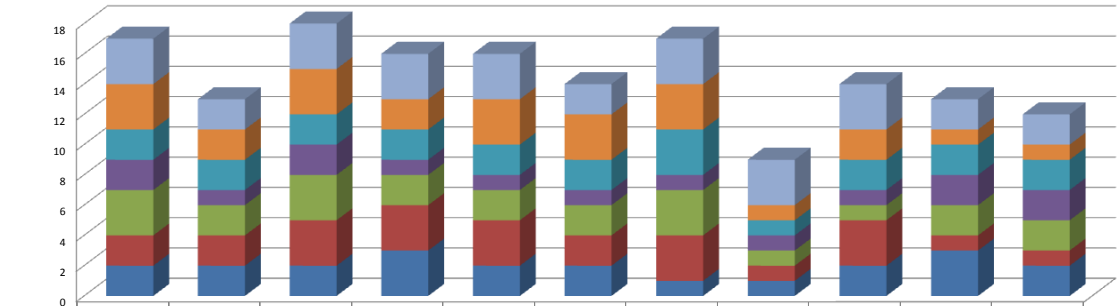
En la tabla 3-12 se muestra una comparativa de los criterios definidos para la evaluación de propuestas de investigadores analizadas previamente. Cada criterio ha sido ponderado de tal manera que puede comparse las propuestas presentadas. En la figura 3-3 puede verse el valor o peso asignados luego de estudiada cada investigación. En el apartado 3.4.1, se analizará por separado cada criterio, así como el peso asignado a las investigaciones previamente estudiadas. Se analiza en cada tabla y gráfico la ponderación asignada a nuestra propuesta para la Monitorización Móvil (MoMo), con respecto a las demás estudiadas.

[7] Java Platform, Micro Edition. http://www.java.com/es/download/faq/whatis_j2me.xml

Propuestas	Diseño						Adaptabilidad						Comunicación		Seguridad				Costos			
	Cohesión		*Acoplamiento*		*Usabilidad*		*Capacidad evolutiva*		*Dominio de aplicación*		*Migración Tecnológica*		*Transmisión de datos*		*Tratamiento de datos*		*Transferencia de datos*		*De implementación*		*De mantenimiento*	
Roy	MD	2	MD	2	MD	2	AT	3	MD	2	MD	2	BJ	1	BJ	1	MD	2	BJ	3	MD	2
Broens	MD	2	MD	2	AT	3	AT	3	AT	3	MD	2	AT	3	BJ	1	AT	3	AT	1	AT	1
Preuveneers	AT	3	MD	2	AT	3	MD	2	MD	2	MD	2	AT	3	BJ	1	BJ	1	MD	2	MD	2
Kebler	MD	2	AT	1	MD	2	BJ	1	BJ	1	BJ	1	BJ	1	BJ	1	BJ	1	MD	2	MD	2
Gentag	MD	2	MD	2	MD	2	MD	2	MD	2	MD	2	AT	3	BJ	1	MD	2	MD	2	MD	2
Dagtas	AT	3	MD	2	AT	3	MD	2	AT	3	AT	3	AT	3	BJ	1	MD	2	AT	1	AT	1
MoMo	AT	3	MD	2	AT	3	AT	3	AT	3	MD	2	AT	3	AT	3	AT	3	MD	2	MD	2

Tabla 3-12. *Comparativa de los criterios definidos para las propuestas de investigadores evaluadas.*

El criterio más altamente contemplado en el desarrollo de las propuestas y aplicaciones para la monitorización de pacientes es el de usabilidad, seguido del tipo de transmisión de datos. Ambos criterios son muy importantes ya que hablamos de monitorización móvil desarrollada para dominios médicos, en donde los usuarios son personas en su mayoría de edades avanzadas.



	Cohesión	Acoplamiento	Usabilidad	Capacidad evolutiva	Dominio de aplicación	Migración Tecnológica	Transmisión de datos	Tratamiento de datos	Transferencia de datos	De implementación	De mantenimiento
	Diseño			Adaptabilidad			Comunicación	Seguridad		Costos	
MoMo	3	2	3	3	3	2	3	3	3	2	2
Dagtas	3	2	3	2	3	3	3	1	2	1	1
Gentag	2	2	2	2	2	2	3	1	2	2	2
Kebler	2	1	2	1	1	1	1	1	1	2	2
Preuveneers	3	2	3	2	2	2	3	1	1	2	2
Broens	2	2	3	3	3	2	3	1	3	1	1
Roy	2	2	2	3	2	2	1	1	2	3	2

Figura 3-3. *Comparativa de los criterios definidos para la evaluación de propuestas de investigadores, según cada subcriterio.*

3.4.1 Análisis de criterios de evaluación de algunas propuestas de investigadores

El análisis que se ha hecho a cada investigación referenciada en el estado del arte, tiene un carácter cualitativo. Sin embargo, puede hacerse una aproximación cuantitativa de cada una de estas investigaciones. Dicho análisis está basado en los criterios previamente definidos, los cuales han sido ponderados para conocer el grado de pertenencia que tienen con la propuesta desarrollada.

1. Diseño

En la tabla 3-13, se explican en detalle los sub-criterios evaluados en el criterio diseño y sus respectivos sub-criterios.

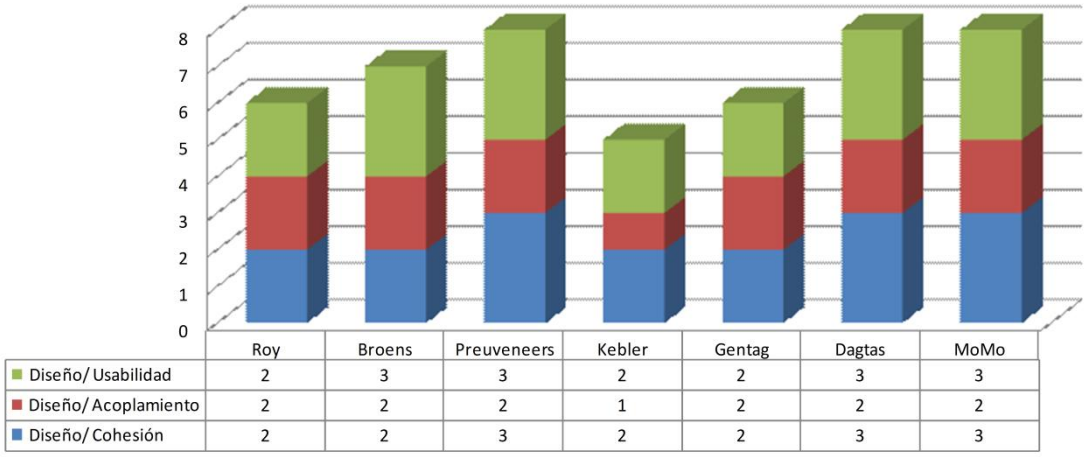
	Valoración	Descripción	Peso
Cohesión	Alta (AT)	La investigación posee un alto grado de cohesión.	3
	Media (MD)	La investigación posee mediano grado de cohesión.	2
	Baja (BJ)	La investigación posee un bajo grado de cohesión.	1

	Valoración	Descripción	Peso
Acoplamiento	Alto (AT)	La investigación posee un alto grado de acoplamiento.	1
	Medio (MD)	La investigación posee un mediano grado de acoplamiento.	2
	Bajo (BJ)	La investigación posee un bajo grado de acoplamiento.	3

	Valoración	Descripción	Peso
Usabilidad	Alta (AT)	La investigación presenta un entorno altamente usable, es decir, ofrece un entorno amigable al usuario.	3
	Media (MD)	La investigación presenta un entorno medianamente usable, es decir, ofrece un entorno medianamente amigable al usuario.	2
	Baja (BJ)	La investigación presenta un entorno poco usable, es decir, ofrece un entorno poco amigable al usuario.	1

Tabla 3-13. *Descripción de las valoraciones del criterio diseño y sub-criterios para de las propuestas de investigadores evaluadas.*

Cuantificando el grado del criterio *diseño* y sus sub-criterios, se muestra la figura 3-4 la evaluación de cada propuesta de investigadores analizadas en esta tesis.



	Roy	Broens	Preuveneers	Kebler	Gentag	Dagtas	MoMo
Diseño/ Usabilidad	2	3	3	2	2	3	3
Diseño/ Acoplamiento	2	2	2	1	2	2	2
Diseño/ Cohesión	2	2	3	2	2	3	3

Figura 3-4. *Nivel de diseño de las propuestas de investigadores evaluadas.*

Podemos notar que el sub-criterio de diseño, es un criterio medianamente contemplado al momento de desarrollar aplicaciones móviles para entornos médicos. Esto puede ser debido a que la mayoría de las aplicaciones desarrolladas, dependen en gran medida de módulos de control inicial, lo que dificulta la distribución de la funcionalidad.

Preuveneers, *Dagtas y MoMo*, se aproximan más a un nivel de cohesión de grado alto con respecto a las otras investigaciones. Mientras que el aspecto más contemplado a la hora de diseñar aplicaciones móviles para el seguimiento de pacientes es el de *usabilidad*. Esto es debido a la importancia que se le da al diseño para que pueda ser más fácil de usar por los usuarios finales. Dentro de los sub-criterios analizados nos interesa más el sub-criterio de *usabilidad*.

2. Adaptabilidad

En la tabla 3-14, se explican en detalle los sub-criterios evaluados en el criterio adaptabilidad y sus respectivos sub-criterios.

	Valoración	Descripción	Peso
Capac. Evolutiva	Alta (AT)	La investigación tiene alta capacidad de adaptarse a nuevos cambios.	3
	Media (MD)	La investigación tiene mediana capacidad de adaptarse a nuevos cambios.	2
	Baja (BJ)	La investigación tiene baja capacidad de adaptarse a nuevos cambios.	1

	Valoración	Descripción	Peso
Mig. dominio de aplicación	Alta (AT)	La investigación posee una alta capacidad para adaptarse a otros dominios de aplicación que el propuesto inicialmente.	3
	Media (MD)	La investigación posee una mediana capacidad para adaptarse a otros dominios de aplicación que el propuesto inicialmente.	2
	Baja(BJ)	La investigación posee una baja capacidad para adaptarse a otros dominios de aplicación que el propuesto inicialmente.	1

	Valoración	Descripción	Peso
Migración Tecnológica	Alta (AT)	La investigación tiene altas posibilidades de adaptarse a nuevos estándares tecnológicos de *hardware* y *software*.	3
	Media (MD)	La investigación tiene medianas posibilidades de adaptarse a nuevos estándares tecnológicos de *hardware* y *software*.	2
	Baja (BJ)	La investigación tiene bajas posibilidades de adaptarse a nuevos estándares tecnológicos de *hardware* y *software*.	1

Tabla 3-14. *Descripción de las valoraciones del criterio adaptabilidad y sub-criterios para las propuestas de investigadores evaluadas*

Cuantificando el grado del criterio *adaptabilidad* y sus sub-criterios, se muestra la figura 3-5, la evaluación de cada propuesta de investigadores analizadas en esta tesis.

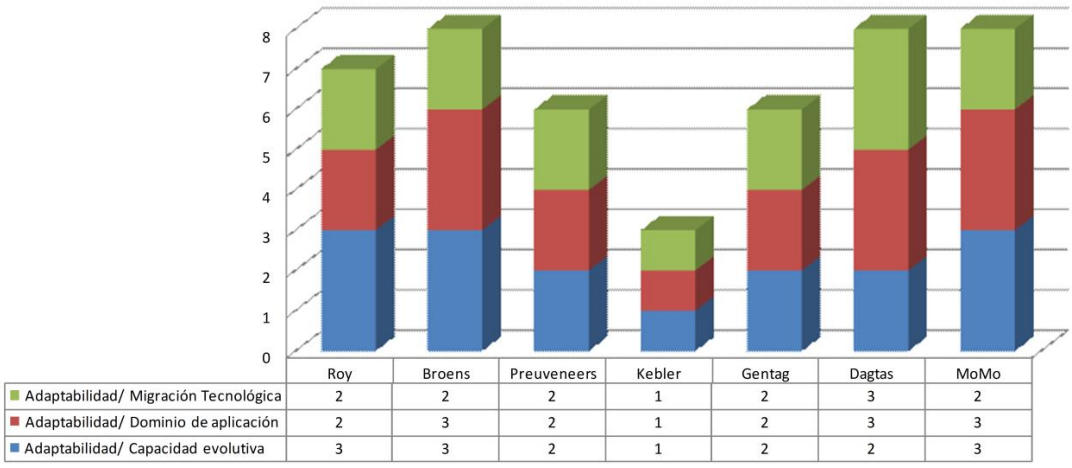


Figura 3-5. *Nivel de adaptabilidad de las propuestas de investigadores evaluadas.*

El nivel de adaptabilidad, como lo hemos mencionado previamente, permite a las aplicaciones desarrolladas poder convivir ante nuevos cambios en el dominio de aplicación, dominio tecnológico y la capacidad de ir cambiando con el tiempo. Las propuestas de *Broens y MoMo* son las que mayor grado de adaptabilidad poseen con respecto a las demás propuestas. Mientras que la de *Kebler* se mantiene en un grado de adaptabilidad *bajo*. Esto nos hace suponer que la mayoría de las aplicaciones desarrolladas han sido pensadas para entornos y tecnologías específicas. Es cierto que desarrollar aplicaciones para plataformas tecnológicas diferentes es un arduo trabajo, y que quizás no se contemplen desde el inicio, pero una correcta distribución e identificación de todos los elementos de la arquitectura, favorece una futura adaptación de éstas.

3. Comunicación

En la tabla 3-15, se explican en detalle los sub-criterios evaluados en el criterio comunicación.

	Valoración	Descripción	Peso
Comunicación de datos	Alta (AT)	La investigación propone o utiliza tecnologías de comunicación entre dispositivos.	3
	Media (MD)	La investigación propone o utiliza medianamente tecnologías de comunicación entre dispositivos.	2
	Baja(BJ)	La investigación no propone o no utiliza tecnologías de comunicación entre dispositivos.	1

***Tabla 3-15**. Descripción de las valoraciones del criterio comunicación y sub-criterios para las propuestas de investigadores evaluadas*

Cuantificando el grado del criterio *comunicación* y sus sub-criterios, se muestra la figura 3-6, la evaluación de cada propuesta de investigadores analizadas en esta tesis.

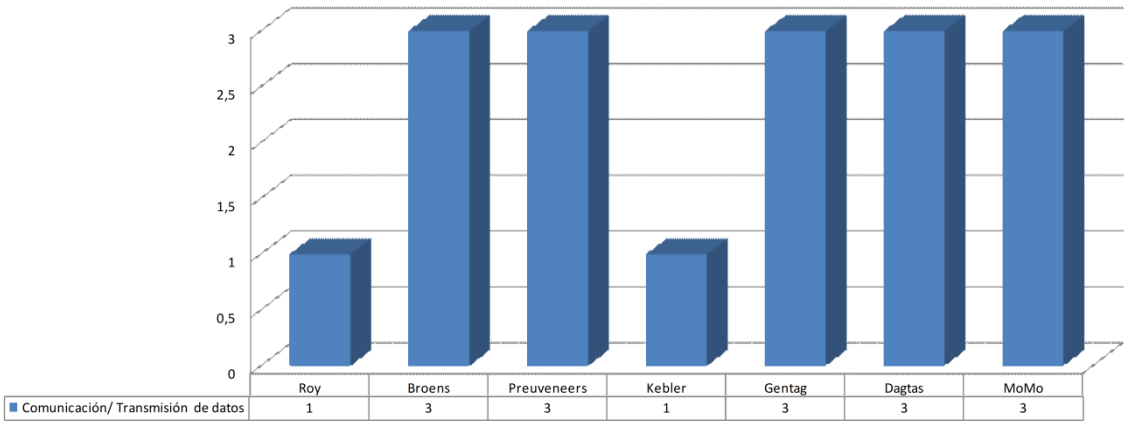


***Figura 3-6.** Nivel de comunicación de las propuestas de investigadores evaluadas.*

Las propuestas de *Broens*, *Preuveeners*, *Gentag*, *Dagtas y MoMo* contemplan aspectos de comunicación (transmisión) de datos. Es decir, utilizan algún tipo de transmisión de datos, basadas en las tecnologías de comunicación existentes. Mientras que *Roy* y *Kebler* han desarrollado propuestas en donde el dispositivo de procesamiento de datos no depende de

ningún tipo de conexión a través de tecnologías de comunicación. Esto no quiere decir que en sus trabajos posteriores no se contemplen, pero al momento de realizar este análisis no lo presentan o existe muy poca relación entre sus propuestas y el criterio evaluado.

4. Seguridad

En la tabla 3-16, se explican en detalle los sub-criterios evaluados en el criterio seguridad y sus respectivos sub-criterios.

	Valoración	Descripción	Peso
Tratamiento de datos	Alta (AT)	La investigación propone un alto tratamiento de los datos de los participantes en la aplicación desarrollada o propuesta.	3
	Media (MD)	La investigación propone medianamente el tratamiento de los datos de los participantes en la aplicación desarrollada o propuesta.	2
	Baja (BJ)	La investigación no propone tratamiento de los datos de los participantes en la aplicación desarrollada o propuesta.	1

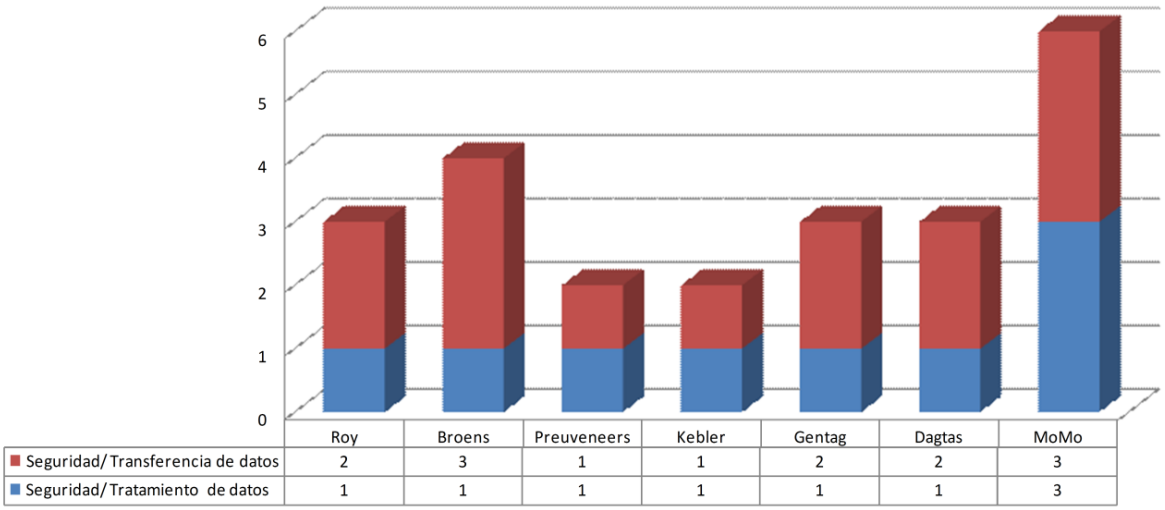
	Valoración	Descripción	Peso
Transferencia de datos	Alta (AT)	La investigación propone aspectos de seguridad en la transferencia de datos.	3
	Media (MD)	La investigación propone aspectos medianos de seguridad en la transferencia de datos.	2
	Baja(BJ)	La investigación no propone aspectos de seguridad en la transferencia de datos.	1

***Tabla 3-16**. Descripción de las valoraciones del criterio seguridad y sub-criterios para las propuestas de investigadores evaluadas.*

Cuantificando el grado del criterio *seguridad* y sus sub-criterios, se muestra la figura 3-7, la evaluación de cada propuesta de investigadores analizados en esta tesis. Dentro de este criterio existen dos subcriterios importantes al momento de desarrollar aplicaciones móviles. Como nuestro dominio de análisis, es entornos médicos, además de mantener la seguridad al momento de transferir datos entre cada una de las entidades que interactúan con la arquitectura desarrollada, también se contempla el aspecto de la seguridad en el tratamiento de los datos de los participantes.

Ninguna de las propuestas analizadas contempla el aspecto de seguridad en el tratamiento de datos, o no lo mencionan en la explicación de su propuesta. *MoMo* propone altos niveles de seguridad tanto en la transferencia de datos como en su tratamiento. De igual manera, solo la propuesta de *Broens* especifica aspectos de seguridad en la transferencia de datos. Es decir utilizan algún tipo de tecnología para mantener la integridad de los datos al momento de ser trasmitidos.

Es importante definir explícitamente los aspectos de seguridad en ambos casos, ya que facilitan la relación entre los participantes y la tecnología a utilizar, respetando así, las normas y leyes que rigen a cada uno.



	Roy	Broens	Preuveneers	Kebler	Gentag	Dagtas	MoMo
Seguridad/ Transferencia de datos	2	3	1	1	2	2	3
Seguridad/ Tratamiento de datos	1	1	1	1	1	1	3

Figura 3-7. *Nivel de seguridad de las propuestas de investigadores evaluadas.*

5. Costos

En la tabla 3-16, se explican en detalle los sub-criterios evaluados en el criterio costos y sus respectivos sub-criterios.

	Valoración	Descripción	Peso
De Implementación	Alto (AT)	La investigación tiene un alto costo de desarrollo y de implementación.	1
	Medio (MD)	La investigación tiene un mediano costo de desarrollo y de implementación.	2
	Bajo (BJ)	La investigación tiene un bajo costo de desarrollo y de implementación.	3

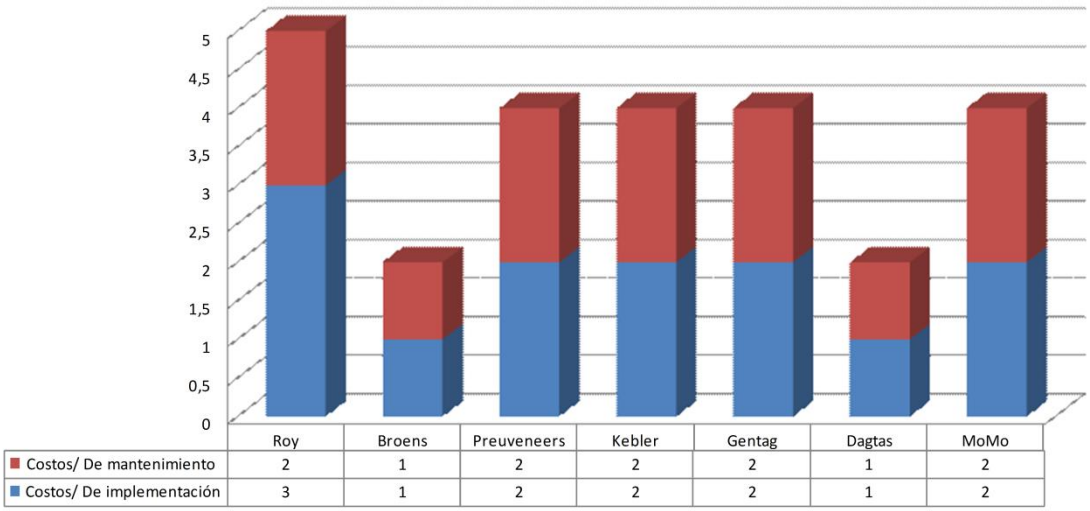
	Valoración	Descripción	Peso
De Mantenimiento	Alto (AT)	La investigación tiene un alto costo de mantenimiento posterior al desarrollo de la aplicación.	1
	Medio (MD)	La investigación tiene un mediano costo de mantenimiento posterior al desarrollo de la aplicación.	2
	Bajo(BJ)	La investigación tiene un bajo costo de mantenimiento posterior al desarrollo de la aplicación.	3

Tabla 3-17*. Descripción de las valoraciones del criterio costos y sub-criterios para las propuestas de investigadores evaluadas.*

Cuantificando el grado del criterio *costos* y sus sub-criterios, se muestra la figura 3-8, la evaluación de cada propuesta de investigadores analizadas en esta tesis.

Los costos para el desarrollo de aplicaciones es otro criterio importante para analizar. No sólo hablamos de costos económicos, que son altos a la hora de integrar tecnologías, sino también el tiempo y dificultad que conlleva actualizar las aplicaciones previamente

desarrolladas. La propuesta que más *bajo* costo de implementación tiene (uso de tecnologías para su desarrollo), es la de *Roy*, mientras que la de *Broens* y *Dagtas* son las que más *alto* costo de implementación poseen. En cuanto al costo de mantenimiento (facilidad para modificar la aplicación desarrollada), la mayoría de las propuestas tiene un grado medio, siendo *Broens* y *Dagtas* las que poseen un grado alto de mantenimiento. En *MoMo* se categoriza los niveles de costos en un nivel medio, esto es debido a la utilización de tecnologías actuales cuya relación costo/beneficio es muy alta.



	Roy	Broens	Preuveneers	Kebler	Gentag	Dagtas	MoMo
Costos/ De mantenimiento	2	1	2	2	2	1	2
Costos/ De implementación	3	1	2	2	2	1	2

***Figura 3-8**. Nivel de costos de las propuestas de investigadores evaluadas.*

La percepción de costos puede ser analizada desde diversos puntos de vistas, pudiendo ser más altamente valorado por unos investigadores, mientras que para otros tiene un nivel de importancia bajo. En nuestra percepción una valoración aceptable para este sub-criterio podría ser *media*.

3.5 APLICACIONES PARA LA MONITORIZACIÓN MÓVIL DE PACIENTES.

Estas investigaciones han traído consigo la creación de diferentes plataformas tecnológicas que han ofrecido una solución puntual a problemas de cuidados asistenciales, dentro de los cuales podemos encontrar:

Health Buddy System. (BOSCH, 2011) es un sistema que provee control sanitario de pacientes reduciendo las posibilidades de hospitalización. En la figura 3-9, se muestra que este sistema conecta a los pacientes en sus casas a los proveedores de su cuidado. Lo que lo diferencia de los demás es su capacidad no sólo para comunicar el histórico de información de los pacientes con enfermedades crónicas, sino también para facilitar la educación del paciente y fomentar el cumplimiento de la medicación y el estilo de vida.

Cada día, los pacientes responden a una serie de preguntas sobre su salud y el bienestar usando el simple aparato de *Health Buddy*. Los datos se envían a través de

una línea telefónica o una conexión *Ethernet* a un centro de datos seguro, los datos están a disposición para revisar en la web de *Health Buddy*. La aplicación está diseñada para estratificar el riesgo de forma rápida y presentar los resultados del paciente, permitiendo a los proveedores intervenir proactivamente antes de que la condición del paciente se agudice.

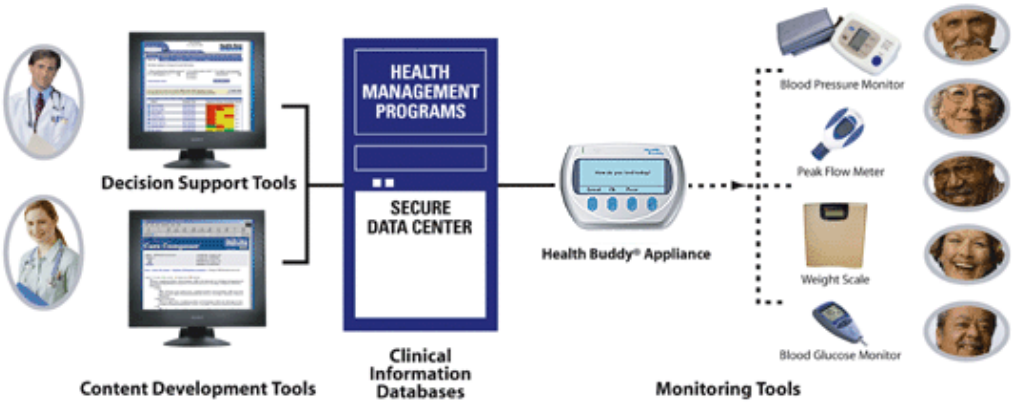


Figura 3-9. *Funcionamiento de Health Buddy System*

Nuestra Propuesta: busca la mínima interacción con el dispositivo móvil, es decir que todo el procesamiento se haga desde el proceso inicial de captura de las señales vitales. De allí en adelante, se trata de minimizar esa interacción, para facilitarle su uso al paciente ofreciéndole respuestas y mensajes automáticamente.

AirStrip Patient Monitoring. Es una plataforma de desarrollo de *software* con una visión de envío seguro de información crítica del paciente directamente desde los sistemas de vigilancia del hospital, dispositivos de cabecera, y los registros clínicos de salud para dispositivos móviles (AirStrip, 2011). *AppPoint* también fue diseñado para resolver los principales retos en el desarrollo de *software* para móviles, como el desarrollo de aplicaciones nativas que proporcionan los requisitos de una rica experiencia de usuario, mientras que al mismo tiempo ser capaz de escalar y adaptarse a un mundo siempre cambiante de los sistemas operativos y dispositivos móviles. Esta aplicación ha sido desarrollada para dispositivos como *iPhone*, *iPod* y *iPad*. Como se muestra en a figura 3-10, se almacena información relacionada a valores obtenidos de electrocardiogramas, signos vitales, medicación, resultados de laboratorios, listas de alergias, entre otras.

Nuestra Propuesta: evita depender de una plataforma en particular, y contempla al igual que esta aplicación, información adicional del paciente como resultados de analíticas y exámenes en general. Inicialmente desarrollamos una arquitectura software para una plataforma en particular, de la manera más generalizada, permitiendo adaptarla a otra realizando los mínimos cambios.

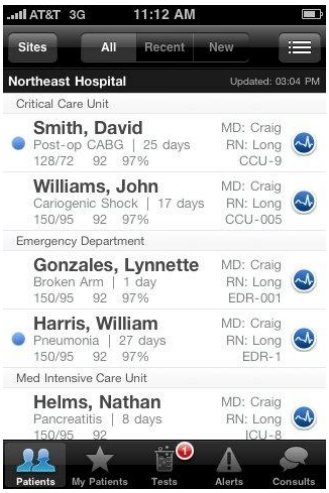


Figura 3-10. *Funcionamiento de AirStrip Patient Monitoring*

WellDoc. Es una aplicación diseñada para ser un servicio de monitoreo para diabéticos (WellDoc, 2011), integrado con *Ford Sync*, diseñada para *iOS*, que permite supervisar la condición actual de un paciente mediante el registro manual de alimentación y glucosa. Gracias a su integración con *Sync,* la misma sincronizará con este servicio vía *Bluetooth*, el cual detectará si no hemos introducido ningún registro recientemente, y mediante un sistema de preguntas (sí o no), se asegurará de ver si nuestros niveles sanguíneos están correctos. De no ser así, sugerirá la próxima acción recomendada a tomar, o en casos extremos, enviará un *SMS* al contacto que tengamos previamente seleccionado como de emergencias, con la opción de enviar otro mensaje al llegar a casa, confirmando que estamos seguros (ver figura 3-11).

Nuestra Propuesta: esta implementada independientemente de la enfermedad a controlar, es decir, se ha desarrollado para interactuar con gran número de enfermedades que pueda ser monitorizada a través de señales vitales. Implementaremos casos de estudios particulares para conocer su funcionalidad.

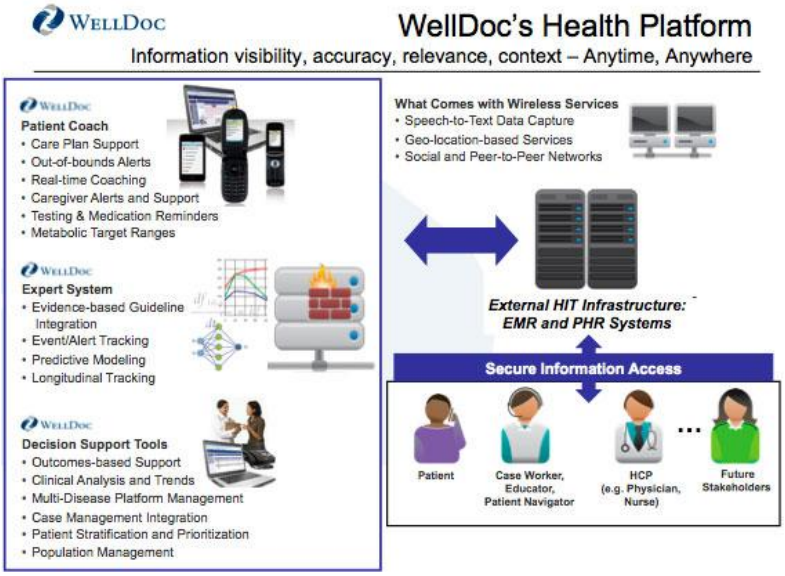


Figura 3-11. *Visión de alto nivel de la arquitectura WellDoc*

METABO (Georga, Protopappas et al., 2009) es un sistema de monitoreo y administración de diabetes que tiene como objetivo la grabación y la interpretación del contexto del paciente, así como, prestar apoyo a las decisiones para el paciente y el médico. El sistema METABO se compone de (a) un dispositivo móvil del paciente (*PMD*), (b) los diferentes tipos de biosensores discretos, (c) un subsistema central (*CS*) ubicado en forma remota en el hospital y (d) el panel de control (*CP*) desde el que los médicos pueden dar seguimiento de sus pacientes y además tener acceso al *CS*. METABO proporciona un sistema de vigilancia que facilita la grabación eficaz y sistemática de la dieta, actividad física, medicación e información médica (mediciones de glucosa continua y discontinua). Basado en toda la información contextual registrada, esquemas de minería de datos que se ejecutan en el *PMD* son responsables mediante el modelo de metabolismo de los pacientes de predecir eventos de hipo/hyper-glucémico y proporcionar al paciente alertas cortas y a largo plazo. Además, se analizan todos los datos pasados y recientemente grabados para extraer los patrones de comportamiento, descubrir nuevos conocimientos y proporcionar explicaciones al médico a través del *CP*. Herramientas avanzadas en el *CP* permiten al médico prescribir los planes de tratamiento personalizado y cuantificar con frecuencia la adhesión del paciente al tratamiento.

Nuestra Propuesta: sigue contemplando el aspecto de multimonitorización, es decir, que se desarrolla una arquitectura *software* que permita la monitorización y el control de pacientes independiente del tipo de enfermedad que padece, ajustándose cada uno de los elementos de la arquitectura *software* desarrollada a los requisitos funcionales planteados inicialmente.

Ambulation. En (Ryder, Longstaff et al., 2009) se presenta una herramienta importante para evaluar la salud de los pacientes que sufren de enfermedades crónicas que afectan a la movilidad como esclerosis múltiple (MS), la de Parkinson y la distrofia muscular a través de la evaluación cuando caminan. **Ambulation** es un sistema de monitorización de movilidad que utiliza teléfonos móviles Android y Nokia N95 para detectar automáticamente el modo de movilidad del usuario. La interacción que el usuario necesita con el teléfono es encenderlo y mantenerlo con él durante todo el día, con la intención de que podría utilizarse como su teléfono móvil cotidiano para voz, datos y otras aplicaciones, mientras **Ambulation** se ejecuta en segundo plano. El teléfono carga la información recopilada de la movilidad y la ubicación en un servidor. Luego se visualiza los datos a través de un servicio basado en Web seguro e intuitivo, quedando disponible para el usuario, cualquier familiar, amigos o cuidadores. Esta información permite identificar tendencias en el progreso de la movilidad y la medida según el tiempo y respuesta a diversos tratamientos.

Nuestra Propuesta: mantiene el comportamiento en *segundo plano* sugerido por Ryder, es decir que la aplicación de monitorización del paciente se ejecute sin interferir con el uso cotidiano que se le dé al dispositivo móvil. Estos hace transparente la funcionalidad para el usuario y minimizando la interacción.

SenSAVE. (Lorenz, Mielke et al., 2007) con su proyecto senSave desarrolló un sistema para la monitorización móvil de los parámetros de señales vitales. Definió así la interface de usuario y la interacción adaptada específicamente para pacientes. Se toman las medidas de presión sanguínea y el pulso de personas relacionadas con problemas cardiovasculares en tiempo real. En resumen este sistema, estudió la forma en que se presenta la información, las ventanas de diálogos y la función de alarmas, demostrando la usabilidad de los dispositivos móviles. Se buscaba con esto facilitar la interacción de los ancianos con el teléfono móvil (ver figura 3-12).

Figura 3-12. *Esquema funcional de SenSAVE*

Nuestra Propuesta: propone la comunicación entre múltiples dispositivos biométricos y móviles, basada en un perfil de paciente específico y en la enfermedad que padece. A partir de estas señales vitales se genera la información de control para los pacientes.

U-Health. En (Wan-Young, Seung-Chul et al., 2008) ha diseñado y desarrollado un sistema móvil de cuidados de la salud con capacidad de medición de varios signos fisiológicos en tiempo real. Este sistema realiza un análisis de datos de las constantes vitales a través del teléfono móvil y transmite esos datos a través de una red de sensores inalámbricos. Electrocardiograma (ECG) y las señales de la presión arterial de los pacientes son continuamente controlados, procesados y analizados localmente en el teléfono móvil para producir información médica útil para el diagnóstico y con fines de seguimiento. El teléfono móvil realiza algunos análisis de datos simples primero y, a continuación, transmite inmediatamente estas señales a un servidor en el hospital para el diagnóstico de los médicos. La aplicación de tecnología inalámbrica en el sistema de diagnóstico permite que el paciente pueda ser supervisado en cualquier lugar, en cualquier momento y no podría verse obstaculizado por las restricciones físicas impuestas por los cables. Esta función puede resultar útil en el cumplimiento de la visión de cuidados de salud pervasiva (*Pervasive Healthcare*).

Nuestra Propuesta: facilita la monitorización continua del paciente, es decir, ofrece servicios de control cada vez que se obtienen los valores de las mediciones de las señales vitales. A partir de allí, puede generar recomendaciones asociadas a esa medición, ofreciendo un entorno de seguimiento continuo.

Healthwear. En (Paradiso, Alonso et al., 2008) se aborda una investigación que permite la supervisión de las condiciones de salud por medio de electrocardiogramas, frecuencia cardiaca, la saturación de oxígeno, impedancia *pneumography* y patrones de actividad. El servicio de *Healthwear* se basa en el sistema de prototipo *Wealthy*. Se hizo un nuevo diseño para aumentar la comodidad del paciente durante sus actividades cotidianas. La tela está conectada a una unidad electrónica portátil del paciente (*PPU*) que adquiere y elabora las señales de los sensores. La *PPU* transmite la señal a un sitio central de procesamiento mediante el uso de la tecnología inalámbrica *GPRS*. Este servicio se aplica a tres contextos clínicos distintos: rehabilitación de pacientes cardíacos, después de un evento agudo; el programa de descarga temprana en pacientes de respiración crónica y la promoción de la actividad física en pacientes ambulatorios cardio-respiratorios estables.

Nuestra Propuesta: evita el uso de equipos en el cuerpo, tan sólo se hace necesario contar con un dispositivo biométrico para la obtención de las señales vitales. El uso de equipos en el cuerpo limita la comodidad del paciente y cambia su día a día. Queremos minimizar la alteración en el diario vivir del paciente, minimizando la intrusión.

MobiHealth. (Mei, Widya et al., 2006) propuso el desarrollo de un *framework* que representa las señales vitales de pacientes. Este *framework* facilita el almacenamiento de las diferentes notaciones existentes para representar señales vitales (FDA (Brown, Kohls et al., 2002), CEN (CEN, 1999), HL7 (HL7, 2005), DICOM (DICOM, 2004)). Para ello proponen un esquema en XML para definir el *framework* de representación de señales vitales, especificando gran cantidad de estas representaciones. Nuestra propuesta no está basada en una representación de señales vitales, pero sí en el control e interpretación de éstas. En este trabajo vamos más allá de representar, por lo tanto tratamos de generar aplicaciones para el uso de paciente.

En la figura 3-13, se puede observar la representación de dicho *framework* a través del sistema MobiHealth (MobiHealth, 2003), desde dos puntos de vistas, el primero (externo) en donde el usuario se encuentre fuera de BAN (Body Area Network), el *Mobi* captura todos los datos del sensor enviándolo al Móvil (MBU Mobile Base Unit) que actúa como una puerta de enlace de la BAN hacia Internet (como se puede notar en el enlace 2). Por otra parte, el segundo (interno), la BAN del paciente que está dentro del hogar puede alternativamente usar el MBU como puerta de enlace hacia la "Home Gateway" (HG) a través del enlace 7, este envía la información de signos vitales al Centro de llamada de Healthcare a través de la línea 8. Las señales no críticas se almacenan temporalmente hasta que se localice el origen.

Nuestra Propuesta: en este sistema se propone solamente una representación de las señales vitales recibidas por los dispositivos móviles creando documentos de datos de las representaciones de señales vitales que resultan de la movilidad de los usuarios (pacientes) en ambientes heterogéneos. Nuestra propuesta no está basada en una representación de señales vitales, pero sí en el control e interpretación de éstas.

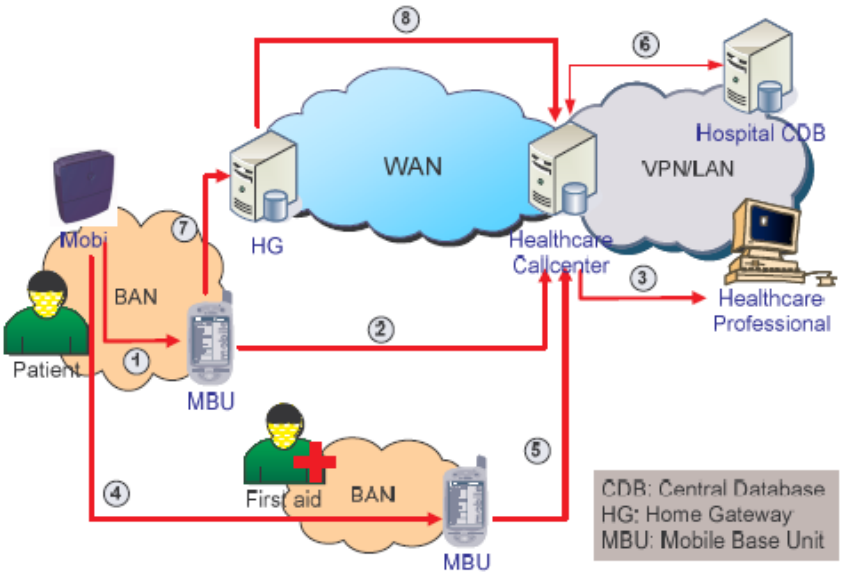


Figura 3-13. *Transmisión de señales vitales, en el sistema MobiHealth*

eDiab. Fernández-Luque (Fernández-Luque, Sevillano et al., 2006) desarrolló un sistema para monitorización, asistencia y educación de personas con diabetes, llamado *eDiab*. Un nodo central (*PDA* o teléfono móvil) es usado por el paciente para la transmisión de información médica, consejos de salud, alarmas y recordatorios. El sensor de glucosa es conectado a un nodo central a través de enlaces inalámbricos (*ZigBee/Bluetooth*) y la comunicación entre el nodo central y el servidor es establecida con conexión *GPRS/GSM*. Finalmente un subsistema para educación de salud envía información médica y consejos como pueden ser recordatorios de tratamientos. Como se muestra en la figura 3-14 el proyecto *eDiab* está siendo diseñado con vistas multidisciplinarias: tecnología asistida, educación en telemedicina y diabetes. Está dividido en dos subsistemas, uno para la monitorización y control de diabetes y otro para educación en salud.

Nuestra Propuesta: contempla aspectos de control de pacientes, con la diferencia, de que nuestra propuesta está basada en la definición del perfil del paciente. Este perfil de paciente asociado a las mediciones obtenidas de una enfermedad, rige el comportamiento de la aplicación final.

LATIS Pervasive *Framework* (LAPERF). Tadj (Tadj and Ngantchaha, 2006) con LATIS Pervasive *Framework* (LAPERF) provee un *framework* de base y herramientas automáticas para el desarrollo e implementación de aplicaciones de computación pervasiva. Su principal aplicación se ha demostrado en el uso de aplicaciones de cuidados médicos. Se intenta con esto obtener mejor integridad en los sistemas pervasivos. Tadj propone el manejo de un perfil general del paciente, en nuestro caso, proponemos una clasificación del perfil, en perfil individual (con información propia a cada paciente) y perfil común del paciente (con información genérica para todos los pacientes), que permite un mayor control. Nuestra propuesta se basa en una descripción de actividades asociadas a la localización del paciente; la comunicación se establece entre el paciente y el

especialista médico. En figura 3-15 se puede ver la estructura del *framework LATIS Pervasive*, a nivel inferior se encuentra el *Access Device Manager* en donde se encuentran todos los dispositivos para acceder al *framework* (*Smartphone*, *PDA*, computadoras de escritorio, etc.). El *Security and Privacy Manager*, se encarga del acceso no autorizado definiendo políticas de seguridad al sistema.

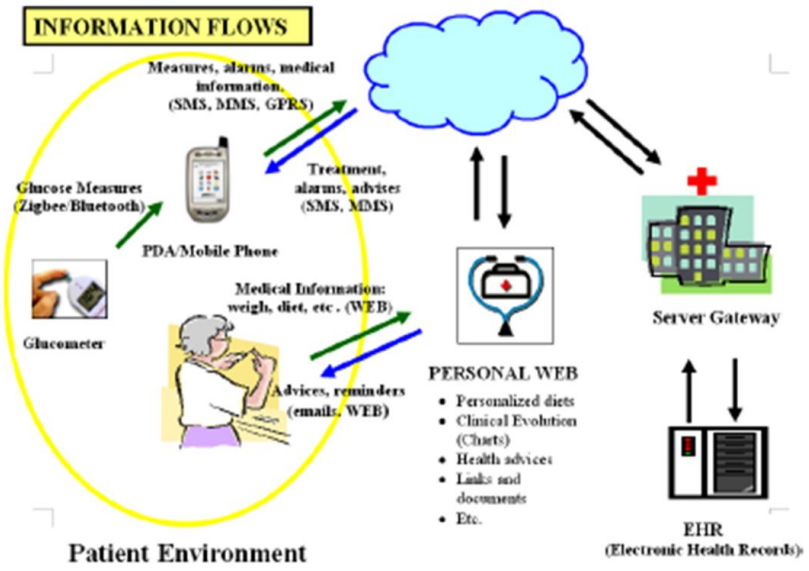


Figura 3-14. *Flujo de información en eDiab*

Nuestra Propuesta: al igual que esta solución, proponemos el desarrollo de un *framework*, con la diferencia de que estará orientado a la monitorización móvil de pacientes. Tadj propone el manejo de un perfil general del paciente, en nuestro caso, proponemos una clasificación del perfil, en perfil individual y perfil común del paciente, que permite mayor seguimiento de la enfermedad.

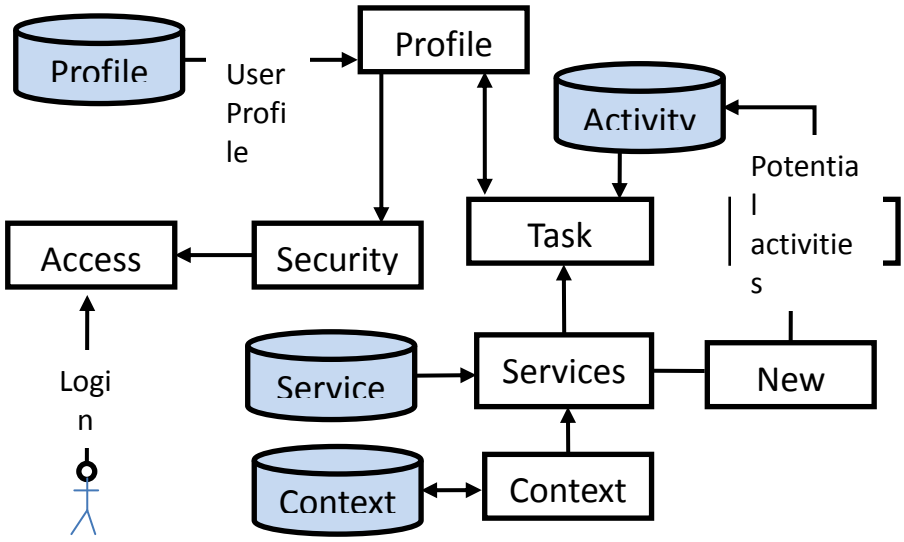
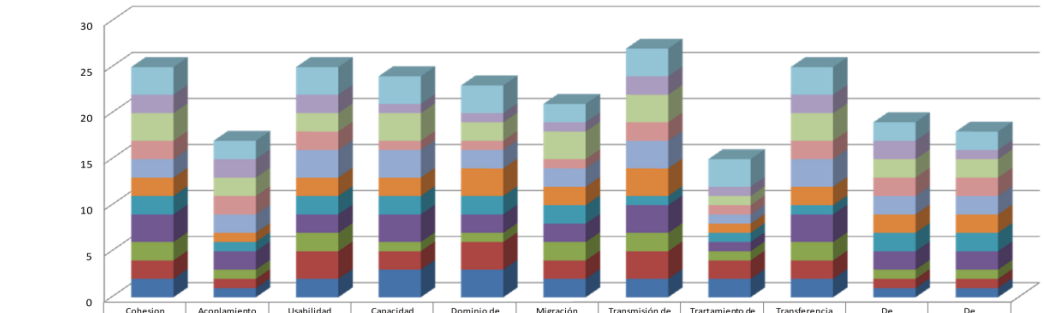


Figura 3-15. *Arquitectura del Sistema Pervasivo LATIS*

MAHI. Mamykina (Mamykina, Mynatt et al., 2006) desarrolló MAHI (*Mobile Access to Health Information*), una aplicación que monitoriza a pacientes diagnosticados con diabetes en donde adquieren habilidades de pensamiento reflexivo por la interacción social con educadores de diabetes. El manejo del análisis reflexivo de experiencias pasadas es una de las más esenciales habilidades en el manejo de diabetes. MAHI es una aplicación móvil distribuida que incluye un glucómetro convencional (*LifeScan's OneTouch Ultra*), un móvil con soporte *Java* (*Nokia N80*) y un adaptador *Bluetooth* que comunica al móvil y el glucómetro. A través de ello graban sus niveles de glucosa en la sangre y los cambios relacionados a la diabetes, como preguntas, problemas y actividades de interés, usando la captura de imágenes y sonidos con el móvil.

Nuestra Propuesta: no se pretende crear de un sitio web, sino que la información va a estar contenida en el dispositivo móvil del paciente, transmitiendo los datos desde y hacia un servidor remoto. Esta información es generada según las actividades desarrolladas por el paciente y las medidas obtenidas de los dispositivos biométricos. El dispositivo móvil puede ser utilizado en dos roles: el rol paciente, específicamente para el control y uso de cada paciente y el rol médico, en donde los médicos tendrán acceso a la información de todos los pacientes asignados a él.

En la tabla 3-18, se muestra la comparativa de los criterios definidos para la evaluación de aplicaciones analizadas previamente. En la figura 3-16 puede verse el valor o peso asignado luego del análisis de cada aplicación. En el apartado 3.5.1 se analizará por separado cada criterio, así como el peso asignado a las aplicaciones previamente estudiadas. Como mencionamos previamente, el criterio más altamente contemplado en el desarrollo de propuestas y aplicaciones para la monitorización de pacientes es el de usabilidad, seguido del tipo de transmisión de datos. Ambos criterios son muy importantes ya que hablamos de monitorización móvil desarrolladas para dominios médicos, en donde los usuarios son personas en su mayoría de edades avanzadas.



	Cohesion	Acoplamiento	Usabilidad	Capacidad Evolutiva	Dominio de aplicación	Migración Tecnologica	Transmisión de datos	Trartamiento de datos	Transferencia de datos	De implementación	De mantenimiento
	Diseño/				Adaptabilidad/		Comunicación/	Seguridad/		Costos/	
MoMo	3	2	3	3	3	2	3	3	3	2	2
MAHI	2	2	2	1	1	1	2	1	2	2	1
LAPERF	3	2	2	3	2	3	3	1	3	2	2
eDiab	2	2	2	1	1	1	2	1	2	2	2
MobiHealth	2	2	3	3	2	2	3	1	3	2	2
Healthwear	2	1	2	2	3	2	3	1	2	2	2
SenSave	2	1	2	2	2	2	1	1	1	2	2
Ambulation	3	2	2	3	2	2	3	1	3	2	2
METABO	2	1	2	1	1	2	2	1	2	1	1
AirStrip PM	2	1	3	2	3	2	3	2	2	1	1
Health Buddy System	2	1	2	3	3	2	2	2	2	1	1

Figura 3-16. *Comparativa de los criterios definidos para la evaluación de aplicaciones desarrolladas, según cada su-criterio.*

Propuestas	Diseño						Adaptabilidad						Comunicación		Seguridad				Costos			
	Cohesión		*Acoplamiento*		*Usabilidad*		*Capacidad evolutiva*		*Dominio de aplicación*		*Migración Tecnológica*		*Transmisión de datos*		*Tratamiento de datos*		*Transferencia de datos*		*De implementación*		*De mantenimiento*	
Health Buddy System	MD	2	AT	1	MD	2	AT	3	AT	3	MD	2	MD	2	MD	2	MD	2	AT	1	AT	1
AirStrip Patient Monitoring	MD	2	AT	1	AT	3	MD	2	AT	3	MD	2	AT	3	MD	2	MD	2	AT	1	AT	1
METABO	MD	2	AT	1	MD	2	BJ	1	BJ	1	MD	2	MD	2	BJ	1	MD	2	AT	1	AT	1
Ambulation	AT	3	MD	2	MD	2	AT	3	MD	2	MD	2	AT	3	BJ	1	AT	3	MD	2	MD	2
SenSave	MD	2	AT	1	MD	2	MD	2	MD	2	MD	2	BJ	1	BJ	1	BJ	1	MD	2	MD	2
U-Health Healthwear	MD	2	AT	1	MD	2	MD	2	AT	3	MD	2	AT	3	BJ	1	MD	2	MD	2	MD	2
MobiHealth	MD	2	MD	2	AT	3	AT	3	MD	2	MD	2	AT	3	BJ	1	MD	3	MD	2	MD	2
eDiab	MD	2	MD	2	MD	2	BJ	1	BJ	1	BJ	1	MD	2	BJ	1	MD	2	MD	2	MD	2
LAPERF	AT	3	MD	2	MD	2	AT	3	MD	2	AT	3	AT	3	BJ	1	AT	3	MD	2	MD	2
MAHI	MD	2	MD	2	MD	2	BJ	1	BJ	1	BJ	1	MD	2	BJ	1	MD	2	MD	2	AT	1
MoMo	AT	3	MD	2	AT	3	AT	3	AT	3	MD	2	AT	3	AT	3	AT	3	MD	2	MD	2

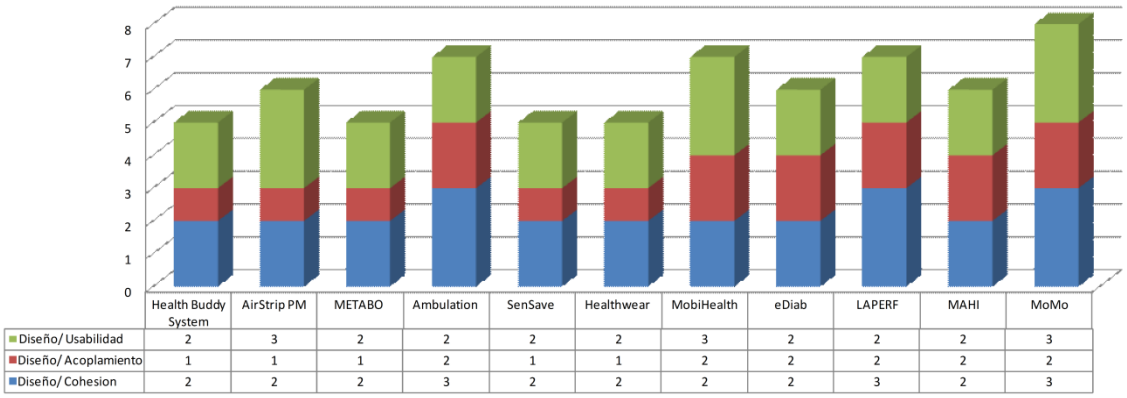
Tabla 3-18. Comparativa de los criterios definidos para la evaluación de aplicaciones desarrolladas.

3.5.1 Análisis de criterios de evaluación de las aplicaciones evaluadas.

Para el análisis de las aplicaciones estudiadas, nos basaremos en la descripción y peso de los criterios y sub-criterios definidos en el apartado 3.4.1. Estos criterios nos permiten analizar este bloque de trabajos relacionados, que han sido clasificados como aplicaciones desarrolladas por empresas o desarrolladores particulares.

1. Diseño

Cuantificando el grado del criterio *diseño* y sus sub-criterios, se muestra la figura 3-17, la evaluación de cada aplicación analizada en el estado del arte de esta tesis.



	Health Buddy System	AirStrip PM	METABO	Ambulation	SenSave	Healthwear	MobiHealth	eDiab	LAPERF	MAHI	MoMo
Diseño/ Usabilidad	2	3	2	2	2	2	3	2	2	2	3
Diseño/ Acoplamiento	1	1	1	2	1	1	2	2	2	2	2
Diseño/ Cohesion	2	2	2	3	2	2	2	2	3	2	3

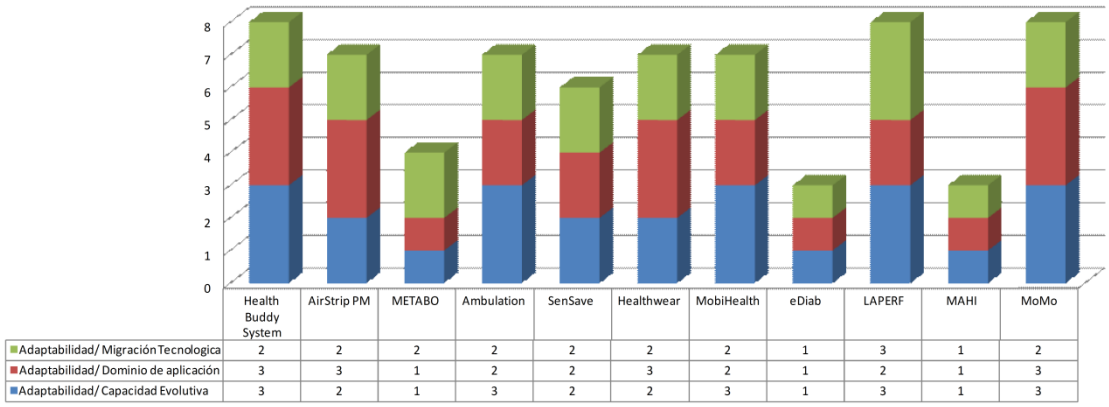
***Figura 3-17**. Nivel de diseño de las aplicaciones analizadas.*

Este criterio, ha sido analizado según las funcionalidades especificadas en la documentación presentada en cada aplicación desarrollada. Las aplicaciones evaluadas poseen un grado *medio* según el criterio de diseño. El sub-criterio de acoplamiento ha sido valorado como el criterio más bajo en el diseño, dándole más importancia a la usabilidad. Con respecto a estas propuestas, *MoMo* ofrece resultados de diseño acordes a los criterios de cohesión, acoplamiento y usabilidad. Se trata de buscar una definición generalizada de la arquitectura *software* minimizando el acoplamiento pero maximizando la cohesión. Con respecto a la usabilidad, nuestra arquitectura *software* es fácilmente usable, ya que se ha tratado de minimizar las interacciones innecesarias por parte del usuario.

2. Adaptabilidad

Cuantificando el grado del criterio *adaptabilidad* y sus sub-criterios, se muestra la figura 3-18 la evaluación de cada aplicación analizada en el estado del arte de esta tesis.

La adaptabilidad de las aplicaciones analizadas, van de la mano con el criterio de diseño, puede notarse que se valora con un grado medio a este criterio. Gran parte de las aplicaciones pueden ser adaptadas para cubrir otros aspectos tecnológicos existentes y otros dominios de aplicación. *MAHI* ha sido la aplicación con más bajo grado de adaptabilidad encontrada, mientras que *Health Buddy System* la que mayor adaptabilidad posee. *MoMo* ofrece un alto nivel de adaptabilidad tanto en tecnologías *hardware* como en tecnologías *software.*

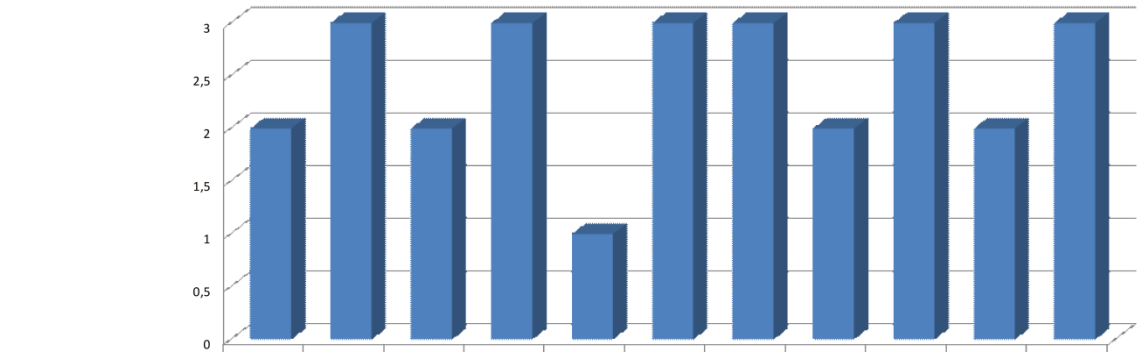


	Health Buddy System	AirStrip PM	METABO	Ambulation	SenSave	Healthwear	MobiHealth	eDiab	LAPERF	MAHI	MoMo
Adaptabilidad/ Migración Tecnologica	2	2	2	2	2	2	2	1	3	1	2
Adaptabilidad/ Dominio de aplicación	3	3	1	2	2	3	2	1	2	1	3
Adaptabilidad/ Capacidad Evolutiva	3	2	1	3	2	2	3	1	3	1	3

***Figura 3-18**. Nivel de adaptabilidad de las aplicaciones analizadas.*

3. Comunicación

Cuantificando el grado del criterio *comunicación* y sus sub-criterios, se muestra la figura 3-19, la evaluación de cada aplicación analizada en el estado del arte de esta tesis.



	Health Buddy System	AirStrip PM	METABO	Ambulation	SenSave	Healthwear	MobiHealth	eDiab	LAPERF	MAHI	MoMo
Comunicación/ Transmisión de datos	2	3	2	3	1	3	3	2	3	2	3

***Figura 3-19**. Nivel de comunicación de las aplicaciones analizadas.*

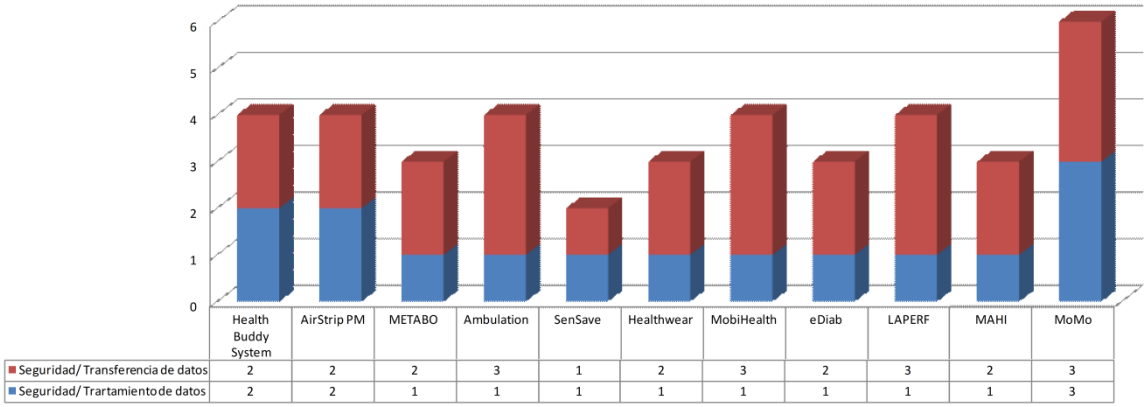
En promedio, las aplicaciones evaluadas utilizan alguna tecnología de comunicación. Estas pueden ser las que ofrecen las redes de telefonía móvil, *Ethernet*, *WiFi*, entre otras. *SenSave* ha sido la aplicación que más bajo grado de comunicación (transmisión de datos) presenta, es decir, no utiliza tecnología de comunicación existente al momento de realizar el análisis. *MoMo* sigue ofreciendo, al igual que la mayoría de las investigaciones, un alto nivel de comunicación y transmisión de datos.

4. Seguridad

Cuantificando el grado del criterio *seguridad* y sus sub-criterios, se muestra la figura 3-20, la evaluación de cada aplicación analizada en el estado del arte de esta tesis.

Otros de los aspectos analizados en las aplicaciones encontradas, es el de seguridad. Sólo dos del total de las aplicaciones, toman en cuenta el tratamiento de datos de los pacientes, es decir, que especifican algún protocolo o lineamiento para mantener la privacidad de estos datos. Por otra parte, la seguridad en la transferencia de datos ha sido un aspecto

medianamente contemplado, ya que algunas de las aplicaciones, dejan este aspecto a la tecnología de comunicación, en el caso que las utilicen y otras no consideran necesario profundizar en este sub-criterio. En este aspecto *MoMo* ofrece mayores servicios de seguridad en comparación con las otras propuestas.

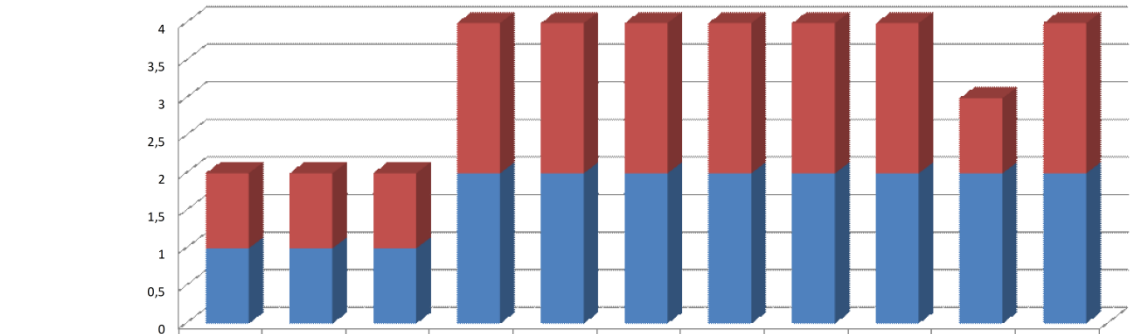


	Health Buddy System	AirStrip PM	METABO	Ambulation	SenSave	Healthwear	MobiHealth	eDiab	LAPERF	MAHI	MoMo
Seguridad/ Transferencia de datos	2	2	2	3	1	2	3	2	3	2	3
Seguridad/ Trartamiento de datos	2	2	1	1	1	1	1	1	1	1	3

***Figura 3-20**. Nivel de seguridad de las aplicaciones analizadas.*

5. Costos

Cuantificando el grado del criterio *costos* y sus sub-criterios, se muestra la figura 3-21, la evaluación de cada aplicación analizada en el estado del arte de esta tesis.



	Health Buddy System	AirStrip PM	METABO	Ambulation	SenSave	Healthwear	MobiHealth	eDiab	LAPERF	MAHI	MoMo
Costos/ De mantenimiento	1	1	1	2	2	2	2	2	2	1	2
Costos/ De implementación	1	1	1	2	2	2	2	2	2	2	2

***Figura 3-21**. Nivel de costos de las aplicaciones analizadas.*

La implementación de las aplicaciones en su mayoría presenta un costo medio tanto de mantenimiento como de implementación. Esto es debido a que son muy dependientes de las tecnologías para las que fueron desarrolladas. Es complicado desligarse de este criterio ya que la mayoría de las aplicaciones son desarrolladas para tecnologías y plataformas específicas. Un grado medio de ponderación minimiza más los costos de cambios que se quieran realizar en el futuro.

3.6 OTRAS APORTACIONES TECNOLÓGICAS. ALGUNOS ESTUDIOS SECUNDARIOS

De igual manera se han hecho algunas otras aproximaciones mediantes sistemas, aplicaciones, algoritmos para el control y seguimiento de pacientes. Algunos de ellos son:

Sistema Clínico de alertas en tiempo real para el cuidado respiratorio de pacientes. Hay muchas condiciones críticas que puede ocurrir que deben tratarse adecuadamente. Estar preparado y saber cómo reaccionar antes de tiempo podría prevenir una situación se convierta en una situación de emergencia. Hoy en día, muchos hospitales utilizan sistemas de información clínica (CIS) para mejorar la gestión de información y atención de la salud del paciente. CIS recupera y administra la información del paciente, tales como administración de pacientes, los datos de laboratorio y medicamentos. En (Mohanalakshmi and Jacob, 2007) se ha diseñado e implementado un sistema clínico de alertas en tiempo real para el cuidado respiratorio de pacientes, que ejecuta algoritmos de alertas dependiendo de la criticidad de los problemas respiratorios del paciente. El paciente es supervisado en primer lugar mediante un sistema de tele observación, la señal se procesa para mostrar el patrón de respiración. Cuando se detecta una condición de alerta sobre un paciente específico, el sistema da avisos a los médicos a través de un mensaje de texto en sus teléfonos móviles. Con esta funcionalidad, hasta la fecha se proporciona información para uso del doctor que juzga la condición del paciente y toma decisiones del tratamiento instantáneamente.

Framework para la supervisión de la condición del paciente. En (Koutkias, Chouvarda et al., 2010) se presenta un *framework* para la supervisión de la condición del paciente y la seguridad en relación con el tratamiento de la medicación administrada. Para este propósito, considerando la posibilidad de una red de área de cuerpo (BAN) con sensores avanzados y una unidad móvil de base como el concentrador central de comunicaciones de un lado y el entorno clínico desde el otro lado, se desarrolló una arquitectura, ofreciendo la definición de patrones de vigilancia para la detección de eventos adversos y la evaluación de la respuesta de la medicación, apoyado por mecanismos que permitan la comunicación bidireccional entre la BAN y la clínica. Se dieron especial énfasis sobre comunicación y aspectos del flujo de información que han sido abordados las estructuras de información, así como el paradigma de la arquitectura orientada al servicio. El marco propuesto se ilustra a través de un escenario de aplicación relativa a la gestión de hipertensión.

Algoritmo de detección y clasificación de movimiento. Hoy nuevas aplicaciones médicas evolucionan desde grandes dispositivos estacionarios para sistemas móviles pequeños e inteligentes que permitirán, por ejemplo, más eficiente atención de la salud post operativa. Estos sistemas móviles se benefician de la miniaturización y del ahorro de energía en *hardware* lo cual permitirá un monitoreo continuo de los pacientes de y apoyo a terapia así como detectar situaciones de emergencia. Adicionalmente a los datos vitales, la carga fisiológica o el contexto de los pacientes son importantes para analizar y comprender los datos registrados de pacientes móviles. Enfoques actuales para la clasificación de movimiento pretenden detectar patrones de movimiento muy

específicos y son dependientes de colocaciones de algún sensor preciso y por lo tanto no son adecuados para uso diario. Por lo tanto, en (Hellbruck, Hua et al., 2009) se ha desarrollado un algoritmo de detección y clasificación de movimiento que puede integrarse fácilmente en los dispositivos existentes. Utilizando los datos de un acelerómetro único incrustado en el dispositivo, el algoritmo puede clasificar entre los patrones de circulación diferentes - el "contexto" - de la persona supervisada. Describirá el *hardware* y el algoritmo proporcionará los primeros resultados de la evaluación demostrando la eficacia de este enfoque para proporcionar conciencia del contexto en aplicaciones médicas móviles en tiempo real.

Infraestructura de red de sensor para teléfonos móviles. Tecnología móvil y ambiental puede utilizarse para evaluar el comportamiento humano y la salud mental de trayectorias fuera de los laboratorios y en configuración ecológicamente pertinentes. Para lograr el máximo beneficio, el conjunto de equipos y los patrones de supervisión deben ser personalizados al respetar las necesidades individuales y encajan en estilos de vida individuales. En (Blum and Magill, 2010) se ha desarrollado una infraestructura de red de sensor para teléfonos móviles y cuidados en el hogar usando una arquitectura de programación orientada a reglas para monitorear las actividades de las personas con trastorno bipolar (BD). Han evaluado la eficacia de la tecnología en un ensayo técnico en curso con participantes de control para estudiar la eficacia del sistema para su uso con personas con BD.

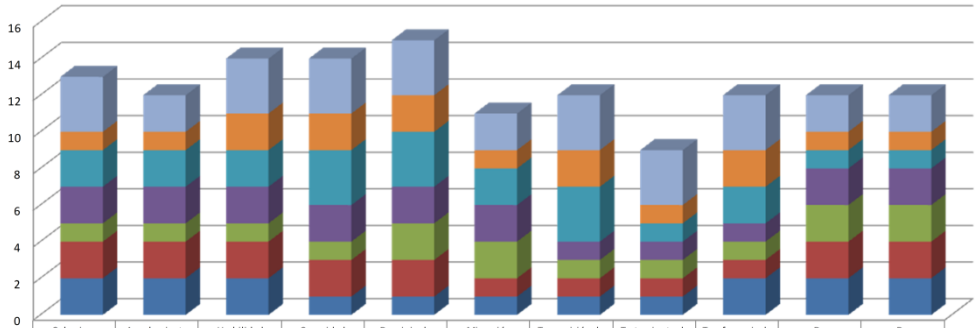
Sistema de Monitorización de señales fisiológicas en tiempo real. En (Sing-Hui, Seung-Chul et al., 2008) se describe un sistema de monitoreo de salud robusto que poseen múltiples mediciones de señales fisiológicas en tiempo real y que se aplica a diversos entornos que integra la tecnología de red de sensor inalámbrico (*WSN*) y técnica digital de acceso múltiple por división de códigos (*CDMA*) con la implementación de un algoritmo en el teléfono móvil. Una simple base de datos se crea dentro del teléfono móvil que facilita a los pacientes almacenar, evaluar y gestionar sus registros de salud personal y la información a través de esta herramienta de administración de información integrada en cualquier lugar y en cualquier momento. Cualquier cambio en los patrones de señales se enviará inmediatamente al servidor del hospital a través teléfono móvil para la evaluación de los médicos. Esta función permite a los pacientes estar informados de la importancia del auto-cuidado de tal forma que las acciones preventivas pueden tomarse antes. Un apropiado autocontrol puede curar la enfermedad y aliviar el dolor especialmente a los pacientes que sufren de enfermedades crónicas que necesitan observación a largo plazo.

Control de diabetes. En (Roudsari, Zhao et al., 2000) han desarrollado dos prototipos de sistemas para diabetes, que soportan las recomendaciones de análisis de datos y la terapia para el paciente individual, cuyo objetivo es hacer coincidir las prácticas iterativas de la profesión médica con necesidades del paciente. Métodos avanzados de análisis de series temporales se utilizan para extraer los patrones e interpretar los cambios abruptos en los patrones de origen de datos. El primer sistema se centra en ajustes de visita, basados en el conjunto de datos recogidos entre visitas de clínica; esencialmente un prototipo para un auxiliar médico se implementa como parte de un

sistema de gestión de la diabetes con inteligencia distribuida. El segundo, evolucionó a partir del primero, incorpora los últimos avances tecnológicos de un sistema basado en la *Web*. Proporciona apoyo a la gestión diaria, acceso a los registros electrónicos de diabetes para los usuarios finales e información en tiempo real de profesionales de la salud a fin de facilitar la toma de decisiones. Integra un glucómetro portátil, ordenador de mano, teléfono móvil y acceso a Internet, y la solución informática móvil para el manejo de la diabetes. Este sistema presenta un concepto innovador de la gestión de la diabetes, por ejemplo, la cita en línea y la consulta, permitiendo a los usuarios acceso a la información de gestión de la diabetes sin limitación de la ubicación, tiempo y las preocupaciones de seguridad.

En la tabla 3-19, puede mostrarse la comparativa de los criterios definidos para la evaluación de otras aportaciones tecnológicas analizadas.

Estas aportaciones se ubican en este apartado ya que en la documentación encontrada no especifican si forman parte de una aplicación en particular o si se han implementado. Estas aportaciones son un valor agregado al desarrollo de aplicaciones para la monitorización móvil.



	Cohesion	Acoplamiento	Usabilidad	Capacidad evolutiva	Dominio de aplicación	Migración tecnológica	Transmisión de datos	Tratamiento de datos	Tranferencia de datos	De Implementación	De Mantenimiento
	Diseño/			Adaptabilidad/			Comunicación/	Seguridad/		Costos/	
MoMo	3	2	3	3	3	2	3	3	3	2	2
Control de Diabetes	1	1	2	2	2	1	2	1	2	1	1
Monitorización de señales	2	2	2	3	3	2	3	1	2	1	1
Infr. de red de sensor	2	2	2	2	2	2	1	1	1	2	2
Algoritmo de detección	1	1	1	1	2	2	1	1	1	2	2
Supervisión de cuidado de pacientes.	2	2	2	2	2	1	1	1	1	2	2
Sistema clínico de alertas.	2	2	2	1	1	1	1	1	2	2	2

Figura 3-22. *Comparativa de los criterios definidos para la evaluación de otras aportaciones tecnológicas,* según cada su-criterio.

En la figura 3-22 puede verse el valor o peso asignados luego de del análisis de cada aportación tecnológica. En el apartado 3.6.1 se analizará por separado cada criterio.

Propuestas	Diseño						Adaptabilidad						Comunicación		Seguridad				Costos			
	Cohesión		*Acoplamiento*		*Usabilidad*		*Capacidad evolutiva*		*Dominio de aplicación*		*Migración Tecnológica*		*Transmisión de datos*		*Tratamiento de datos*		*Transferencia de datos*		*De implementación*		*De mantenimiento*	
Sistema clínico de alertas.	MD	2	MD	2	MD	2	BJ	1	BJ	1	MD	1	MD	1	BJ	1	MD	2	MD	2	MD	2
Supervisión de cuidado de pacientes.	MD	2	MD	2	MD	2	MD	2	MD	2	BJ	1	BJ	1	BJ	1	BJ	1	MD	2	MD	2
Algoritmo de detección	BJ	1	AT	1	BJ	1	BJ	1	MD	2	MD	2	BJ	1	BJ	1	BJ	1	MD	2	MD	2
Infr. de red de sensor	MD	2	MD	2	MD	2	MD	2	MD	2	MD	2	BJ	1	BJ	1	BJ	1	MD	2	MD	2
Monitorización de señales	MD	2	MD	2	MD	2	AT	3	AT	3	MD	2	AT	3	BJ	1	MD	2	AT	1	AT	1
Control de Diabetes	BJ	1	AT	1	MD	2	MD	2	MD	2	BJ	1	MD	2	BJ	1	MD	2	AT	1	AT	1
MoMo	AT	3	MD	2	AT	3	AT	3	AT	3	MD	2	AT	3	AT	3	AT	3	MD	2	MD	2

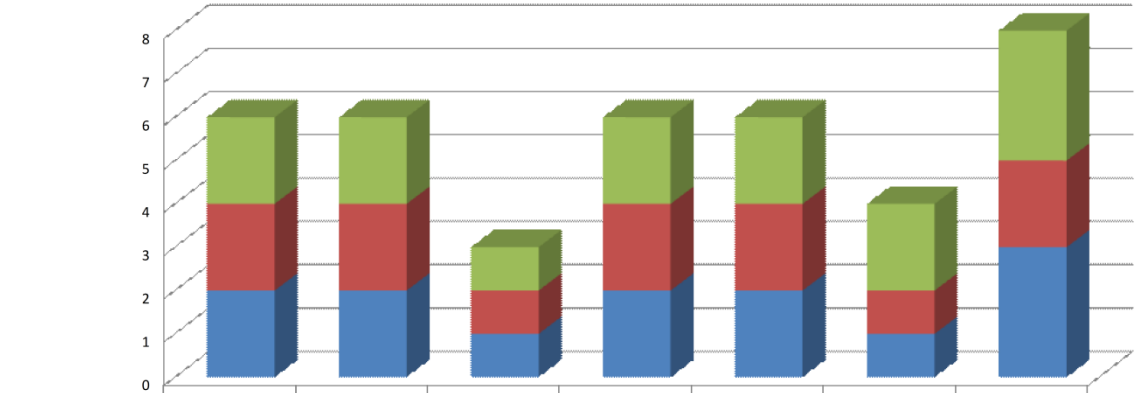
Tabla 3-19. Comparativa de los criterios definidos para la evaluación de otras aportaciones tecnológicas.

3.6.1 Análisis de criterios de evaluación de otras aportaciones tecnológicas.

Además de los estudios analizados previamente, se ha hecho una clasificación de otras aportaciones tecnológicas que se han encontrado, referentes al desarrollo de aplicaciones para la monitorización de pacientes. Este análisis está basado en los criterios mencionados en el apartado 3.4.1.

1. Diseño

Cuantificando el grado del criterio *diseño* y sus sub-criterios, se muestra la figura 3-23, la evaluación de otras aportaciones analizada en el estado del arte de esta tesis.



	Sistema clínico de alertas.	Supervisión de cuidado de pacientes.	Algoritmo de detección	Infr. de red de sensor	Monitorización de señales	Control de Diabetes	MoMo
Diseño/ Usabilidad	2	2	1	2	2	2	3
Diseño/ Acoplamiento	2	2	1	2	2	1	2
Diseño/ Cohesion	2	2	1	2	2	1	3

***Figura 3-23**. Nivel de diseño de otras aportaciones tecnológicas.*

En este criterio se nota que las aportaciones tecnológicas evaluadas, presentan un promedio medio en cuanto al diseño. Esto puede ser debido a que la mayoría de ellas no han sido definidas como aplicaciones finales, sino como un aporte a estudios ya desarrollados previamente. En todo caso, estas aportaciones significan un apoyo al momento de desarrollar nuevas aplicaciones. *MoMo* sigue ofreciendo un alto grado de diseño en todos los aspectos evaluados.

2. Adaptabilidad

Cuantificando el grado del criterio *adaptabilidad* y sus sub-criterios, se muestra la figura 3-24 la evaluación de otras aportaciones analizada en el estado del arte de esta tesis.

Gran parte de las aportaciones estudias presentan un grado medio de adaptabilidad tanto tecnológica como de dominio de aplicación. Es decir, pueden ser implementadas en otras arquitecturas planteadas. El criterio más altamente valorado ha sido el de *dominio de aplicación*, ya que cada aportación contempla la implementación de sus propuestas en diversos entornos.

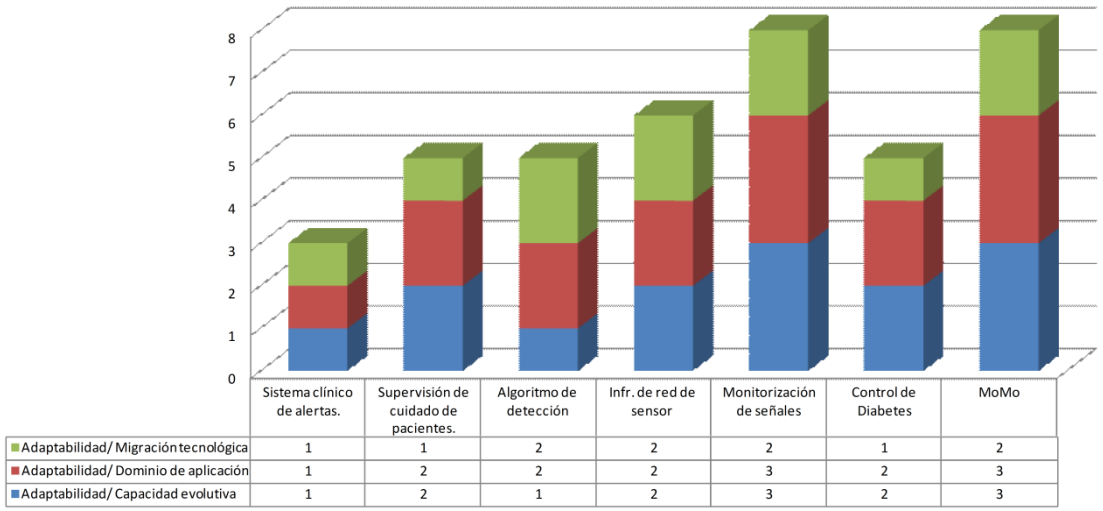


***Figura 3-24**. Nivel de adaptabilidad de otras aportaciones tecnológicas.*

3. Comunicación

Cuantificando el grado del criterio *comunicación* y sus sub-criterios, se muestra la figura 3-25 la evaluación de otras aportaciones analizada en el estado del arte de esta tesis.

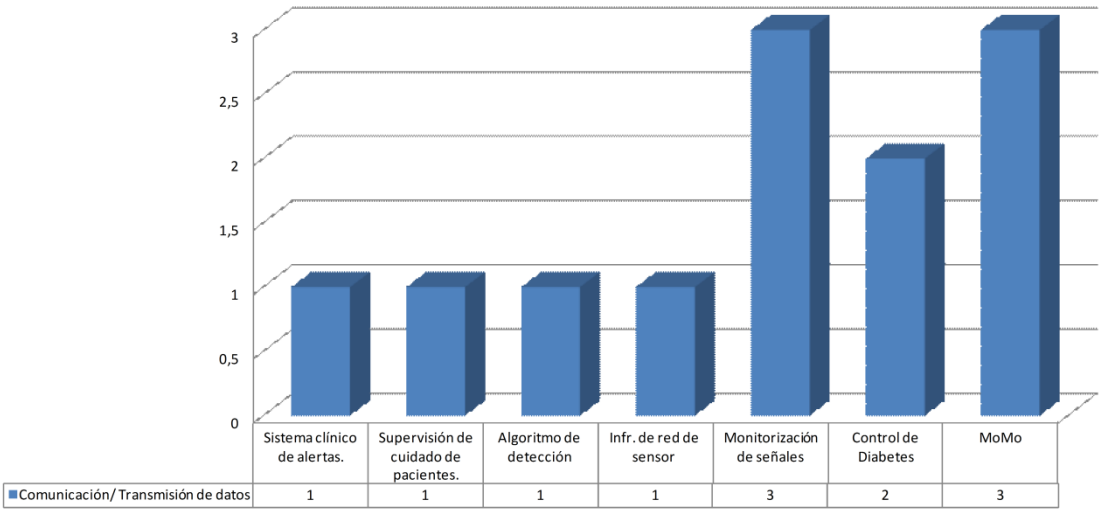
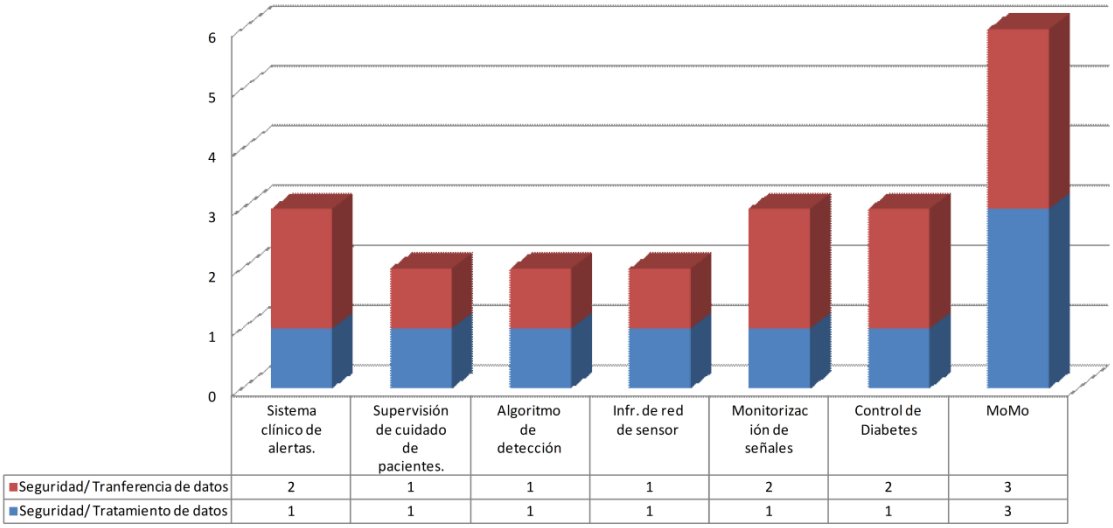


***Figura 3-25**. Nivel de comunicación de otras aportaciones tecnológicas.*

La mayoría de las aportaciones analizadas, no fueron desarrolladas para entornos de trasmisión de datos a través de las tecnologías de comunicación existentes. Sólo una de ellas contempla este criterio dentro de su especificación.

4. Seguridad

Cuantificando el grado del criterio *seguridad* y sus sub-criterios, se muestra la figura 3-26 la evaluación de otras aportaciones analizada en el estado del arte de esta tesis.



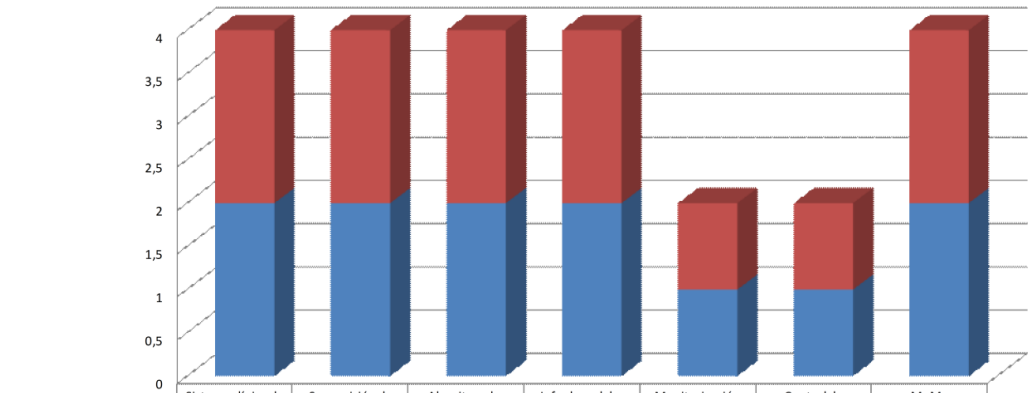
	Sistema clínico de alertas.	Supervisión de cuidado de pacientes.	Algoritmo de detección	Infr. de red de sensor	Monitorización de señales	Control de Diabetes	MoMo
Seguridad/ Tranferencia de datos	2	1	1	1	2	2	3
Seguridad/ Tratamiento de datos	1	1	1	1	1	1	3

***Figura 3-26**. Nivel de seguridad de otras aportaciones tecnológicas.*

De igual manera que los criterios anteriores, dentro de las aportaciones analizadas no se encuentra ninguna que haga referencia a la seguridad de los datos. Todas ellas acotan factores de procesamiento de información, sin especificar la trasmisión de las mismas.

5. Costos

Cuantificando el grado del criterio *costos* y sus sub-criterios, se muestra la figura 3-27 la evaluación de otras aportaciones analizada en el estado del arte de esta tesis. Los costos analizados están en un rango medio con respecto al mantenimiento e implementación. El criterio de costos puede ser muy variante y depende de las especificaciones de cada propuesta.



	Sistema clínico de alertas.	Supervisión de cuidado de pacientes.	Algoritmo de detección	Infr. de red de sensor	Monitorización de señales	Control de Diabetes	MoMo
Costos/ De Mantenimiento	2	2	2	2	1	1	2
Costos/ De Implementación	2	2	2	2	1	1	2

***Figura 3-27**. Nivel de costos de otras aportaciones tecnológicas.*

En la *Tabla 3-20* se muestra una comparativa más resumida entre las principales investigaciones y propuestas evaluadas en nuestro trabajo. Se explica su funcionamiento, el

tipo de enfermedad para la que fue desarrollada, qué tipo de tecnología *software* y *hardware* utiliza, y muy importante, para qué áreas dentro del entorno médico fue desarrollada.

Dentro de la tabla se han especificado las características del *framework* **MoMo (Mo**bile **Mo**nitoring**)**, que se ha desarrollado, notándose que se cubren en una misma aportación, aspectos que son contemplados en propuestas separadas. De igual manera se proponen otras funcionalidades que le dan mayor robustez a nuestra propuesta.

3.7 CONCLUSIONES

En este capítulo se presentaron las diferentes aportaciones que han hecho algunos investigadores en el campo de monitorización de pacientes, ya sea móvil o alguna aproximación a este tipo.

Se han definido un número de criterios de evaluación para conocer el nivel de integración, composición y funcionamiento que tiene cada propuesta, para encontrar las deficiencias y comprender mejor el ámbito de desarrollo de la nuestra. Algunas de las investigaciones analizadas cubren ciertos aspectos propuestos inicialmente en este tema de tesis, pero algunos no han sido implementados. Los criterios fueron evaluados de forma objetiva, basándonos en las especificaciones de cada investigación. Algunas de las propuestas documentadas no explicaban al detalle la distribución de sus elementos, lo que nos hace pensar, que se quedaron como intensiones de desarrollo.

Podemos notar que aunque son muchas las investigaciones que se han desarrollado, todas fueron desarrolladas orientándose a entornos particulares, es decir para solucionar una situación puntual. Se nota con ello que algunas investigaciones tienen un tiempo de vida perentorio, lo que limita el concepto de "*multi-monitorización"* que buscamos en esta tesis. Esto nos lleva a desarrollar una arquitectura software basada en el framework que no es dependiente de un ámbito de estudio en particular, sino que puede ser ajustada a diversas áreas.

Buscamos con esto facilitar una monitorización múltiple, es decir que integre diferentes tipos de enfermedades o padecimientos y que puedan ser monitorizadas por diversos dispositivos móviles y biométricos.

Autor	Nombre que lo identifica	¿Cómo es su funcionalidad?	¿Qué monitoriza?	¿Cómo se implementa?	¿Cuáles tecnologías utiliza?	¿Qué aspectos médicos cubre?
(Georga, Protopappas et al., 2009)	METABO	Interpretación y la grabación de contexto del paciente y prestar apoyo a decisiones médico-paciente.	- Diabetes (Glucosa continua y discontinua).	- Minería de datos.	- Teléfono móvil. - Biosensores. - Subsistema Central.	- Dieta. - Actividad física. - Medicación. - Inf. Médica.
(Mohanalakshmi and Jacob, 2007)	Sistema para el cuidado respiratorio.	Sistema de alertas en tiempo real	- Cuidados respiratorios.	- Algoritmos de alertas.	- Sistema de tele-observación. - Teléfono móvil.	- Aviso a especialistas médicos.
(Wan-Young, Seung-Chul et al., 2008)	No especificado	Sistema móvil de cuidado de salud, mide varios signos fisiológicos	- Constantes vitales. - Presión arterial.	No especificado.	- Teléfono móvil. - Red de sensores. - Electrocardiograma. - Servidor central.	- Mecanismos adversos de la medicación.
(Koutkias, Chouvarda et al.)	*Framework* para la supervisión de la cond. Física.	Supervisión de la condición del paciente y la seguridad en relación al tratamiento de la medicación.	- Hipertensión.	- Patrones de vigilancia.	- Red de área corporal (BAN) con sensores. - Unidad Móvil. - Concentrador central	- Administración de la medicación.
(Hellbruck, Hua et al., 2009)	Algoritmo de detección de movimiento.	Detección y clasificación de movimiento que pueden integrarse en dispositivos móviles.	- El contexto del paciente.	- Algoritmo.	- Algoritmo de detección. - Acelerómetro.	- Movimiento de pacientes.
(Blum and Magill)	Inf. de red de sensores para teléfonos móviles.	Infraestructura que monitoriza las actividades de las personas con trastornos bipolares.	- Trastornos bipolares.	-Programación basada en reglas. - Patrones de supervisión.	- Teléfono móvil. - Sensores.	- Comportamiento de pacientes.
(Ryder, Longstaff et al., 2009)	AMBULATION	Presenta una herramienta importante para evaluar la salud de los pacientes que sufren de enfermedades crónicas que afectan a la movilidad.	- Parkinson. - Distrofia muscular.	- Aplicación en Web.	- Teléfono móvil (Android y Nokia N95).	- Movilidad de pacientes.
(Sing-Hui, Seung-Chul et al., 2008)	Sistema de monitorización de señales fisiológicas en tiempo real.	Sistema de monitoreo de salud robusto que poseen múltiples mediciones de señales fisiológicas en tiempo real y que se aplica a diversos entornos que integra la tecnología de red de sensor	- Auto-cuidado en general. - Auto-monitorización en general.	- Algoritmo.	- Teléfono móvil. - Red de sensor inalámbrico (WSN). - Técnica digital de acceso múltiple por	- Almacenar, evaluar y gestionar registros de salud personal.

		inalámbrico (WSN) y técnica digital de acceso múltiple por división de códigos (CDMA).			división de códigos (CDMA).	
(Paradiso, Alonso et al., 2008)	HEALTHWEAR	Permite la supervisión de las condiciones de salud de un paciente. Se hizo un nuevo diseño para aumentar la comodidad de vestir del sistema durante las actividades cotidianas del paciente.	- Saturación de oxígeno. - Impedancia pneumography. - Patrones de actividad. - ECG.	- Patrones de Actividad.	- Unidad electrónica portátil. - Sensores. - Servidor central. - GPRS.	- Rehabilitación de pacientes cardiacos. - Pacientes de respiración crónica. - Promoción de actividad física con problemas cardio-respiratorios.
(Roudsari, Zhao et al., 2000)	Sistema de control de Diabetes	Hacer coincidir las prácticas iterativas de la profesión médica y necesidades médicas del paciente.	- Diabetes.	- Análisis de datos y terapia - Inteligencia distribuida - Sistema basado en Web.	- Glucómetro portátil. - Ordenador de mano. - Teléfono móvil. - Internet.	- Gestión diaria del paciente. - Cita en línea. - Consultas.
(Villarreal, Bravo, et al, 2010)	MoMo (*Framework* para la monitorización móvil de pacientes)	Obtención de señales vitales, seguimiento de pacientes, módulos de autocontrol, educación para pacientes, a través de dispositivos móviles con conectividad a dispositivos biométricos.	- Señales vitales para enfermedades crónicas. - Diabetes, tensión arterial, temperatura, etc.	- Distribución de datos a través de ontologías. - Diseño de interfaces y funcionalidades a través de patrones. - Distribución de elementos a través de capas.	- Dispositivos móviles (móvil, Tablet, Smartphone) - Dispositivos biométricos/sensores (glucómetro, tensiómetro, etc.) - Conectividad Bluetooth - WiFi, Red de Telefonía Móvil. - Servidor Central.	- Movilidad de pacientes. - Obtención de señales vitales. - Generación de recomendaciones, mensajes de educación, dieta, ejercicios. - Gráficos de autocontrol. - Gestión diaria del paciente. - Mensajes de alertas al médico.

Tabla 3-20. *Estudios relacionados al desarrollo de arquitecturas para monitorización de pacientes.*

4

CAPÍTULO CUARTO

4 MOMO: DISEÑO CONCEPTUAL DEL FRAMEWORK.

En este capítulo se describen los elementos del *framework* desarrollado que guiará el desarrollo de una arquitectura software para la generación de aplicaciones móviles parametrizadas, que facilitan la monitorización móvil de pacientes que será explicada en el capítulo cinco.

En foco de nuestra tesis es crear una arquitectura completa con diferentes servicios que permiten al paciente a través de dispositivos biométricos proporcionar datos e información a los dispositivos móviles (por ejemplo, teléfonos móviles, *PDA*, *Tablet* o *Smartphones*). No solamente se recopila información sino que también se puede evaluar tendencias de las mediciones y prestar asesoramiento sobre salud, dieta, sugerencias sobre la prevención de ciertos aspectos de una enfermedad, todo esto a través de un enlace de datos no redundantes y transparente entre el paciente y su dispositivo móvil personal.

En la figura 4-1, se muestra la distribución de todos sus elementos funcionales, lo que permitirá un desarrollo estandarizado de esta tesis.

Los componentes del *famework* se agrupan de la siguiente manera:

1. **Elementos de distribución del *framework* (MoMoLayers):** Hemos distribuido los elementos del *framework* en capas, lo que permite una mayor interoperabilidad y facilita la

normalización en el desarrollo. Cada capa y su respectivas sub-capas definen aspectos como:

- ***Comunicación***: esto se refiere a la posibilidad técnica de comunicación entre dispositivos biométricos y móviles.

- ***Seguridad***: otro aspecto considerado durante el desarrollo de nuestro modelo en capas, ha sido la seguridad de la transferencia de datos y del tratamiento de datos.

- ***Desarrollo de aplicaciones***: cad uno de los componentes las aplicaciones generadas mediante nuestro modelo en capas se ubicarán en un mismo nivel independientemente de la tecnología *hardware*. Por eso la ubicación de las aplicaciones desarrolladas (módulos integrados en una sola aplicación final) debe colocarse en un nivel identificable.

- ***Heterogeneidad***: muchos dispositivos con capacidades de comunicación deben convivir en el mismo entorno o poder comunicarse con los dispositivos biométricos sin ninguna dificultad.

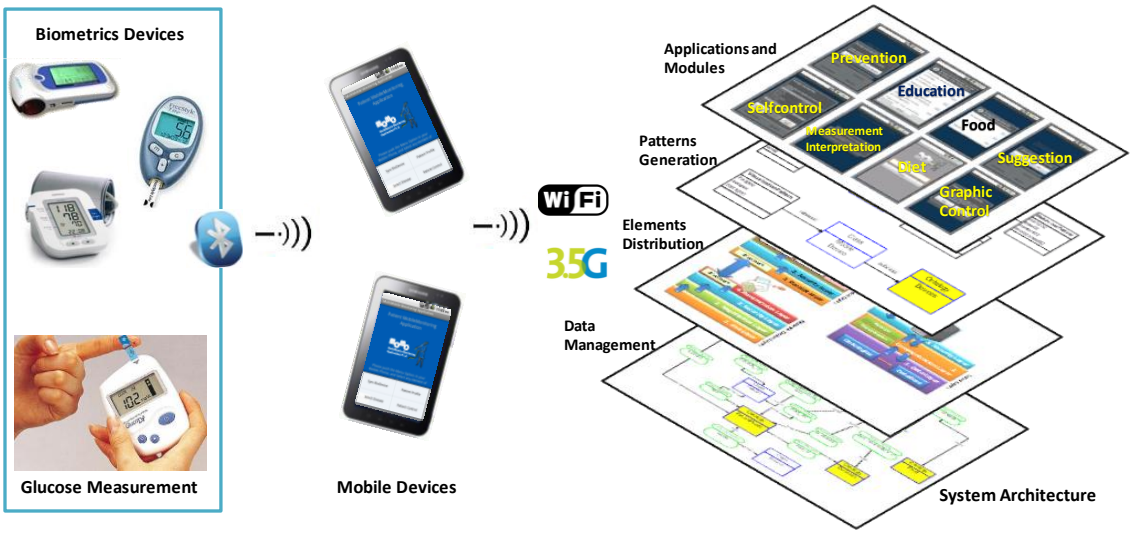


Figura 4-1. *Distribución de los elementos del framework desarrollado.*

2. **Definición de ontologías (MoMOntologies)**: se define un conjunto de ontologías que proporcionan la información base de desarrollo del *framework*, lo que permitirá en un futuro generar un motor de inferencia que permite la obtención de nueva información, a partir de las situaciones previas de un paciente.

3. **Patrones de generación de aplicaciones parametrizadas (MobiPatterns):** El *framework* que hemos desarrollado permite la construcción o la generación de aplicaciones interactivas para ser integradas en el dispositivo móvil. Para la creación de estos módulos y la integración de cada uno y la generación de aplicaciones móviles, hemos definido y desarrollado un conjunto de patrones, llamado *MobiPattern*. Para la definición de cualquier *MobiPatterns* debemos considerar todas las representaciones generadas después de

realizar una medición. Cada MobiPattern permite la generación de cada módulo que integra la aplicación final.

Los detalles funcionales de los elementos conceptuales definidos en este capítulo, se explicarán en el capítulo cinco (capítulo de implementación), por lo que en este capítulo nos limitaremos a explicar la conceptualización de los aspectos desarrollados inicialmente. En la figura 4-2, se muestra un la distribución de todos los elementos del *framework*.

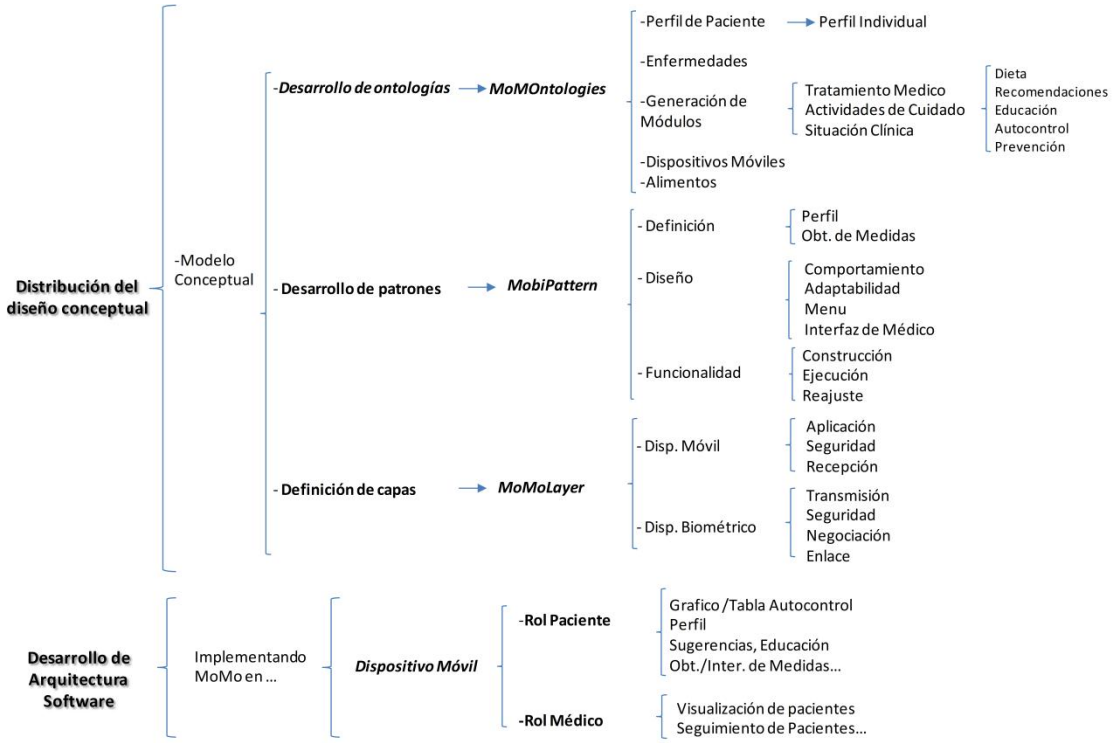


Figura 4-2. *Distribución gráfica de los aspectos relacionados al diseño conceptual del framework y el desarrollo de la arquitectura software.*

4.1 MOMOLAYER: DISTRIBUCIÓN EN CAPAS PARA EL DESARROLLO DE APICACIONES MÓVILES.

4.1.1 Introducción

Algunas investigaciones y trabajos relacionados con monitorización en otros ámbitos, han definido una estructura en capas de toda su arquitectura. Esto ha permitido el diseño más ordenado y la interpretación estandarizada en aspectos de comunicación y trasmisión de datos. Aunque no se ha definido un modelo en capas para la monitorización de pacientes, se toman de ejemplo otras áreas en donde esta tecnología ha funcionado.

Para el desarrollo del *framework*, nos basaremos en la ideología de desarrollo de aplicaciones basadas en n-capas, la cual ajustaremos a las necesidades de nuestra investigación. Mantendremos el concepto de la funcionalidad de la capa de datos, mientras que las capas de presentación y de negocio serán subdivididas en varias capas con funcionalidades lógicas y físicas. Las funcionalidades lógicas definen el desarrollo del *framework* a nivel de sus componentes, es decir, cómo se organiza la aplicación final. Mientras

que las funcionalidades físicas, especifican la comunicación, transmisión y recepción entre cada elemento que interviene en el proceso de monitorización.

El diseño en n-capas ofrece servicios de adaptación y diseño de los elementos que componen el *framework* desarrollado. El desarrollo y distribución en capas ofrece los siguientes beneficios:

- Permite una distribución estructurada de cada uno de los elementos funcionales de la arquitectura *software* a desarrollar;
- Facilita la identificación de elementos estructurales dentro del *framework*;
- Facilita el desarrollo de cada elemento en particular, ubicándolos en niveles identificables para el desarrollo;
- Facilita el mantenimiento y el soporte posterior;
- Facilita la migración tecnológica ante posibles cambios de *software* y *hardware*;
- Ofrece alto grado de escalabilidad, es decir la integración de nuevos elementos sin tener que modificar la arquitectura inicial.

4.1.2 Capas del Framework

Para facilitar la comunicación entre cada uno de los elementos *software* y se ha diseñado un modelo en capas para distribuir los elementos del *framework*. Esta distribución en n-capas, además facilita el manteniendo y actualización del *framework*, organizando todos los componentes, de tal manera que se identifique que parte de la arquitectura necesita ajustes (Villarreal, Urzaiz et al., 2011). Como se muestra en la figura 4-3, dicho modelo ha sido distribuido entre todos los elementos que componen el *framework*, dotándole de funcionalidad a cada una de ellas, estos son:

- los *dispositivos móviles* (teléfono móvil, PDA y demás Smartphone);
- los *dispositivos biométricos* (glucómetros, tensiómetros, termómetros, entre otros) y
- el *servidor principal* donde se encuentran integrado todos los elementos distribuidos del *framework*.

Cada uno de ellos define una funcionalidad específica relacionada a cada dispositivo que interviene en el *framework*, dichas funcionalidades se pueden agrupar en los siguientes componentes:

1. Comunicación
2. Seguridad
3. Desarrollo de aplicaciones
4. Heterogeneidad

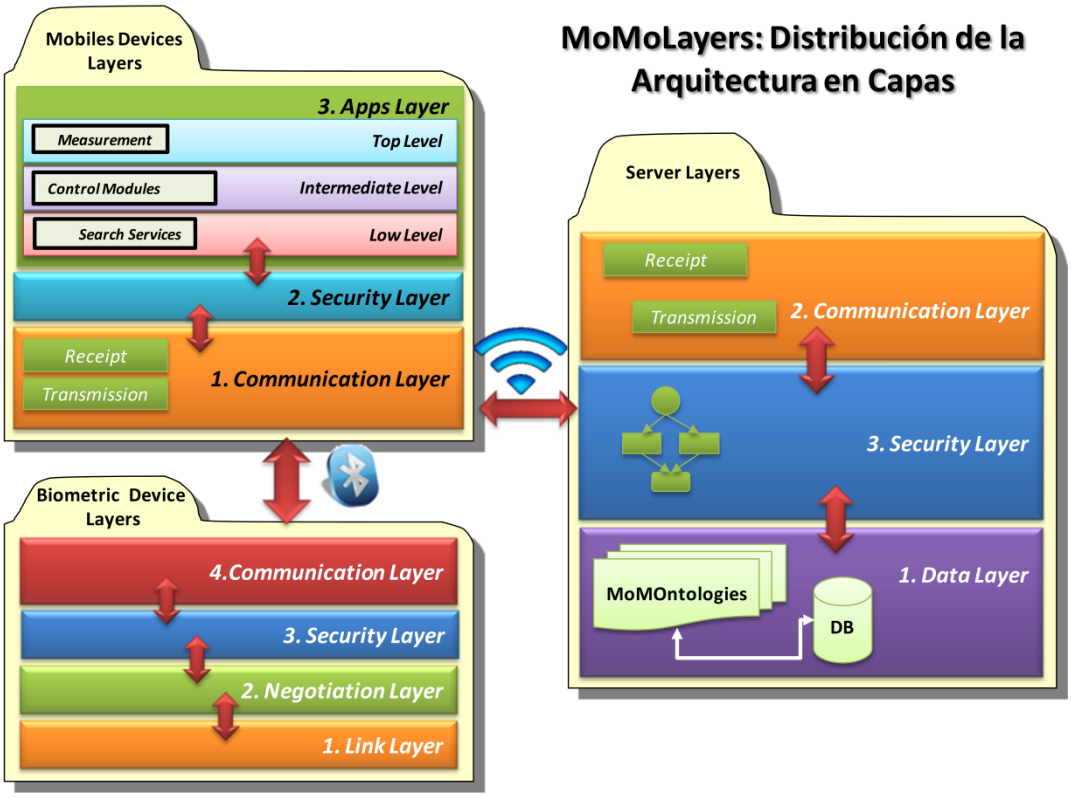


Figura 4-3. *Distribución del modelo en capas para el framework*

1. **Comunicación**: se refiere a la posibilidad técnica de que los dispositivos tanto biométricos como móviles de comunicarse entre sí. Esta comunicación está basada en tres sub-capas que son: *de negociación, enlace* y *de transmisión*.

   Para facilitar la comunicación entre dispositivos se cuenta con diversos medios físicos. Entre ellos está la comunicación Bluetooth que se utiliza para la comunicación entre dispositivo biométrico y dispositivo móvil, y la comunicación de red (en algunos casos red de área global, WiFi y 3G) para la transmisión entre los dispositivos móviles y el servidor.

   Para que un dispositivo pueda comunicarse con otro, previamente debe habilitarse un canal de comunicación. Este canal de comunicación negociará los términos de la comunicación, en donde se establecen los aspectos de requerimiento de cada dispositivo y las especificaciones del paquete a entregar (tamaño, cabeceras, tipo de encriptación, etc.). Luego de negociados estos aspectos entre ambos dispositivos, se procede a la transferencia de los datos requeridos, convirtiéndose el dispositivo biométrico en el emisor y el dispositivo móvil como receptor.

2. **Seguridad**: otro de los aspectos considerado al momento de desarrollar nuestra arquitectura por capas, es el concerniente a la seguridad de transferencia de los datos. Algunos investigadores han propuestos algunas soluciones que hemos evaluado antes de elegir la solución más apropiada para nuestro *framework*.

   La seguridad en nuestro *framework* está relacionado a la protección de envío de los datos capturados por el dispositivo biométrico, es decir, al envío de una (s) medida (s) que deben llegar de manera segura al dispositivo biométrico. Estos es, debido a que al utilizar

tecnología de comunicación entre dispositivo, existe la posibilidad de pérdida o intersección de paquetes en el momento en que la aplicación se esté ejecutando. Para ello se identifica y define capas que permitan una transferencia transparente y a la vez segura para el usuario.

3. **Desarrollo de aplicaciones**: se debe poder localizar a un mismo nivel e independientemente de la tecnología *hardware*, todas las aplicaciones con que cuenta nuestra arquitectura por capas. Es por ello que las aplicaciones desarrolladas (módulos integrados en una sola aplicación final) deben poder ubicarse en un nivel identificable dentro de la arquitectura *software*.

   Para el desarrollo de las aplicaciones, se definen tres niveles de funcionamiento. Estos niveles especifican la ubicación de cada uno de los módulos generados con base en los MobiPatterns diseñados y que se explican en el capítulo 5 de este documento. Es decir la capa de aplicación está compuesta por un conjunto de sub-capas con características funcionales de la propia aplicación.

4. **Heterogeneidad:** muchos dispositivos con capacidades de comunicación variadas deben convivir en un mismo entorno, es de decir, deben poder comunicarse con los dispositivos biométricos sin ninguna dificultad; sin importar la tecnología de comunicación con que cuente dicho dispositivo. Este aspecto se ha analizado al momento de integrar nuevas tecnologías de comunicación y transmisión de datos en los niveles más bajos de nuestra arquitectura por capas.

   La heterogeneidad permite realizar mínimos cambios a la arquitectura, al momento de integrar nuevos dispositivos biométricos. Siempre y cuando el dispositivo biométrico cuente con alguna tecnología de comunicación, debe poder ajustarse su integración a la arquitectura de *software* desarrollada. En el caso de que un dispositivo biométrico cuente con otra tecnología de comunicación, debería realizarse un cambio en la capa de enlace y de transmisión para parear este nuevo dispositivo con la aplicación instalada en el móvil. También se ofrece la escritura de las señales vitales a través del teclado del dispositivo móvil, esto permite que otros dispositivos biométricos que no tengan capacidad de comunicación/transferencia puedan interactuar con la arquitectura.

4.1.2.1 Capas del Servidor

El servidor es el encargado de gestionar la base de datos, ofrecer servicios de comunicación y procesamiento y ajustar las ontologías con que cuenta el *framework*, de tal manera que se puedan generar las aplicaciones que serán instaladas en el dispositivo móvil.

Además recoge la información enviada por los dispositivos móviles que utiliza cada uno de los pacientes que van a ser monitorizados por el sistema; pone esa información a disposición de las capas de seguridad y comunicación, actuando de pasarela entre estas capas y las capas del dispositivo móvil. Para ello cuenta con las siguientes sub-capas principales, mostradas en la figura 4-4.

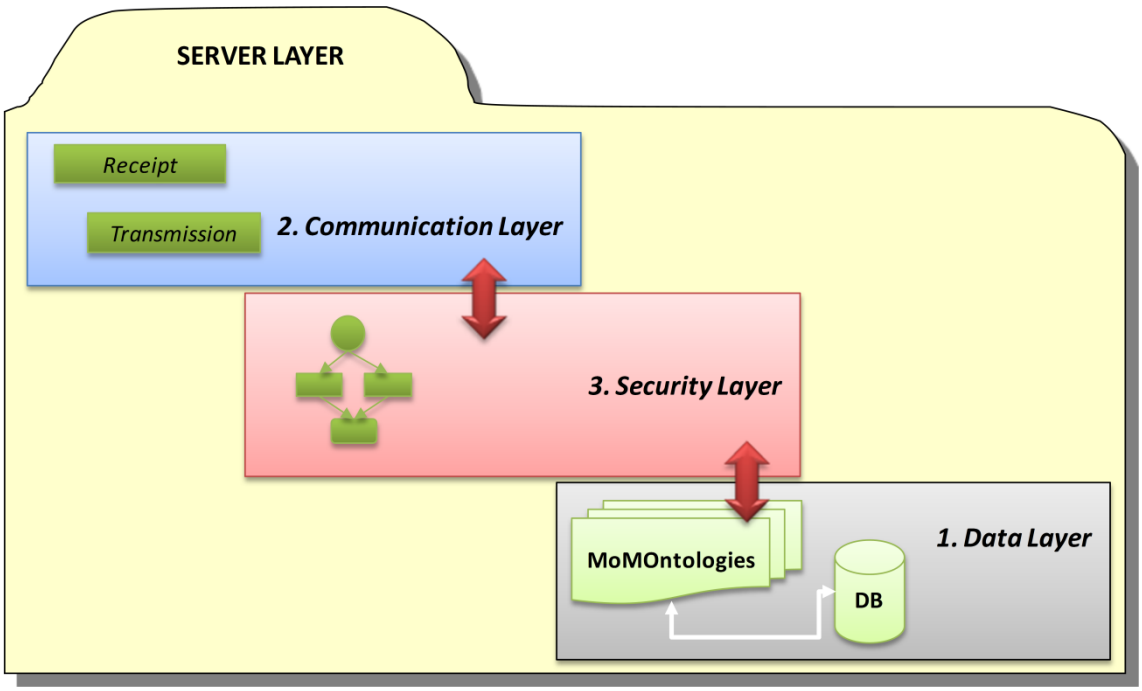


Figura 4-4. *Distribución de las capas que componen el Servidor.*

El objetivo principal de las capas del servidor es centralizar y estructurar la información de los pacientes y actuar de pasarela entre el paciente que accede remotamente a su datos y los dispositivos biométricos que permiten la recopilación de la información del estado y mediciones de los pacientes. Entre sus funciones cabe destacar:

- Almacenar la información (parámetros médicos o señales vitales) enviada por todos los dispositivos móviles de los pacientes.
- Generar servicios de procesamiento y trasmisión de esas señales vitales, con el objetivo de facilitar el control médico del paciente basado en recomendaciones según medidas puntuales.
- Actualizar y rediseñar las ontologías desarrolladas, de tal manera que pueda facilitar información actualizada ubicada en la capa de aplicación del dispositivo móvil.
- Ofrecer servicios de seguridad tanto para la transferencia de información como para el procesamiento interno de los datos.

Capa de Datos

En esta capa se especifican todos los elementos que deben tomarse en cuenta al momento de generar las aplicaciones. Para ello se han definido previamente todos los componentes necesarios a través de ontologías, las cuales establecen las relaciones entre cada módulo y define los parámetros de lectura y escritura que se necesiten para cada caso.

Estas ontologías están representadas en la base de datos que almacena toda la información relacionada por un lado a todos los datos obtenidos de un paciente (perfil, lecturas de

medidas) y los datos asociados a una enfermedad y el funcionamiento para cada uno de los módulos generados para ese tipo de enfermedad.

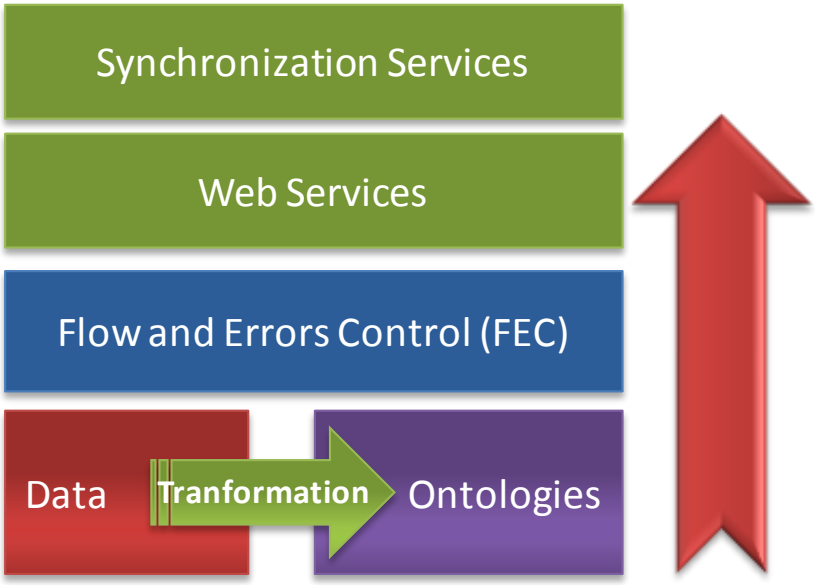


Figura 4-5. *Elementos que componen la capa de datos del servidor.*

La figura 4-5 muestra la distribución de la capa de datos, entre sus funciones tenemos:

- Iniciar la identificación de roles y accesos a la aplicación;
- Enviar la respuesta de sincronización solicitada por la aplicación móvil;
- Establecer los aspectos de control de flujo y acceso de cada módulo de la aplicación final;
- Establecer las relaciones entre los elementos de cada ontología definida;
- Ofrecer servicios a los diferentes módulos al momento que lo soliciten;
- Procesar los datos solicitados en las consultas generadas desde el dispositivo móvil;
- Ofrecer los servicios web, para cada enlace de los dispositivos móviles;
- Actualizar la base de datos según las mediciones obtenidas desde el dispositivo móvil;
- Gestionar y coordinar la comunicación entre el servidor y el dispositivo móvil a través de servicios web definidos.

Capa de Seguridad

En esta capa se especifican todos los parámetros de seguridad para la transferencia de los datos hacia el dispositivo móvil correspondiente. Es decir, se encriptan/desencriptan los datos para que no sean captados por otros dispositivos al momento que la aplicación del paciente es instalada en su dispositivo móvil.

Para mantener la seguridad en los datos es importante mantener la integridad de los datos, evitando alterarlos al momento que son enviados/recibidos entre el dispositivo móvil y el servidor. Al momento de seleccionar un método de criptografía se deben tener en cuenta los siguientes aspectos:

- *Confidencialidad*: evita que los datos sean accedidos por usuarios no autorizados.
- *Integridad*: asegura la integridad de los datos, es decir la capacidad de detectar la manipulación (inserción, supresión, y sustitución) de los datos por terceros no autorizados.
- *Autenticación*: la información prestada debe ser autenticada desde origen, tomando en cuenta el contenido de los datos, el tiempo de envío, fecha de envío, entre otros.

Estos aspectos nos permite la evaluación de múltiples algoritmos, los cuales han sido evaluados a través de los siguientes criterios:

- Nivel de seguridad
- Funcionalidad
- Métodos de operación
- Rendimiento
- Facilidad de aplicación

En la figura 4-6 se muestra los elementos que conforman la capa de seguridad del servidor central. Dentro de sus funciones tenemos:

- Asegurar la integridad de los datos a trasmitir procedentes de la capa de datos;
- Evaluar cada solicitud de acceso a los datos, según el rol (paciente/médico) que solicita el acceso;
- Ofrecer servicios acceso al **FEC** (*Flow and Errors Control*) de la capa de datos;
- Limitar el acceso de personas sospechosas;
- Ofrecer servicios criptografía para asegurar la integridad de los datos recibidos desde la capa de datos.

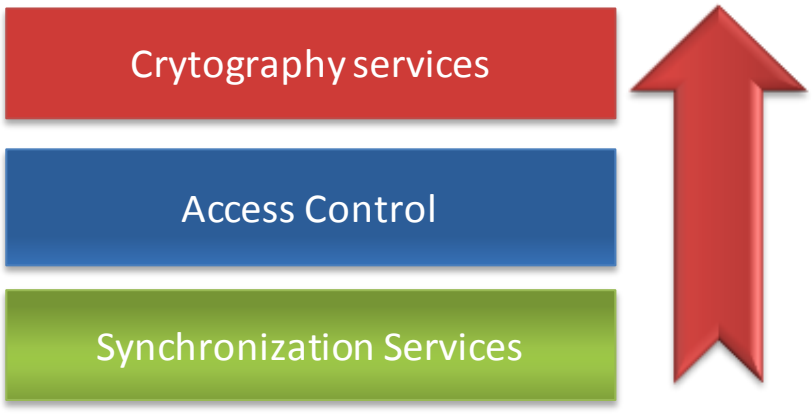


Figura 4-6. *Elementos que componen la capa de seguridad del servidor.*

Capa de Comunicación

En esta capa se establecen los parámetros de comunicación entre el servidor principal y los dispositivos móviles de cada paciente, manteniendo la información actualizada de forma constante.

Esta capa además facilita la actualización de las aplicaciones previamente generadas, dependiendo de las necesidades en cuanto a la evolución del paciente, es por ello que se pueden realizar ajustes en el caso que una enfermedad se complique o el paciente mejore con su tratamiento.

En la figura 4-7, se muestra los elementos de la capa de comunicación del servidor, resumidas en las funciones que tiene la capa de comunicación:

- Habilitar el canal de comunicación entre el servidor y el dispositivo móvil;
- Ofrecer servicios de seguridad para la transmisión de datos;
- Ofrecer al dispositivo móvil acceso a los servicios ofertados por la capa de datos;
- Recibir las solicitudes encriptadas para pasárselas a la capa de seguridad;
- Habilitar n-números de canales que permitan el acceso simultáneo de diversos dispositivos móviles;

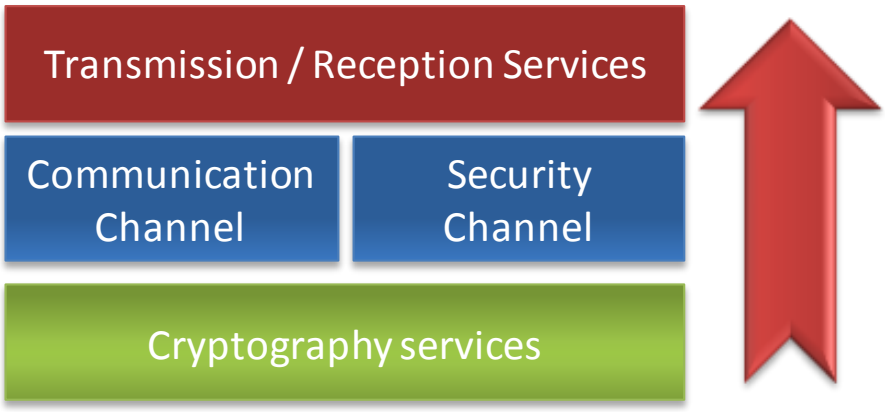


Figura 4-7. *Elementos que componen la capa de comunicación del servidor.*

4.1.2.2 Capas de los Dispositivos Móviles

Para facilitar la distribución de los elementos tanto de aplicación como de seguridad y comunicación de los dispositivos móviles, se ha generado un modelo en capas para los dispositivos móviles que interactúan con los dispositivos biométricos oportunos. En la figura 4-8 se pueden ver la ubicación de cada uno de los elementos y sub-capas que la componen.

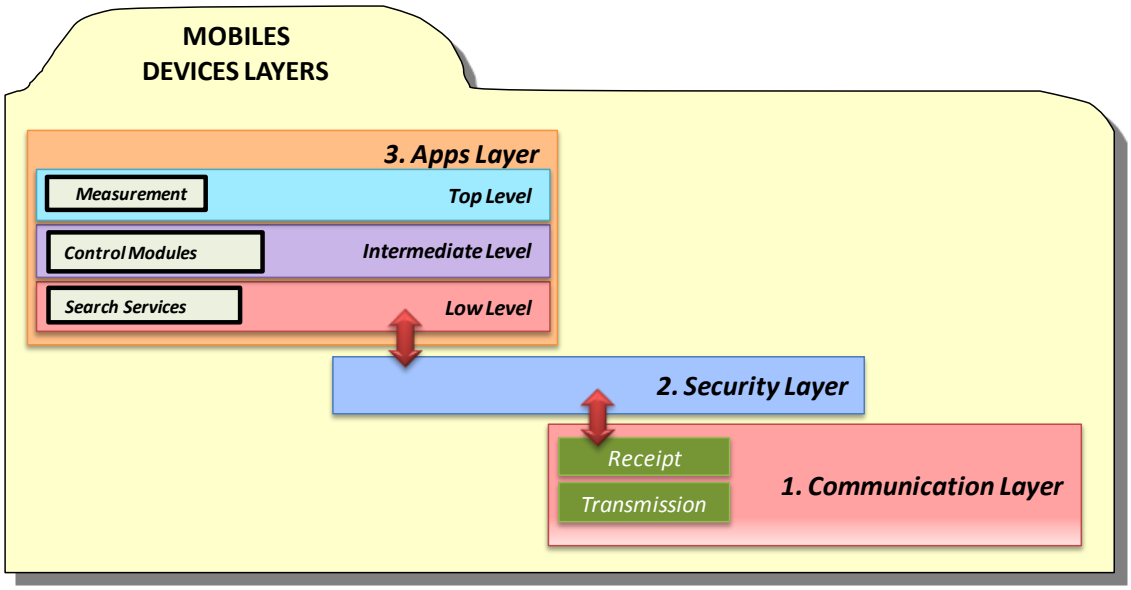


Figura 4-8. *Distribución de las capas que componen los dispositivos móviles*

Capa de Comunicación

Esta capa es la encargada de establecer la comunicación entre el dispositivo biométrico y el dispositivo móvil. Es a través de ella que se reciben los paquetes de datos generados de las medidas de las señales vitales de un paciente. Además de recibir los datos de los dispositivos biométricos, la capa de recepción gestiona la comunicación entre el servidor central, a través de la capa de transmisión desarrollada para la instalación y actualización de aplicaciones desde y hacia el dispositivo móvil correspondiente.

La capa de comunicación tiene la misma estructura que la capa de comunicación del servidor, ésta se encarga de:

- La recepción de las medidas obtenidas de los dispositivos biométricos;
- La recepción de los datos enviado por la capa de trasmisión del servidor;
- Enlazar todos los servicios definidos para acceder a los datos que están almacenados en el servidor principal y que son gestionados por la capa de datos del servidor.

Es en esta capa donde se implementan las librerías necesarias para integrar algún tipo de tecnología de comunicación entre dispositivos; ya sea que la aplicación móvil trabaje con tecnología Bluetooth u otra tecnología de comunicación. En el caso de que quiera integrar otra tecnología de comunicación, sólo es necesario el cambio de la capa de comunicación del dispositivo móvil.

Capa de Seguridad

La capa de seguridad en el dispositivo móvil tiene la misma funcionalidad que en los otros dispositivos. La capa de seguridad del dispositivo móvil tiene las siguientes funciones:

- Es la encarga de desencriptar los paquetes de datos que son recibidos desde el servidor central, cuando es instalada la aplicación en el dispositivo móvil y los datos recibidos desde el dispositivo biométrico (en este caso las medidas).
- Asegurar la integridad de los datos recibidos y enviados desde y hacia el servidor central;
- Facilitar el control de acceso de los servicios web al servidor principal;
- Asegurar el autenticamiento desde el dispositivo móvil.

Capa de Aplicaciones

En esta capa se encuentran las aplicaciones generadas por el *framework* para la monitorización y autocontrol del paciente. Estas aplicaciones están basadas en las ontologías propias a cada enfermedad y al perfil individual de cada paciente. Además cuenta con módulos, que se actualizan cada vez que se genera una lectura desde el dispositivo biométrico. Este es el encargado de solicitar cambios a futuro en las aplicaciones previamente generadas para el dispositivo móvil.

Esta capa provee al usuario de una interfaz gráfica que le permite mantener una monitorización constante y continua del paciente incluyendo los parámetros médicos o señales vitales obtenidas de los dispositivos biométricos, así como controlar el proceso de monitorización.

Además, la capa de aplicación hace referencia a la organización y distribución de todos los módulos, que componen la capa de aplicación. Esta distribución permite la correcta ejecución de cada uno de los módulos, otorgándoles funcionalidades específicas dentro de la aplicación.

Además de las funciones mencionadas anteriormente, tenemos las siguientes:

- Distribuir los elementos de cada módulos;
- Establecer las relaciones existentes entre cada elemento;
- Ofrecer servicios de conexión interna en la aplicación;
- Ubicar en niveles, según el grado de interacción, cada uno de los elementos de la aplicación.

En la figura 4-9, se muestra la distribución de cada módulo dependiendo del nivel en el que han sido ubicados inicialmente. Esta distribución hace referencia a la ubicación en memoria de cada sub-módulo. Esta capa está compuesta por:

- **Nivel Superior**: en este nivel se ubica el módulo de medidas, que es el encargado de recibir las mediciones del paciente.
- **Nivel Intermedio**: en este nivel se ubican todos los demás módulos de la aplicación. Dentro de ella se ubican el perfil, las recomendaciones, prevención, alarmas, dietas, etc.

- **Nivel Bajo**: en este nivel de ubica el motor de búsqueda, encargado de la interpretación de la información actualizada, para pasársela a cada módulo.

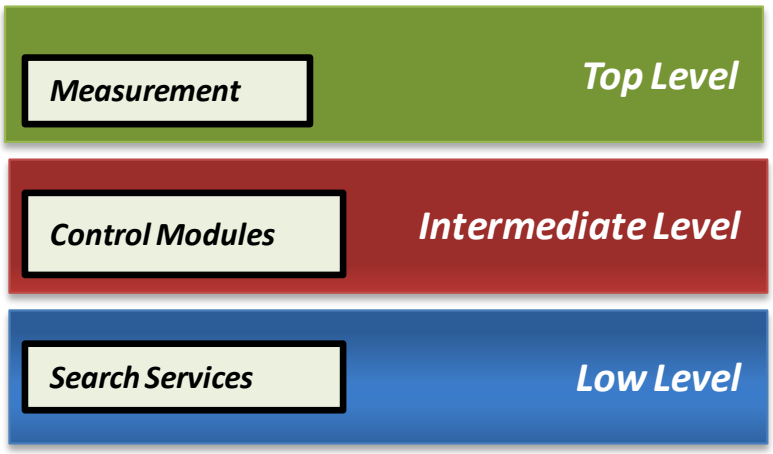


Figura 4-9. *Elementos que componen la capa de aplicación de los dispositivos móviles*

4.1.2.3 Capas de los Dispositivos Biométricos

Las capas que componen un dispositivo biométrico no forman parte del diseño de esta tesis, pero si vemos oportuno definir los posibles elementos que componen un dispositivo biométrico. Esto nos permite conocer la manera en que estos dispositivos trasmiten datos, obtenidos a partir de una solicitud hecha por el dispositivo móvil.

Otra ventaja de definir capas en los dispositivos biométricos, es que facilitan la adecuación de éstos a la arquitectura desarrollada, basados en la tecnología de comunicación con que cuenten. Para el desarrollo de la arquitectura software presentada en esta tesis utilizaremos *Bluetooth* para la recepción de datos desde el dispositivo biométrico, pero existen en el mercado otros dispositivos con tecnologías de comunicación diferente. Es aquí donde se puede observar más claramente la funcionalidad de dividir en capas los dispositivos biométricos.

Cada tecnología de comunicación tiene elementos de trasmisión/recepción de datos, aspectos de seguridad en esa trasmisión/recepción, elementos de negociación en donde se establecen los parámetros antes de iniciar la trasmisión/recepción y elementos de enlace. Estos elementos están claramente definidos en las capas que componen el dispositivo móvil, permitiendo relacionarlos entre sí. Esta comunicación facilita la apertura de canales de trasmisión/recepción basados en normas propias a cada tecnología de comunicación.

Tratamos con ello relacionar cada elemento del *framework* de tal manera que no sea necesaria la modificación posterior al momento en que un nuevo dispositivo entre a formar parte de la arquitectura. Es evidente que ante cambios más significativos esta funcionalidad tendría que ser rediseñada.

Podemos decir entonces que la distribución en capas de los dispositivos biométricos facilita la comunicación entre el dispositivo móvil y el dispositivo biométrico, a través de la tecnología

de comunicación con que cuente ambos dispositivos. En esta capa intervienen cuatro subcapas mostradas en la figura 4-10.

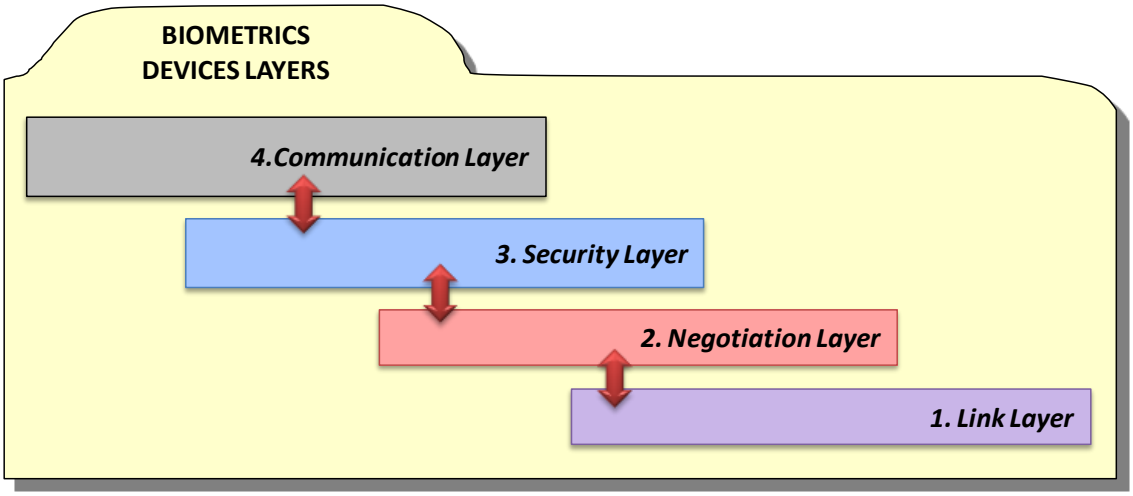


Figura 4-10. *Distribución de las capas de los dispositivos biométricos.*

Se ha definido cada elemento para facilitar posteriormente la integración de nuevas tecnologías de comunicación, en donde cada tecnología define un enlace entre dispositivos, una negociación para la transmisión y recepción de datos. Establece además, niveles de seguridad que aseguran la integridad de la transmisión; y por último, se define el comportamiento de la transmisión de datos.

Capa de Enlace

Esta capa se encarga de establecer el enlace inicial entre el dispositivo biométrico y el dispositivo móvil. Esta comunicación puede hacerse a través de cualquier tecnología de comunicación, bien propia del dispositivo hecho por su fabricante o en otro caso a través de una arquitectura desarrollado por otros.

La estructura de esta capa depende de la tecnología de comunicación que se utilice, establecida para cada una de ellas. Por ejemplo se puede utilizar comunicación Bluetooth, Infrarrojo, etc.

Capa de Negociación

Esta capa se encarga de negociar la comunicación entre el dispositivo biométrico y el dispositivo móvil. Es aquí en donde se abre un canal seguro de transmisión en donde el dispositivo móvil solicita los datos generados por el dispositivo biométrico.

De igual manera que en la capa de enlace, la estructura de esta capa depende de la tecnología de comunicación que se utilice, establecida para cada una de ellas.

Capa de Seguridad

Esta capa es la encargada de asegurar que los datos enviados a través de la tecnología de comunicación lleguen de manera fiable al dispositivo móvil. Generalmente esta capa viene integrada en los diferentes dispositivos biométricos desarrollados (glucómetros, tensiómetros, etc.). Corresponde al *framework* definir cómo captar los datos generados de una medición y asignárselo al módulo de medidas (encargado de interpretar cada medida recibida en tiempo real).

Cada tecnología de comunicación define sus protocolos de seguridad, lo que facilita la transmisión de la información entre cada uno de ellos.

La intervención en los dispositivos biométricos está limitada por el desarrollo propietario que trae cada una de ellas. Esto dificulta integrar nuevas funcionalidades a estos dispositivos, limitándonos a utilizar las que ya están integradas en ellas.

Capa de Transmisión

Luego de establecido el enlace y negociado el canal de transmisión, la capa de transmisión se encargará de enviar los datos medidos hacia el dispositivo móvil para su posterior interpretación.

El dispositivo móvil recibe la medición resultante, en forma de paquete, de tal manera que no pueda ser alterada en el camino. De igual manera que las capas anteriores, la capa de transmisión cuenta con sus propias librerías de trasmisión de datos, definidas por la tecnología de comunicación que ha sido integrada en ella.

4.1.3 Conclusiones

Se ha definido un modelo en capas, para facilitar el desarrollo y mantenimiento posterior de la aplicación. Se han identificado cada una de las partes asociando cada capa según la funcionalidad en el *framework* en general.

Además de facilitar el desarrollo de los elementos del *framework*, la distribución en capas facilita la identificación de lugares concretos donde se necesitan cambios al momento de migrar la arquitectura a otros entornos. Se facilita también la reutilización de todos sus componentes, fácilmente identificables.

El diseño de capas presentada es un aporte al diseño estandarizado y organizado, ofreciéndole más robustez e integridad a la arquitectura a desarrollar a partir del *framework*.

4.2 MOMONTOLOGY: DISEÑO DE ONTOLOGÍAS PARA LA MONITORIZACIÓN MÓVIL DE PACIENTES.

4.2.1 Introducción

Luego de un análisis funcional de las ventajas que ofrece cada metodología de diseño de ontologías del capítulo dos, se ha elegido **METHONTOLOGY** como el enfoque metodológico utilizado para la construcción de las ontologías. Entre las razones de la elección se cuentan su

utilización en la construcción de ontologías en distintos ámbitos (Fernandez-Lopez, 1999), la disponibilidad de documentación y por ser la metodología recomendada por la "*Fundación para los agentes Físicos Inteligentes*" (**FIPA**[8]), la cual promueve la interoperabilidad entre las aplicaciones basadas en agentes.

METHONTOLOGY tiene sus raíces en las actividades identificadas por el proceso de desarrollo de *software* propuesto por la IEEE y en otras metodologías de ingeniería de conocimientos. A continuación se describe brevemente cada una de estas actividades:

- *Especificación* permite determinar por qué se construye la ontología, cuál será su uso, y quiénes serán sus usuarios finales.

- *Conceptualización* se encarga de organizar y convertir una percepción informal del dominio en una especificación semi-formal, para lo cual utiliza un conjunto de representaciones intermedias, basadas en notaciones tabulares y gráficas, que pueden ser fácilmente comprendidas por los expertos de dominio y los desarrolladores de ontologías. El resultado de esta actividad es el modelo conceptual de la ontología.

-*Formalización* es la actividad encargada de la transformación de dicho modelo conceptual en un modelo formal o semi-computable.

-*Implementación* construye modelos computables en un lenguaje de ontologías (Ontolingua, RDF Schema, OWL, etc.). La mayor parte de las herramientas de ontologías permiten llevar a cabo esta actividad de manera automática.

- *Mantenimiento* se encarga de la actualización y/o corrección de la ontología.

METHONTOLOGY también propone actividades de gestión (*planificación, control y aseguramiento de la calidad*), y de soporte (*adquisición de conocimientos, integración, evaluación, documentación y gestión de la configuración*).

4.2.2 Clasificación de MoMOntology y sus elementos

Basándonos en los pasos que conforman el desarrollo de ontologías definidos por **METHONTOLOGY** y para una mejor comprensión de cada uno de los elementos del *framework*, se presenta una clasificación ontológica llamada "**MoMOntology**" que comprenden la definición de los elementos *perfil de paciente*, *definición de módulos, definición de enfermedades* y *definición de dispositivos móviles*, lo que nos permite una visión más profunda de cada una de las funcionalidades que la componen. Además, las ontologías son interpretadas por los módulos para permitir la interoperabilidad entre cada uno de ellos y la aplicación final. En la figura 4-11 se muestra la clasificación del diagrama inicial que demuestra la relación de cada uno de sus componentes.

Mediante esta clasificación, cada uno de los elementos que componen la definición de un determinado módulo está relacionado con la definición inicial del perfil del paciente. Esa definición de módulos, nos permite generar la estructura de la aplicación tanto para el doctor

[8] http://www.fipa.org/

como para el paciente, con base a cada uno de esos patrones y relaciones de la estructura de definición de módulos.

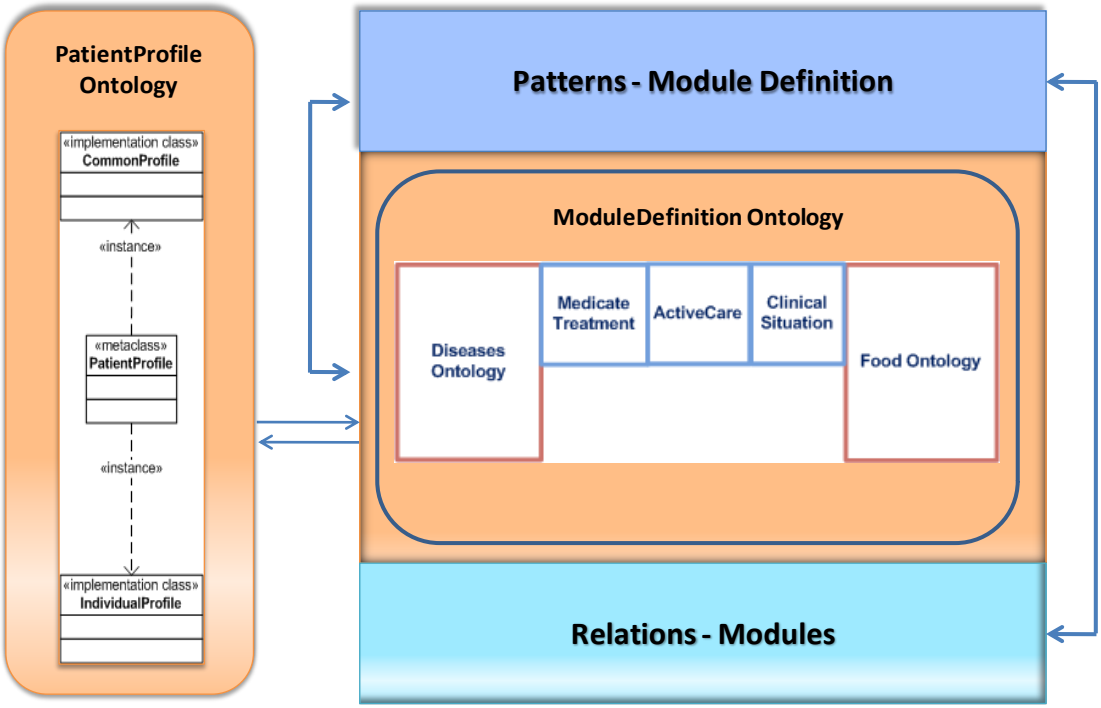


Figura 4-11. *Estructura ontológica del diagrama de framework inicial*

Es por ello que se ha desarrollado una clasificación ontológica que nos permita la extracción del conocimiento necesario para el *framework* (Villarreal, Hervás et al., 2009). En la figura 4-12, se establece la relación que existe entre cada una de las ontologías propuestas y los patrones y relaciones que nos permitan la generación de aplicaciones en la monitorización móvil de pacientes, dichos patrones serán definidos posteriormente.

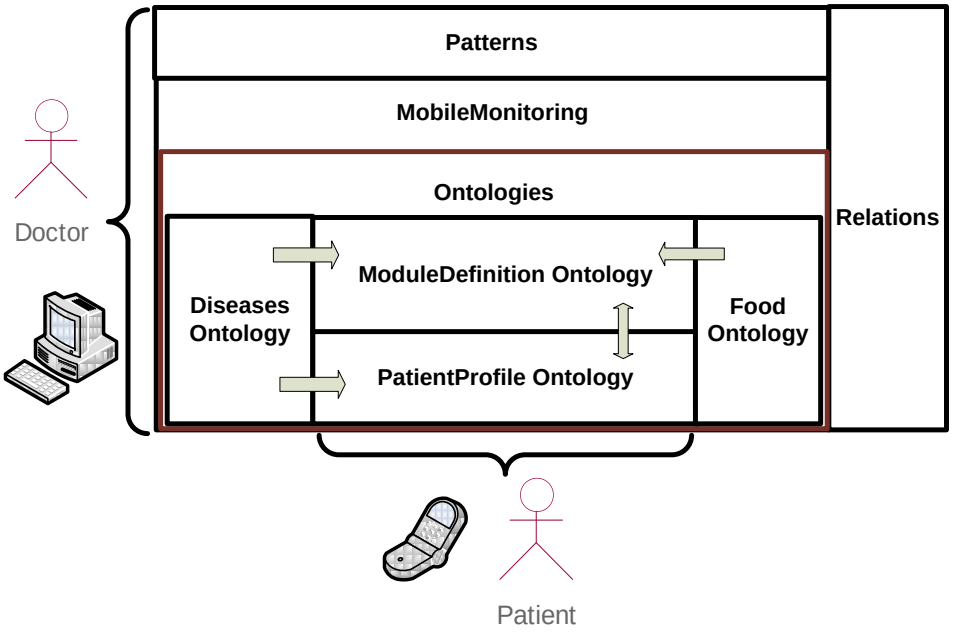


Figura 4-12. *Interacción entra las ontologías y los elementos del framework.*

Existe una ontología **Enfermedades** (*Diseases*) y una ontología **Alimentos** (*Food*) que se relacionan con las ontologías principales (*ModuleDefinition y PatientProfile Ontologies*).

En la figura 4-13, se especifican los elementos que forman las ontologías **Perfil de Paciente** (*PatientProfile*) y **Definición de Módulos** (*ModuleDefinition*). La primera está compuesta por dos elementos principales: la definición de un **Perfil Común** (*CommonProfile*) y un **Perfil Individual** (*IndividualProfile*) de cada paciente. Esta clasificación nos permite generar conceptos de niveles más especializados y establecer los mecanismos de relación entre cada uno de ellos a la hora de definir los módulos asociados a un determinado perfil de paciente.

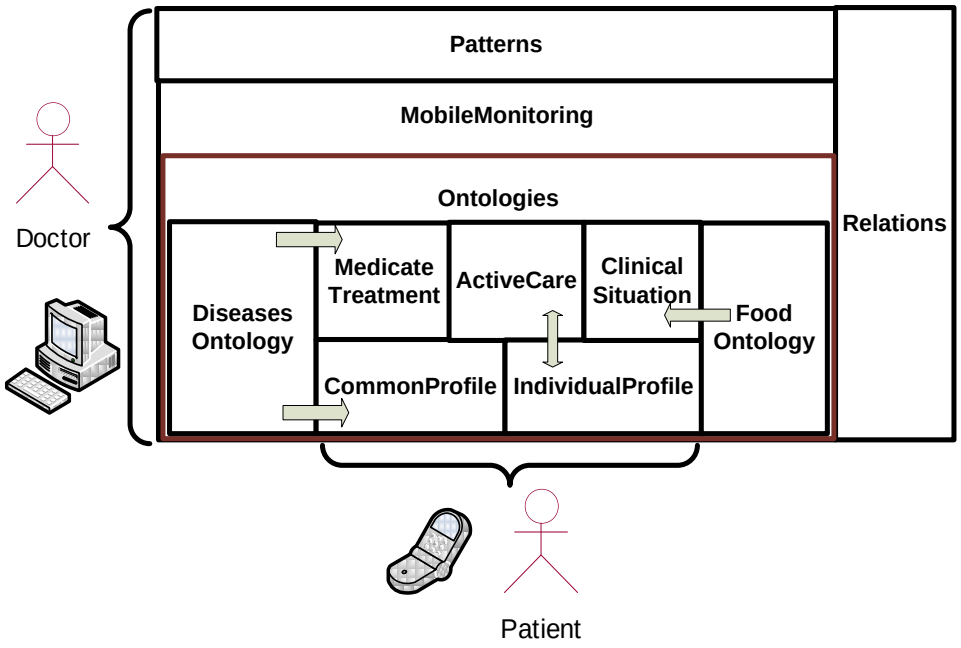


Figura 4-13. *Especificación de las ontologías de perfil de paciente y definición de módulos.*

Se muestra la distribución de cada uno de los elementos que intervienen en la utilización y desarrollo de esta arquitectura, así como la descripción ontológica de los ejes centrales de nuestro diseño. La comunicación entre el doctor (*doctor*), el paciente (*patient*), el dispositivo o sensor (*sensor*) y la propia monitorización móvil (*MobileMonitoring*) se presentan en relación a las ontologías de definición de módulos (*ModuleDefinition*), definición de perfil de paciente (*PatientProfile*), enfermedades (*Diseases*) y alimentos o comidas (*Food*)

Se describe una ontología de alto nivel de abstracción que representa la estructura ontológica diseñada para apoyo al *framework*, en cuya estructura se destacan los siguientes elementos:

a) **Monitoring**: se define en (Kellogg, 2008) como el control de las constantes vitales de un paciente a través de monitores. Para nuestra arquitectura nos centraremos en la monitorización de pacientes a través dispositivos móviles.
b) **MobileMonitoring**: Define los elementos que intervienen en la monitorización móvil de pacientes que padecen una o más enfermedades. Esta arquitectura está compuesta por tres entidades externas: *dispositivos biométricos* (sensor), *paciente* y *doctor*. La

monitorización móvil permite el seguimiento y control de pacientes bajo la supervisión de un doctor, a través del dispositivo móvil y tecnologías de comunicación.

En la figura 4-14 y figura 4-15, se puede observar la clasificación y organización de cada unas de las ontologías, definidas por **MoMOntology**.

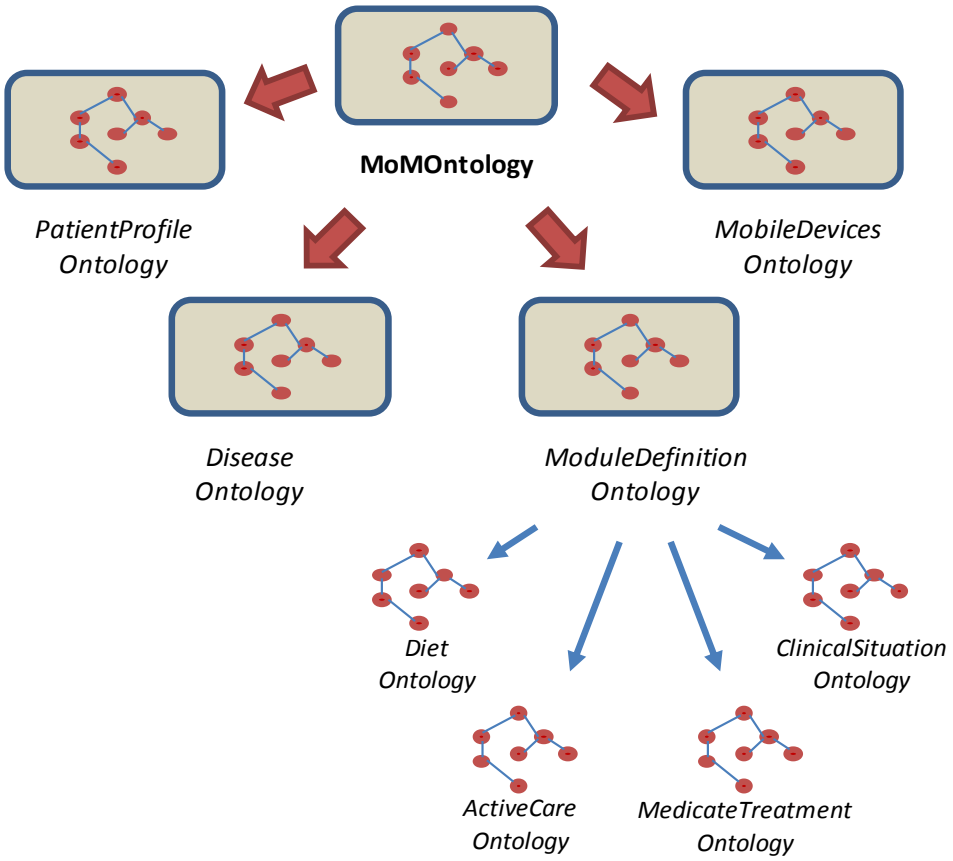


Figura 4-14. *MoMOntology y la relación con las ontologías desarrolladas.*

c) **Entities**: se han descrito las entidades que intervienen en el proceso de monitorización móvil de pacientes y que interactúan con el *framework*. Estas son:

- **Doctor**: es aquella persona que se encarga de las actividades médicas (tratamientos, diagnostico, evaluación, etc.) de un paciente. Dentro de la arquitectura es aquél que define el perfil del paciente y asigna tratamientos y actividades a seguir.
- **Patient**: es aquella persona que padece de alguna enfermedad. Además en esta arquitectura es la persona que suministra información personalizada sobre una determinada enfermedad.
- **Dispositivo Biométrico (Sensor)**: cualquier dispositivo que captura medidas vitales de un paciente y los envía al teléfono móvil para su correcta interpretación. Dentro de la arquitectura propuesta se tiene el dispositivo biométrico, que obtiene medidas de alguna constante vital del paciente (por ejemplo, glucosa y tensión arterial, obtenida a través de un glucómetro y tensiómetro respectivamente) y el teléfono móvil, que captura la información (a través de NFC, Bluetooth, etc.) y procesa e interpreta los valores de esas mediante una aplicación embebida en el

dispositivo. Además se transmitirán esos mismos datos a la aplicación del médico (a través de Wi-Fi GPRS).

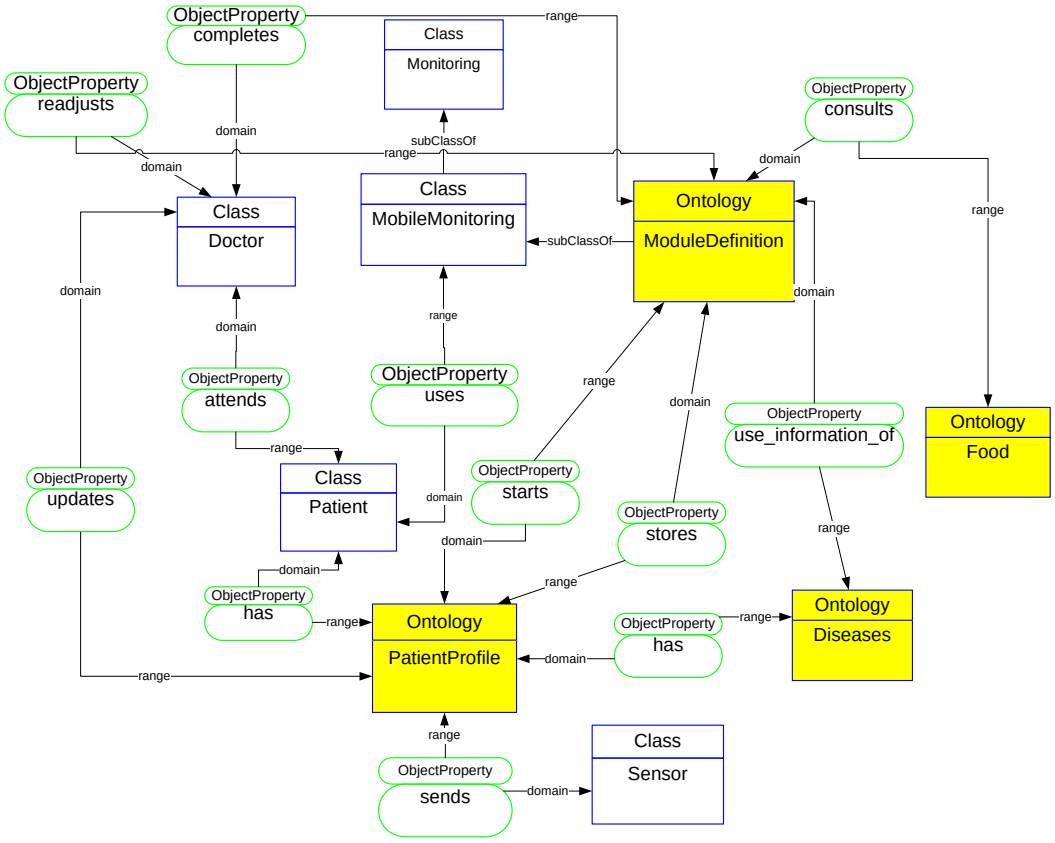


Figura 4-15. *Diagrama general de ontologías para la monitorización móvil de pacientes.*

En la tabla 4-1, se define y explican las clases que componen la ontología de alto nivel.

Clase	Ontología	Conceptos relacionados	Valores
MobileMontoring	Upper	ModuleDefinition, PatientProfile.	Cualquier referencia RDF válida
	Descripción: Ontología que define la organización más abstracta del proceso de monitorización.		
PatientPofile	Upper	Diseases, sensor, Doctor ModuleDefinition, CommonProfile, IndividualProfile, Prevention.	Cualquier referencia RDF válida
	Descripción: Ontología que componen la información básica de un paciente.		
ModuleDefinition	Upper	Doctor, PatientProfile, Diseases, Food, ActiveCare, MedicateTreatment, ClinicalSituation.	Cualquier referencia RDF válida
	Descripción: Ontología que permite la definición de módulos para la monitorización		
Diseases	Upper	PatientProfile, ModuleDefinition,	Cualquier referencia RDF válida

		ClinicalSituation, MedicateTreatment.	
	Descripción: Ontología que define los tipos de enfermedades a monitorizar.		
Food	Upper	ModuleDefinition.	Cualquier referencia RDF válida
	Descripción: Ontología que permite la clasificación de los alimentos para consumir.		
Doctor	Entidad externa	Patient, PatientProfile, ModuleDefinition.	Cualquier referencia RDF válida
	Descripción: Actor externo que interactúa con arquitectura bajo el rol de doctor.		
Patient	Entidad externa	PatientProfile, Doctor, MobileMonitoring.	Cualquier referencia RDF válida
	Descripción: Actor externo que interactúa con arquitectura bajo el rol de paciente.		
Sensor	Entidad externa	PatientProfile.	Cualquier referencia RDF válida
	Descripción: Dispositivo externo que interactúa con arquitectura.		

Tabla 4-1. *Clases de la ontología de alto nivel.*

En la tabla 4-2, se define y explican las propiedades que componen la ontología de alto nivel.

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
uses	ObjectProp	Patient, Sensor	MobileMonitoring, FormatVisualization	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen se sirve de un miembro de la clase destino.			
attends	ObjectProp	Doctor	Patient	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen atiende a un miembro de la clase destino.			
has	ObjectProp	Patient, PatientProfile.	PatientProfile, Diseases,	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen tiene un elemento de la clase destino			
updates	ObjectProp	Doctor	PatientProfile	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen actualiza o modifica un miembro de la clase destino			
sends	ObjectProp	Device, Measure.	PatientProfile, Sensor.	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen envía un elemento a un miembro de la clase destino			
completes	ObjectProp	Doctor	ModuleDefinition	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase completa o rellena elementos de un miembro de la clase destino			
readjusts	ObjectProp	Doctor	ModuleDefinition	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen reajusta o redistribuye algún elemento de un miembro de la clase destino			

consults	ObjectProp	ModuleDefinition	Food	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen consulta información de un miembro de la clase destino			
use_ information_ of	ObjectProp	ModuleDefinition	Diseases	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen obtiene información de un miembro de la clase destino			
stores	ObjectProp	ModuleDefinition	PatientProfile	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen almacena información de un miembro de la clase destino			
starts	ObjectProp	PatientProfile	ModuleDefinition	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen inicializa a un miembro de la clase destino			

Tabla 4-2. *Propiedades de la ontología de alto nivel.*

4.2.2.1 Estructura Ontológica del Perfil de Paciente (PatientProfile)

PatientProfile: define las la información de cada paciente y está compuesta por el *CommonProfile* y el *IndividualProfile*. Como se muestra en la figura 4-16, se ha desarrollado una estructura de clasificación ontológica del perfil de cada paciente.

CommonProfile: Este perfil almacena toda la información común de paciente. Esta información está definida por sus datos personales (*PersonalData*), donde se registra el *nombre*, *dirección*, *fecha de nacimiento* y *sexo* de cada paciente.

IndividualProfile: A diferencia del perfil común, el perfil individual (*IndividualProfile*) cuenta con información médica asociada a cada uno de los pacientes. Además tiene un historial donde se almacenan las medidas (*Measure*) y tendencias (*Trend*) obtenidas de los dispositivos sensoriales (teléfono móvil, glucómetro, tensiómetro, etc.). Como se muestra en la figura 4-16, cada medida o tendencia tiene asociada una actividad que el paciente ha desarrollado en un momento determinado, es por ello que, según el tipo de actividad que esté desarrollando, sus valores en medidas y tendencias varían continuamente (existe la posibilidad que los valores estén por encima o debajo de los valores permitidos según una enfermedad).

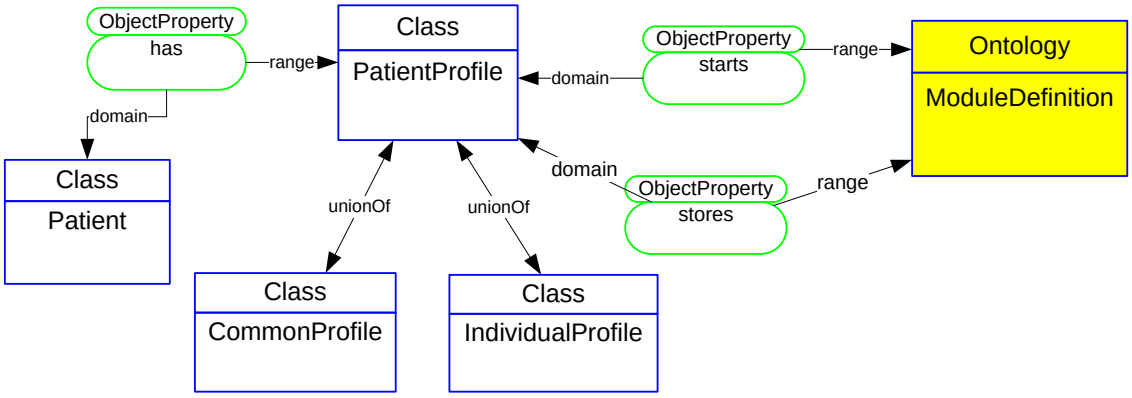


Figura 4-16. *Diagrama Ontológico general del Perfil del Paciente.*

En la tabla 4-3, se define y explican las clases que componen la ontología de perfil de paciente (PatientProfile).

Clase	Ontología	Conceptos relacionados	Valores
CommonProfile	PatientProfile	PatientProfile	Cualquier referencia RDF válida
	Descripción: Elemento que define la información generalizada para los pacientes		
IndividualProfile	PatientProfile	PatientProfile	Cualquier referencia RDF válida
	Descripción: Elemento que define la información específica a cada paciente		

Tabla 4-3. *Clases de la ontología de perfil de paciente (PatientProfile).*

Las aplicaciones generadas por el *framework* deben ser capaces de interpretar estas medidas y permite la visualización de datos en formatos variados (gráficos, tablas, texto, etc.) para el autocontrol del paciente.

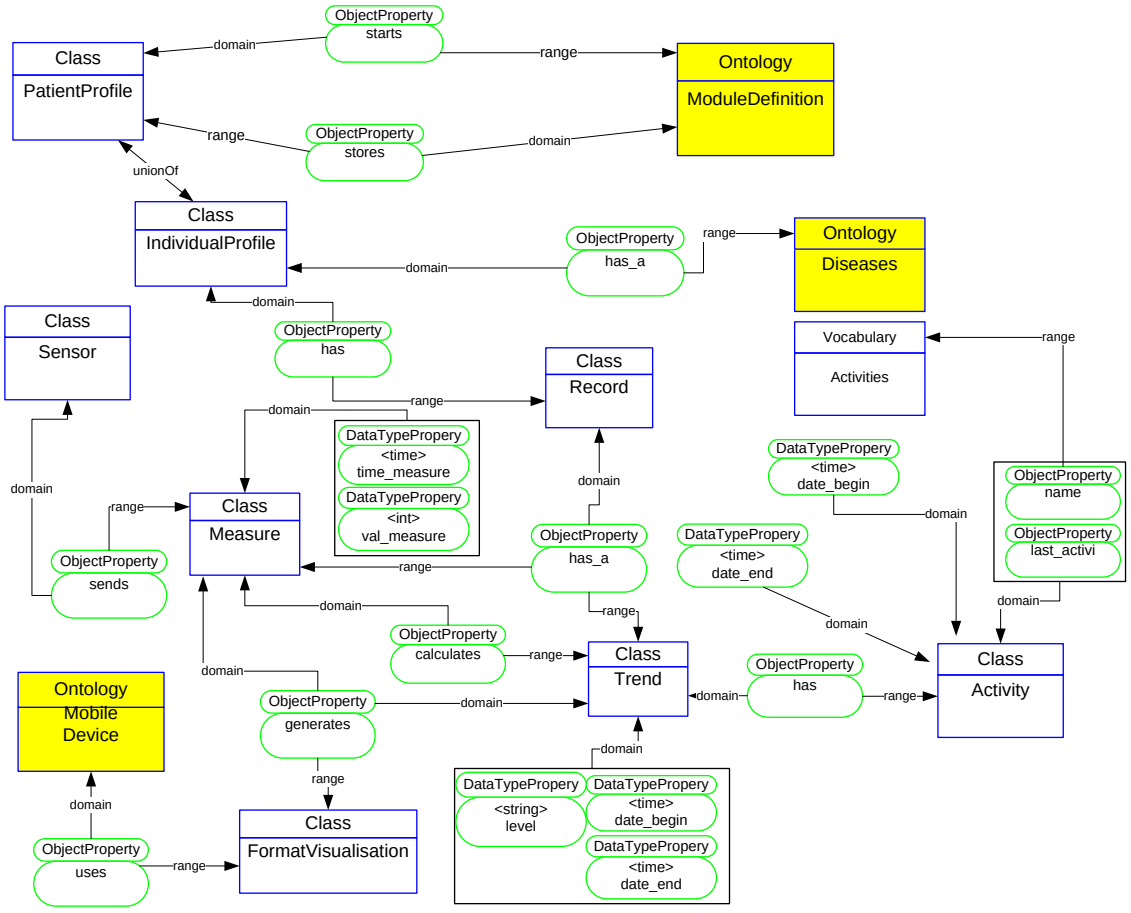


Figura 4-17. *Diagrama ontológico de la definición de perfil individual del paciente.*

En la tabla 4-4, se define y explican las clases que componen la ontología de perfil individual de paciente (IndividualProfile).

Clase	Ontología	Conceptos relacionados	Valores
Record	IndividualProfile	Trend, Measure	Cualquier referencia RDF válida
	Descripción: Elemento que almacena los registros de cada paciente		
Measure	IndividualProfile	Record, Trend, FormatVisualization, Sensor,	Cualquier referencia RDF válida
	Descripción: Elemento que almacena las medidas obtenidas de cada paciente.		
FormatVisualization	IndividualProfile	Measure, trend, Device	Cualquier referencia RDF válida
	Descripción: Elemento que define la salida de la información.		
Trend	IndividualProfile	Record, measure, FormatVisualization, Activity,	Cualquier referencia RDF válida
	Descripción: Elemento que define la tendencia de las medidas obtenidas según el rango de distribución.		
Activity	IndividualProfile	Trend	Cualquier referencia RDF válida
	Descripción: Elemento que almacena la actividades realizadas por el paciente.		
MobileDevice	Upper	FormatVisualization	Cualquier dispositivo que interactúa con la aplicación
	Descripción: Ontología que define las características de un dispositivo móvil.		
Activities	IndividualProfile	Activity	Cualquier referencia RDF válida
	Descripción: Vocabulario que define un conjunto de actividades a desarrollar por un paciente.		

Tabla 4-4. *Clases de la ontología de perfil individual de paciente (IndividualProfile).*

En la tabla 4-5, se define y explican las propiedades que componen la ontología de perfil de individual de paciente (IndividualProfile).

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
generates	ObjectProp	Measure	FormatVisualization	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen crea o genera un miembro de la clase destino.			
calculates	ObjectProp	Measure	Trend	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen obtiene o calcula un valor a un miembro de la clase destino			
date_end	DataTypeProp	Activity, Trend	time	[0 - 24 horas]
	Descripción: Tipo de dato que define el tiempo de finalización de las actividades y la obtención de una tendencia en el tiempo.			
date_begin	DataTypeProp	Activity, Trend	time	[0 – 24 horas]
	Descripción: Tipo de dato que define el tiempo de inicio de las actividades y la obtención de una tendencia en el tiempo.			

level	DataTypeProp	Trend.	string	[low, mediun, high]
	Descripción: Tipo de dato que define el rango de una tendencia obtenida.			
time_measure	DataTypeProp	Measure	time	[0 – 24 horas]
	Descripción: Tipo de dato que define la hora de obtención una medida.			
val_measure	DataTypeProp	Measure	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que define el valor exacto de la medida obtenida.			

Tabla 4-5. *Propiedades de la ontología de perfil individual de paciente (IndividualProfile).*

4.2.2.2 Estructura Ontológica de la Definición de Enfermedades (Diseases)

Diseases: define un rango de clasificación de enfermedades en las cuales nuestro *framework* se puede aplicar. Para este caso y según se muestra en la figura 4-18, se ha desarrollado una ontología de clasificación de enfermedades según criterios de agrupación.

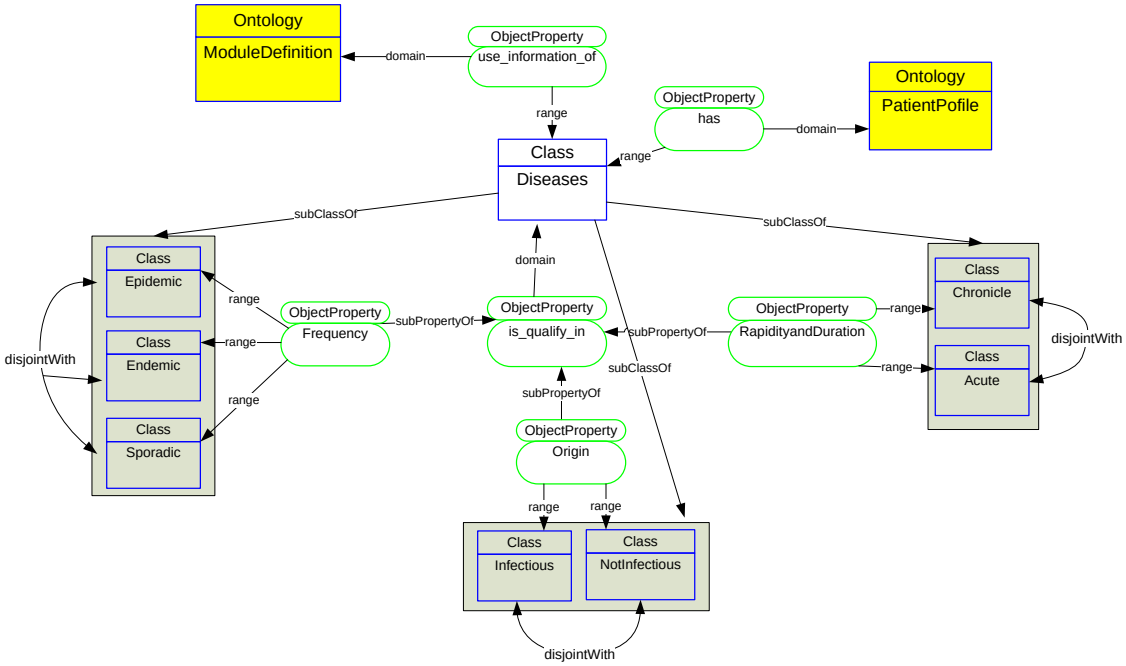


Figura 4-18. *Diagrama ontológico de la definición de enfermedades.*

Estos criterios se han agrupado de la siguiente manera:

a) Por la rapidez con la que aparecen y por su duración (*RapidityandDuration*):
   - **Agudas** (*Acute*),
   - **Crónicas** (*Chronicle*).

b) Por la frecuencia con que se producen (Frequency):
   - **Esporádicas** (*Sporadic*),
   - **Endémicas** (*Endemic*),
   - **Epidémicas** (*Epidemic*).

c) Por su origen (*Origin*):
   - **Infecciosa** (*Infectious*),
   - **No Infecciosa** (*NotInfectious*).

Con el fin de delimitar nuestro campo de estudio, hemos hecho una clasificación de las enfermedades crónicas. Esto nos permite evaluar la clasificación propuesta en una enfermedad específica y además extenderla a otras enfermedades existentes. El modelo queda abierto a todas las enfermedades con que el *framework* vaya a interactuar.

En la tabla 4-6, se define y explican las clases que componen la ontología de enfermedades (Diseases).

Clase	Ontología	Conceptos relacionados	Valores
Chronicle	Diseases	RapidityandDuration	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad crónica.		
Acute	Diseases	RapidityandDuration	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad aguda.		
NotInfectious	Diseases	Origin	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad no infecciosa.		
Infectious	Diseases	Origin	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad no infecciosa.		
Endemic	Diseases	Frequency	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad endémica.		
Epidemic	Diseases	Frequency	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad epidémica.		
Sporadic	Diseases	Frequency	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características de una enfermedad esporádica.		
Diseases	Upper	PatientProfile, ModuleDefinition, Frequency, Origin, RapidityandDuration	Cualquier referencia RDF válida.
	Descripción: Elemento que define las características básica de una enfermedad		

Tabla 4-6. *Clases de la ontología de enfermedades (Diseases).*

En la tabla 4-7, se define y explican las propiedades que componen la ontología de enfermedades (Diseases).

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
use_information_of	ObjectProp	ModuleDefinition	Diseases	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen se sirve de un miembro de la clase destino.			
Is_qualify_in	ObjectProp	Diseases	Frequency Origin RapidityandDuration	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen se clasifica en algún miembro de la clase destino			
Frequency	subPropOf	Diseases	Epidemic Endemic Sporadic	Cualquier referencia RDF válida
	Descripción: Subpropiedad que define las enfermedades según la frecuencia con la que aparecen.			
Origin	subPropOf	Diseases	Infectious NotInfectious	Cualquier referencia RDF válida
	Descripción: Subpropiedad que define las enfermedades según su origen.			
Rapidityand Duration	subPropOf	Diseases	Chronicle Acute	Cualquier referencia RDF válida
	Descripción: Subpropiedad que define las enfermedades según la rapidez y la duración con la que aparecen.			

Tabla 4-7. *Propiedades de la ontología de enfermedades (Diseases).*

4.2.2.3 Estructura Ontológica de la Definición de Módulos (ModuleDefinition)

ModuleDefinition: Elementos generados a partir del perfil de cada paciente. Como se muestra en la figura 4-19, contiene la siguiente información:

- **Actividades de cuidado (***ActivitiesCare***)** se encarga de la manipulación de los siguientes módulos: *Prevention*, *Education*, *SelfControl*, *Suggestion* and *Diet*, definidos para una patología en particular para cada paciente.
- **Situación Clínica (***ClinicalSituation***)** define las características propias a una enfermedad (*diseases ontology*), y establece controles para la interpretación de datos obtenidos de los diferentes sensores biométricos.
- **Tratamiento Médico (***MedicateTreatment***)** los tratamientos pueden ser de actividades (*ofActivities*) y farmacológicos (*Pharmacology*) siempre prescritos por un especialista. Éstos últimos se pueden suministrar vía oral (*OralMedicate*) o vía inyectables (*InjectableMedicate*) dependiendo de cada enfermedad. Estos conceptos se podrían ampliar para nuevos tipos de enfermedades y/o necesidades.

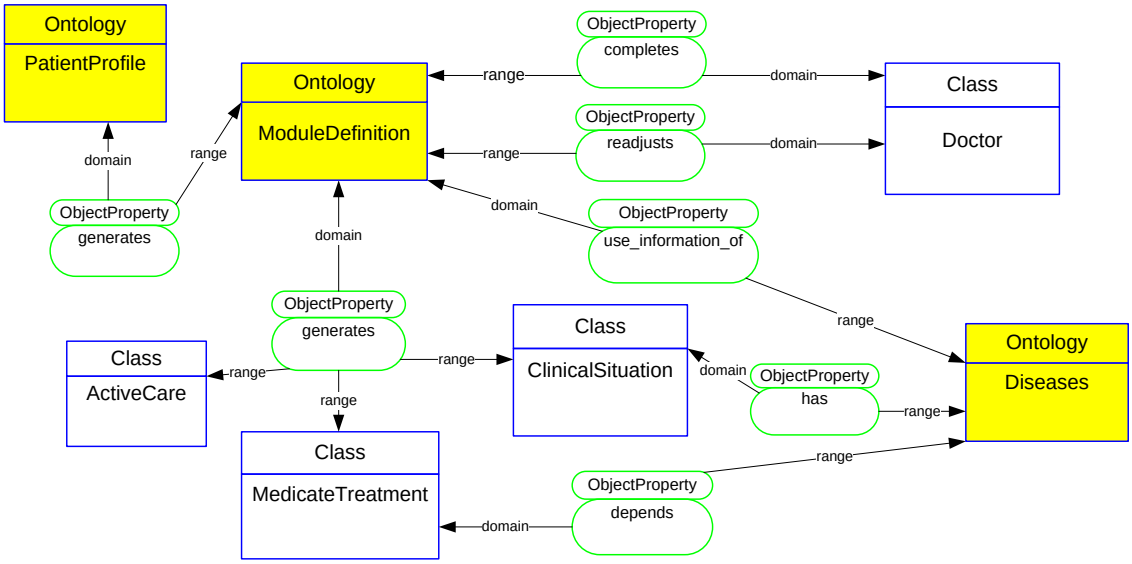


Figura 4-19. *Diagrama ontológico general para la definición de módulos.*

En la tabla 4-8, se define y explican las clases que componen la ontología de definición de módulos (ModuleDefinition).

Clase	Ontología	Conceptos relacionados	Valores
ActiveControl	ModuleDefinition	ModuleDefinition	Cualquier referencia RDF válida
	Descripción: Elemento que define las actividades de control para un paciente.		
MedicateTreatment	ModuleDefinition	ModuleDefinition, Diseases, Suggestion, ofActivities, Pharmacology.	Cualquier referencia RDF válida
	Descripción: Elemento que define el tratamiento médico a seguir por un paciente.		
ClinicalSituation	ModuleDefinition	ModuleDefinition, Diseases.	Cualquier individuo que utiliza la aplicación
	Descripción: Elemento que define las características básicas de la enfermedad que padece un paciente.		

Tabla 4-8. *Clases de la ontología de definición de módulos (ModuleDefinition).*

En la tabla 4-9, se define y explican las propiedades que componen la ontología de definición de módulos (ModuleDefinition).

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
depends	ObjectProp	MedicateTreatment	Diseases	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen depende de un miembro de la clase destino.			

Tabla 4-9. *Propiedades de la ontología de definición de módulos (ModuleDefinition).*

Definición de las actividades de Cuidado (ActiveCare)

En la figura 4-20, se especifican los elementos que intervienen en las **Actividades de Cuidado** (*ActiveCare*). Una actividad de cuidado no es más que aquellas actividades que el médico recomienda y que son detalladas en la generación de módulos del *framework*.

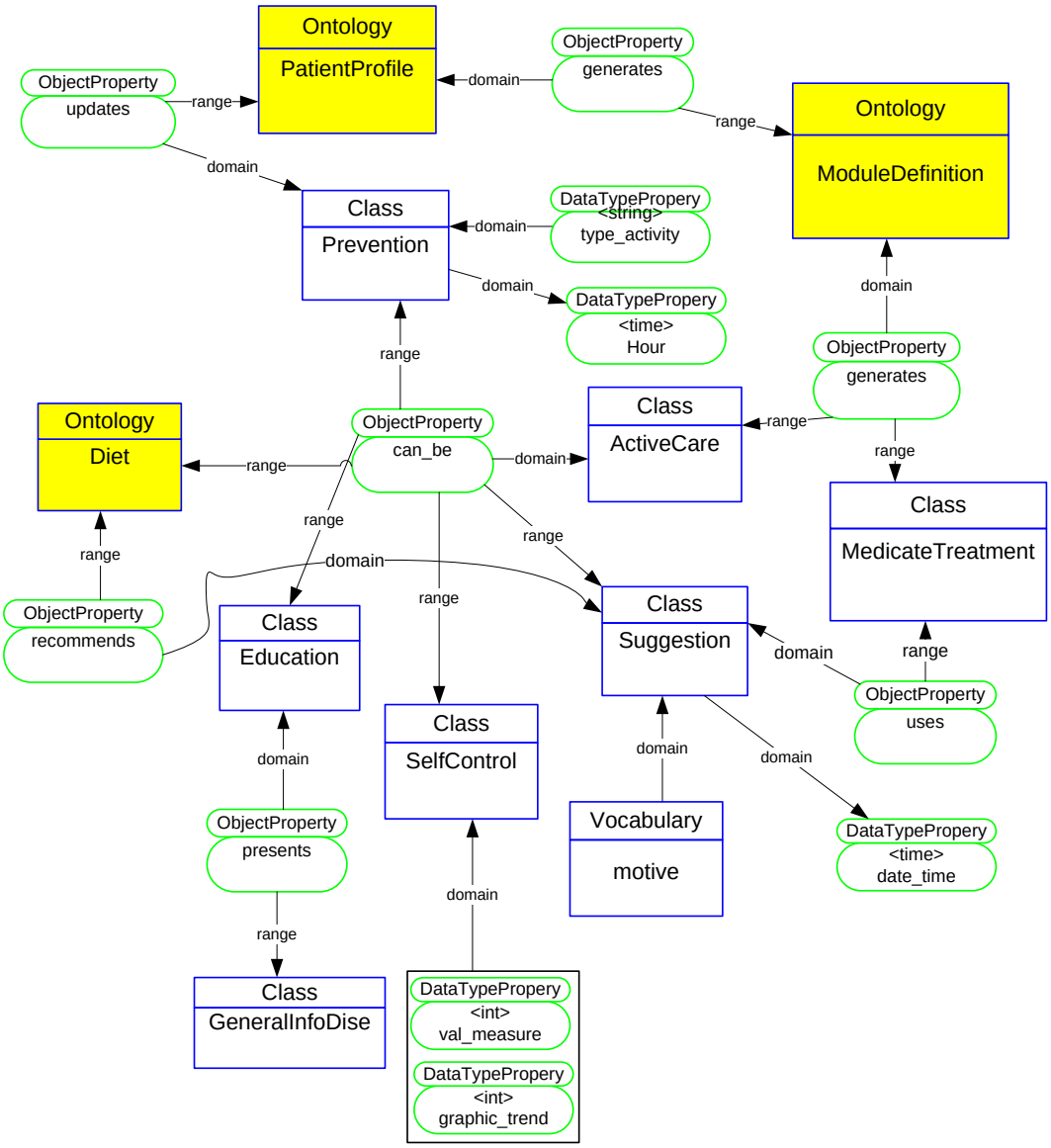


Figura 4-20. *Diagrama ontológico de la definición del Módulo ActiveCare.*

Estas actividades nos permiten la definición futura de los módulos ofrecidos a un paciente para su monitorización. Están compuestas por:

- **Prevención** (*Prevention*) el cual trabaja en función de un horario de actividad del paciente, y en función del aprendizaje del propio sistema en cuanto al comportamiento del paciente en ciertas actividades. Se establece una relación entre las medidas, tendencias y la ejecución de una actividad.
- **Educación (Education)** en donde se le ofrece al paciente información adicional a la enfermedad que padece. Esto no estará asociado a ninguna medida o tendencia

obtenida del sensor biométrico, sino que funciona como un valor agregado a nuestro *framework*.

- **Auto-control (SelfControl)** obtiene información del historial de medidas y tendencias de un paciente almacenada en el perfil individual. Aquí podrá visualizar información por medio de gráfico, tablas, y otros formatos de salida en donde pueda conocer las variaciones de medidas obtenidos de algún dispositivo sensor (glucómetro, tensiómetro, etc.).
- **Recomendación (Suggestion)** el sistema ofrece al paciente un conjunto de recomendaciones que pueden ser presentadas dependiendo ya sea de una actividad desarrollada por el paciente o por variaciones en las medidas obtenidas en un momento determinado. Las recomendaciones pueden presentar una dieta a seguir, pueden usar un tratamiento médico propuesto, entre otras.

En la tabla 4-10, se define y explican las clases que componen la ontología de actividades de cuidado (ActiveCare).

Clase	Ontología	Conceptos relacionados	Valores
Diet	Upper	Suggestion, ActiveCare, RestrictedFood, ForbiddenFood, AdvisableFood.	Cualquier referencia RDF válida.
	Descripción: Ontología que define la dieta para un paciente.		
ActiveCare	ModuleDefinition	ModuleDefinition, Prevention, Diet, Education, SelfControl, Suggestion.	Cualquier referencia RDF válida.
	Descripción: Elemento que define las actividades de cuidado del paciente.		
Prevention	ActiveCare	ActiveCare, PatientProfile	Cualquier referencia RDF válida.
	Descripción: Elemento que define las reglas de prevención de un paciente.		
Education	ActiveCare	ActiveCare, GeneralInfDise	Cualquier individuo que utiliza la aplicación.
	Descripción: Elemento que define los parámetros generales de una enfermedad.		
SelfControl	ActiveCare	ActiveCare	Cualquier referencia RDF válida.
	Descripción: Elemento que define el autocontrol del paciente, según su enfermedad.		
Suggestion	ActiveCare	ActiveCare, Diet MedicateTreatment,	Cualquier referencia RDF válida.
	Descripción: Elemento que define los pasos a seguir para el control de una enfermedad.		
GeneralInfoDise	Education	Education	Cualquier referencia RDF válida.
	Descripción: Elemento que define las información básica de cada enfermedad.		
Motive	Suggestion	No definido	Cualquier referencia RDF válida.
	Descripción: Elemento que define el motivo de una recomendación.		

Tabla 4-10. *Clases de la ontología de actividades de cuidado (ActiveCare).*

En la tabla 4-11, se define y explican las propiedades que componen la ontología de actividades de cuidado (ActiveCare).

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
presents	ObjectProp	Education	GeneralInfoDise	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen tiene un elemento de la clase destino			
can_be	ObjectProp	ActiveCare	Prevention, Education, SelfControl, Suggestion	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase puede ser un miembro de la clase destino			
recommends	ObjectProp	Suggestion	Diet	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase genera un recomendación a un miembro de la clase destino			
hour	DataTypeProp	Prevention	time	time
	Descripción: Tipo de dato que hace referencia a la hora en que se ha desarrollado una actividad.			
date_time	DataTypeProp	Suggestion	time	[0 – 24 horas]
	Descripción: Tipo de dato que hace referencia a la hora en que se ha generado una recomendación.			
grapgic_trend	DataTypeProp	SelfControl	int	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase genera un recomendación a un miembro de la clase destino			
type_activity	DataTypeProp	Prevention	string	[corta duración, mediana duración, larga duración]
	Descripción: Relación en la que un miembro de la clase genera un recomendación a un miembro de la clase destino			

Tabla 4-11. *Propiedades de la ontología de actividades de cuidado (ActiveCare)*

Definición de la Situación Clínica (ClinicSituation)

La **Situación Clínica** (*ClinicSituation*) define las características propias de una enfermedad, aquí se hace una calificación teórica de los conceptos asociados a ella, y se establecen los controles para la interpretación de datos o valores según el grado del padecimiento.

Asociamos la interpretación de datos, ya que según las características de un padecimiento, podemos conocer que tipos de actividades de cuidado podemos asociar, según el valor de las medidas resultantes de esa interpretación. La situación clínica depende directamente de la ontología enfermedad (*Diseases*) en donde se han definido con anterioridad cada una de las características propias a esa enfermedad.

Definición del Tratamiento Médico (MedicateTreatment)

El **tratamiento médico** (*MedicateTreatment*) está especificado por las actividades (*ofActivities*) y la farmacología (*Pharmacology*) que el médico recomienda a sus pacientes. Este

tratamiento depende inicialmente de la enfermedad que el paciente padece y luego del comportamiento de dicha enfermedad en las actividades que este desarrolla.

Una **recomendación** (*suggestion*) hace uso de esta sección en cuanto a las *actividades* y *medicamentos* que debe ingerir un paciente según avance la enfermedad. El médico recomienda una **actividad** (*ofactivities*) que debe desarrollar el paciente con una frecuencia y tiempo establecido. Además el médico recomienda el uso de fármacos los cuales pueden ser de prescripción **oral** (*OralMedicate*) o **inyectable** (*InjectableMedicate*) cada uno de ellos con su respectivo nombre, el tipo (si es el caso que exista más de uno) y la dosis. En la figura 4-21 se especifica la relación entre cada uno de ellos.

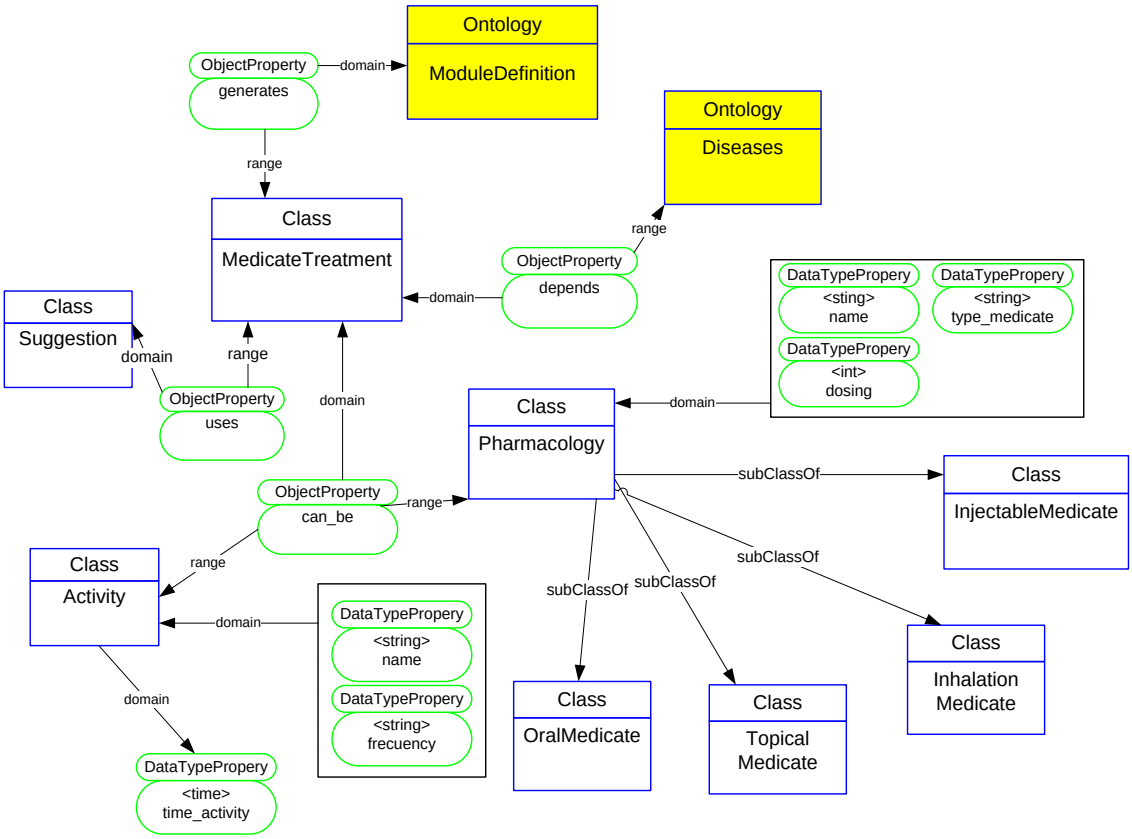


Figura 4-21. *Diagrama ontológico de la definición del Módulo MedicateTreatment***.**

En la tabla 4-12, se define y explican las clases que componen la ontología de Tratamiento Médico (MedicateTreatment).

Clase	Ontología	Conceptos relacionados	Valores
Pharmacology	MedicateTreatment	OralMedicate, InjectableMedicate	Cualquier referencia RDF válida.
	Descripción: Elemento que define tratamiento farmacológico de una enfermedad.		
OralMedicate	Pharmacology	Pharmacology	Cualquier referencia RDF válida.
	Descripción: Elemento que define tratamiento farmacológico oral de una enfermedad.		
InjectableMedicate	Pharmacology	Pharmacology	Cualquier referencia RDF válida.

	Descripción: Elemento que define tratamiento farmacológico inyectable de una enfermedad.		
TopicalMedicate	Pharmacology	Pharmacology	Cualquier referencia RDF válida.
	Descripción: Elemento que define tratamiento con medicamentos de uso tópico para una enfermedad.		
InhalationMedicate	Pharmacology	Pharmacology	Cualquier referencia RDF válida.
	Descripción: Elemento que define tratamiento de medicamentos inhalables para una enfermedad.		

Tabla 4-12. *Clases de la ontología de Tratamiento Médico (MedicateTreatment).*

En la tabla 4-13, se define y explican las propiedades que componen la ontología de Tratamiento Médico (MedicateTreatment).

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
time_activity	DataTypePop	Activity	time	[0 – 24 horas]
	Descripción: Tipo de dato que hace referencia a la hora en que se realizó una actividad.			
frecuency	DataTypePop	Activity	string	Cualquier referencia RDF válida.
	Descripción: Tipo de dato que hace referencia a la frecuencia con que se realiza una actividad.			
type_medicate	DataTypePop	Pharmacology	string	Cualquier referencia RDF válida.
	Descripción: Tipo de dato que hace referencia al tipo de medicamento a ingerir.			
dosing	DataTypePop	Pharmacology	int	Cualquier referencia RDF válida.
	Descripción: Tipo de dato que hace referencia a la dosis del medicamento a ingerir.			

Tabla 4-13. *Propiedades de la ontología de Tratamiento Médico (MedicateTreatment)*

Definición de Dieta (Diet)

Define una clasificación de los diferentes alimentos que un paciente puede consumir. La dieta es un complemento al tratamiento médico recomendado, por ello debe ser incluida como un módulo más del *framework*.

En la figura 4-22, se presenta el diagrama ontológico **dieta** (*Diet*). La ontología propuesta por Cantais (Cantais, Dominguez et al., 2005) hace una clasificación de alimentos para diabéticos, según el contenido energético de cada uno de ellos. Por otro lado (Hellín, Sánchez et al., 2009) (Hellín, Fuentes et al., 2010), define una ontología de alimentos (Food) que permite generar dietas para pacientes, asociadas a los requerimientos y gastos de energía en cada consumo.

Usamos ambas ontologías generalizándolas a otras enfermedades y además agregamos el componente de **alimentos recomendables** (*RecommendedFood*), **alimentos prohibidos**

(*ForbiddenFood*) y **alimentos restringidos** (*RestrictedFood*) para generar los menús a cada paciente, basándose en componentes como: *número de calorías, grasas, vitaminas, etc*.

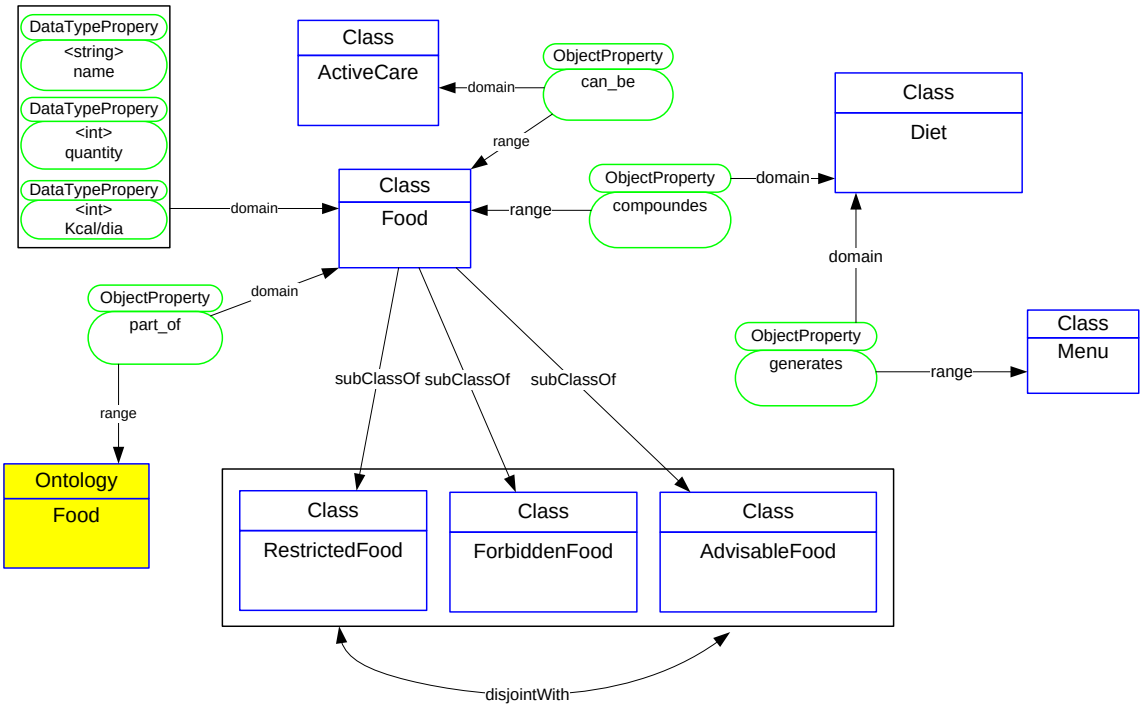


Figura 4-22. *Diagrama ontológico de la definición de dieta.*

En la tabla 4-14, se define las clases que componen la ontología de definición de dieta.

Clase	Ontología	Conceptos relacionados	Valores
Menu	Food	AdvisableFood	Cualquier referencia RDF válida.
	Descripción: Elemento que define la organización de comidas de los pacientes		
Food	Upper	Diet	Cualquier referencia RDF válida.
	Descripción: Ontología que organiza los alimentos a consumir por un paciente		
RestrictedFood	Diet	Diet	Cualquier referencia RDF válida.
	Descripción: Elemento que define los alimentos restringidos para el consumo de un paciente.		
ForbiddenFood	Diet	Diet	Name.
	Descripción: Elemento que define los alimentos prohibidos para el consumo de un paciente.		
AdvisableFood	Diet	Diet, Menu	Name, quantity.
	Descripción: Elemento que define los alimentos permitidos para el consumo de un paciente.		

Tabla 4-14. *Clases de la ontología de definición de dieta (Diet).*

En la tabla 4-15, se define y explican las propiedades que componen la ontología de definición de dieta (Diet).

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
part_of	ObjectProp	Diet	Food	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen es parte de un miembro de la clase destino.			
compoundes	ObjectProp	Diet	Food, RestrictedFood, ForbiddenFood, AdvisableFood	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase origen está compuesto por un elemento de la clase destino			
can_be	ObjectProp	ActiveCare	Food	Cualquier referencia RDF válida
	Descripción: Relación en la que un miembro de la clase puede ser un miembro de la clase destino			
quantity	DataTypePop	Food	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que hace referencia a la cantidad de alimento a consumir en la dieta.			
Kcal/dia	DataTypePop	Food	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que hace referencia a la cantidad de kilocalorías de un alimento.			

Tabla 4-15. *Propiedades de la ontología de definición de dieta (Diet)*

4.2.2.4 Estructura Ontológica de la Definición de Dispositivos Móviles

El desarrollo de ontologías para dispositivos ha sido abordado en algunas ocasiones, cada una de ellas ajustándose a las necesidades del momento.

Por un lado la **FIPA** (*Foundation for Intelligent Physical Agents*) (FIPA, 2000) ha definido una estructura ontológica, para facilitar la comunicación entre agentes, decidiendo si se usan ontologías explícitas, declarativamente representadas o las que implícitamente son codificadas con la puesta en marcha del *software* real. Hay también los ejemplos de modelos de dispositivo *genérico* (Vazquez, López-de-Ipiña et al., 2006) (Terrenghi, Quigley et al., 2009) que incluye mecanismos para descripciones de dispositivo crecientes. Para nuestra intención cada una de las ontologías evaluadas, carecen de una funcionalidad específica al momento de ser aplicadas exclusivamente a un entorno de monitorización móvil de pacientes.

Es por ello que basándonos en los estudios que se han analizado, hemos desarrollado una ontología de dispositivos móviles específicamente considerando los parámetros necesarios para la generación de aplicaciones para esos dispositivos móviles. Esta ontología puede observarse en la figura 4-23 y es una subclase derivada de una clase superior llamada **dispositivos**, en los que se engloban otras funcionalidades de diversos dispositivos. En la tabla 4-16, se define y explican las clases que componen la ontología de dispositivos móviles.

Clase	Ontología	Conceptos relacionados	Valores
MobileDevice	Devices	Devices, *Hardware*Specification, SoftwareSpecification, CommunicationSpecification	Cualquier referencia RDF válida
	Descripción: Elemento que clasifica cada uno de los dispositivos móviles que pueden ser usados en la arquitectura.		
***Hardware* Specification**	MobileDevice	MobileDevice, ResolutionSpecification, UserIntSpecification, MemorySpecification	Cualquier referencia RDF válida
	Descripción: Elemento que clasifica cada uno de los especificaciones del *hardware* de los dispositivos móviles que pueden ser usados en la arquitectura.		
Software Specification	MobileDevice	MobileDevice	Cualquier referencia RDF válida
	Descripción: Elemento que clasifica cada uno de los especificaciones del *software* de los dispositivos móviles que pueden ser usados en la arquitectura.		
CommunicationSpecification	MobileDevice	MobileDevice	Cualquier referencia RDF válida
	Descripción: Elemento que clasifica cada uno de los especificaciones de comunicación de los dispositivos móviles que pueden ser usados en la arquitectura.		
Resolution Specification	HardwareSpecification	HardwareSpecification	Cualquier referencia RDF válida
	Descripción: Elemento *hardware* que especifica las características de visualización de dispositivos móviles que pueden ser usados en la arquitectura.		
Memory Specification	HardwareSpecification	HardwareSpecification	Cualquier referencia RDF válida
	Descripción: Elemento *hardware* que especifica las características de memoria de dispositivos móviles que pueden ser usados en la arquitectura.		
Screen Specification	UserIntSpecification	UserIntSpecification	Cualquier referencia RDF válida
	Descripción: Elemento *hardware* que especifica las características de pantalla de dispositivos móviles que pueden ser usados en la arquitectura.		
UserInt Specification	HardwareSpecification	HardwareSpecification, ScreenSpecification, Audio	Cualquier referencia RDF válida
	Descripción: Elemento *hardware* que especifica las características de interfaz de usuario de los dispositivos móviles que pueden ser usados en la arquitectura.		
Audio	UserIntSpecification	UserIntSpecification	Cualquier referencia RDF válida
	Descripción: Elemento *hardware* que especifica las características de sonido de dispositivos móviles que pueden ser usados en la arquitectura.		
Devices	Upper	MobileDevice, BiometricDevice	Cualquier referencia RDF válida
	Descripción: Ontología que define los tipos de dispositivos electrónico que intervienen en la arquitectura.		

Tabla 4-16. *Clases de la ontología de dispositivos móviles.*

En la tabla 4-17, se define y explican las propiedades que componen la ontología de dispositivos móviles.

Propiedad	Tipo	Dominios	Rangos	Estado/ Valores
width	DataTypeProp	ResolutionSpecification , ScreenSpecification	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica el ancho de un miembro de la clase destino.			
heigth	DataTypeProp	ResolutionSpecification , ScreenSpecification	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica la altura de un miembro de la clase destino.			
unit	DataTypeProp	ResolutionSpecification , ScreenSpecification	string	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica unidad de medida de un miembro de la clase destino.			
suppgragraphic	DataTypeProp	ResolutionSpecification	string	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica el tipo de grafico de un miembro de la clase destino.			
amount	DataTypeProp	MemorySpecification	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica la cantidad de memoria de un miembro de la clase destino.			
available	DataTypeProp	MemorySpecification	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica la cantidad de memoria libre de un miembro de la clase destino.			
maximun	DataTypeProp	MemorySpecification	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica la cantidad de memoria máxima de un miembro de la clase destino.			
model	DataTypeProp	MobileDevice	string	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica nombre del modelo de un miembro de la clase destino.			
color	DataTypeProp	SreenSpecification	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica el número de pixeles de un miembro de la clase destino.			
nameOS	DataTypeProp	SoftwareSpecification	string	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica nombre de un elemento de un miembro de la clase destino.			
version	DataTypeProp	SoftwareSpecification	int	[0 a 1000]
	Descripción: Tipo de dato que especifica la versión de un miembro de la clase destino.			
volumen	DataTypeProp	Audio	int	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica el nivel de volumen de un miembro de la clase destino.			
vmachine	DataTypeProp	SoftwareSpecification	string	Cualquier referencia RDF válida
	Descripción: Tipo de dato que especifica el tipo de máquina virtual de un miembro de la clase destino.			

Tabla 4-17. *Propiedades de la ontología de dispositivos móviles.*

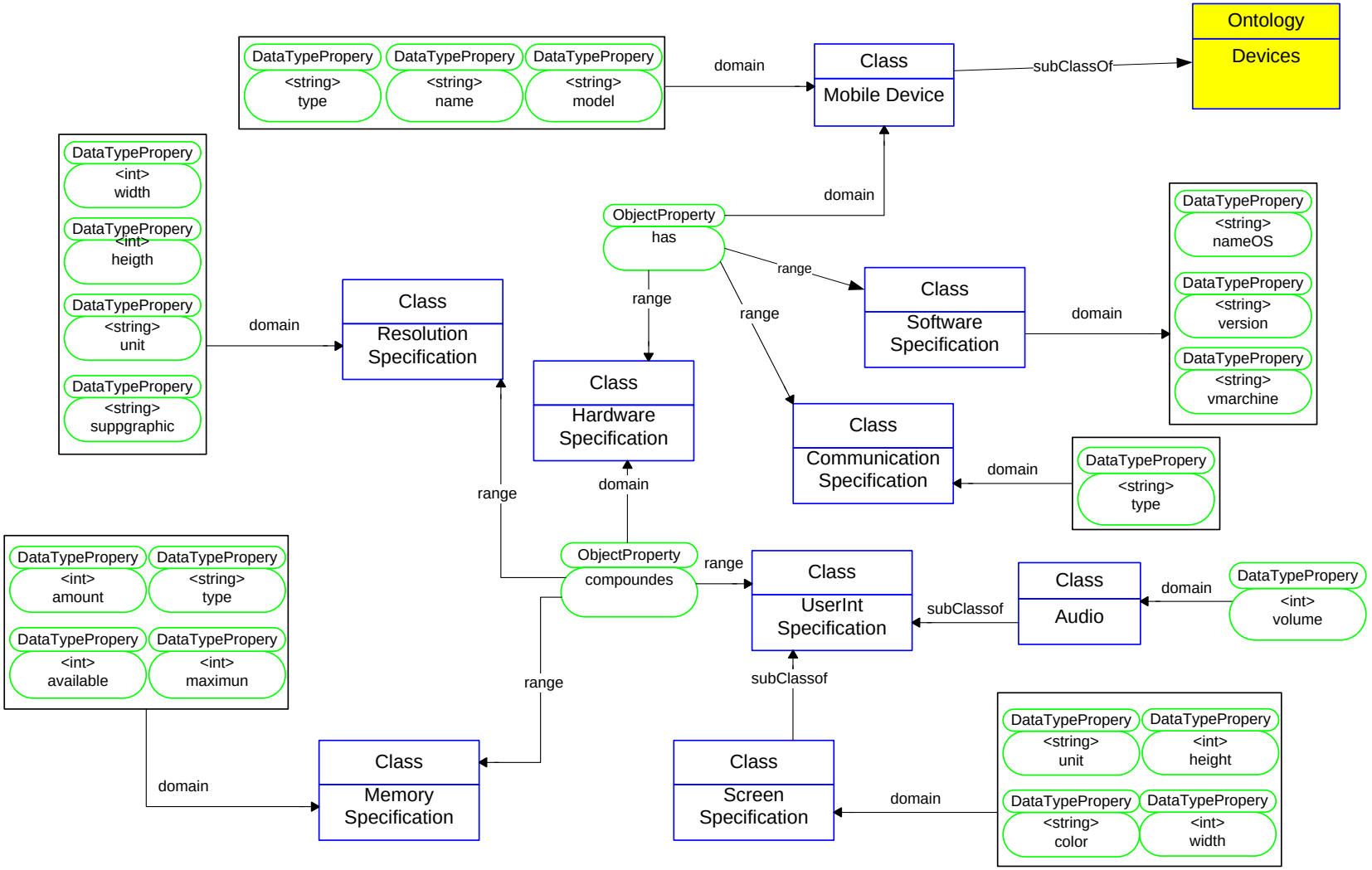


Figura 4-23. *Diagrama ontológico de Dispositivos Móviles.*

4.2.3 Conclusiones

En este apartado se ha presentado la estructura ontológica definida, desarrollada e implementada en nuestro *framework*. Cada uno de los elementos que lo componen, define tanto la funcionalidad como la organización de cada una de las partes.

En cuanto a la funcionalidad, define las situaciones en las cuales el *framework* puede mostrar información sobre situaciones suscitadas al paciente. Estas situaciones dependen de las medidas obtenidas de la enfermedad a monitorizar. En cuanto a la organización, se definen las propiedades de pertenencia que tiene cada ontología en su entorno funcional. Cómo son utilizadas cada una de ellas y cómo se organizan para manipular, de forma estructurada, la información que contienen.

Además de utilizar las ontologías desarrolladas para entender el entorno en el que hemos trabajado, son utilizadas para obtener información tanto de forma implícita o explícita. Esta información está relacionada con los parámetros de control médico del paciente. Las ontologías servirán en un futuro para hacer las aplicaciones desarrolladas interoperables con otras aplicaciones externas y para poder implementar motores de inferencia que doten de mayor inteligencia a las aplicaciones de monitorización.

4.3 MOBIPATTERNS: DESARROLLO DE APLICACIONES MÓVILES BASADO EN PATRONES

4.3.1 Introducción

El *framework* que hemos definido permite la construcción o generación de aplicaciones interactivas parametrizadas, que denominaremos arquitectura *software* en adelante, que será instalada en el dispositivo móvil (PDA, teléfono móvil, etc.). Para ello el usuario contará con la funcionalidad que necesite utilizar en el programa, dependiendo de parámetros previamente seleccionados (Villarreal, Fontecha et al., 2011).

Las aplicaciones móviles difieren en gran medida de las aplicaciones de escritorio. Los usuarios no la utilizan de la misma manera, no esperan las mismas capacidades y en algunos casos suelen utilizarlas de manera complementarias. Poseen características propias que la hacen completamente diferente del resto de aplicaciones, especialmente en el entorno de ejecución. Estas son algunas de las situaciones que nos lleva a diseñar y desarrollar elementos apropiados para estas aplicaciones, dependiendo de la funcionalidad y requerimientos con que se cuente.

En esta sección se analizan la generación de aplicaciones móviles para la monitorización de pacientes dentro de un alto nivel de abstracción.

4.3.2 Aplicando Model-driven Architecture (MDA)

Basándose en las especificaciones de *Model-driven Architecture* (**MDA**) explicado en el capítulo 2, definimos las especificaciones requeridas para el diseño de los *MobiPattern* utilizado para la generación de cada uno de los módulos de nuestra arquitectura. El proceso que MDA propone para el desarrollo de aplicaciones para una plataforma concreta

típicamente consiste en la definición de un modelo PIM, independiente de la plataforma de ejecución, para ser transformado a un modelo PSM dependiente de la plataforma en cuestión, y por último, ser transformado a código (o a otras estructuras de datos).

Las transformaciones que se ejecutan idealmente deben estar automatizadas y ser ejecutadas por herramientas, aunque en muchos casos debe haber cierta intervención humana en el proceso. El uso de los modelos CIM y la automatización de sus transformaciones apenas están descritas por la especificación de MDA, no quedando del todo claro qué se entiende como modelo CIM. En general se considera que estos modelos se usan sólo para especificar requisitos y obtener, de forma manual, los modelos PIM, aunque no se rechaza que esta transformación pueda ser automatizada.

Se explica de forma breve y resumida algunos aspectos analizados con MDA en el desarrollo de esta arquitectura.

4.3.2.1 Definición del Modelo Independiente de la Computación (CIM)

Se ha clasificado una lista de actividades independientes de la plataforma, desde el punto de vista del problema a resolver, abstrayéndose de detalles específicos. Esto se hace a través de la definición de **CIM**.

PARA EL PACIENTE:

1. Obtener las medidas de señales vitales para el control del paciente.
2. Actualizar el perfil individual del paciente con la información de contacto y control.
3. Almacenar los resultados de esas mediciones para futuro control.
4. Definir actividades de control como recomendaciones en caso de variaciones de medidas.
5. Definir actividades de prevención para el control de medidas de signos vitales.
6. Generar alarmas ante situación de emergencia.
7. Generar recomendaciones basadas en situaciones pasadas.
8. Enviar mensaje a familiar de contacto ante mediciones de señales vitales extremas.
9. Generar mensajes de educación, para mantener actualizado al paciente de datos referentes a la enfermedad que padece.
10. Generar gráficos de autocontrol de las mediciones obtenidas.
11. Generar historial visible para médicos.

PARA EL MÉDICO:

1. Visualizar historial médico de todos los pacientes.

2. Seleccionar un paciente en particular.
3. Observar el comportamiento de mediciones de cada paciente.
4. Ponerse en contacto con el paciente, ante situaciones de emergencia.
5. Seleccionar el comportamiento irregular del historial de mediciones de un paciente.
6. Enviar una recomendación puntual ante mediciones variantes.

4.3.2.2 Definición del Modelo Independiente de la Plataforma (PIM)

En este apartado se definen cada uno de los modelos explicados en **CIM**. Se especifica cada modelo definido antes, con un alto nivel de abstracción independiente de la tecnología. Se describe la arquitectura desde el punto de vista de los procesos de negocios que van a soportar.

PARA EL PACIENTE:

1. La obtención de las señales vitales se hará de forma manual o automática. La primera ingresando el valor obtenido por pantalla y la segunda leyendo a través de la tecnología de comunicación desde el dispositivo biométrico (siempre que este lo permita o posea dicha tecnología de comunicación). Se crea para esta y todas las funcionalidades un rol *paciente* que es accedido por todos los pacientes.
2. El paciente a través de la pantalla de perfil podrá actualizar sus datos que serán almacenados en el servidor de forma permanente. Esto es, si ha cambiado algún dato luego de instalada la aplicación. Todo esto de forma inalámbrica entre el servidor y el dispositivo móvil.
3. Las mediciones serán almacenadas en la base de datos, que está alojada en el servidor central.
4. Las recomendaciones generadas son módulos definidos en la arquitectura propuesta. Estas recomendaciones se encuentran ubicadas en el servidor, y son la base de futuras inferencias.
5. Se definen estructuras que eviten las alteraciones de las señales vitales en el futuro. Esto es a través de comparaciones entre medidas que pertenecen a los módulos de control definidos en la aplicación.
6. Se generarán alarmas para las personas de contacto en caso de emergencia. Además de alarmas mediante mensajes para el médico de cabecera.
7. Para generar las recomendaciones basada en situaciones pasadas, se compara las mediciones actuales de un paciente con las previas y generando respuestas a través de la aplicación que utiliza.
8. Se envía un SMS a familiares de un paciente en caso que la aplicación así lo requiera. Estos mensajes son trasmitidos a través de la red móvil.

9. Se crean módulos educativos, a través de información almacenada en el servidor, acerca de una enfermedad en particular. Los mensajes pueden ser sobre cualquier tipo, entre ellos, información básica sobre la enfermedad, cuidados generales, etc.

10. Se generarán gráficos utilizando librerías ofrecidas por la plataforma de desarrollo. Estos gráficos muestran el histórico del comportamiento de las mediciones de un paciente.

11. Un informe detallado con información importante para el médico que será visualizado en el rol "médico".

PARA EL MÉDICO:

1. El historial médico se visualizará a través de los registros históricos de cada paciente. Estos registros están presentes en el perfil de cada paciente. Se crea para esta y todas las funcionalidades un rol "*médico*" que es accedido por cada médico, en donde se muestra la lista de pacientes que se atienden.

2. Se muestra una lista de los pacientes que monitoriza, con esta lista se muestra información básica del perfil del paciente y sus mediciones en formato tabular.

3. Puede visualizar de forma gráfica o tabular el comportamiento de las medidas en el tiempo para cada paciente; puede ver al detalle cada medida, los valores y rangos obtenidos, las recomendaciones y mensajes generados.

4. Se generará un mensaje de alerta con los cuales el médico o algún familiar cercano puede ponerse en contacto con el paciente, en momentos de emergencias.

5. Mediante un gráfico el médico puede seleccionar un valor pico, que muestre alteraciones de mediciones de un paciente. Esto le permite conocer el motivo de dicha alteración.

6. Además de las recomendaciones, el doctor puede enviar mensajes al paciente una vez la aplicación le haya comunicado el estado del paciente. Ya sea una recomendación o un mensaje de prevención.

4.3.2.3 Definición del Modelo Específico de la Plataforma (PSM)

En este apartado se especifica cada modelo en términos de las construcciones que se van a implementar en el desarrollo. Se generan modelos **PSM**, a partir de los modelos **PIM** definidos previamente. Aunque este capítulo solo se centra en explicar la arquitectura desde un punto de vista conceptual, se mencionarán algunas especificaciones puntuales para PSM. En el capítulo quinto se profundiza más en detalles de implementación.

A pesar de que **PSM** es dependiente de la plataforma, el *framework* no impone una en particular, pero hace recomendaciones de plataformas y tecnologías que se pueden implementar.

PARA EL PACIENTE:

1. Se implementa el módulo de "*measure*" en A*ndroid*, con comunicación a través de servicios web en *java*.

2. El perfil se actualiza mediante llamadas a servicios web, implementados en el servidor de MySQL.

3. Como se menciono en el punto anterior, se ha implementado una base de datos en MySQL para almacenar toda la información de la arquitectura desarrollada.

4. Cada módulo de recomendación ha sido definido como una clase independiente de las otras, manteniendo una relación funcional definida por el "*XML Manifiest*" de Android.

5. Las alarmas son definidas como un módulo independiente de la arquitectura, basadas en Android y con comunicación a través de servicios web a la base de datos MySQL, genera mensaje en tiempo real distinguiendo entre cada paciente que use la aplicación.

6. Se utiliza la red móvil para el envío de SMS del paciente a un familiar de contacto, en caso que muestre altas variaciones en las mediciones obtenidas del dispositivo biométrico

7. Se generan gráficos a través de librerías para entorno gráfico desarrolladas para Android así como tablas que organizan los datos de las mediciones y rangos obtenidos de un paciente.

8. Se crea una lista con información relevante basada en Android.

PARA EL MÉDICO:

1. Se desarrolla la interfaz gráfica basada en librerías implementadas en Android.

2. Se utiliza el sistema de telefonía móvil para el envío de mensajes desde el médico hacia el paciente y viceversa. Esto es para generar alertas en caso de emergencia.

3. Mediante librería de gráficos para Android el médico puede visualizar las mediciones de un paciente basándose en una línea de tiempo, es decir, el comportamiento puntual de una medida pasada.

4.3.3 Clasificación estructural de los MobiPatterns

Algunas investigaciones para el desarrollo de aplicaciones móviles han utilizado los patrones como una alternativa en el proceso de desarrollo. Un patrón de recopilación de datos es un esquema genérico para adquirir los datos necesarios para los servicios de monitorización. (Cheun, Lee et al., 2010) desarrollaron un riguroso análisis sobre los patrones de comunicación y el intercambio de datos incluidos en (Clements, Bachman et al., 2002) y (Gamma, Johnson et al., 1994), y personalizaron cuatro patrones, que pueden ser utilizados para el seguimiento efectivo, estos son: patrón de *tirar (pulling)*, *pizarra (blackboard)*, *empujar*

(pushing), y el *observador (observer)*. Cada modelo se especifica por su *estructura*, la *situación de su caso*, sus *beneficios* y sus *limitaciones*.

En nuestro estudio hemos sugerido la elaboración de módulos específicos con funciones definidas para cada uno de los requerimientos que se tengan al momento de crear una aplicación para un dispositivo móvil.).

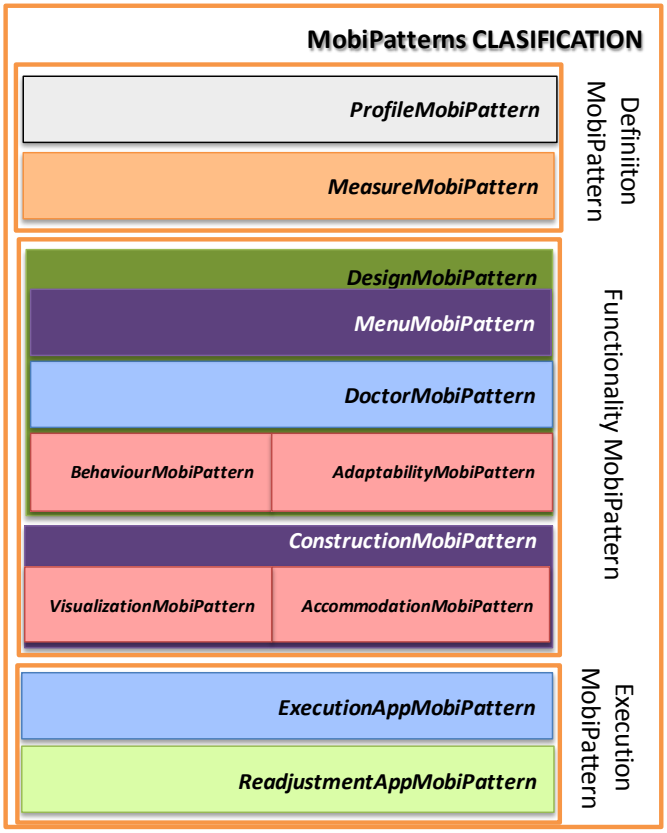


Figura 4-24. *Pila de Patrones usados para la generación de aplicaciones.*

Para la creación de esos módulos y la integración de cada uno de ellos para la generación de aplicaciones móviles se han definido y desarrollado un conjunto de patrones denominados "*MobiPatterns*" (Patrones Móviles). Para la definición de estos patrones integramos la definición inicial que comprende los patrones de arquitectura y de diseño, es decir, se hace una nueva definición integradora de ambos tipos de patrones, clasificándolos en los siguientes elementos mostrados en la figura 4-24:

a. **MobiPattern de Definición**: son patrones que permite definir los elementos que interactúan con los módulos de control. Son considerados de definición ya que definen el funcionamiento completo de la aplicación final.

   a.1 MobiPattern de Perfil: Patrón que define el módulo de adición y modificación del perfil de cada paciente.

   a.2 MobiPattern de Medidas: Patrón que define el módulo de adquisición de las medidas a monitorizar.

b. **MobiPattern de Funcionalidad**: son los patrones que definen la funcionalidad de cada módulo, la forma en que son construidos, la manera en que se visualizan y la manera en que se organizan o comunican entre sí.

b.1 MobiPattern de Diseño: patrones que definen el comportamiento de los módulos y la forma en que se adaptan nuevos módulos.

- **MobiPattern de menú**: patrón que define el diseño y comportamiento del menú principal dentro de la aplicación.
- **MobiPattern de Interfaces para el médico**: patrón que define el diseño y comportamiento de los módulos utilizados por el médico, para el seguimiento de sus pacientes.
- **MobiPattern de Comportamiento**: patrón que define el comportamiento funcional de los módulos de control médico. Utilizar este patrón facilita la implementación de la arquitectura desarrollada en diferentes tipos de enfermedades.
- **MobiPattern de Adaptabilidad**: patrón que define la forma en que deben ser adaptados nuevos módulos de control, siguiendo el patrón de comportamiento definido previamente.

b.2 MobiPattern de Construcción: patrones que definen la forma en que se visualizan y acomodan los elementos funcionales de la aplicación final.

- **MobiPattern de Acomodación**: patrón que define la manera en que se acomodan todos los módulos para la generación de la aplicación final. Definir la acomodación como un patrón facilita la identificación de todos los módulos, conociendo cuáles se ejecutan en primer o segundo plano.
- **MobiPattern de Visualización**: patrón que define la estructura visual de cada módulo ajustada a los diferentes dispositivos móviles. Como desarrollamos aplicaciones para diferentes dispositivos móviles con características diferentes, la aplicación debe ser capaz de ajustarse a las dimensiones de visualización en cada dispositivo (móviles, tablet, PDA, etc.).

c. **MobiPattern de Ejecución**: son los patrones que definen el proceso de ejecución de la aplicación final, es decir, definen las relaciones de ejecución entre cada uno de los módulos funcionales.

c.1 MobiPattern de Ejecución de aplicación: patrón que define el orden de ejecución de cada uno de los elementos de la aplicación.

c.2 MobiPattern de Reajuste de aplicación: patrón que define el proceso de reajuste de la aplicación una vez haya sido generada. Algunas características que pueden cambiar son: la enfermedad a monitorizar, perfil del paciente o las sugerencias o recomendaciones generadas.

Para la definición de cualquier *MobiPattern* debemos tener en cuenta que todas las interpretaciones que se hagan son generadas luego de realizada una medición. Buscando la "***multimonitorización"***, todas las lecturas de las mediciones hechas por un determinado dispositivo biométrico contarán con cinco niveles de rangos, mostrados en la tabla 4-18.

Nivel de Rango	Descripción
Alerta *(Alert)*	Nivel que debe generar una alerta inmediata al momento que se hace una lectura de las señales vitales de un paciente. Este caso ya es atendido por especialistas y emergencias médicas
Bajo *(Low)*	Nivel en donde se hacen recomendaciones, sugerencias, se reajustan dosificaciones, entre otras opciones de control que se le ofrece al paciente. Hay una relación entre el paciente y el dispositivo que lo controla (PDA, Teléfonos móviles)
Aceptable *(Acceptable)*	Nivel en donde aun manteniéndose valores estables en las señales vitales del paciente se hacen algunas recomendaciones para evitar cambios repentinos de niveles. Se activan elementos de educación sobre la enfermedad para evitar cambios abruptos.
Ideal *(Ideal)*	Nivel en donde todos los elementos de la medición se encuentran en el ideal buscado, sin alteración de ningún tipo. El dispositivo que te monitoriza sigue funcionando a la espera de cualquier variación
Elevado *(High)*	Nivel superior en donde las señales vitales leídas muestran un crecimiento elevado. Se activan elementos de control o en casos extremos de alerta inmediata.

Tabla 4-18. *Clasificación de rangos de lectura de señales vitales de un determinado paciente.*

4.3.4 MobiPatterns de Definición

Este patrón es el encargado de definir las funcionalidades iniciales de la aplicación para la monitorización de pacientes. Se ha mencionado inicialmente que el perfil de paciente y sus valores de las señales vitales, son los puntos de partida para definir la funcionalidad de la aplicación final.

Es por ello que se han definido dos patrones dentro del patrón de diseño. Estos patrones se encargan de la definición del perfil (*ProfileMobiPattern*) y la obtención de las señales vitales (*MeasureMobiPattern*). Se define aquí la estructura física de cada uno de los módulos mencionados.

4.3.4.1 MobiPattern de Perfil (ProfileMobiPattern)

El perfil del paciente es uno de los elementos más importantes de nuestro *framework*, ya que según las especificaciones que este tenga, se generarán cada uno de los módulos necesarios para su monitorización.

Es por ello que se ha definido un *MobiPattern* llamado *ProfileMobiPattern*, que se encarga de generar el perfil del paciente como un módulo independiente, de tal manera que facilite la generación de los módulos respectivos, especialmente asociados a la enfermedad que tiene el paciente.

En la tabla 4-19 se definen las especificaciones de este *MobiPattern*.

Nombre	***ProfileMobiPattern***
Contexto	El perfil del paciente es uno de los elementos de gran importancia a la hora de la generación de módulos y aplicaciones. Representan la información única a cada paciente.
Problema	Se definen un número de módulos, cada uno asociados a un tipo de enfermedad; pero cada paciente tiene un historial clínico asociado al comportamiento de esa enfermedad en sus actividades diarias. La medicación, recomendación o cualquier tratamiento que funcione para un determinado paciente puede no funcionar para otro paciente, aun que tengan la misma enfermedad con características similares.
Usos	En todos los elementos de las aplicaciones para monitorización móvil de un paciente. Definición de módulos asociados a un perfil único de cada paciente.
Solución	La solución consiste en definir una única funcionalidad al módulo del perfil de paciente, de esta manera podremos asociar los demás módulos que se generen en situaciones posteriores, evitando la generación innecesaria de aplicaciones poco funcionales para un paciente (ver figura 4-25).
Fundamentos	Patrón que define las características propias a cada paciente asociados a los módulos de control, monitorización, etc.
Relaciones	Patrón de medida (MeasurePattern).

Tabla 4-19. *Especificaciones del MobiPattern de Perfil* ***(ProfileMobiPattern).***

En la figura 4-25, se especifica la estructura del **MobiPattern** del perfil del paciente, con los siguientes elementos: *ID, name, address, phonenum, housenum, contact, email*

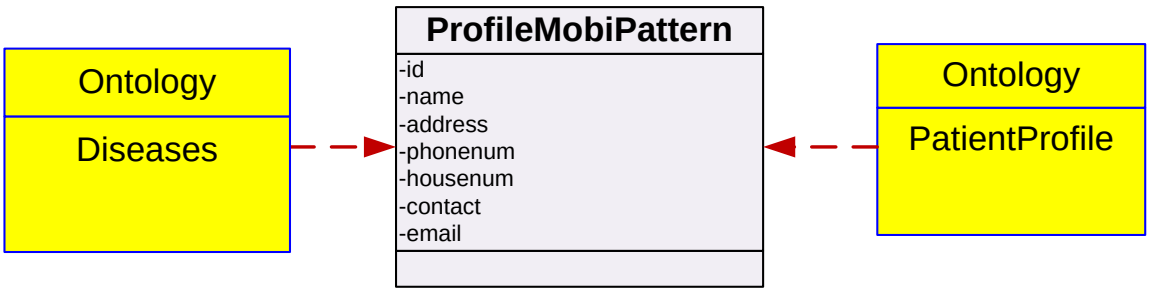


Figura 4-25. *Estructura del MobiPattern Perfil y su relación con la Ontología PatientProfile.*

En la figura 4-26, se muestra la estructura física del MobiPattern de perfil.

Los elementos que la componen son:

- **Picture**: fotografía del paciente
- **Name**: Nombre completo del paciente
- **Basic Information**: Incluye todos los bloques correspondientes a la información básica del paciente.

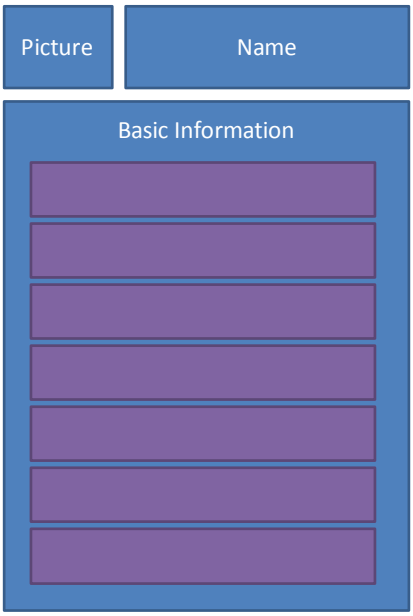


Figura 4-26. *Representación visual del ProfileMobiPattern.*

4.3.4.2 MobiPattern de Medidas (MeasureMobiPattern)

Además del perfil del paciente, se ha definido un *MobiPattern* que facilite la generación del módulo de medición. Para ello se ha creado el *MobiPattern* de Medida llamado *MeasureMobiPattern***.** Este *MobiPattern* se encarga de manipular las lecturas de cada una de las señales vitales generadas por los dispositivos biométricos, de tal manera que la aplicación monitorice constantemente al paciente y sepa en su momento que módulos de la aplicación *lanzar* cuando se necesiten.

En la tabla 4-20 se definen las especificaciones de este *MobiPattern*.

Nombre	***MeasureMobiPattern***
Contexto	La medida es un elemento de constante cambio a la hora de iniciar la monitorización del paciente. Por tal motivo se ha definido un patrón independiente para este elemento, ya que serán esas variaciones en las unidades de medidas las que llamaran a los respectivos módulos de la aplicación del paciente.
Problema	Es muy importante la constante evaluación de la medida obtenida del dispositivo biométrico, por tal motivo, se hace necesario definir elementos que se encarguen del control constante de esa medida. Si se llega a perder en algún momento de la monitorización el resultado de una lectura de una medida, la aplicación no estaría realizando la función para la cual fue generada.
Usos	Para generar un historial de medidas que se utilizarán en el modelo predictivo al momento de generar nuevos módulos. Para el uso de los módulos funcionales que ha sido definido.
Solución	La solución consiste en definir un patrón que nos permita la generación de ese módulo de medición, el cual posea características asociadas a los valores obtenidos de los dispositivos biométricos (ver figura 4-27).
Fundamentos	Patrón que define las características que deben tener todas las lecturas emitidas por el dispositivo biométrico para su interpretación
Relaciones	Patrón de Perfil (ProfileMobiPattern) y patrones funcionales (Functionality MobiPattern).

Tabla 4-20. *Especificaciones del MobiPattern de Medidas* ***(MeasureMobiPattern).***

En la figura 4-27, se especifica la estructura del *MobiPattern* de Medida, con los siguientes elementos:

- **Disease:** dependiendo del tipo de enfermedad se definen las unidades de medidas asociadas a cada enfermedad. Antes de utilizar el *MobiPattern* de medida, se debe seleccionar previamente el tipo de enfermedad que el paciente quiere medir. Estos se hace debido a que un paciente con una determinada enfermedad puede padecer de dos o más enfermedades ya sean derivadas de la enfermedad inicial. Este *MobiPattern* basado en ese tipo de enfermedad, establece los siguientes valores:

    - **MeasurerangeH**: define el rango de una medida para el valor alto de una determinada enfermedad.

    - **MeasurerangeL**: define el rango de una medida para el valor bajo de una determinada enfermedad.

- **Level**: se basa en los niveles delimitados por la tabla 4-18, referente a los rangos permitidos de los valores de una determinada enfermedad. El paciente además de saber cuáles son sus rangos de medida para su enfermedad, se le presenta una interpretación más fácil de entender por él, sin necesidad de saber conocer mucho sobre su enfermedad.

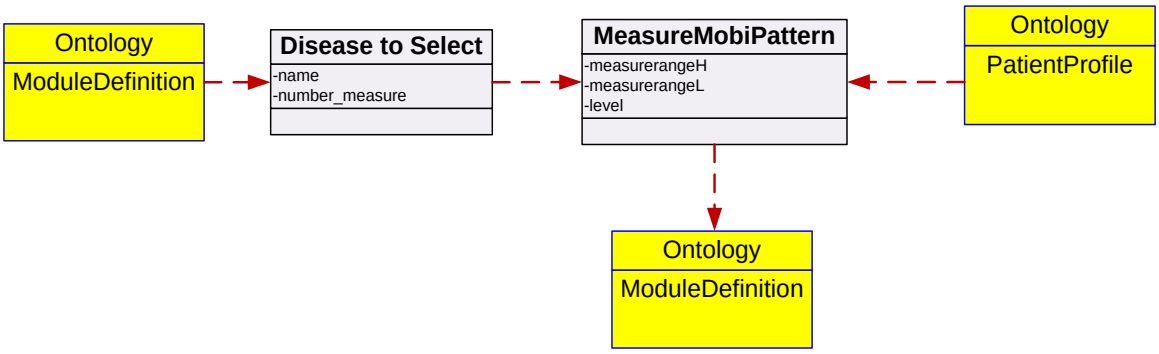


Figura 4-27. *Estructura del MeasureMobiPattern y su relación con la Ontología PatientProfile, Diseases y ModuleDefinition.*

En la figura 4-28, se muestra la estructura física del MobiPattern de medidas. Los elementos que la componen son:

- **Disease Name**: Nombre de la enfermedad a monitorizar.
- **Conditions**: tipo de captura de las señales vitales (automática o manual) o tipos de ejercicio a desarrollar.
- **Value**: valor de la medida obtenida o tipo de ejercicio a desarrollar.
- **"Action":** botón de acción para almacenar la medida.

4.3.5 MobiPatern de Funcionalidad

El patrón de funcionalidad define las características funcionales de cada módulo, es decir, la función que tienen dentro de la aplicación final. Para ello se distingue entre el diseño y la construcción de la aplicación, cada uno de ellos con patrones claramente definidos.

Un patrón de diseño está compuesto por el patrón de comportamiento y el patrón de adaptabilidad. Mientras que el patrón de construcción, está compuesto por los patrones de acomodación y visualización. Cada patrón se definirá a continuación.

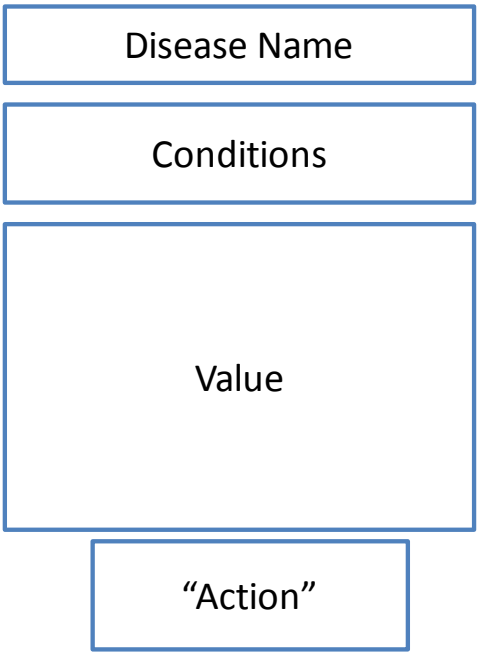


Figura 4-28. *Representación visual del MeasureMobiPattern.*

4.3.5.1 MobiPattern de Diseño (DesignMobiPattern)

El *MobiPattern* de diseño llamado *DesignMobiPattern* nos permite definir las estructuras de funcionalidad de nuestra aplicación, es decir, cómo la aplicación será ejecutada en el dispositivo móvil.

A diferencia de la definición inicial del concepto inicial de *diseño* en el uso de patrones, este patrón no hace referencia al diseño visual, sino a definir el diseño funcional de cada módulo.

Tomando en cuenta que aunque las prestaciones técnicas de nuestro dispositivo móvil es lo suficientemente poderosa como para procesar gran cantidad de información, no puede compararse a las prestaciones que ofrecen un ordenador convencional. Es por ello que las aplicaciones generadas deben poder ajustarse a cada dispositivo en el que se ejecutará.

En este sentido se han definido dos *MobiPattern* esenciales al momento de generar aplicaciones para dispositivos móviles basados en aspectos de funcionalidad y adaptabilidad.

MobiPattern de Comportamiento (BehaviourMobiPattern)

En este *MobiPattern* (*BehaviourMobiPattern*) se definen las diferentes funcionalidades de cada módulo. Es aquí en donde se establece qué parámetros son necesarios para un módulo, cuales son los procesos que se ejecutan y cuál es la salida final de cada uno de ellos. Es decir ***¿Qué hace cada módulo?*** En la tabla 4-21 se definen las especificaciones de este *MobiPattern*.

Nombre	***BehaviourMobiPattern***
Contexto	La funcionalidad de cada módulo o elemento de la aplicación está definida por este patrón, es decir qué hace cada uno, cómo funcionan al momento que son ejecutados.
Problema	Hay que definir muchas funcionalidades que dependen en su gran mayoría del previo análisis de las ontologías que se han definido para el generador de aplicaciones móviles (*framework*).
Usos	En todos los elementos de las aplicaciones para monitorización móvil de un paciente. Definición de muchos elementos que requieran un orden específico al momento de utilizarse.
Solución	La solución consiste en definir una única funcionalidad a cada elemento del *framework*, de tal manera que al momento que se necesite implementar ese elemento en una aplicación, se diferencie de las demás.
Fundamentos	Evitar la duplicidad funcional de elementos dentro del *framework*, asignando parámetros específicos a cada uno.
Relaciones	Este patrón no tiene ninguna relación con otro patrón.

Tabla 4-21. *Especificaciones del MobiPattern de Comportamiento* ***(BehaviourMobiPattern).***

En la figura 4-29, se muestra la relación que existe entre el *MobiPattern* comportamiento con las ontologías previamente definidas.

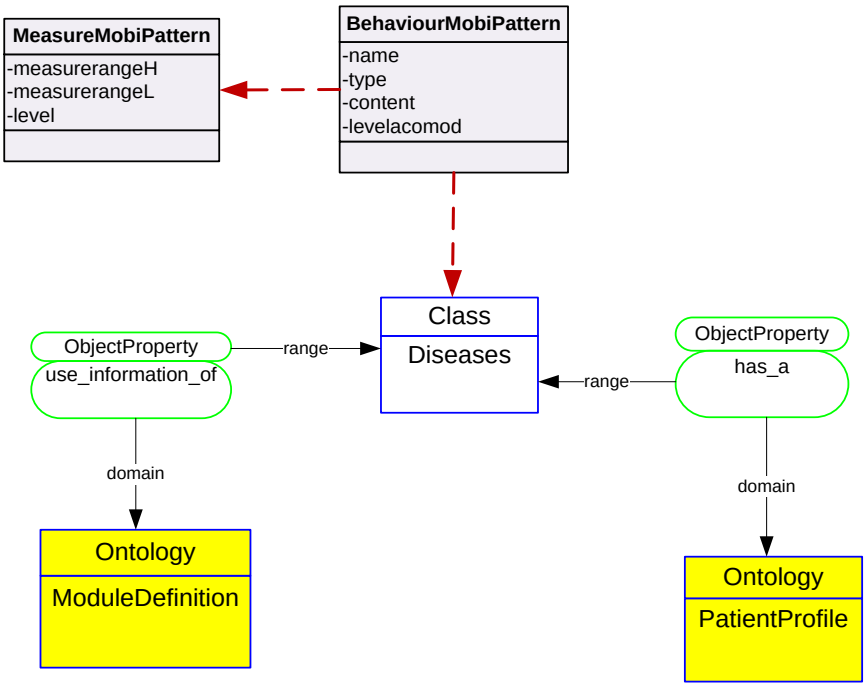


Figura 4-29. *Estructura del MobiPattern Comportamiento para cada enfermedad y su relación con la ontología correspondiente.*

Este *MobiPattern* posee cuatro elementos importantes que son:

- **Name**: en donde se define el nombre del módulo que puede ser por ejemplo dieta.
- **Type**: se define el tipo de módulo especificado según el rango especificado en la lectura de la medida.
- **Content**: se especifica la información específica del módulo, según su funcionalidad.

- **Levelacomod**: este campo define la secuencia de acomodación que será utilizado en el *MobiPattern* de acomodación (*AccomodationMobiPattern*).

Todos los módulos que se creen posteriormente deben seguir la estructura física del patrón de adaptabilidad observado en la figura 4-30. Los elementos que componen esta MobiPattern son:

- **Name**: Nombre del módulo a agregar.
- **Type**: a qué tipo de módulo corresponde: recomendación, prevención, educación, etc.
- **Message o content**: comportamiento del módulo (funcionalidad mostrada).
- **"Action 1" y "Action 2"**: botones de acción del MobiPattern.

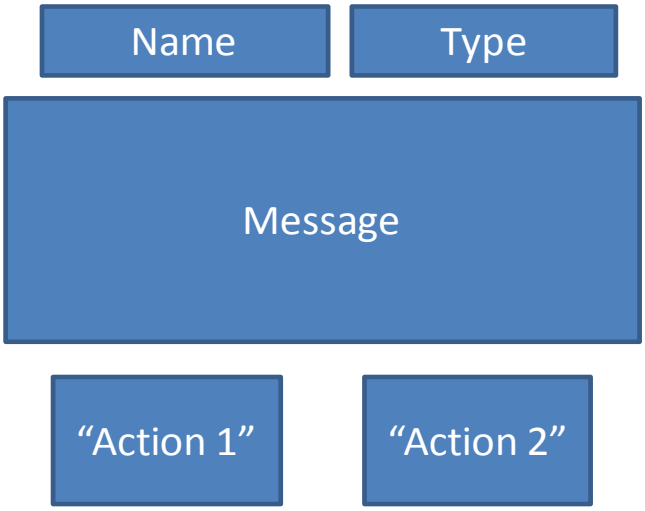


Figura 4-30. Representación visual del MobiPattern de Comportamiento.

MobiPattern de Adaptabilidad (AdaptabilityMobiPattern)

Define la posibilidad de adaptar nuevas funcionalidades a las aplicaciones, para que convivan con las ya existentes. Es decir ***¿Cómo se integra una funcionalidad nueva a un módulo?*** O ***¿Cómo crear un nuevo módulo a partir de los ya existentes?*** En la tabla 4-22 se definen las especificaciones de este **MobiPattern**.

Nombre	***AdaptabilityMobiPattern***
Contexto	Existen algunas funcionalidades que no pertenecen a un módulo en especial, y la aplicación debe poder adaptar esas nuevas funcionalidades cuando se necesiten.
Problema	El *framework* debe poder redefinir nuevas funcionalidades que no se encuentren definidas en los módulos previos.
Usos	Para controlar el estado del paciente ante situaciones de alarma o necesidad de atención oportuna.
Solución	La solución consiste en ofrecer una posibilidad de adaptar nuevas funcionalidad según la necesidad del paciente si tener que hacer severos cambios a toda la aplicación.
Fundamentos	Facilitar la integración sin realizar muchos cambios de nuevas funciones a ciertas áreas de la aplicación. Facilita el desarrollo de elementos que dependan de otros ya existentes.
Relaciones	Patrón de Comportamiento *(BehaviorMobiPattern)*

Tabla 4-22. *Especificaciones del MobiPattern de Adaptabilidad* ***(AdaptabilityMobiPattern).***

El MobiPattern de adaptabilidad tiene el mismo comportamiento físico que el MobiPattern de comportamiento. Es decir, si se quiere adaptar un nuevo módulo a la aplicación debe integrarse siguiendo el mismo diseño del MobiPattern de comportamiento. Este MobiPattern está formado por los siguientes elementos: NAME, TYPE, MESSAGE o CONTENT, and ACTIONS.

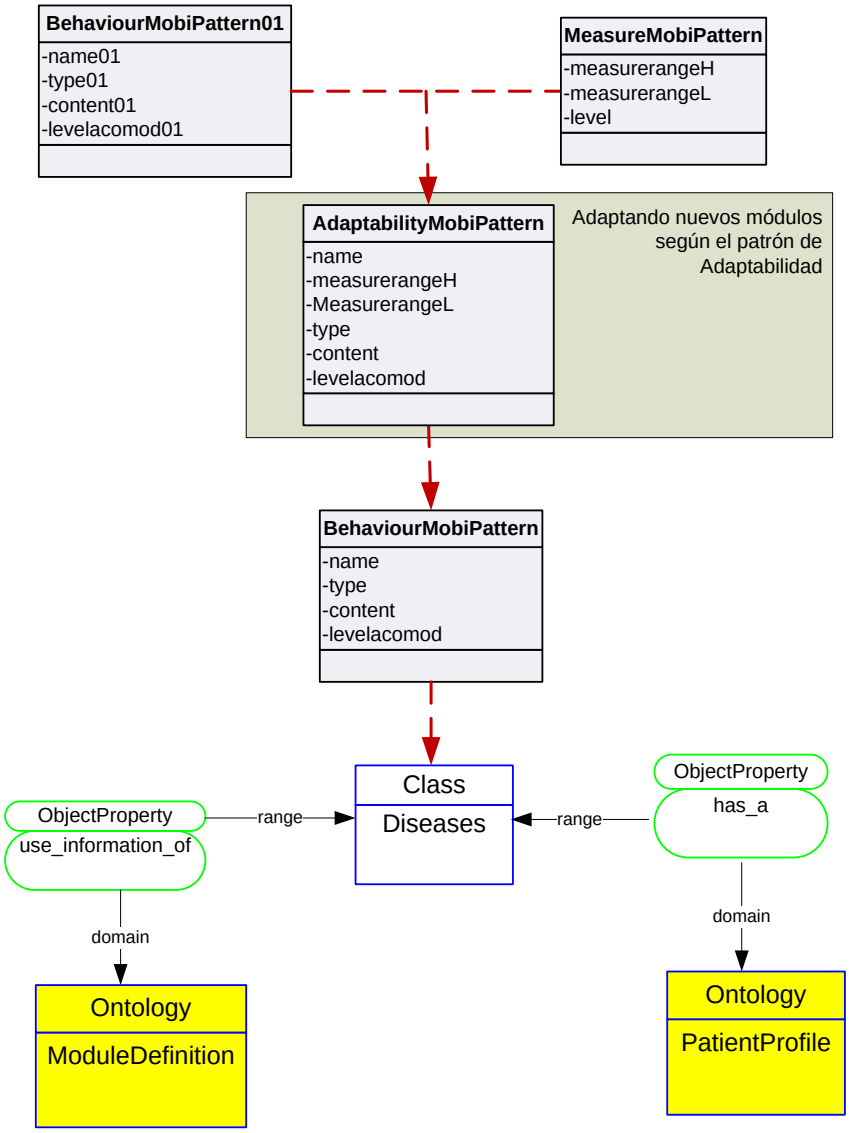


Figura 4-31. *Estructura del MobiPattern de Adaptabilidad para cada enfermedad y su relación con la ontología correspondiente.*

En la figura 4-31, se muestra la relación que existe entre el *MobiPattern* Adaptabilidad con las ontologías previamente definidas y los otros MobiPattern que intervienen como son el *BehaviourMobiPattern* y el *MeasreMobiPattern*. Para ello el ***MobiPattern*** posee cinco elementos importantes, tres de los cuales son heredados del ***MobiPattern*** de comportamiento, estos son:

- **Name**: en donde se define el nombre del nuevo módulo adaptado a partir del o de los patrones relacionados.
- **MeasurerangeH**: define el rango de una medida para el valor alto de una determinada enfermedad.

- **MeasurerangeL**: define el rango de una medida para el valor bajo de una determinada enfermedad.
- **Type**: se define el tipo de módulo especificado según el rango especificado en la lectura de la medida según la tabla 4-1.
- **Content**: se especifica la información específica del módulo, según su funcionalidad.
- **Levelacomod:** este campo define la secuencia de acomodación que será utilizado en el **MobiPattern** de acomodación (**AccomodationMobiPattern**).

La funcionalidad de este *MobiPattern* se comprende en el siguiente caso: existen dos módulos creados previamente a través de su *MobiPattern* de comportamiento y se tiene la necesidad de crear un nuevo módulo que tenga una funcionalidad nueva, que utilice funcionalidades de estos dos módulos anteriores.

Aplicando el *MobiPattern* de adaptabilidad, se ajustan estos valores para generar un nuevo módulo con un *MobiPattern* de comportamiento nuevo. Cuando se necesite el mismo módulo nuevamente no será necesario volver adaptar este, ya que ha sido creado previamente para ese paciente.

4.3.5.2 MobiPattern de Diseño interfaz de médico (DoctorMobiPattern)

El *MobiPattern* diseño de interfaz para los médico (DoctorMobiPattern), define la estructura de las interfaces de diseño de las pantallas principales para la gestión de pacientes por parte de los médicos.

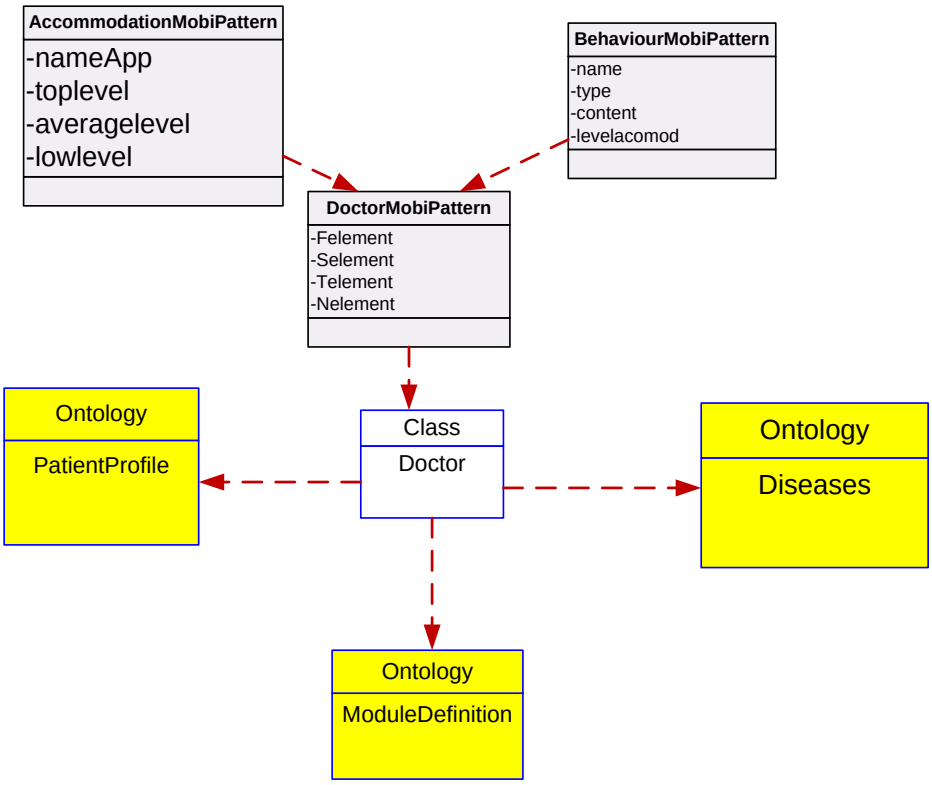


Figura 4-32. *Estructura del DoctorMobiPattern para la generación de interfaces del médico.*

Todas las interfaces que interactúan con el médico deben seguir la misma forma o diseño de tal manera que sea fácil de identificar a todos los pacientes, así como obtener la información más importante para su seguimiento.

En la figura 4-32, se puede mostrar la relación y estructura del **DoctorMobiPattern** que tiene relación con las ontologías de enfermedades, perfil de paciente y de definición de módulos. Este el único patrón que depende de dos patrones previos como son el patrón de acomodación (**AccommodationMobiPattern**) y el patrón de comportamiento (**BehaviourMobiPattern**). Para la aplicación del médico no se hace necesario crear las interfaces de comportamiento y acomodación, ya que han sido creadas previamente en la aplicación del paciente.

Con este *MobiPattern* se define la ubicación o acomodación de cada módulo en la aplicación y se eligen y configuran los parámetros de visualización para así adaptarlos al dispositivo móvil que se vaya a utilizar. Este patrón aclara la pregunta de **¿cómo organizar la información para el que médico puede observa de manera rápida todos y cada uno de los pacientes que tiene?**

En la figura 4-33 se muestra la manera en que se organiza la información para un mejor manejo por los médicos.

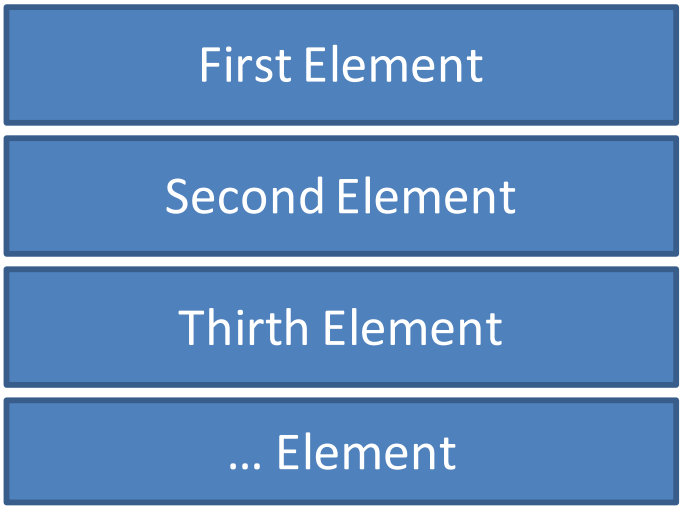


Figura 4-33. Representación visual del MobiPattern de interfaces para el médico.

Cada uno de los elementos (1, 2, 3, 4…) define la ubicación de dos tipos de información de la siguiente manera:

- **Definición de la lista de pacientes**: aquí se puede listar todos los pacientes que tiene un médico bajo su control, de tal manera que si se quieren ir agregando más el patrón es el mismo, es decir, el MobiPattern definido siempre tendrá la misma estructura física, cambiando la funcionalidad.
- **Especificación de la información de cada paciente**: aquí se pueden listar las opciones visibles de la información de un paciente. Se pueden definir alguna información inicial y cada vez que se quiera agregar nuevas funcionalidades a las interfaces del médico, se debe seguir la organización según el MobiPattern.

Mantener un mismo patrón para todo el diseño de las interfaces del médico facilita el desarrollo y reutilización de sus elementos, familiarizando al médico por un lado, y por el otro facilitando el diseño. En la tabla 4-23, se definen las especificaciones de este MobiPattern:

Nombre	***DoctorMobiPattern***
Contexto	Definir el diseño de las interfaces desarrolladas para el médico, de tal manera que se mantenga la uniformidad en la distribución de sus elementos.
Problema	El *framework* debe poder definir estructuras funcionales tanto para el paciente como para el médico, integrando características funcionales reutilizables.
Usos	Para el diseño de las interfaces del médico, lo que permite una uniforme distribución de sus elementos.
Solución	La solución consiste en ofrecer una posibilidad que permita la reutilización y estandarización de los elementos que definen las interfaces de los médicos.
Fundamentos	Facilitar el desarrollo de cada elemento, estableciendo un modelo específico a seguir al momento de integrar nuevos elementos basados en la reutilización de los ya existentes.
Relaciones	Patrón de Comportamiento *(BehaviorMobiPattern), Patrón de acomodación (AccommodationMobiPattern), Patrón de perfil de paciente (ProfileMobiPattern)*

Tabla 4-23. *Especificaciones del MobiPattern de la interfaces para el médico* ***(DoctorMobiPattern).***

MobiPattern de Menú (MenuMobiPattern)

En este *MobiPattern* (*MenuMobiPattern*) se define la organización de todos los elementos del menú principal y la relación funcional con los demás patrones. La definición y uso de este patrón permite adaptar nuevas funcionalidades al menú sin la necesidad de modificar su estructura. Se responde a la pregunta ***¿Cómo organizar el acceso a los diferentes módulos de la aplicación, haciéndolo más amigable y fácilmente identificable?*** En la tabla 4-24 se definen las especificaciones de este *MobiPattern*. En la figura 4-34, se muestra la relación que existe entre el *MobiPattern* menú con las ontologías previamente definidas.

Nombre	***MenuMobiPattern***
Contexto	La estructura física del menú de la aplicación está definida por este patrón, es decir cómo organizar los elementos visuales para ser fácilmente identificados.
Problema	Hay que mantener la consistencia al momento de desarrollar las aplicaciones, de tal manera que el usuario identifique de manera uniforme todas las funcionalidades que ésta ofrece. Muchas veces no se conoce la manera de distribuir los elementos en un menú por lo cual se hace necesario definir una estructura física estandarizada.
Usos	En todos los elementos de las aplicaciones para monitorización móvil de un paciente. Definición de muchos elementos que requieran un orden específico al momento de utilizarse.
Solución	La solución consiste en definir una única funcionalidad a cada elemento del *framework*, de tal manera que al momento que se necesite implementar ese elemento en una aplicación, se diferencie de las demás.
Fundamentos	Evitar la duplicidad funcional de elementos dentro del *framework*, asignando parámetros específicos a cada uno.
Relaciones	ProfileMobiPattern.

Tabla 4-24. *Especificaciones del MobiPattern de Comportamiento* ***(BehaviourMobiPattern).***

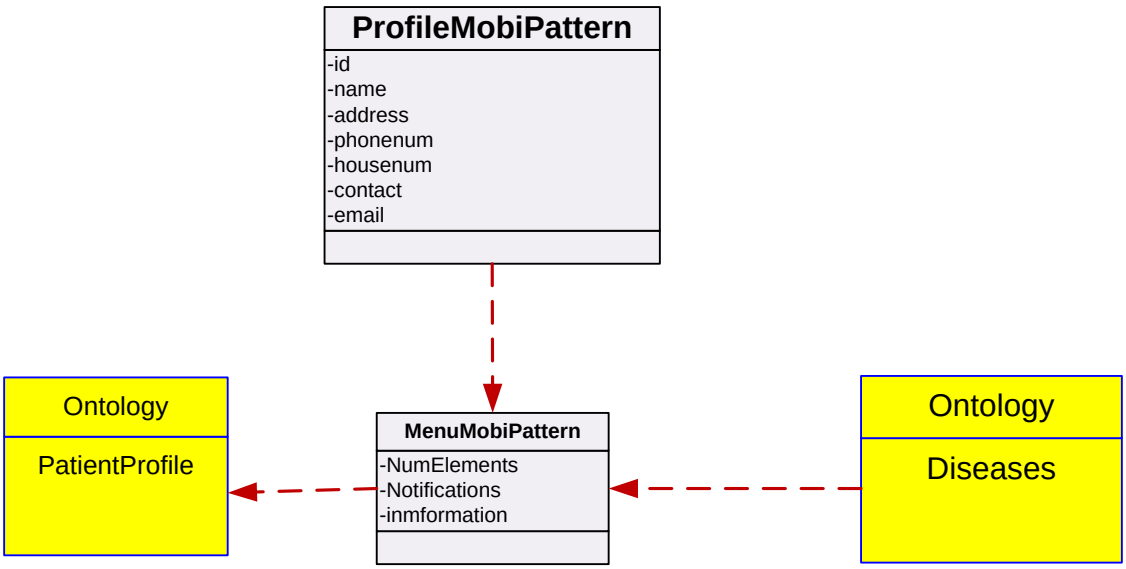


Figura 4-34. *Estructura del MobiPattern Menú y su relación con la ontología correspondiente.*

Figura 4-35. *Representación visual del MobiPattern Menú.*

En la figura 4-35, se muestra la estructura física del MobiPattern de Menú, sus elementos son:

- **Information:** este elemento del menú muestra información relacionada a la pantalla en donde se está ubicado. Al desarrollar aplicaciones para personas con edades variadas, se debe tener en cuenta la facilidad de identificar en dónde se encuentran en un momento determinado.
- **NumElements (Menu #):** define cada uno de los posibles elementos que tiene el menú. Cada elemento hace referencia a las funcionalidades definidas para esa opción

del menú. Para pasos posteriores, se pueden ir agregando nuevas opciones de menú siguiendo la estructura del patrón.

- **Notifications:** muestra información de mensajes de alertas generados por la aplicación. La ubicación de estos elementos permite la identificación inmediata de elementos prioritarios en la aplicación. Todos los mensajes de notificación pueden ubicarse en la posición definida por el patrón, aunque su reubicación no afectaría la funcionalidad del menú.

4.3.5.3 MobiPattern de Construcción (ConstructionMobiPattern)

El *MobiPattern* de construcción (*ConstructionMobiPattern*) define la forma en que los módulos, luego de diseñados, serán integrados en una única aplicación para ser ejecutados en el dispositivo móvil que se necesite.

Debe tomarse en cuenta que las aplicaciones generadas, que serán embebidas en el dispositivo móvil, deben tener una funcionalidad delimitada a cada paciente de tal manera que no se carguen elementos que no se vayan a utilizar.

MobiPattern de Acomodación (AccommodationMobiPattern)

Define la estructura de ubicación de cada módulo o elemento en la aplicación. Se debe poder visualizar en una misma pantalla diferentes funcionalidades para un paciente sin la necesidad de tener que navegar excesivamente en la aplicación. Para ello se han definido niveles de ubicación de los elementos, con la función de cada uno de ellos. Es decir ***¿Dónde ubicar cada elemento?*** En la tabla 4-25 se definen las especificaciones de este *MobiPattern*.

Para facilita el manejo y construcción de este **MobiPattern**, se han definido tres niveles de acomodación al momento de generar aplicaciones, como se ve en la figura 4-36.

- **Nivel Inferior o de Reajustes (*Low Level*): a**quí se encuentran todos los módulos correspondientes a ajustes o cambios que haya que hacerle a la aplicación de tal manera que si las lecturas de señales vitales varían mucho, debe este nivel saber comunicar esta necesidad al momento en que se requiera. Aquí también se ubica el modelo predictivo que recibe información constante al momento de leer en periodos de tiempo las variaciones de las señales vitales recibidas del dispositivo biométrico.

- **Nivel Medio o de Control (*Average Level*)** aquí se ubican todos los demás módulos que corresponde a la monitorización del paciente, es decir si un paciente sufre una determinada enfermedad, en este nivel solamente existirán los módulos que lo ayudaran a controlar esa enfermedad.

- **Nivel Superior o de Medición (*Top Level*)** es el nivel más alto en el *MobiPattern* de acomodación, es decir, aquí se ejecutará con mayor prioridad la lectura e interpretación constante de las variaciones de las señales vitales de un paciente, comunicando a las capas más inferiores que debe ejecutarse al momento en que se necesite.

Nombre	**AccommodationMobiPattern**
Contexto	Cómo ubicar cada módulo en la aplicación final es muy importante a la hora de definir secuencias de ejecución y prioridades en esas aplicaciones. Saber qué evento debe ejecutarse en un momento determinado facilita la intercomunicación entre cada módulo minimizando el tiempo de respuesta ante una solicitud hecha por el usuario.
Problema	La problemática radica en saber cómo ordenar la secuencia de ubicación de cada uno de los elementos que componen la aplicación. Es decir, asignar niveles prioritarios para saber en qué plano debe correr un elemento de nuestra arquitectura.
Usos	Para la generación de aplicaciones de todo tipo, en donde se requiera el menor consumo tanto de procesamiento como de almacenamiento del dispositivo.
Solución	Se definirán niveles prioritarios a cada módulo de la aplicación relacionándolos con otros, en donde su ejecución dependerá de elementos propios a cada módulo.
Fundamentos	Similar a la asignación de un modelo en capa, lo que facilita la comunicación y reasignación de todos sus elementos.
Relaciones	Patrón de ejecución, patrón de visualización y patrón de acomodación

Tabla 4-25. *Especificaciones del MobiPattern de Acomodación* ***(AccommodationMobiPattern)***

Al momento de desarrollar aplicaciones nos basaremos en este patrón para ubicar en tres niveles los módulos correspondientes a cada aplicación. Esto se hace con el objetivo de asignar prioridades de ubicación en el dispositivo para luego utilizarlas en el *MobiPattern* de ejecución.

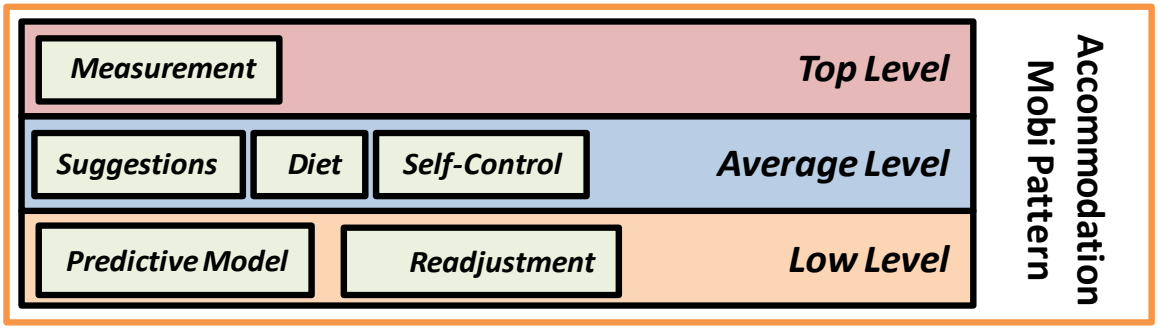


Figura 4-36. *Definición de la Pila de Niveles del MobiPattern de Acomodación.*

En la figura 4-37 se muestra los elementos que componen este *MobiPattern*.

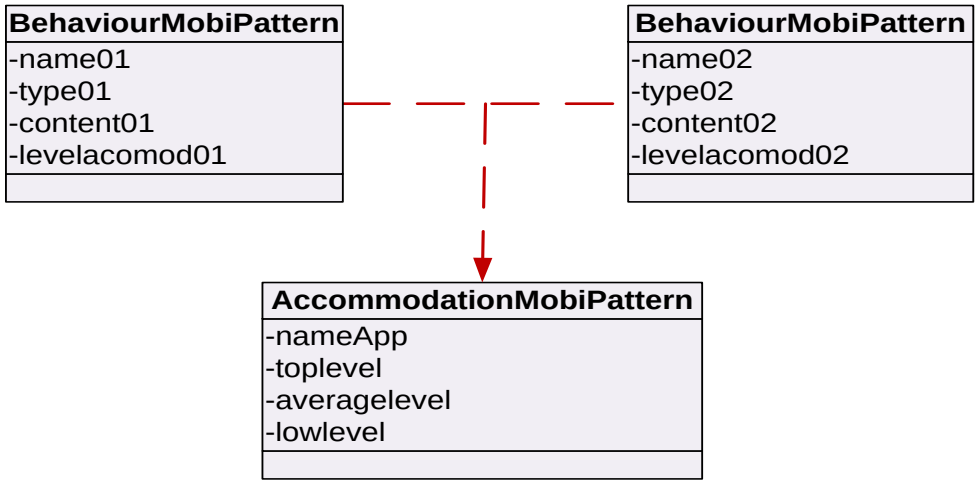


Figura 4-37. *Estructura del MobiPattern de Acomodación y la relación con otros MobiPattern.*

Este MobiPattern está formado por:

- **NameApp**: aquí se define el nombre de la aplicación que se va a generar para un determinado paciente.
- **TopLevel**: define la ubicación de un módulo en el nivel superior.
- **AverageLevel**: define la ubicación de un módulo en el nivel medio.
- **LowLevel**: define la ubicación de un módulo en el nivel inferior.

MobiPattern de Visualización (VisualizationMobiPattern)

Define los parámetros visuales de la aplicación al momento de ser generada para los diversos tipos de dispositivos móviles. Por un lado los elementos a evaluar son dimensión, resolución, tipo de dispositivo, y por el otro, hace referencia a las estructuras de salidas (Pantallas).

Nombre	***VisualizationMobiPattern***
Contexto	Definir el tamaño de la pantalla de salida, así como la resolución son algunos de los elementos importantes que se debe tener en cuenta al momento de generar aplicaciones para dispositivos móviles.
Problema	Hay que adaptar todos los elementos de salida para que puedan ser visualizados en cualquier dispositivo móvil, de tal manera que el paciente pueda utilizar la aplicación sin ningún problema.
Usos	Visualización de la información de forma correcta y organizada en dispositivos móviles pequeños. Mejora la interfaz de usuario en aplicaciones que requieren ser ajustadas a pequeñas dimensiones.
Solución	Definir elementos de ajustes para mostrar todos los elementos de la aplicación en el correspondiente dispositivo móvil. Una vez que esté generada la aplicación, este indistintamente del dispositivo debe poder ajustarse a las necesidades visuales del usuario.
Fundamentos	Similar a la distribución visual de dispositivos con mayores prestaciones, facilitando la navegación visual en pantalla.
Relaciones	Patrón de Ejecución y patrón de reajuste.

Tabla 4-26. *Especificaciones del MobiPattern de Visualización* ***(VisualizationMobiPattern).***

Estos elementos deben ser adaptables para ser visibles en cualquier dispositivo. Es decir ***¿Cual formato de salida es el más apropiado?*** En la tabla 4-26 se definen las especificaciones de este **MobiPattern**.

Este **MobiPattern** está relacionado a la ontología de dispositivos móviles que hemos definido en el capítulo anterior, de tal manera que se deben conocer todos los tipos, marcas, modelos y especificaciones de cada dispositivo para así adaptar la salida de la aplicación ajustada a exigencias técnicas de cada usuario.

En la figura 4-38 se observa la estructura del *MobiPattern* de visualización para la correspondiente generación de la aplicación que será instalada en el dispositivo móvil. Estos elementos son:

- **hwspec**: aquí se definen las características de *hardware* para el dispositivo móvil, definidas en la ontología de dispositivos móviles.

- **swspec**: aquí se definen las características del *software* que tiene el dispositivo móvil, también definida en la respectiva ontología del dispositivo móvil.
- **comspec**: aquí se definen las características de comunicación (envío y recepción de información) con que cuenta el dispositivo móvil.

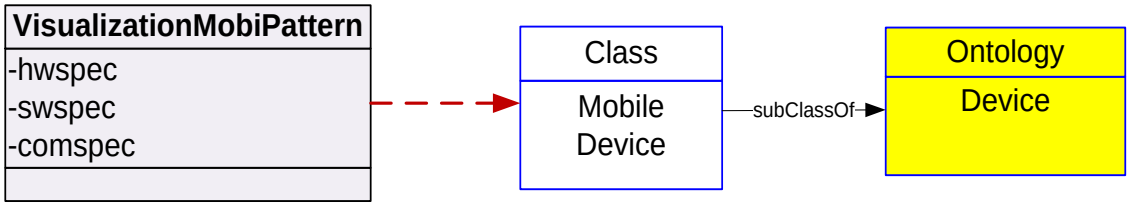


Figura 4-38. *Estructura del MobiPattern de Visualización y la relación con la ontología de dispositivos móviles.*

Para este MobiPattern, hemos definido una estructura genérica para todos los tipos de dispositivos móviles y que se ajusta a las prestaciones técnicas de los mismos. Generalmente en la mayoría de los dispositivos móviles nos preocupan los aspectos de visualización de la información, es decir cómo se comportan las interfaces y como se muestran los datos obtenidos de una consulta.

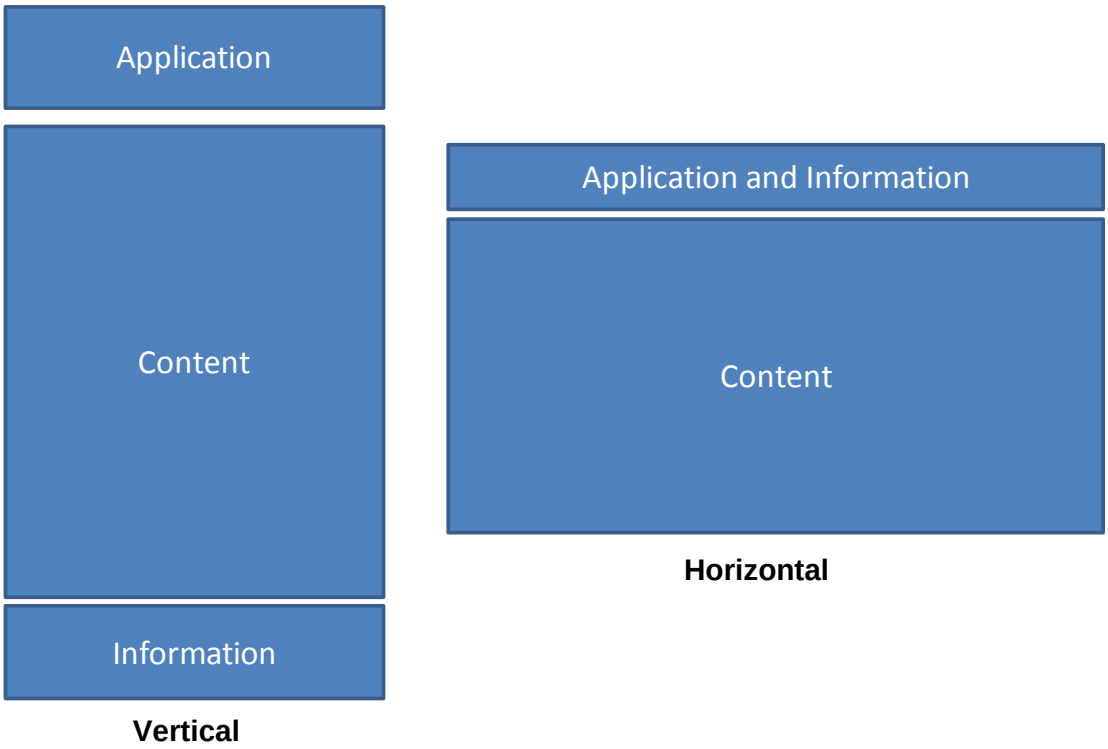


Figura 4-39. *Representación visual del MobiPattern de Visualización: vertical (izquierda) y horizontal (derecha.)*

En la figura 4-39, se muestra la estructura física del MobiPattern de visualización. Para este patrón se definen dos estructuras físicas dependiendo de la orientación del dispositivo: vertical y horizontal.

Los elementos que conforman esta estructura son:

- **Application**: especifica el nombre del paciente y datos principales.
- **Content:** muestra el comportamiento de la enfermedad según el formato elegido.
- **Information**: primera vista elegida (gráfica, tabla, etc.).

4.3.6 MobiPattern de Ejecución

El MobiPattern de ejecución determina el funcionamiento de la aplicación cuando este ejecutándose. Aquí se contemplan dos aspectos, el de ejecución que especifica la relación que existe entre cada módulo en ejecución y el de reajuste, es decir, hacer cambios a la aplicación inicialmente generada.

4.3.6.1 MobiPattern de Ejecución (ExecutionAppMobiPattern)

Este *MobiPattern* es de gran ayuda, ya que permite definir qué elementos se relacionarán entre sí luego de generada la aplicación para el dispositivo móvil. Se deben definir qué módulos son considerados para análisis, cual para procesamiento y cual para visualización de información.

Es aquí donde se ubica las funcionalidades previamente diseñadas por los *MobiPattern* de comportamiento. Es decir ***¿Qué elementos ejecutar para monitorizar al paciente?***

Aquí se definen los *MobiPattern* para generar dietas, autocontrol, sugerencias, recomendaciones, alertas, etc. En la tabla 4-27 se definen las especificaciones de este *MobiPattern*.

Nombre	***ExecutionAppMobiPattern***
Contexto	Luego de definida la funcionalidad de cada módulo, es necesario definir el orden de ejecución de esos elementos en la aplicación final. Definiendo que elementos se ejecutan en primer plano y cuáles no.
Problema	Una aplicación debe estar bien desarrollada de tal manera que cada proceso sepa en qué orden de ejecución se encuentra e identificar sus procesos predecesores y antecesores para el envío y recepción de datos. Si no se cuenta con una correcta definición de flujo de ejecución, la aplicación podría aumentar su grado de latencia e inclusive generar errores.
Usos	Para definir secuencia y niveles de ejecución al momento de generar aplicaciones para cualquier dispositivo.
Solución	La solución consiste en definir niveles de ejecución prioritarias para cada elemento de la aplicación de tal manera cada uno de los módulos tenga definido esa prioridad a la hora de generar la aplicación. De igual forma se define la relación y comunicación entra cada uno de esos módulos.
Fundamentos	El principal fundamento de la definición de patrones de ejecución radica en la organización de todo el procesamiento a través de un flujo organizado de cada uno de los procesos. Se evita con ello desbordamiento o ruptura de la aplicación.
Relaciones	Patrón de Acomodación y Patrón de Visualización

Tabla 4-27. *Especificaciones del MobiPattern de Ejecución* ***(ExecutionAppMobiPattern).***

En la figura 4-40, se muestra la estructura del *MobiPattern* de ejecución, en donde existe un número de módulos con funcionalidades de determinadas (*BehaviourPattern01,*

BehaviourPattern02, BehaviourPattern03, BehaviourPattern04) las cuales deben tener un alto grado de cohesión, facilitando su funcionamiento.

En la tabla 4-27 se han definidos las siguientes relaciones existentes entre cada módulo de tal manera que se pueda delimitar las funcionalidades al momento de generar aplicaciones:

Relación		Módulos que intervienen
recommended_a		Suggestion, MedicateTreatment, Diet, PatientProfile
present_a		Education, SelfControl, PatientProfile
organized_to		Diet, Food
compounded_by		PatientProfile
is_part_of		Diet, Suggestions, Education, prevention, Disease.
has_a		PatientProfile, Diseases, MedicateTreatment
depend_of		Measure, Suggestion, Prevention, Education, Diet.

Tabla 4-28. *Relación entre módulos utilizando el MobiPattern de ejecución.*

Para ello se utiliza un *ExecutionAppMobiPattern* que define esas relaciones a nivel de ejecución. Estos elementos son:

- **Typerelation**: define el tipo de relación que se va a utilizar al momento de enlazar cada módulo.
- **Typepattern**: aquí se define que elementos o módulos se van a comunicar o enlazar con este tipo de relación. Se define previamente relaciones para cada módulo, lo que facilita la generación previa de la aplicación final.
- **Numberelement**: define los elementos que intervienen en esa relación, es decir si existe la posibilidad de tener otras relaciones luego de realizadas las anteriores. Una relación entre un módulo y otro puede generar una nueva relación dependiente de otra.

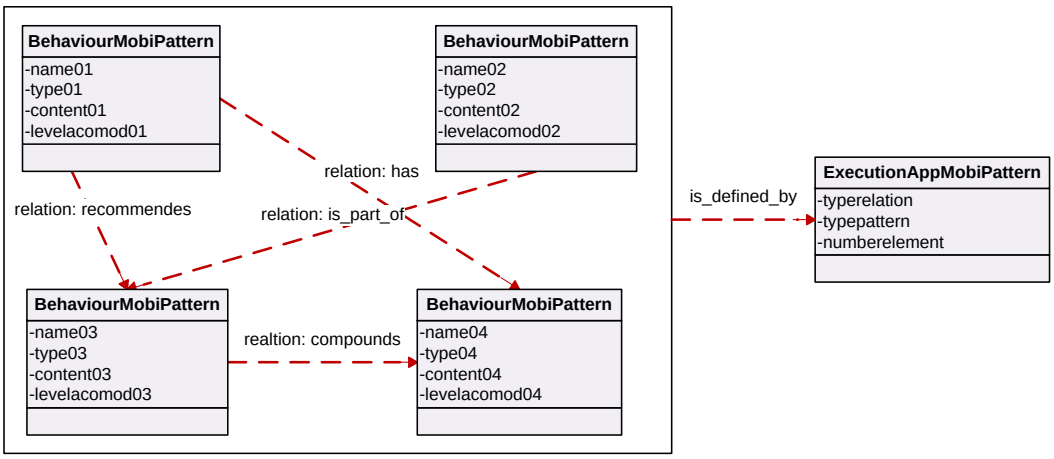


Figura 4-40. *Estructura del MobiPattern de Ejecución y la relación con el MobiPattern Comportamiento.*

4.3.6.2 MobiPattern de Reajuste (ReadjustmentAppMobiPattern)

Luego de generada la aplicación de monitorización para un paciente, y tomando en cuenta que pueden surgir variaciones o alteraciones significativas de las alteraciones en la situación médica del paciente, el *framework* debe ser capaz de redefinir o reajustar la aplicación previamente generada en el dispositivo móvil. Es decir ***¿Qué cambiar o actualizar?*** y ***¿Cuándo hacerlo?*** Esto permite tener una aplicación actualizada ante el comportamiento de la enfermedad de cada paciente. En la tabla 4-29 se definen las especificaciones de este *MobiPattern*.

Como se muestra en la figura 4-41, el *MobiPattern* reajuste tiene la función de generar nuevos módulos o reajustes a la aplicación previamente generada para el dispositivo móvil, de tal manera que si se necesita hacer cambios a la aplicación original, ésta sepa en donde hacer esos cambios. Mencionando esto último, los cambios se pueden hacer a toda la aplicación, generándola nuevamente o a cierto módulos, actualizándolos en el dispositivo móvil. Este reajuste se puede hacer en dos momentos determinados, ya sea que se genere una alerta de suma urgencia y se solicite el reajuste inmediato o que los cambios no sean de tanto nivel de importancia y que se pueda esperar la próxima visita al doctor.

En ambos casos la aplicación de monitorización, almacena cualquier necesidad de cambio o reajuste que haya que hacerse. Luego el *framework* revisa si se ha generado algún módulo de reajuste consultándole a la aplicación, y esta automáticamente comunica si se necesita o no (**needchange**, *true* o *false*) y se hacen las actualizaciones oportunas (**namemodule**). Cuando se ha actualizado todos los elementos necesarios, se procede a actualizar la aplicación o módulos previamente generados en el dispositivo móvil.

Nombre	***ReadjustmentAppMobiPattern***
Contexto	Muchas veces las aplicaciones luego de generadas necesitan ser reajustadas y más cuando estas aplicaciones se encargan de la monitorización de pacientes. Esos reajustes se hacen en la funcionalidad de cada módulo ya sea debido a que las lecturas de las señales del paciente hayan mejorado o por el contrario hayan tenido complicaciones.
Problema	Si se genera una aplicación que se embebe en un dispositivo móvil una única vez, la funcionalidad de nuestro *framework* sería muy limitada de tal manera que la aplicación se quedaría obsoleta inmediatamente. Para monitorización de pacientes, una aplicación obsoleta no cubre las necesidades para las cuales fue creada.
Usos	Se hacen cambios a cada módulo que así lo requieran ya sea ante cambios abruptos o mediante actualizaciones periódicas programadas al momento de hacer una visita al doctor.
Solución	La solución consiste en analizar la frecuencia con que un módulo ha sido llamado y evaluar la posibilidad de reajustarse. Luego si ese reajuste es muy importante se decide si se solicita el cambio al servidor o si se espera la próxima visita con el doctor. Esta decisión depende de cómo se han comportado los niveles de las señales vitales en un determinado tiempo.
Fundamentos	El principal fundamento de este patrón radica en la facilidad que ofrece el hacer ajustes a una aplicación ya generada previamente, ofreciendo nuevas posibilidades ante cambios en la funcionalidad inicial de cada módulo.
Relaciones	Patrón de Comportamiento

Tabla 4-29. *Especificaciones del MobiPattern de Reajuste* ***(ReadjustmentAppMobiPattern)***

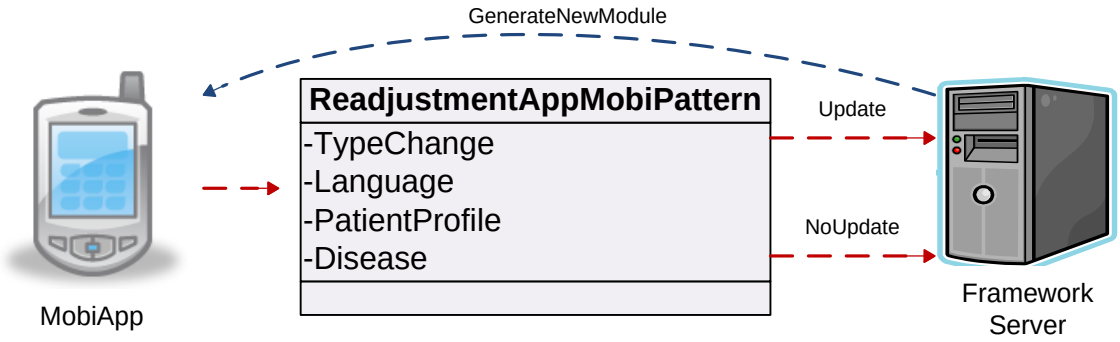


Figura 4-41. *Estructura del MobiPattern de Reajuste y su funcionamiento en el framework.*

En la figura 4-42, se muestra la estructura física del MobiPattern de reajustes. Un ejemplo de aplicación del MobiPattern de reajuste está compuesto por los siguientes elementos:

- **Language**: selecciona el leguaje de la aplicación.
- **Diseases:** selecciona la enfermedad a monitorizar, en caso que se tenga que reajustar una nueva enfermedad al control de monitorización.
- **Profile: actualizar** datos del perfil de paciente.
- "**Action**": botón de acción para el patrón.

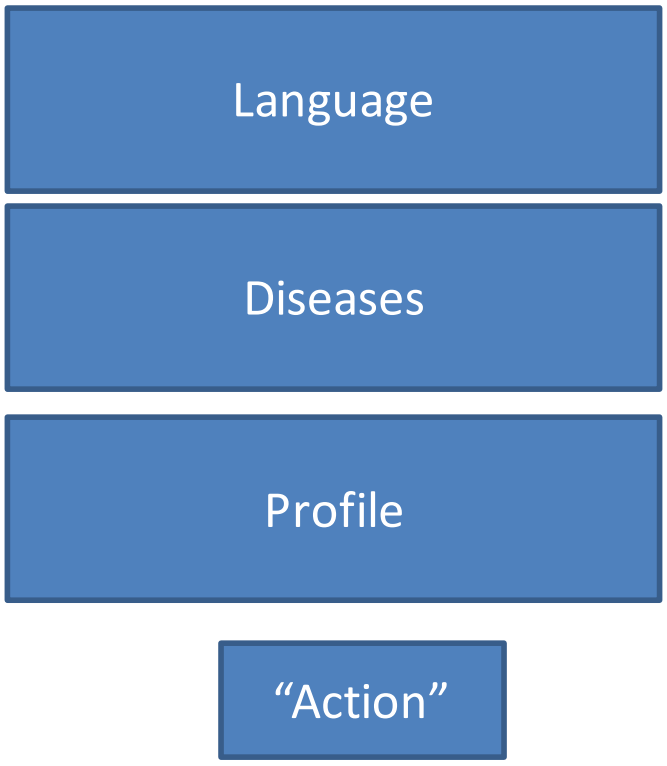


Figura 4-42. *Representación visual de reajuste de aplicación.*

4.3.7 Conclusiones

En este apartado se explicaron aspectos referentes al diseño de patrones para el desarrollo de aplicaciones móviles para la monitorización de pacientes.

Cada patrón especifica la arquitectura *software* completa en aspectos de *diseño*, *funcionalidad* y *ejecución*. Para cada patrón se ha definido una estructura funcional y la relación existente entre cada patrón y las ontologías desarrolladas.

Nos hemos basado en las definiciones conceptuales sugeridas por los expertos en cuanto al uso de patrones, ubicándonos en una organización más superior que contempla aspectos de patrones de arquitectura y de diseño.

5

CAPÍTULO QUINTO

5 IMPLEMENTACIÓN DE LA ARQUITECTURA SOFTWARE A PARTIR DEL FRAMEWORK

En este capítulo, presentamos la implementación de la arquitectura *software* desarrollada a partir del *framework* **MoMo** para la generación de aplicaciones móviles. Relacionaremos las definiciones de *MobiPattern* utilizados y las ontologías definidas para cada caso, así como un esquema del módulo que relaciona ambos aspectos.

La arquitectura software es una solución tecnológica formada por un servidor y una aplicación móvil adaptable y parametizable para múltiples contextos. La arquitectura desarrollada en esta tesis, está distribuida en dos áreas específicos. Una corresponde a una aplicación desarrollada para ser instalada en el teléfono móvil del paciente y del médico. Y la otra correspondiente a los elementos distribuidos en el servidor principal, que contiene los datos y ontologías definidas previamente. La arquitectura móvil ha sido desarrolla en Android 2.2, como sistema operativo para dispositivos móviles. Además se ha implementado la conectividad a través de una base de datos en MySQL, la cual es accedida por servicios web definidos. Estos servicios web han sido desarrollados basados en ksoap2[9,10].

El *framework* tiene un carácter general, lo que permite su aplicación en diferentes entornos médicos y usando diversas tecnologías. Para estas primeras aproximaciones, el desarrollo de la

[9]Fuente: http://ksoap2.sourceforge.net/
[10]Fuente: http://code.google.com/p/ksoap2-android/

arquitectura ha sido realizado para la monitorización de enfermedades con un máximo de dos unidades de medidas.

En este capítulo se definen las relaciones existentes entre cada uno de los elementos mencionados en el diseño conceptual. Se detalla el diseño de las interfaces, basada en los *MobiPattern* funcionales y de diseño, la relación ontológica y la estructura de comunicación que se ha desarrollado. Además se establecen los siguientes conceptos adicionales:

- **Zona Contenedora**: es la zona que contendrá el *MobiPattern* de diseño explicado en el capítulo anterior. Cada interfaz de usuario cuenta con una zona contenedora, que se diferencia de las demás interfaces de usuario, por el *MobiPattern* que se implemente. Se ha definido esta zona para la mayoría de la interfaces para facilitar el diseño posterior y la escalabilidad de la arquitectura.
- **Esquema y relación del patrón**: aquí se define el esquema que compone la interfaz de usuario, basada en los patrones definidos. Se agrega cada patrón a la zona contenedora para generar las interfaces propias a cada módulo.
- **Origen y Flujo de datos**: aquí se definen los flujos de datos que intervienen en cada una de los módulos o interfaces desarrollados. La estructura ontológica define en gran parte esta estructura, lo que nos permite identificar la relación patrón-interfaz-ontologías.

En la figura 5-1, se muestra el proceso de desarrollo a seguir. Dicho proceso se basa en la integración de todos sus elementos hasta la obtención de un prototipo, que nos permite evaluar la funcionalidad de la arquitectura.

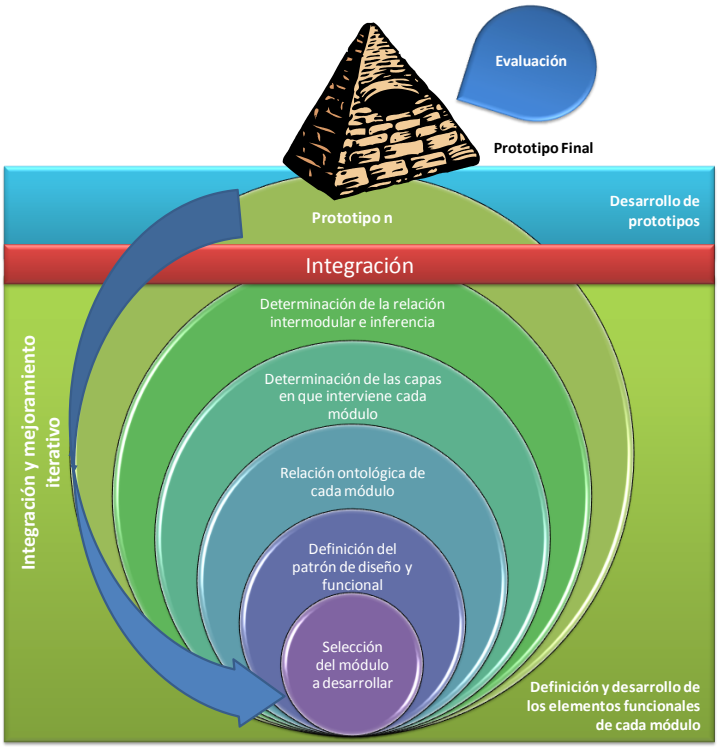


Figura 5-1. *Modelo de desarrollo de aplicaciones según los elementos del framework.*

En el proceso de implementación se obtienen diferentes prototipos funcionales, que definen cada elemento o módulo que compondrá la aplicación final. Los pasos de implementación son los siguientes:

1. **Selección del módulo a implementar**: se diseña la estructura funcional de cada módulo que formará parte de la aplicación final. Cada módulo tiene asociada una funcionalidad específica, basada en un diseño general.

2. **Definición de patrones de diseño**: se definen las estructuras físicas de cada patrón asociado a cada módulo que se ejecutará. Se especifica el diseño visual de cada uno de los módulos.

3. **Definición de patrones funcionales:** se definen las funciones y relaciones de cada uno de los módulos de la aplicación.

4. **Relación ontológica de cada módulo**: se especifican las ontologías que intervienen o son utilizadas por cada módulo, así como la relación entre otros elementos de la arquitectura.

5. **Determinación de las capas en que interviene el módulo**: se define la capa funcional de cada módulo, relacionándolo con el modelo en capas, definido por el *framework*.

6. **Determinación de la relación intermodular**: se definen las relaciones entre cada uno de los módulos desarrollados, permitiendo la interoperabilidad entre cada uno de ellos.

7. **Integración de todos los elementos**: para la obtención del prototipo a evaluar. Se obtiene en este paso, un prototipo inicial.

8. **Evaluación del prototipo**: esto permite la evaluación funcional y de diseño visual de la aplicación generada, facilitando la realimentación para el mejoramiento de la arquitectura.

9. **Rediseño de los elementos para la generación de un nuevo prototipo**: es aquí en donde se analiza la funcionalidad del prototipo obtenido, para su rediseño según los pasos iniciales. Se inicia el proceso completo, cada vez que se encuentren problemas de diseño.

Luego de implementados todos los elementos que componen la arquitectura, se mostrarán algunos escenarios en donde se probará el prototipo final, permitiendo así demostrar y comprobar los objetivos previamente definidos en esta tesis.

El *framework* puede ayudar en gran parte al desarrollo organizado de arquitecturas *software*, ya sea para desarrolladores que quieran implementar algunos de sus elementos, modificar los ya existente o simplemente utilizar la metodología de desarrollo que planteamos con los pasos definidos.

5.1 DEFINICIÓN DE REQUISITOS FUNCIONALES

En este apartado definimos los requisitos funcionales para la generación de aplicaciones basadas en el *framework* desarrollado. Los requisitos de la aplicación móvil se muestran en la siguiente tabla:

Código	Requisito funcional	Descripción
RF01	Obtención de las medidas provenientes de los distintos dispositivos biométricos	La aplicación debe obtener las medidas provenientes de los distintos dispositivos biométricos ya sea de manera automática o manual.
RF02	Introducción de aspectos relacionados con la toma de medidas	La aplicación permitirá la introducción de aspectos significativos a la toma de medidas para su mejor análisis.
RF03	Configuración de la aplicación	La aplicación permitirá configurar ciertos aspectos de la aplicación así como de parámetros relacionados a la toma de medidas.
RF04	Comunicación con el servidor remoto	La aplicación podrá comunicarse con el servicio web alojado en un servidor remoto para el envío y recepción de datos. Si la aplicación no tuviera conexión a Internet, la aplicación guardará de manera local dichos datos y los sincronizará cuando pueda.
RF05	Envío de mensajes en situaciones de riesgo	La aplicación detectará posibles situaciones de riesgo e informará de ellas tanto al paciente como a terceras personas.
RF06	Visualización de recomendaciones y avisos	La aplicación mostrará al usuario mensajes con sugerencias y recomendaciones.
RF07	Visualización de las últimas tomas de medidas	La aplicación mostrará (en forma de gráfica o de tabla) las últimas mediciones introducidas por el paciente.
RF08	Configuración de enfermedades para monitorizar	El paciente podrá configurar, añadir y eliminar las enfermedades que se monitorizarán desde la aplicación móvil.
RF09	Configuración de datos del paciente	El paciente podrá insertar, visualizar y modificar sus datos personales en relación a la aplicación móvil.
RF10	Configuración de dispositivos biométricos	La aplicación soportará la adición, eliminación y visualización de los distintos dispositivos biométricos que se emparejarán con la aplicación móvil.

Tabla 5-1.*Especificación de los requisitos funcionales de la aplicación móvil.*

A continuación se describen los requisitos funcionales y su correspondencia con los distintos casos de usos.

Requisito funcional	Casos de uso	Descripción del caso de uso
RF 01	CU-01 Realizar medición automática	La aplicación se conectará al dispositivo biométrico adecuando y obtendrá la medición
	CU-02 Realizar medición manual	La aplicación recibe de manera manual (por teclado) por el usuario la medición.
RF 02	CU-03 Introducción del momento en el que se toma la medida	El usuario escogerá el momento en el que se produjo la medición manual.
RF 03	CU-04 Selección de la metodología de entrada	Selección de la manera en la que la aplicación recibirá los datos de las mediciones.
	CU-05 Selección del idioma de	Se podrá escoger el idioma en el que se muestra la

	la aplicación	interfaz de la aplicación.
RF 04	CU-06 Captura de datos desde el servidor remoto	En este caso la aplicación recibirá datos desde el servidor remoto a través del servicio web.
	CU-07 Envío de datos al servidor remoto	La aplicación enviará datos al servidor remoto por medio del servicio web.
RF 05	CU-08 Envío de SMS a contacto del paciente	En el caso en el que la aplicación lo requiera, la aplicación enviará un SMS al un contacto.
RF 06	CU-09 Visualización de mensajes de prevención relacionados con la alimentación	La aplicación mostrará mensajes de prevención sobre la enfermedad escogida y en relación a la alimentación.
	CU-10 Visualización de mensajes de prevención relacionados con el ejercicio físico	La aplicación mostrará mensajes de prevención sobre la enfermedad escogida para introducir la medida y en relación al ejercicio físico.
	CU-11 Visualización de mensajes de prevención relacionados con la medicina	La aplicación mostrará mensajes relacionados con aspectos médicos y con la enfermedad escogida.
	CU-12 Visualización de mensajes de prevención educativos	La aplicación mostrará mensajes educativos con aspectos relacionados con la enfermedad escogida.
RF 07	CU-14 Visualización en forma de tabla de los resultados de las últimas tomas	La aplicación mostrará en forma de tabla la última toma de la enfermedad escogida.
	CU-15 Visualización en forma de gráfica de los resultados de las últimas tomas	La aplicación mostrará en forma de gráfica las últimas medidas obtenidas de la enfermedad seleccionada.
RF 08	CU-16 Adición de enfermedades para monitorizar	La aplicación permite la adición de enfermedades para monitorizar.
	CU-17 Adición de condiciones a la enfermedad introducida	A la enfermedad que se está introduciendo se le podrán añadir condiciones para el cálculo de rangos de las medidas.
	CU-18 Adición de valores barrera a la enfermedad introducida	A la enfermedad que se está introduciendo se le ha de añadir la unidad de medida a la hora de la realización de las tomas.
RF 09	CU-19 Visualización de los datos del paciente	La aplicación mostrará los datos del paciente que se registró en la aplicación.
	CU-20 Modificación de los datos del paciente	La aplicación permite la modificación de parte de los datos del paciente.
	CU-21 Introducción de los datos del paciente en la aplicación móvil	En el inicio, la aplicación exige la introducción de los datos personales del paciente que utilizará la aplicación.
RF 10	CU-22 Búsqueda de dispositivos biométricos para emparejar con la aplicación	La aplicación será capaz de buscar dispositivos biométricos mediante Bluetooth.
	CU-23 Adición de dispositivos biométricos asociados a una enfermedad	El usuario podrá asociar un dispositivo biométrico a varias enfermedades, de tal manera que las medidas de dicha enfermedad se tomarán con el dispositivo asociado.
	CU-24 Eliminación de dispositivos biométricos asociados a una enfermedad	De la misma manera, la aplicación permitirá desasociar un dispositivo de una enfermedad.

Tabla 5-2.*Especificación de los requisitos funcionales de la aplicación móvil y su relación con los diversos casos de uso.*

5.2 DEFINICIÓN DE LOS AJUSTES INICIALES DE LA ARQUITECTURA – ROL PACIENTE

Lo primero que debe configurarse al usar la aplicación por primera vez, son los parámetros iniciales que definen la funcionalidad de la aplicación. Para la definición de estos ajustes o parámetros iniciales se usa el patrón de reajuste (*ReadjustmentAppMobiPattern*).

En la tabla 5-3, se muestran los elementos del *framework* que se utilizan para el desarrollo de los ajustes iniciales de la aplicación, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a desarrollar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Ajustes iniciales	ReadjustmentAppMobiPattern	-Disease	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	**- RF02** (CU-03), **- RF03** (CU-04, CU-05), **- RF04** (CU-07), **- RF08** (CU-16, CU-17, CU-18)
	ProfileMobiPattern	-PatientProfile		**- RF09** (CU-19, CU-21)

Tabla 5-3.*Elementos del framework utilizados para la generación de los ajustes iniciales.*

Este patrón puede ser utilizado al inicio para hacer cambios totales en la instalación de la aplicación y puede ser usado en cualquier otro momento para hacer reajustes a la aplicación final. Algunos ajustes que se pueden hacer serían: *el idioma de uso de la aplicación*, *ajustar los parámetros de una enfermedad* ya agregada o una nueva enfermedad y *cambios al perfil del paciente* inicialmente definidos. En la figura 5-2 se muestra la relación entre el patrón y las ontologías utilizadas

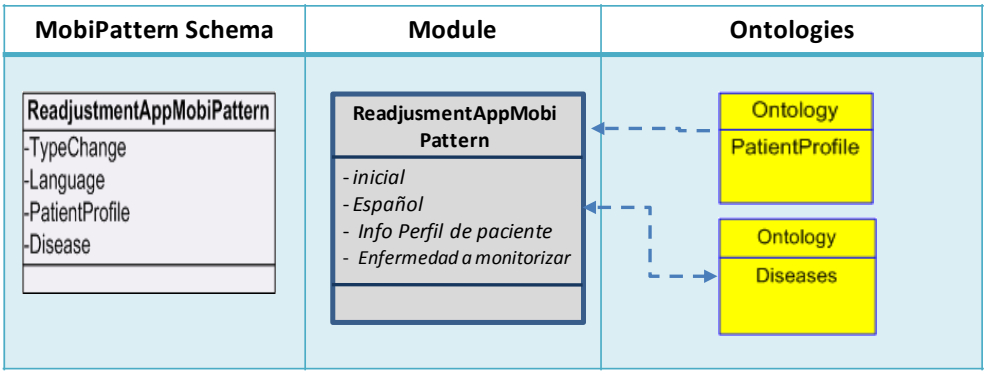


Figura 5-2. *Implementación del MobiPattern de parámetros iniciales y su relación con las ontologías de perfil de paciente y enfermedades.*

En la figura 5-3 se muestra la relación del módulo de parámetros generados y su relación física y estructural con el MobiPattern definido previamente.

Figura 5-3. *Generación parámetros de ajustes iniciales y su patrón correspondiente.*

Un paciente puede definir una o más enfermedades a monitorizar. Cada enfermedad es administrada por la aplicación con las mismas interfaces estructuras gráficas, pero variando la funcionalidad de cada una de ellas. Esto permite a la aplicación implementar múltiple tipos de enfermedades realizando los cambios mínimos o nulos a la aplicación inicialmente desarrollada. Los parámetros que se configuran en el inicio son los siguientes:

- **Idioma de la aplicación**: la aplicación por defecto se muestra en el idioma en el que esté el dispositivo móvil. Además puede elegirse desde este menú uno de los idiomas existentes (inglés y español).
- **Adición de enfermedades**: para el uso de la aplicación, se debe añadir al menos una enfermedad que se monitorizará (pudiendo más tarde añadir una nueva desde el menú principal). Desde esta opción se accederá a una pantalla donde se seleccionará la enfermedad (de una lista inicialmente almacenada) y la manera en que se leerán las medidas (*manual* por teclado o *automática* desde el dispositivo biométrico).
- **Adición del perfil de paciente**: inicialmente, y para el correcto funcionamiento de la aplicación, también se ha de añadir un perfil de paciente. Desde ésta opción se accede a las pantallas de introducción de perfil de paciente.

Para que el usuario tenga una mejor interpretación visual del estado de configuración de éstos parámetros, en la parte derecha de cada elemento del menú se ofrece un icono visual que muestra el estado de configuración de ese elemento.

5.2.1 Esquema y relación del patrón

En este caso, nos encontramos con una interfaz que no sigue el formato del resto de las interfaces. En las pantallas que se explicarán posteriormente, se mostrará información en una barra de navegación en donde se cargan interfaces según ciertos parámetros, y en la zona inferior se encontrarán elementos para la confirmación de ciertas acciones. En el caso de la pantalla que nos ocupa, nos encontramos con una zona de información demasiado grande para que ocupe solo una pequeña barra en la parte superior de la pantalla, y un menú desde el cual se llevarán a cabo diversas acciones. Por lo tanto, el esquema de la pantalla de instrucciones queda definido de la siguiente manera mostrada en la figura 5-4.

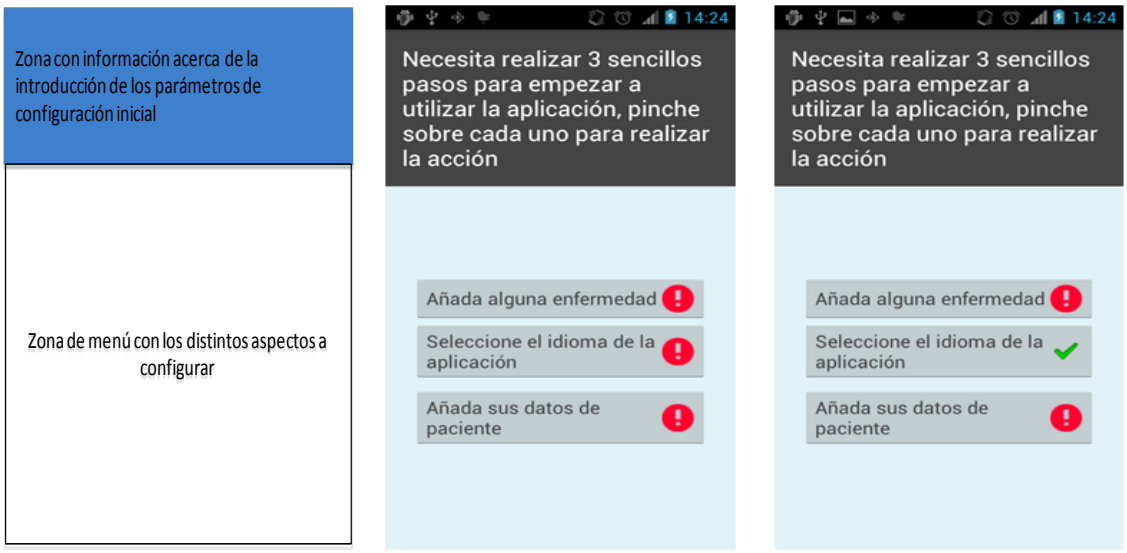


Figura 5-4. *Esquema de interfaz y principales pantallas de la configuración de parámetros iniciales.*

Como se puede apreciar y se comentó anteriormente, los elementos de la configuración de los parámetros iniciales proporcionan información acerca del estado de configuración de las acciones que representan.

En la imagen del medio, ninguna de las acciones del menú está terminada, y se representa con un icono de advertencia. Por el contrario, en la imagen de la derecha se observa como la selección de idioma está terminada, y se representa con un *"tick"* que nos permite identificarlo. En la parte superior de las pantallas, se muestra un texto informativo sobre los pasos que se han de realizar para terminar la configuración inicial.

5.2.2 Origen y flujo de datos.

El modelo define de forma general el comportamiento de la interfaz de usuario y las acciones de los elementos del menú de las configuraciones iniciales. Las opciones de añadir enfermedad y añadir datos del paciente se explicarán más adelante en sus respectivos apartados.

En la opción de seleccionar el idioma de la aplicación se muestra un pequeño diálogo en el que se listan los idiomas disponibles para que el usuario seleccione el que es de su preferencia.

Una vez seleccionado, se graba en la configuración de la aplicación y, en caso de ser elegida una opción distinta a la que viene por defecto en el dispositivo móvil (idioma del dispositivo), se reinicia la pantalla para que los cambios tengan efecto.

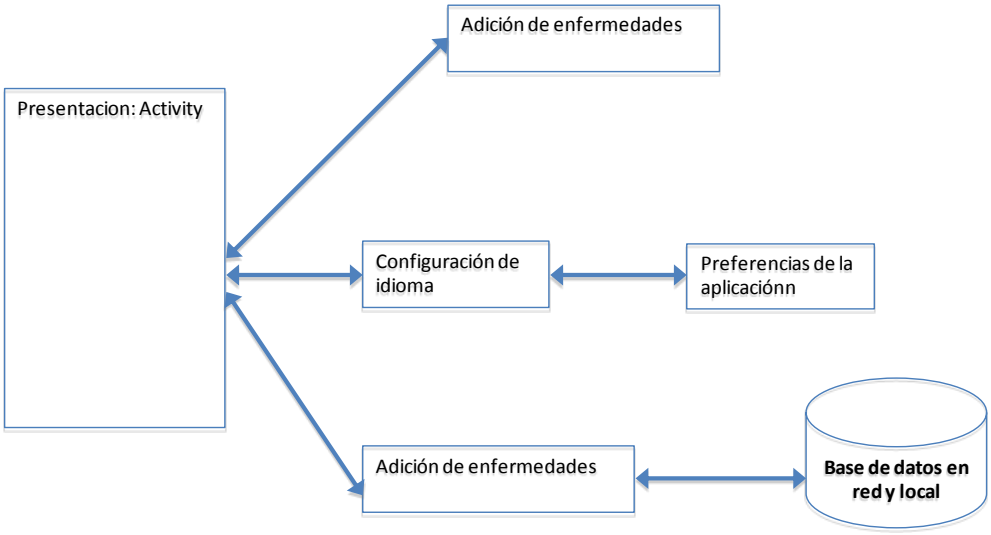


Figura 5-5. *Flujo de datos para la selección de idioma en los parámetros iniciales de la aplicación.*

En la figura 5-5 se muestra el flujo de datos para la opción *selección de idioma* de la aplicación y se dejan señaladas las opciones para añadir una enfermedad y añadir perfil de paciente que como mencionamos previamente se explicarán más adelante.

5.3 GENERACIÓN DEL MENÚ O PANTALLA PRINCIPAL

Una vez definidos los parámetros iniciales de la aplicación, se genera un menú o pantalla principal, que define cada unas de las opciones que tiene la aplicación. En la pantalla principal nos encontramos con seis funciones y algunas más al activar la tecla menú del dispositivo móvil.

En la tabla 5-4, se muestran los elementos del *framework* que se utilizan para el desarrollo del menú de la aplicación, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Desarrollo del Menú	MenuMobiPattern	-Disease -PatientProfile	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	**- RF03** (CU-04)

Tabla 5-4.*Elementos del framework utilizados para la generación del menú principal.*

En la figura 5-6, se muestra la relación del patrón de menú, con las ontologías que intervienen en su desarrollo.

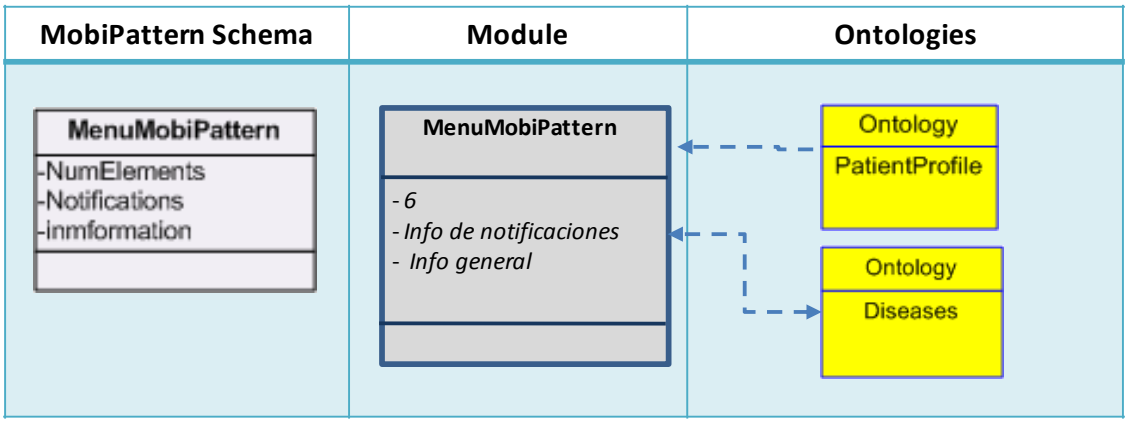


Figura 5-6. *Implementación del MobiPattern de Menú y su relación con las ontologías de perfil de paciente y enfermedades.*

En la figura 5-7 se muestra la interfaz gráfica de la aplicación, es decir, el menú principal y su relación con el patrón definido. Los elementos que componen esta pantalla son:

- **Perfil del paciente**: Permite visualizar y modificar los datos previamente creados del perfil del paciente. Este perfil sirve de información base para cada médico. En esta opción podemos visualizar los datos del usuario de la aplicación así como modificarlos.

- **Nueva enfermedad**: Permite agregar una nueva enfermedad a monitorizar. Esto permite la idea inicial de multi-monitorización de enfermedades. Además de las enfermedades almacenadas en la aplicación, se puede agregar nuevas enfermedades. Sólo el médico puede agregar una nueva enfermedad con sus parámetros en la aplicación.

- **Autocontrol**: Permite visualizar en formato *gráfico* y *tabla* las últimas mediciones obtenidas del dispositivo biométrico. Puede verse el gráfico de cada enfermedad agregada facilitando el seguimiento de las tomas de medidas de las distintas enfermedades que se monitorizan.

- **Sincronizar dispositivos Bluetooth**: Permite la sincronización entre el dispositivo móvil y los dispositivo biométricos, que permitan el acceso a la obtención de los datos. En esta sección se podrán añadir a las distintas enfermedades que monitoriza la aplicación los distintos dispositivos biométricos Bluetooth. De esta manera cuando se activa el dispositivo Bluetooth automáticamente se comunica con la aplicación y añade la medida obtenida a la enfermedad a la que esta enlazada dicho dispositivo. Una vez sincronizado el dispositivo biométrico al dispositivo móvil, la aplicación recibirá las últimas medidas del paciente.

- **Añadir nuevas medidas**: Podremos añadir nuevas medidas desde el apartado *nuevas medidas*. La entrada de dichas tomas puede realizarse de manera automática o manual, según como esté configurada la aplicación. Una vez el dispositivo biométrico obtiene la

medida del paciente, automáticamente se inicia la aplicación para la recepción de estas medidas.

- **Añadir nueva actividad física**: permite agregar actividades físicas que el paciente vaya a desarrollar. Se muestra una lista de las actividades físicas clasificadas por la *duración*, de tal manera que se puedan generar recomendaciones a cada una de ellas.

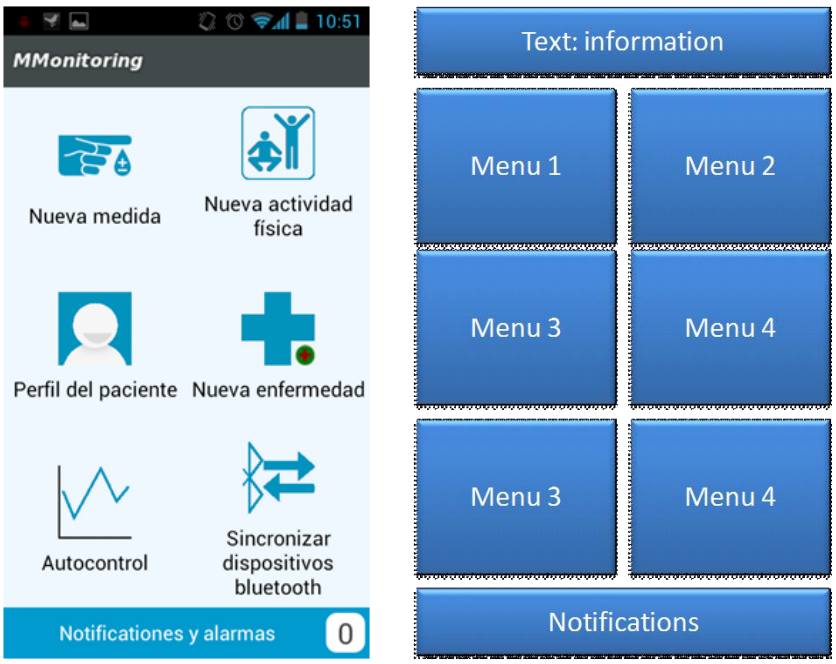


Figura 5-7. *Generación de menú y su respectivo patrón.*

- **Situado en la barra inferior nos encontramos con la zona de notificaciones y alarmas,** que nos muestra a simple vista si tenemos alguna *alarma* pendiente (creada para controlar la entrada de una medida considerada de riesgo) o *notificaciones* (mensajes del médico, etc.). Desde esa zona se puede acceder a una pantalla donde se muestra con más claridad dichas notificaciones.

- **Tecla Menú**: desde la tecla menú se puede acceder a otros apartados de la aplicación. Estos apartados contemplan aspectos como:

    - **Preferencias**: donde se podrán modificar parámetros de la aplicación tales como escoger la entrada de datos (manual o automática) y el idioma de la aplicación.

    - **Acerca de**: desde donde se accederá a la información acerca de la aplicación, ayuda e información sobre sus desarrolladores.

Cabe destacar que en los accesos a la adición de medidas y de autocontrol se despliega un pequeño submenú en el que se selecciona la enfermedad para la que se realizará dicha tarea. Mencionamos previamente que la aplicación puede monitorizar más de una enfermedad, por tal razón debe diferenciar entre cada una de ellas al momento de generar los gráficos y valores de autocontrol.

5.3.1 Esquema y relación del patrón

Vista la estructura general de la interfaz de usuario comparada con el patrón que lo define, podemos observar tres zonas:

- **Barra de navegación**: elemento de la interfaz que muestra información de distinto carácter (esta interfaz se usa principalmente para no romper el "*look&feel*" o estructura visual de la aplicación).
- **Zona de elementos de menú**: zona de la interfaz donde irán alojados los distintos elementos de la interfaz que representan acciones.
- **Zona de alarmas y notificaciones**: zona dedicada a la visualización de las distintas alarmas y notificaciones que la aplicación mostrará.

Barra de navegación

La barra de navegación conforma un pequeño espacio dentro de la interfaz que muestra información relevante de la aplicación. En algunos casos, esta información es de carácter *navegacional* y otras de carácter *informativo*. En esta interfaz, la barra de navegación tiene un carácter únicamente informativo, mostrando simplemente el logotipo de la aplicación como puede verse en la figura 5-8.

Figura 5-8. *Barra de navegación con información de la aplicación.*

La barra de navegación es un elemento visual que se auto añade a la interfaz mediante el uso de librerías que dan soporte a la adición de patrones de diseño de interfaces, por lo que para la adición de la barra de navegación en el menú principal únicamente debemos añadir la siguiente instrucción a nuestro código:

```
setActionBarContentView(R.layout.dashboard);
```

Con esta instrucción, la interfaz añadirá una barra como la que se muestra en la imagen de arriba, independientemente de cuál sea el archivo *XML* que define nuestra interfaz.

Zona de elementos de menú

En esta zona de la interfaz irán alojados las distintas acciones que el usuario/médico podrá realizar, mostrándolas de manera clara y a simple vista.

A nivel de diseño, esta zona está constituida por una *cuadrícula* en la que serán insertados otros elementos de la interfaz que serán mencionados más adelante. Dicha cuadrícula está diseñada para ocupar todo el espacio disponible que queda definido entre la barra de navegación y la zona de notificaciones, y se adapta a los elementos que la conforman. Cada elemento de la cuadrícula está formado por un icono representativo de la acción, y un pequeño texto que lo define.

Listado 5-1. *Estructura XML que define la zona de los elementos del menú, para cualquier enfermedad.*

```
<RelativeLayout
      android:id="@+id/button2"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1" >
      <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_centerInParent="true"
          android:orientation="vertical" >
          <ImageView
              android:id="@+id/ImageB2"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="center"
              android:background="@null"
              android:src="@drawable/sync" />
          <TextView
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_below="@id/ImageB2"
              android:layout_gravity="center"
              android:layout_marginTop="8dip"
              android:gravity="center"
      </LinearLayout>
  </RelativeLayout>
```

En el listado 5-1 podemos identificar cada elemento de la cuadrícula definido por la siguiente estructura *XML*. Esto quiere decir que cada elemento del menú que se quiera adicionar, debe seguir el estándar establecido por el *XML* que lo define, ajustando solamente los elementos que especifican ese nuevo menú.

Zona de notificaciones

Como último elemento visual del menú principal, nos encontramos con la *zona de notificaciones y alarmas*. La funcionalidad de esta zona es muy importante, ya que le ayuda al usuario contar a "*simple vista*" con una zona en la que pueda ver el número de notificaciones y alarmas que el sistema mantiene, y pueda acceder a ellas de manera rápida. Esta zona de la interfaz está compuesta por un pequeño texto y un pequeño contador del número de alarmas y notificaciones pendientes como se muestra en la figura 5-9.

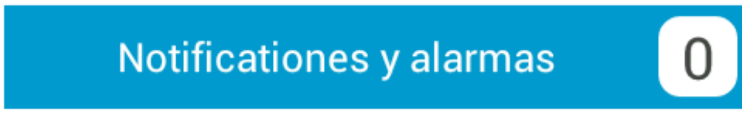


Figura 5-9. *Barra de notificaciones y alarmas del paciente.*

La estructura *XML* que define la zona de notificaciones se muestra en el listado 5-2.

Listado 5-2. *Estructura XML que define la barra de notificaciones, para cualquier enfermedad.*

```
<RelativeLayout
        android:id="@+id/label_bottom"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
```

```
        android:background="#C1D5DD"
        android:paddingBottom="3dip"
        android:paddingTop="3dip" >
    <TextView
            android:id="@+id/number_of_notifications"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:layout_marginBottom="3dip"
            android:layout_marginRight="4dip"
            android:layout_marginTop="3dip"
            android:background="@drawable/square_notifications_gray"
            android:paddingLeft="10dip"
            android:paddingRight="10dip"
            android:text="0"
            android:textColor="@color/dkgray"
            android:textSize="25dip" />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_marginLeft="5dip"
            android:layout_toLeftOf="@+id/number_of_notifications"
            android:text="Notificationes y alarmas"
            android:textColor="@color/dkgray" android:gravity="center"
            android:textSize="17dip" />
</RelativeLayout>
```

5.3.2 Origen y flujo de datos.

Una vez documentado el diseño de la interfaz, procedemos a ver el flujo de datos que tiene dicha vista. Los flujos de datos se realizarán hacia la base de datos alojada en la red. Estos datos serán utilizados por la zona de notificaciones y alarmas.

El área de notificaciones debe mostrar el número de mensajes nuevos que se le han generado al usuario, después de la obtención de una medida, por lo que el flujo de datos va encaminado hacia este fin. En la figura 5-10 se muestra el esquema que representa el flujo de datos que realizará la vista para el área de notificaciones.

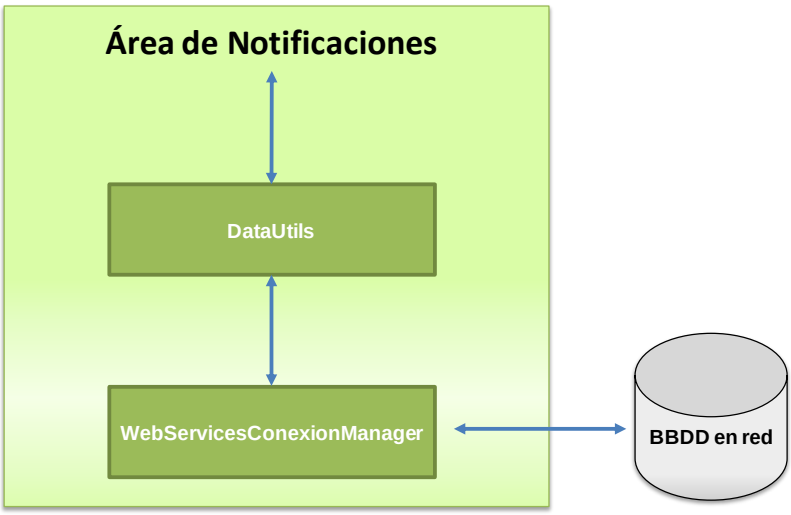


Figura 5-10. *Flujo de datos que define el área de notificaciones del menú.*

Flujo de datos dentro de la aplicación

En la figura 5-11, se muestra el esquema de flujo de datos mostrando tanto las clases implicadas como los métodos utilizados en el menú principal. Al basarse la aplicación en un modelo de tres capas, tenemos como intermediaria la clase *DataUtils*, que hace de enlace entre la interfaz y la persistencia (clase *WebServicesConexionManager*).

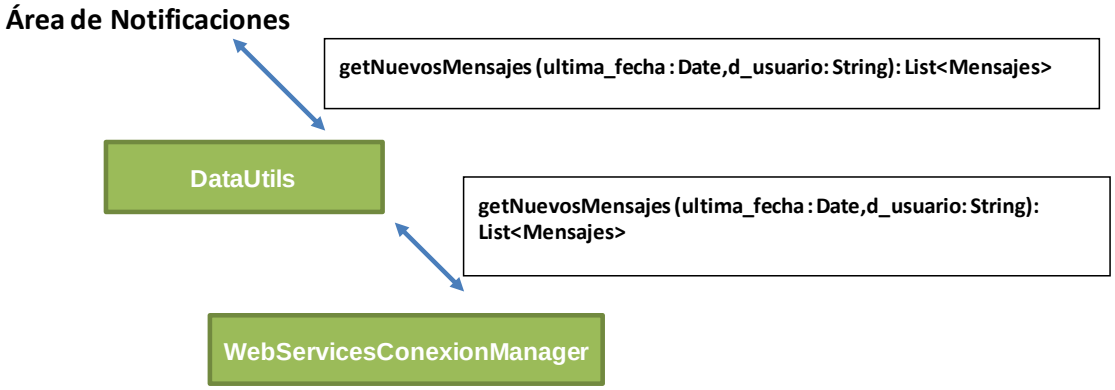


Figura 5-11. *Flujo de datos al recibir o enviar una notificación.*

Dentro de la base de datos alojada en la red, debemos obtener todos los mensajes nuevos que el usuario pudiera tener. Estos mensajes nos permitirán generar el módulo de autocontrol, la zona de notificaciones y alarmas. Para tal motivo, existe una tabla en la base de datos denominada *mensajes* con la siguiente estructura:

Nombre	Id_sender	Id_receiver	date	message
Tipo de dato	VAR_CHAR(9)	VAR_CHAR(9)	DATE	TEXT

5.4 DEFINICIÓN DEL PERFIL DEL PACIENTE

El primer elemento que debe definirse antes de utilizar la aplicación para la monitorización de pacientes es el *perfil del paciente*. Esto es debido a que almacena la información básica de la enfermedad a monitorizar.

En la tabla 5-5, se muestran los elementos del *framework* que se utilizan para el desarrollo de la interfaz de perfil de paciente, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Desarrollo del Perfil del Paciente	ProfileMobiPattern	-Disease -PatientProfile	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	**- RF09** (CU-19, CU-20, CU-21)

Tabla 5-5.*Elementos del framework utilizados para la generación del perfil de paciente.*

Creación y actualización del Perfil del Paciente (ProfileMobiPattern).

Actualización del Perfil del Paciente

Una de las principales características de la arquitectura es asociar la aplicación a un paciente, de tal manera que la toma de datos, adición de actividades, recomendaciones y otros módulos de control vayan en función del perfil de ese paciente. Por lo tanto, la aplicación debe proveer una forma de insertar, ver y editar la información del paciente que esté utilizando la aplicación. Para explicar de manera más detallada los elementos asociados al perfil del paciente, nos centraremos en los aspectos de: inserción de datos, visualización de esos datos insertados y la edición posterior de un perfil de un paciente.

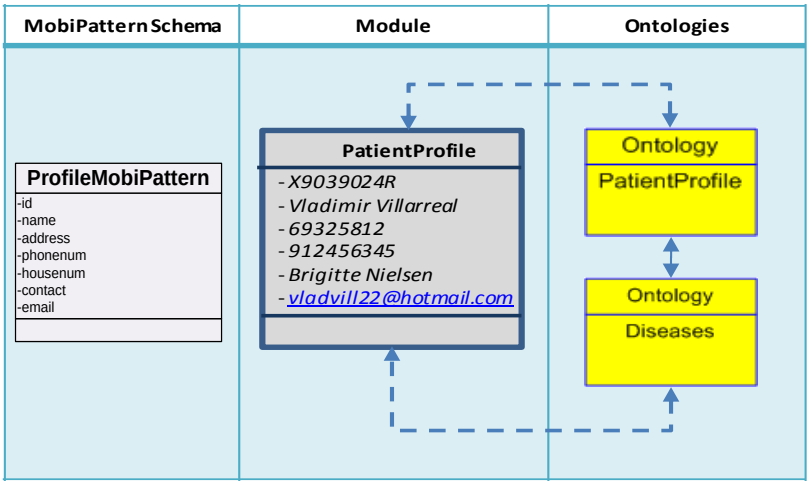


Figura 5-12. *Implementación del MobiPattern de Perfil de Paciente y su relación con la ontología de perfil de paciente y enfermedades.*

Para la creación del perfil del paciente se van a utilizar el **ProfileMobiPattern** previamente definido, creando un nuevo perfil**.** En la figura 5-12 se muestra el esquema general del **MobiPattern** del Perfil, que se usa para generar el módulo de un perfil de paciente, basado en las ontologías que intervienen y que permitirán la generación de los siguientes módulos.

En el listado 5-3, se muestra una estructura del código donde se relaciona el **MobiPattern** de perfil de paciente previamente definido, con la estructura ontológica del perfil de paciente.

Listado 5-3. *Código de la relación entre el MobiPattern de perfil de paciente y su Ontología previamente definida.*

```
// Invocación del patrón de Perfil del paciente
public class PatientProfileAc extends Activity {
       /** Called when the activity is first created. */
       @Override
       public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
        /** Llamada a la estructura xml del MobiPattern Perfil de Paciente. */
              setContentView(R.layout.patientprofilely);
              cargarInterfazGrafica(); }
// Relación de la ontología del perfil de paciente
private void insertPatientProfile(String idpatient, String NamePP,
              String AddressPP, String MphonePP, String LphonePP,
```

```
                String NcontactPP, String EmailPP) {
                SQliteDB sqlitedb = new SQliteDB(this);
                SQLiteDatabase db = sqlitedb.getWritableDatabase();
                ContentValues cv = new ContentValues();
                cv.put(SQliteDB.IDPP, idpatient);
                cv.put(SQliteDB.NAMEPP, NamePP);
                cv.put(SQliteDB.ADDRESSPP, AddressPP);
                cv.put(SQliteDB.MPHONEPP, MphonePP);
                cv.put(SQliteDB.LPHONEPP, LphonePP);
                cv.put(SQliteDB.NCONTACTPP, NcontactPP);
                cv.put(SQliteDB.EMAILPP, EmailPP);
                try {
                db.insert("patientprofile", SQliteDB.IDPP, cv);
                }catch(Exception e){
                        Toast.makeText(getBaseContext(), e.toString(),
Toast.LENGTH_SHORT).show(); }
                        db.close(); }
```

En la figura 5-13, se muestra la pantalla generada para administrar el perfil de paciente, relacionándolo con su respetivo patrón.

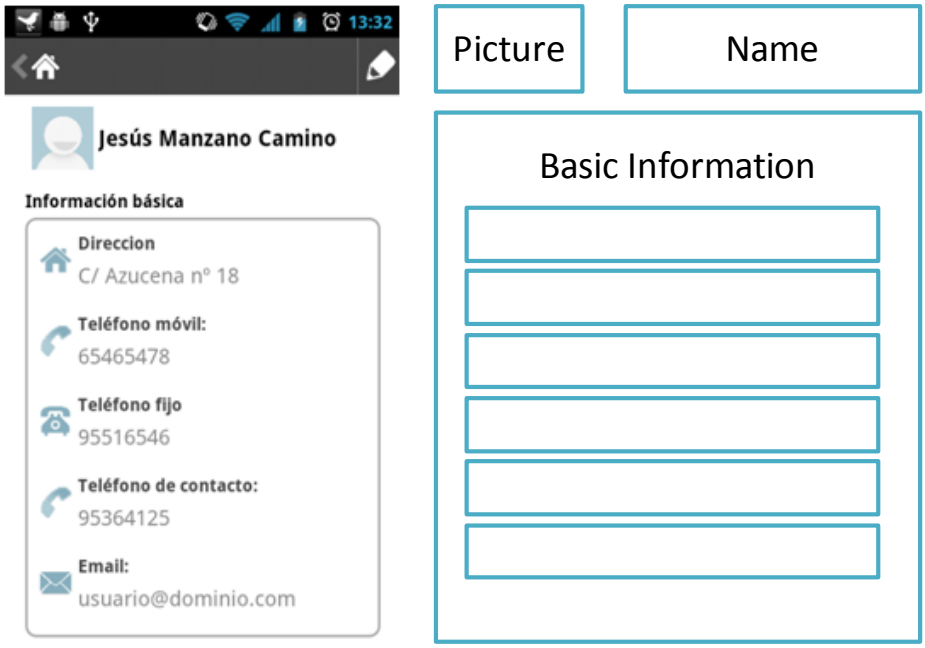


Figura 5-13. *Generación del perfil del paciente y su respectivo patrón.*

5.4.1 Inserción de datos por primera vez. Registro del perfil de un paciente

Muchos de los campos son meramente informativos (dirección, teléfono móvil, correo electrónico, etc.) salvo el *número de contacto*, el cual se usará para enviar un *SMS* a la persona de contacto que el paciente haya definido. Esta opción le permite a la aplicación comunicarle a la persona de contacto sobre situaciones de alto riesgo, manteniéndole informado de la situación del paciente ante variaciones de sus señales vitales monitorizadas.

Como existe la posibilidad de que la aplicación pierda conectividad con el servidor central, donde se almacenan todos los datos de la aplicación, el dispositivo tiene la opción automática de mantener los datos en *una base de datos local*, de tal manera que no se pierda información relevante del paciente. Cuando la aplicación tiene nuevamente conectividad, sincroniza la base

de datos local con la base de datos remota de tal manera que se actualizan los datos para un determinado paciente. La base de datos del dispositivo móvil solo cuenta con los datos del paciente que utiliza el móvil, no integra la base de datos completa de la arquitectura.

5.4.2 Visualización de la inserción de datos

En la figura 5-14(a), se muestra el esquema de la interfaz gráfica para la introducción de los datos del perfil de paciente asociada a la interfaz resultante y basada en el patrón de perfil de paciente previamente definido.

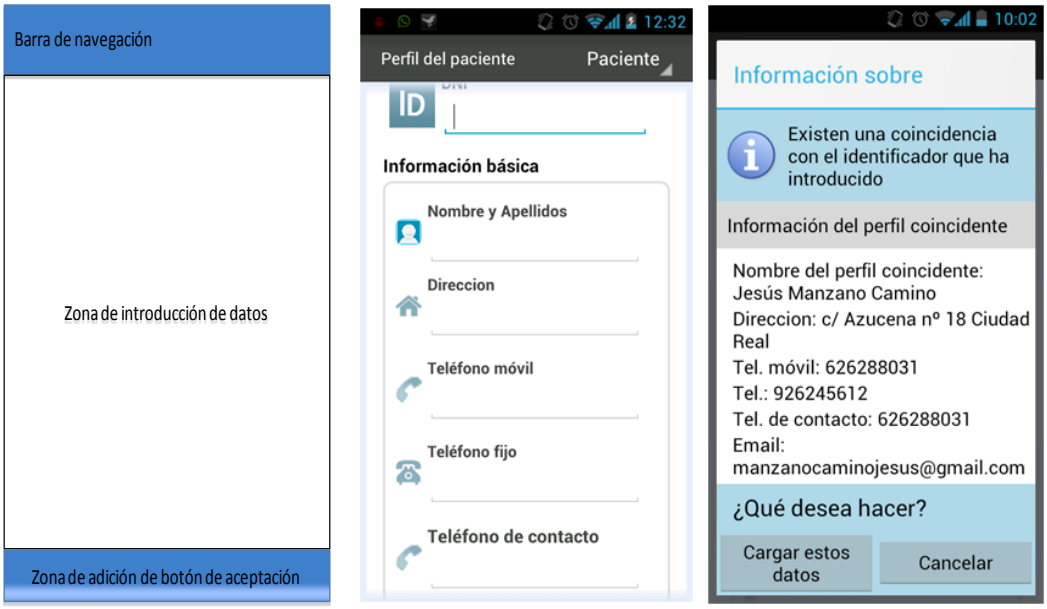


Figura 5-14. *Generación del perfil del paciente y su relación con las interfaces generadas.*

En la parte superior se define la barra de navegación, que al igual que en el resto de pantallas de la aplicación, provee al usuario información relativa al contenido de la interfaz que se está viendo en ese momento.

En la parte derecha está la barra en donde se ubicará el botón para elegir qué tipo de perfil de usuario (paciente o doctor) será introducido. En función de este parámetro, la zona de introducción de datos mostrará un conjunto de datos a rellenar. El paciente solo puede modificar los datos de su perfil, mientras que el médico puede modificar los datos de diferentes pacientes.

La zona de introducción de datos contendrá una serie de campos de texto a rellenar en función del rol seleccionado. En la zona inferior, se ha añadido un botón de confirmación una vez todos los campos hayan sido rellenados, permaneciendo deshabilitado mientras el paciente está completando el formulario.

En la figura 5-14(c) se puede apreciar la funcionalidad que se mencionó anteriormente, la aplicación reconoce el identificador introducido por el usuario, y permite la carga de los datos del perfil que coincide con ese identificador.

5.4.3 Estructura de la interfaz

La estructura que define la interfaz es muy similar al resto de las pantallas de la aplicación en general, se hace uso de fragmentos (*Fragments*) para el relleno, en nuestro caso, para rellenar la *zona contenedora* (zona de introducción de datos). La zona contenedora, incorpora los elementos que componen la estructura del MobiPattern que se esté implementando.

El uso de estos elementos nos permite la generación de interfaces "*hijas*" dentro de una estructura definida e inmutable. La estructura funcional de la interfaz o módulo de perfil de paciente se muestra en la figura 5-15.

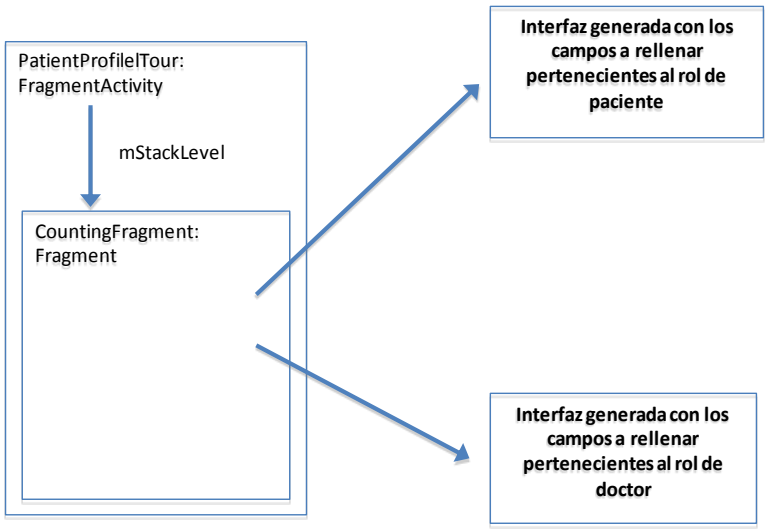


Figura 5-15. *Elementos que componen la interfaz de perfil de paciente.*

5.4.4 Origen y flujo de datos.

En la introducción de datos de perfil de paciente nos encontramos tanto con lectura como escritura en base de datos (tanto local como en red) ya que, cómo se explicó anteriormente, en la introducción de datos existe la funcionalidad de cargar datos ya existentes en la base de datos en red usando una coincidencia del identificador del perfil. Primeramente se expondrá la estructura de datos que "*define*" el perfil de un paciente, y a continuación se esquematizará el flujo de datos que conlleva la introducción de los datos de perfil de ese paciente.

Hay que recordar que la aplicación final podrá ser utilizada por un paciente así como por un médico, por lo cual se han definido un rol para cada uno de ellos. El médico puede ver información del perfil de varios pacientes mientras que el paciente solo puede ver información referente a su perfil. Los elementos que componen la tabla de perfil de paciente es la siguiente:

Camp	**idpatient**	**NamePP**	**AddressPP**	**MphonePP**	**LphonePP**	**Ncontact PP**	**Email PP**	**Photo**
Tipo	Varchar(9)	Text	Text	Int(12)	Int(12)	Int(12)	text	BLOB

Definida la estructura del perfil de paciente, podemos ver el flujo de datos que se produce en la introducción de estos datos, en la figura 5-16.

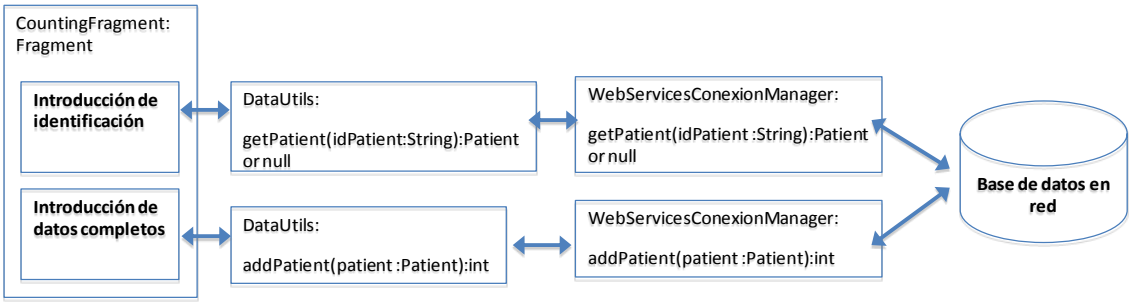


Figura 5-16. *Flujo de datos para la inserción de datos de la interfaz de perfil de paciente.*

A la hora de actualizar el perfil del paciente, el proceso devolverá un entero mayor que "0" si el resultado es satisfactorio, y "-1" indicando que ha existido algún error al momento de actualizar los datos. Lo que nos permite solicitar al usuario que vuelva a actualizar el módulo ya que no se cargaron los datos introducidos.

5.5 CREACIÓN DEL MÓDULO DE MEDIDAS Y EJERCICIOS FISICOS (MEASUREMOBIPATTERN)

Uno de los módulos de mayor importancia es el **módulo de medidas**, es a través de él que se obtienen los valores de las señales vitales generadas por el dispositivo biométrico, y que definirá qué módulo de control debe ejecutarse en la aplicación de monitorización. Como este módulo se relaciona con el tipo de enfermedad, pueden existir múltiples enfermedades que usen este módulo de medida para cada tipo de señal vital, es decir, uno para medir glucosa en sangre otros para medir tensión, otro para medir temperatura y ejecutar procesos para cada uno de ellos.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Obtención de medidas	MeasureMobiPattern	-Disease -PatientProfile	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación) **Del Dispositivo Biométrico** (enlace, negociación, seguridad, comunicación)	**- RF01** (CU-01, CU-02) **- RF09** (CU-19, CU-20, CU-21)
Definición de ejercicio físico	MeasureMobiPattern	-Disease -PatientProfile -Module Definition	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	**- RF06** (CU-10)

Tabla 5-6.*Elementos del framework utilizados para la generación del módulo de medidas y ejercicios físicos.*

En la tabla 5-6, se muestran los elementos del *framework* que se utilizan para el desarrollo de la interfaz de obtención de medidas y selección de ejercicios, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

En la figura 5-15 se evalúa la relación entre el **MeasureMobiPattern**, la relación entre los módulos y las ontologías que interviene o interactúan con él.

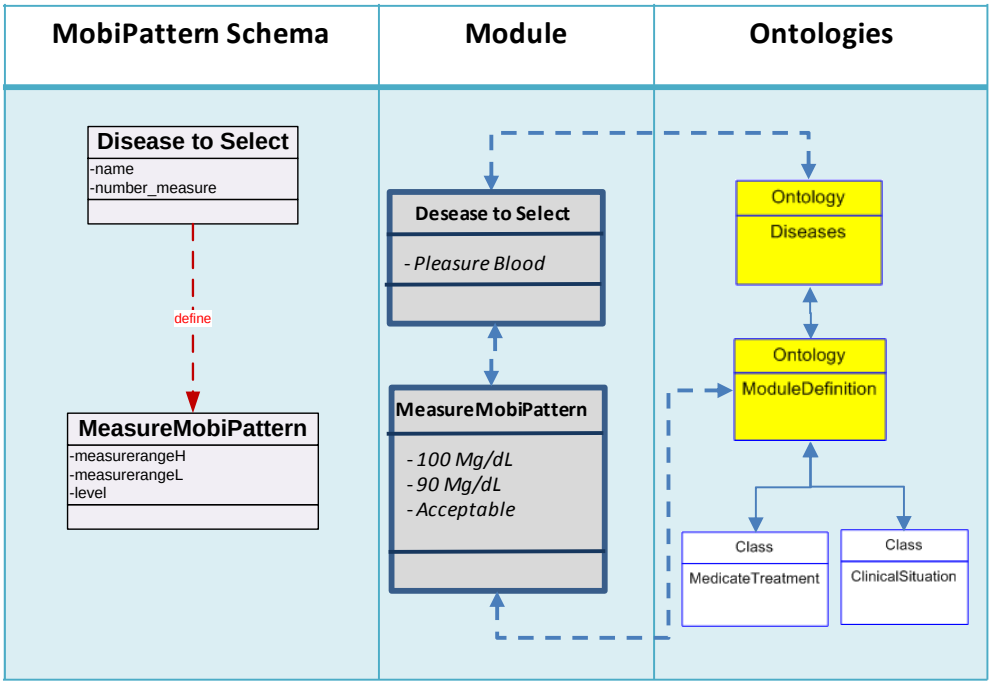


***Figura 5-17**. Implementación del MobiPattern de lectura de medidas (MeasureMobiPattern) y la relación con las ontologías enfermedades y definición de módulos.*

Después de generado el módulo de medidas y creados cada uno de los módulos, la aplicación muestra al paciente algunas opciones dependiendo del nivel obtenido por la lectura de su señal vital, es decir según sea el resultado de la medición, se le mostrará al paciente diferentes opciones que él puede elegir.

Este mismo patrón de medidas se utiliza para agregar actividades físicas, una vez obtenida una medida en un momento determinado. Es decir, el paciente puede por un lado tomarse la medida de control de su enfermedad y además la aplicación le pregunta si va a hacer alguna actividad física, lo cual le permite generar los mensajes de recomendación oportunos. También ofrece la posibilidad de que el paciente seleccione un ejercicio físico a desarrollar, y en este caso la aplicación puede automáticamente aconsejar tomarse una medida antes de realizar el ejercicio, para así poder recomendarle qué hacer. De manera similar se podría enlazar las medidas con la dieta del paciente.

En la figura 5-18, se puede observar las interfaces que se muestran a la hora de introducir las medidas y actividades físicas. Después, se realizará un estudio de sus componentes especificando el origen de los datos.

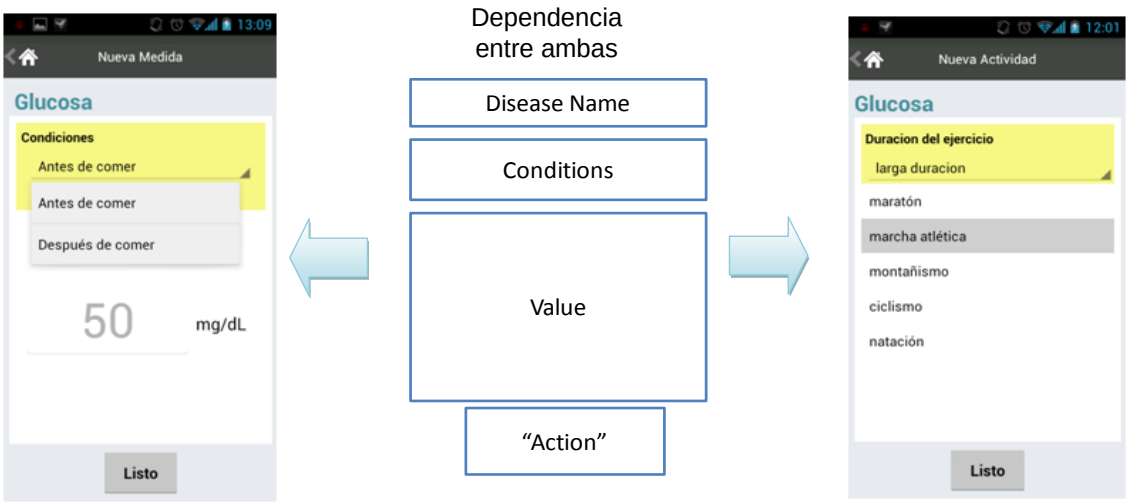


***Figura 5-18**. Relación del MeasureMobiPattern para de obtención de medidas y selección de ejercicios.*

Como se puede observar, ambas comparten una estructura visual semejante. Además, ambas pantallas comparten elementos visuales con el resto de vistas de la aplicación. Dicha característica hace que el usuario tenga una visión de una ***aplicación unificada y sólida***. Más adelante se profundizará en los patrones de diseño utilizados para dicha característica.

Las interfaces de adición, tanto de medidas como de actividades físicas, son creadas de forma genérica, de tal manera que sea cual sea la enfermedad en la que se quiere añadir la medida, la interfaz se comportará de manera similar.

***Listado 5-4.** Código XML de la interfaz contenedora para el MeasureMobiPattern.*

```
<?xml version="1.0" encoding="utf-8"?>
< LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:gravity="center_horizontal"
     android:orientation="vertical" >
<RelativeLayout
     android:id="@+id/label_top"
     android:layout_width="fill_parent"
     android:layout_height="50dip"
     android:layout_alignParentTop="true"
     android:background="@color/dkgray" >
<LinearLayout
     android:id="@+id/home_action_bar"
     android:layout_width="wrap_content"
     android:layout_height="match_parent"
     android:layout_alignParentLeft="true"
     android:layout_alignParentTop="true">
<ImageView
     android:layout_width="wrap_content"
     android:layout_height="match_parent"
     android:src="@drawable/left_arrow_disabled" />
```

5.5.1 Esquema y relación de patrón

Como bien se ve en el apartado anterior, tanto la adición de enfermedades como de medidas comparten elementos de interfaz común. Esto sucede a lo largo de la aplicación, permitiendo la *reutilización de código*.

Se procederá a enumerar las partes de la interfaz que son genéricas, explicando las clases generales y su especificación. Además se muestran los datos con los que se comunican dichas interfaces genéricas.

Generalización para la adición de elementos (obtención de medidas y actividades físicas)

La primera interfaz objeto de estudio es la interfaz que se encarga de la adición de elementos. Dicha interfaz se puede representar mediante el *MobiPattern* de medida definido en el capítulo 4 (*MeasureMobiPattern*). En la figura 5-19 se muestra la estructura física del patrón y su relación con una interfaz de salida, por ejemplo, la pantalla de obtención de ejercicios. Dicho patrón es una generalización del patrón de medidas utilizado.

Como podemos observar en esta aproximación, la interfaz viene representada por un elemento genérico que contiene una barra de navegación e información sobre la vista en la que nos encontramos. La interfaz específica se creará y mostrará dependiendo de ciertos parámetros de inicialización de dicha actividad (*tipo de acción que se va a realizar*).

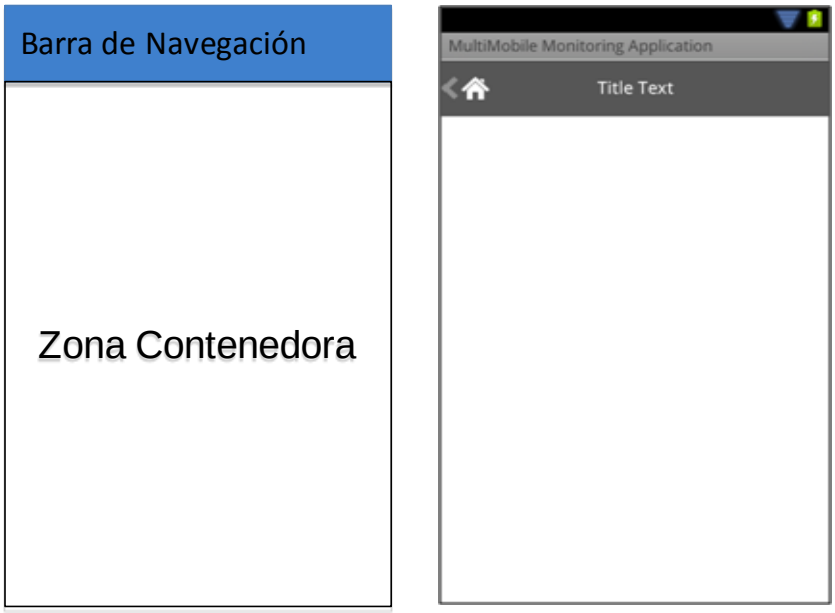


***Figura 5-19**. Estructura de la zona contenedora para el patrón a implementar.*

Desde el punto de vista de la programación, la estructura que se sigue es:

- Para la vista general que contiene la barra de navegación y la zona contenedora donde irá alojada la interfaz específica, se tiene la clase *NuevaMedidaFragment*, que hereda de *FragmentActivity*, permitiendo la adición de elementos del tipo

Fragment[11]. Este tipo especial de *Activiy*[12] permiten incluir o añadir "fragmentos" alojados en otros archivos *java* permitiendo que dichos fragmentos se comporten como *Activitys* independientes. De esta manera, podemos tener varias actividades (*Activitys*) actuando de la misma manera, y permite la generalización y adición de interfaces específicas. No obstante, hay que hacer hincapié que aunque el comportamiento de los *Fragments* se puede comparar con el de un *Activity* independiente, no es así, sigue siendo parte del *Activity* padre que lo invocó y por tanto rinde cuentas ante él.

Se ha implementado también una serie de métodos abstractos en la clase hija que heredará la actividad padre. Tales métodos son *Listener*[13] para las distintas interacciones que el usuario puede realizar con la interfaz generada por los *Fragments*.

- En los *Fragments*, podemos elegir si mostrará una interfaz u otra, dependiendo de parámetros que enviemos a los *Fragments*. En el caso de la adición de medidas, la interfaz que se podrá ver será la adecuada para la adición de medidas, o en el caso de la adición de ejercicios físicos la interfaz mostrada será la adecuada para tal fin.

Para ver mejor la estructura desde el punto de vista de la programación se muestra el siguiente esquema en la figura 5-20:

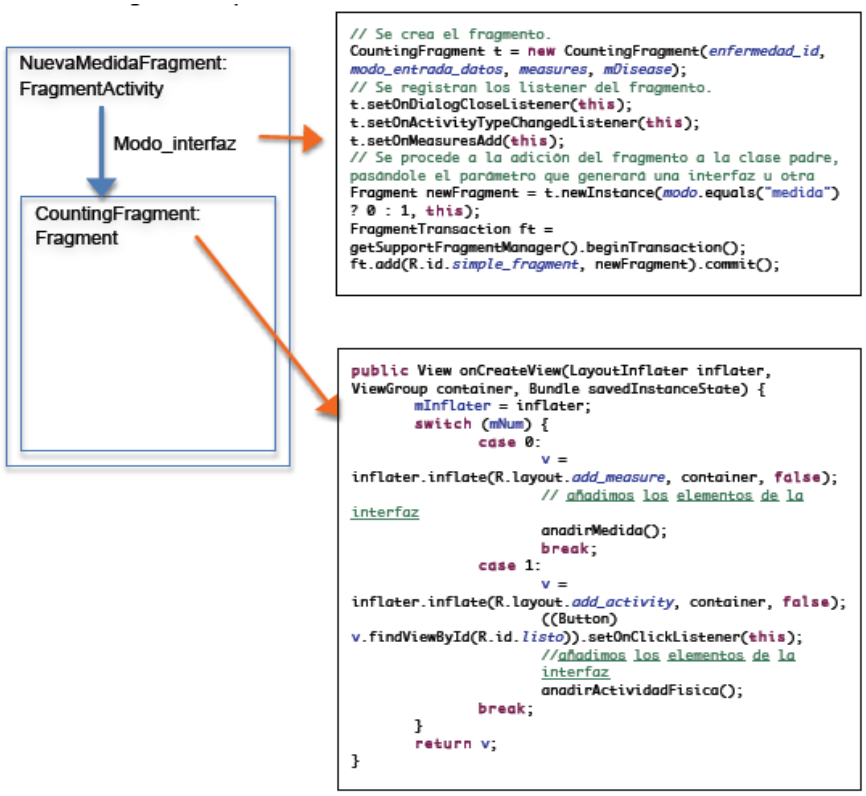


***Figura 5-20**. Elementos que forman el patrón de medidas y actividades (MeasureMobiPattern).*

[11] Un Fragment representa un comportamiento o porción de la interfaz de usuario en una Activity.
[12] Activity es el bloque encargado de construir la interfaz de usuario.
[13] Un Listener respuesta a los eventos que el usuario dispare al interactuar con la aplicación. Un Listener será llamado cada vez que se produzca un evento.

La primera porción de código representa la manera en la que, desde la actividad padre, se procede a la creación del fragmento hijo y la adición de la interfaz genérica ya mostrada al paciente. A la actividad padre desde el menú principal se le envía un parámetro en el que se indica que acción se relazará (si se tomará medida o introducirá un ejercicio físico). Este parámetro queda recogido en la variable de tipo *String*. Al crear una nueva instancia del *Fragment* hijo se envía otro parámetro en forma de número entero según el valor del parámetro *modo*. Los diferentes *Listener* que se registran son métodos abstractos creados en el *Fragment* hijo y que son implementados en la clase padre *CountingFragment*.

El segundo código merece más explicación. Los pasos que se muestran en la segunda porción de código son:

- En la llamada por parte de la actividad padre (*NuevaMedidaFragment*) al constructor del fragmento que será añadido se le inicializan una serie de variables que serán útiles para la creación de una interfaz u otra en el Fragment hijo (destacamos la variable *mNum* de tipo entero que sólo puede tener los valores 1 y 0 que indicarán en qué modo se añadirá el *Fragment* hijo).

- Al heredar el Fragment hijo de la clase Fragment, ha de sobrescribir una serie de métodos. El que más nos interesa es el método *onCreateView* que genera la interfaz específica de dicho *Fragment*. En este método (que es el expuesto en la segunda porción de código) se observa como genera una interfaz según uno de los parámetros provenientes del padre (variable *mNum* que indica que tipo de interfaz se requiere: añadir medida o añadir ejercicio físico).

En la figura 5-21, se muestra la estructura visual de las interfaces generadas para la obtención de una medida y la selección de un ejercicio asociado a esa medida.

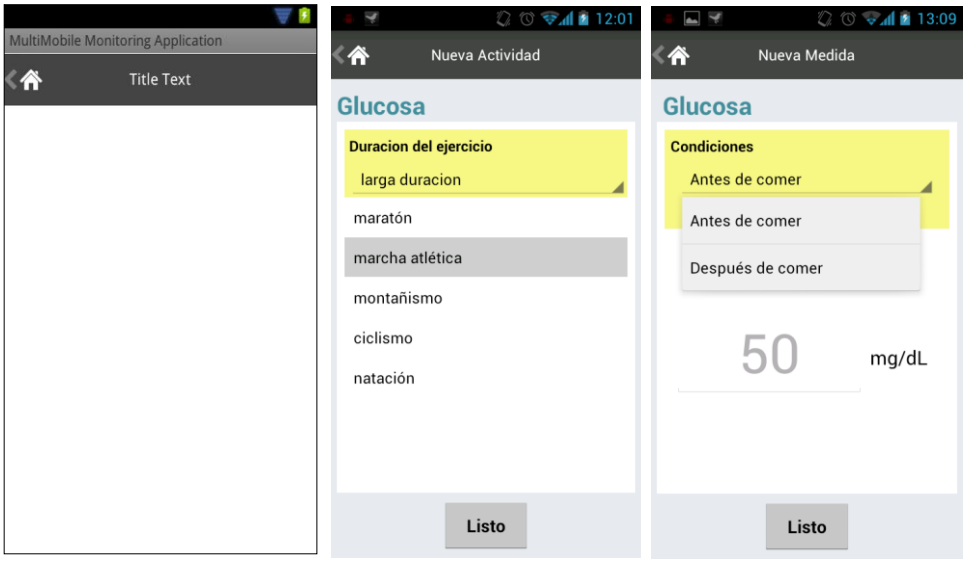


Figura 5-21. *Capturas de pantallas que define la selección de algún tipo de ejercicio a desarrollar antes/después de obtenida la medida.*

La imagen de la izquierda corresponde con la interfaz genérica (asociada al patrón inicialmente definido), en la que nos encontramos la barra de navegación y el contenedor que

se rellenará con la interfaz específica (espacio en blanco). A continuación se ven las interfaces que se obtienen añadiendo los distintos *Fragments* según corresponda el flujo de la aplicación. Las otras dos imágenes corresponden a las interfaces específicas que se crean. Se puede observar como ambas interfaces específicas contienen una estructura y diseño similar, de cara a no perder la integridad de la interfaz, pero cada una se diseña para un fin concreto.

5.5.2 Origen y flujo de datos.

Ya se ha visto cómo se generan las interfaces de manera específica y se ha insertado código de cómo se realiza dicha generación, pero queda definir de qué manera se rellenan las interfaces específicas con datos y qué origen tienen esos datos. Los datos con los que se comunica la interfaz, provienen de dos fuentes distintas pero comunes: *base de datos local* (en el dispositivo móvil) *o la base de datos en red* (en el servidor). Se definen dos bases de datos, por dos razones: la primera, porque existen estructuras de datos que no se necesita obtener desde una base de datos externa, sino que su estructura está definida para procesos mucho más pequeños dentro del dispositivo móvil. La segunda, a través de la base de datos local que almacena momentáneamente los datos obtenidos del dispositivo móvil, y posteriormente actualiza las últimas modificaciones realizadas a la base de datos en red.

Un primer esquema para la aproximación del flujo de datos se muestra en la figura 5-22. Este esquema se mantiene en la mayor parte de la aplicación.

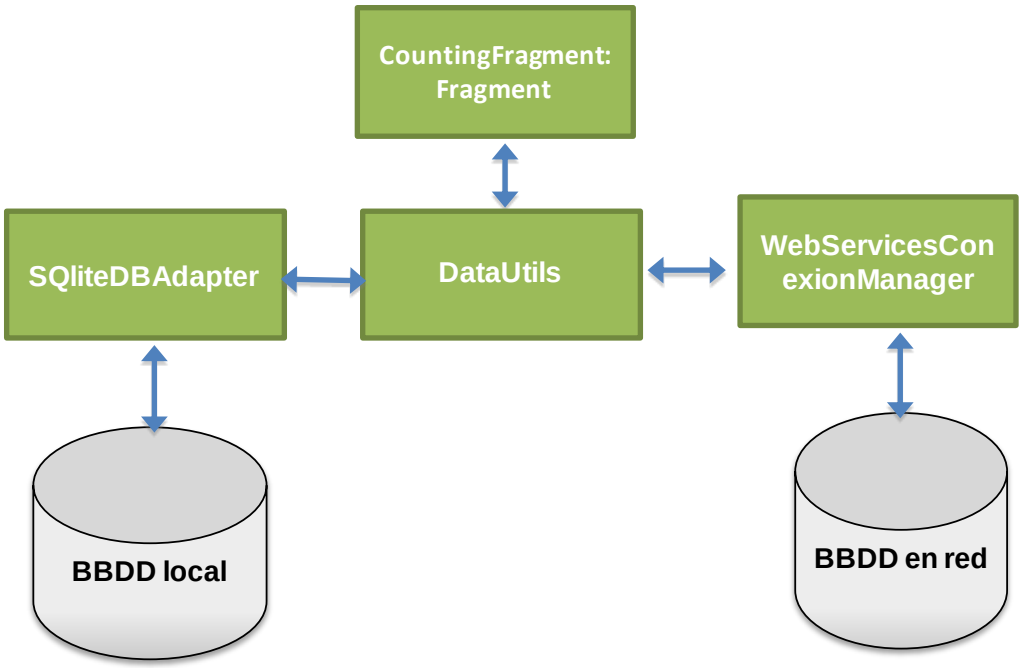


***Figura 5-22**. Esquema de flujo de datos para el almacenamiento de medidas.*

Una vez visto el esquema general de flujo de datos para la generación de las interfaces específicas, podemos entrar más en detalle explicando qué datos aloja cada interfaz.

Adición de medidas

La adición de medidas consta de un apartado en el que se seleccionan las condiciones de la toma de medida (si hubiese condiciones que mostrar) y una serie de campos de texto que se han de rellenar con las medidas correspondientes (pueden ser mas de una). Existen enfermedades que al momento de realizar una obtención de medida cuentan con condiciones previas para interpretar esa medida obtenida. Es el caso de enfermedades en las que se tiene que especificar si la medida se obtuvo antes o después de comer, antes o después de hacer un ejercicio, entre otras. Una vez agregada la medida, ésta es almacenada en la base de datos remota y local. En el listado 5-5 se define la estructura que me permite visualizar las condiciones previas a la obtención de una medida, las cuales están definidas en una tabla de la base de datos.

Listado 5-5. *Código de carga de datos para la selección de medidas y actividades físicas.*

```
// Spinner que carga las condiciones de una enfermedad de la base de datos local
private void cargarSpinner(View l1) {
// TODO Auto-generated method stub
name_condition = (Spinner) l1.findViewById(R.id.spinnerConditions);
name_conditions_data = rellenarCondiciones(l1);
aa = new ArrayAdapter<String>(mContext,
android.R.layout.simple_spinner_item, name_conditions_data);
aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
name_condition.setAdapter(aa);
name_condition.setOnItemSelectedListener(new OnItemSelectedListener()
    {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View arg1,int arg2,long arg3)
        {
        // TODO Auto-generated method stub
        id_condicion = name_conditions_data[arg2]; }
        @Override
        public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub } }
```

Se puede observar que el primer paso en la generación de la interfaz es obtener los datos de la enfermedad a la que se le añadirá la medida. Como se observa se rellena un *Spinner* (lista) con los datos procedentes del método *rellenarCondiciones*, cuyo código es el mostrado en el listado 5-6.

Listado 5-6. *Código de carga de datos de la enfermedad a la que se añadirá una medida.*

```
// Método para obtener datos de una enfermedad según condiciones
private String[] rellenarCondiciones(View l1) {
        // TODO Auto-generated method stub
        String[] condiciones = null;
        TextView t = ((TextView) l1.findViewById(R.id.unidad_medida));
        t.setText(mDisease.getMedida().get(0));
        condiciones = new String[mDisease.getCondiciones().size()];
        for (int j = 0; j < mDisease.getCondiciones().size(); j++) {
        condiciones[j] = mDisease.getCondiciones().get(j).getNombre(); }
        return condiciones; }  }
```

En el listado 5-7, se muestra una estructura del código en donde se relaciona el **MobiPattern** de selección de la enfermedad para un tipo de medida previamente definido, con la estructura ontológica de la enfermedad. El paciente puede tener más de una enfermedad

por ende el patrón de selección del tipo de enfermedad debe permitirle seleccionar la enfermedad que quiere medir.

Listado 5-7. *Código de la relación entre el MobiPattern del tipo de medida y su Ontología de enfermedad previamente definida.*

```
// Invocación del MobiPattern de selección del tipo de medida
public class MeasurePP extends Activity {
	/** Called when the activity is first created. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
/**Llamada a la estructura xml del MobiPattern Medidas seleccionando previamente el
tipo de enfermedad. */
		setContentView(R.layout.measurepp); }
// Relación de la ontología de enfermedad (según tipo de enfermedad)
	public void onAddClick(View control) {
		switch (control.getId()) {
		case R.id.pleasureblood:
			Intent showviewpleasureblood = new Intent();
			showviewpleasureblood.setClass(this, PleasureBloodView.class);
			startActivity(showviewpleasureblood);
			break;
		case R.id.temperature:
			Intent showviewtemperature = new Intent();
			showviewtemperature.setClass(this, TemperatureView.class);
			startActivity(showviewtemperature);
			break;
		case R.id.diabetes:
			Intent showviewdiabetes = new Intent();
			showviewdiabetes.setClass(this, DiabetesView.class);
			startActivity(showviewdiabetes);
			break;
		default:
			break; }
		finish(); }
```

Cada vez que el paciente haga sus tomas de medidas correspondientes a una enfermedad, este **MobiPattern** se ajustará al tipo de enfermedad que quiere medir, para conocer cuántas unidades de medidas tiene esa enfermedad, lo que nos permitirá en un futuro pasarle esos parámetro al módulo de autocontrol. En el Anexo 2, se muestra el código que completa la obtención, procesamiento y almacenamiento de las señales vitales de un paciente.

Adición de actividades físicas

La adición de actividades físicas consta de varias zonas que son rellenadas con datos provenientes de almacenamientos externos. Se tiene la posibilidad de escoger que tipo de actividad física se desea añadir de una serie de tipos predefinidos. Estos tipos de ejercicios son provenientes de una base de datos, utilizando para su obtención la clase *SQliteDBAdapter*. Detallaremos el proceso de obtención de datos paso a paso:

1. Primeramente se rellena el *Spinner* (lista) que mostrará los tipos de ejercicio según el tiempo de duración. Se han desarrollado una clasificación de los ejercicios según su duración de la siguiente manera: *ejercicios de mediana duración*, *ejercicios de larga duración*, *ejercicios breves con mucho esfuerzo* y *ejercicio de actividades intermitentes*. Se muestra la estructura que genera esta lista en el listado 5-8.

Listado 5-8. *Código que muestra los tipos de ejercicios previamente definidos y que puede llevar a cabo un paciente.*

```
private void rellenarSpinnerDuracionEjer(final View l1) {
// TODO Auto-generated method stub
   Spinner duracion_ejercicio = (Spinner)l1 .findViewById(R.id.spinnerTipoEjer
   tipos_ejercicio = rellenarTiposEjercicio(enfermedad_id);
   aa = new ArrayAdapter<String>(mContext,
   android.R.layout.simple_spinner_item, tipos_ejercicio);
   aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
   duracion_ejercicio.setAdapter(aa);
   duracion_ejercicio.setOnItemSelectedListener(new OnItemSelectedListener() {
public void onItemSelected(AdapterView<?> arg0, View arg1,
   int arg2, long arg3) {
// TODO Auto-generated method stub
   tipo_ejercicio_name = tipos_ejercicio[arg2];
   listView = (ListView) l1.findViewById(R.id.list1);
   lv_arr = DataUtils.getInstance(mContext).getActivitiesByType(tipo_ejercicio
   s = new SelectedAdapter(mContext, 0, lv_arr);
   listView.setAdapter(s);
   listView.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> arg0,
    View arg1, int arg2, long arg3) {
    s.setSelectedPosition(arg2);
nombre_ejercicio = s.getItem(arg2).toString(); } }); }
@Override
public void onNothingSelected(AdapterView<?> arg0) {
// TODO Auto-generated method stub } }); }
```

2. A continuación, se cargan los ejercicios que correspondan al tipo de ejercicio seleccionado en el *Spinner* (lista). Se muestra la estructura en el listado 5-9.

Listado 5-9. *Código que muestra los ejercicios de una clasificación de ejercicios seleccionados.*

```
private String[] rellenarTiposEjercicio(String id_enfermedad2) {
       // TODO Auto-generated method stub
       return
DataUtils.getInstance(mContext).getTiposEjercicio(id_enfermedad2);
}
```

Ahora bien, se ha definido el código desde la clase *CountingFragment*, pero si se observa la figura 5-10, todas las peticiones de datos pasan por la clase *DataUtils*. Dicha clase contiene una serie de métodos y hace de puente entre la capa de interfaz y los distintos adaptadores para la comunicación con datos de origen externo. Ahora veamos que necesitamos de dicha clase para poder obtener los datos necesarios para la interfaz de adición de ejercicios (listado 5-13).

Listado 5-10. *Código relaciona la medida con el ejercicio seleccionado.*

```
public String[] getTiposEjercicio(String id_enfermedad) {
       // TODO Auto-generated method stub
       String[] resul = null;
       SQliteDBAdapter.getInstance().open(mContext);
       List<String> a=SQliteDBAdapter.getInstance().
       ObtainTypesActivities(id_enfermedad);
       SQliteDBAdapter.getInstance().close();
       resul = new String[a.size()];
       for (int i = 0; i < a.size(); i++) {
              resul[i] = a.get(i); }
       return resul; }
```

La comunicación y obtención de los datos viene definida por la clase *SQliteDBAdapter*, que es la encargada de la comunicación directa con la base de datos local. Se muestra el código en el listado 5-11.

Listado 5-11. *Código relaciona la medida con el ejercicio seleccionado.*

```
public List<String> obtainTypesActivities(String id_enfermedad) {
       database = dbHelper.getReadableDatabase();
       Cursor c = database.rawQuery("Select tipo ejercicio from  ejercicio  where iddi
?", new String [] {id_enfermedad});
       return parserToListActivityExercise(c); }
private List<String> parserToListActivityExercise(Cursor c) {
       // TODO Auto-generated method stub
       List<String> res = new ArrayList<String>();
       while (c.moveToNext()) {
       if (!res.contains(c.getString(c.getColumnIndex("tipoejercicio")))) {
              res.add(c.getString(c.getColumnIndex("tipoejercicio"))); } }
       return res; }
```

En el Anexo 3, se muestra el código que completa la definición y selección de un tipo de ejercicio a desarrollar por un paciente.

5.6 GENERACIÓN DE MÓDULOS.

Es en esta sección que se procede a la creación de todos los módulos que intervienen en la generación de aplicaciones, basadas en los esquemas de cada uno de los MobiPattern. Para cada tipo de paciente se crean módulos específicos basados en su perfil y el tipo de enfermedad que se padece.

En la tabla 5-7, se muestran los elementos del *framework* que se utilizan para el desarrollo de cada uno de los módulos de control (definición de módulos), el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Recomendaciones **Educación** **Prevención**	BehaviourMobiPattern	-Disease -PatientProfile -ModuleDefinition	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	**- RF05** (CU-08) **- RF06** (CU-09, CU-10, CU-11, CU-12)

Tabla 5-7.*Elementos del framework utilizados para la definición de módulos de control.*

5.6.1 Generación de módulos: Education, Suggestion y Prevention. (BehaviourMobiPattern)

La aplicación posee un sistema de recomendaciones con el cual, al añadir una nueva medida, el usuario podrá ver un conjunto de recomendaciones de distinta índole relacionadas tanto con la enfermedad que se monitoriza en ese momento, como con la última medida añadida. Dicho sistema trata de ofrecer al paciente información importante acerca de posibles

recomendaciones en función de su última medida introducida, de tal manera que esta información ayude al paciente con respecto a su enfermedad. Tanto los módulos de recomendación, educación, alertas usan el mismo patrón de comportamiento, lo que varía es su funcionalidad. Tanto el *framework* como la arquitectura *software* contemplan la posibilidad de realizar recomendaciones basadas en el análisis del historial completo de mediciones.

Las recomendaciones podrán ser de naturaleza variada, y según ésta se mostrarán de manera distinta. El cambio de interfaz según la naturaleza en función del tipo de recomendación ha sido realizado para la comodidad del paciente, ya que existen recomendaciones que pueden ser vistas sin cargar una interfaz distinta y otras que deben ser visualizadas en una interfaz diferente. Por ello podemos distinguir dos tipos de interfaces; una interfaz en la que se mostrarán recomendaciones que son más complejas, y una interfaz tipo ventana emergente que mostrará las recomendaciones que son más simples.

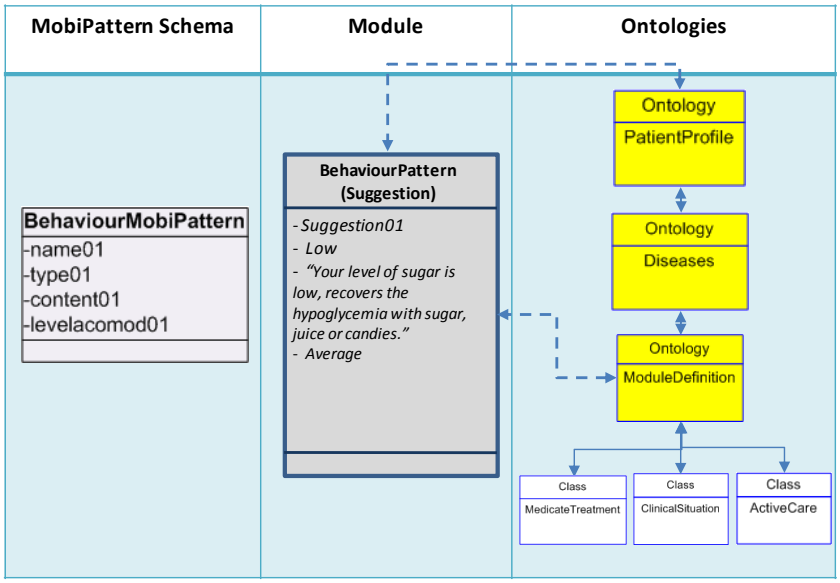


Figura 5-23. *Implementación del BehaviourMobiPattern para la generación del módulo "Recomendación".*

Se procederá a explicar ambos tipos de interfaces para las recomendaciones, viendo las interfaces por separado. El flujo de datos de manera conjunta tiene el mismo comportamiento. En las figuras 5-23 se evalúa la funcionalidad del **MobiPattern** de Comportamiento (**BehaviourMobiPattern**), para los módulos de educación, recomendación y prevención. Estos módulos son considerados funcionales, ya que es la base de la relación *perfil – medida.*

5.6.2 Esquema y patrón de la interfaz

El diseño de las recomendaciones, como se ha comentado anteriormente, cuenta con dos tipos de interfaces, con importantes cambios entre ellas. Por un lado, nos encontramos con una ventana emergente que contendrá algunas recomendaciones y botones de control, y por otra con una nueva pantalla en la que se mostrarán las recomendaciones.

En la figura 5-24 se muestra la zona contenedora del patrón de recomendaciones con ventana emergente. La ventana se divide en tres partes esenciales:

- **La barra de título** donde se mostrará información relacionada con la naturaleza de la recomendación.
- **La zona de visualización de mensajes** donde se mostrará la información de la recomendación.
- **Zona de botones** de control donde se situarán los distintos botones para el control de la pantalla.

Podemos observar cómo la ventana emergente se ve por encima de la pantalla de adición de medidas (en este caso) y muestra los componentes visuales determinados en el punto anterior.

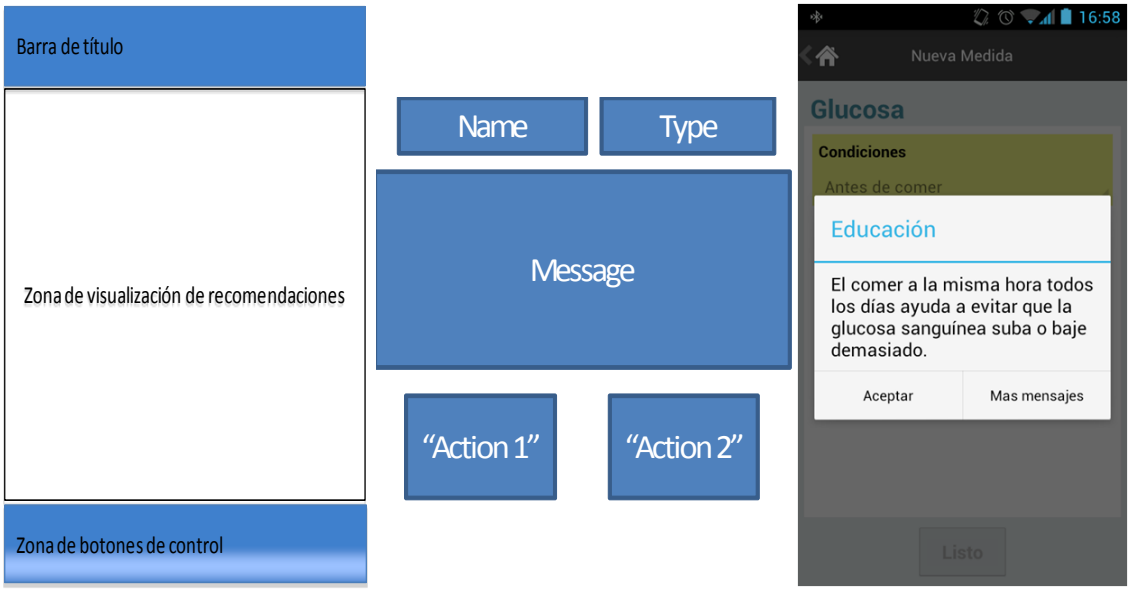


Figura 5-24. *Zona contenedora del patrón de recomendaciones y su interfaz emergente según el BehaviourMobiPattern.*

En la figura 5-25 se muestra la zona contenedora del patrón de recomendaciones con pestañas. En la parte superior de la pantalla, se indica mediante pestañas las distintas naturalezas de las recomendaciones, de tal manera que al seleccionar cada pestaña, en la zona de visualización de recomendaciones se mostrarán las recomendaciones correspondientes con la naturaleza seleccionada.

Podemos notar que el patrón inicialmente definido puede ser redefinido según las necesidades. Esto le da una clara capacidad de adaptación a los elementos de diseño propuestos en el *framework*.

El establecimiento de un patrón de interfaz basado en pestañas favorece la separación de la información, permitiendo que el usuario acceda a las recomendaciones según su naturaleza. El usuario sólo tendrá que seleccionar la pestaña cuyo título mostrará la naturaleza de las recomendaciones que necesita para poder acceder a ella.

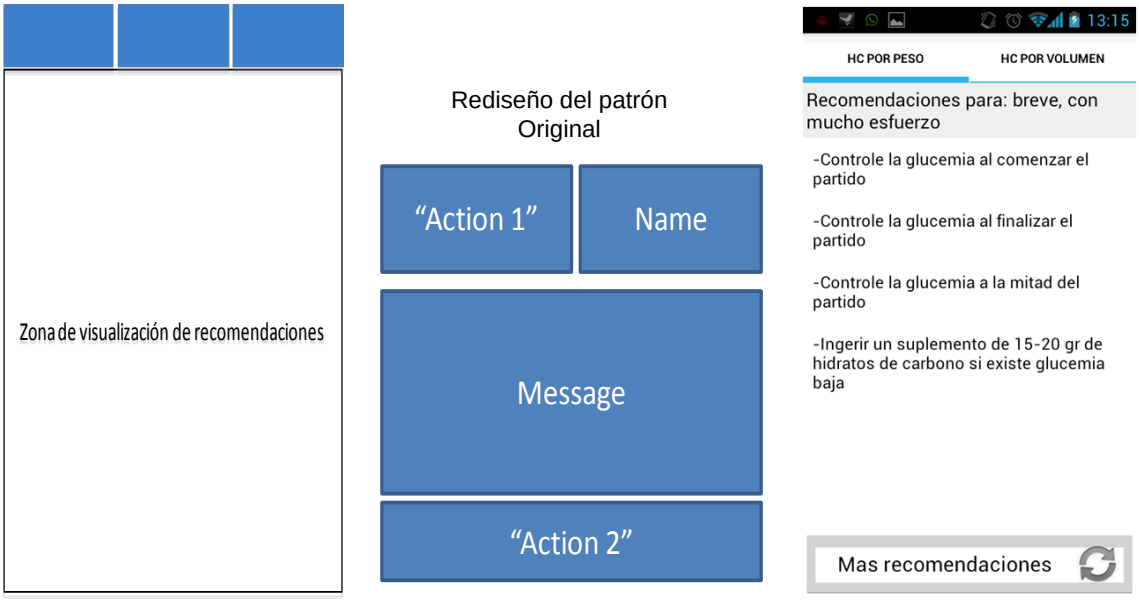


Figura 5-25. *Zona contenedora del patrón de recomendaciones y su interfaz basada en pestañas.*

En esta figura vemos las recomendaciones en pestañas en la que se muestra un resumen de las características de las recomendaciones que contendrá, en la zona de información se visualiza información acerca de los parámetros que generaron las recomendaciones y en la parte inferior controles para manejar la información (en este caso mostrar más recomendaciones).

5.6.3 Origen y flujo de datos.

Antes de definir y ver de manera esquemática los flujos de datos que trabajan con la interfaz, se verá de manera sencilla la comunicación de dichos datos.

Al ser una interfaz que muestra gran cantidad de información, se necesita que la información sea accedida de manera rápida para mejorar el rendimiento de la aplicación, ésta no debe demorar mucho tiempo en mostrar los servicios solicitados. Por tanto, el origen de los datos debe ser de carácter local, no impidiendo eso que en otro instante en el transcurso de la aplicación, ésta cargue contenido proveniente de bases de datos en red a la base de datos local.

Por tanto, el flujo de datos en el esquema irá dirigido hacia la base de datos local, que nos proporcionará la información de manera más rápida que si accediéramos a una base de datos en red. En la figura 5-26, se muestra el esquema que representa el flujo de datos que utiliza la interfaz definida.

Nos encontramos en la parte izquierda con la pantalla que define la introducción tanto de medidas como de actividades. Desde esa pantalla, se accederá a las distintas formas de ver las recomendaciones, las cuales requerirán datos distintos

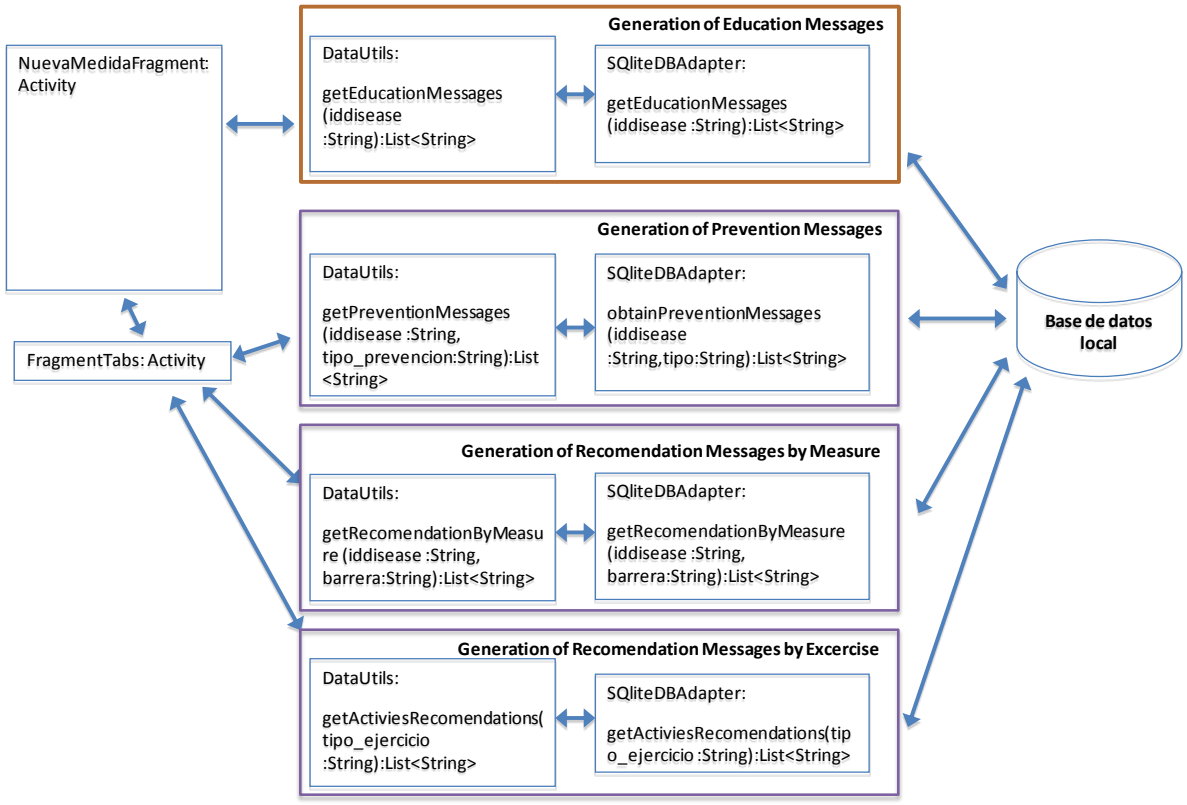


Figura 5-26. *Flujo de datos para el patrón de recomendaciones, educación y prevención.*

Se puede observar con el esquema de flujo de datos cómo los mensajes son mostrados en función de ciertos rangos, que son utilizados para hacer una selección dentro de las tablas correspondientes. Los rangos establecidos para cada enfermedad son: ***alto, normal, bajo***. Cada rango es especificado según los niveles que se han definido para cada enfermedad en particular.

Comportamiento del patrón para otros módulos de control

Para los demás módulos de control, la interfaces tendrá el mismo comportamiento del definido inicialmente por el **BehaviourMobiPattern**, con la única diferencia que cambiará el contenido y propósito de la interfaz contenedora.

En las figuras 5-27 y 5-28 se evalúa la funcionalidad del **MobiPattern** de Comportamiento (**BehaviourMobiPattern**), para los módulos de educación y prevención.

La información que muestra cada módulo forma parte de cada una de las ontologías que han sido descritas previamente, y de las cuales adquiere los valores que se necesiten para cada paciente.

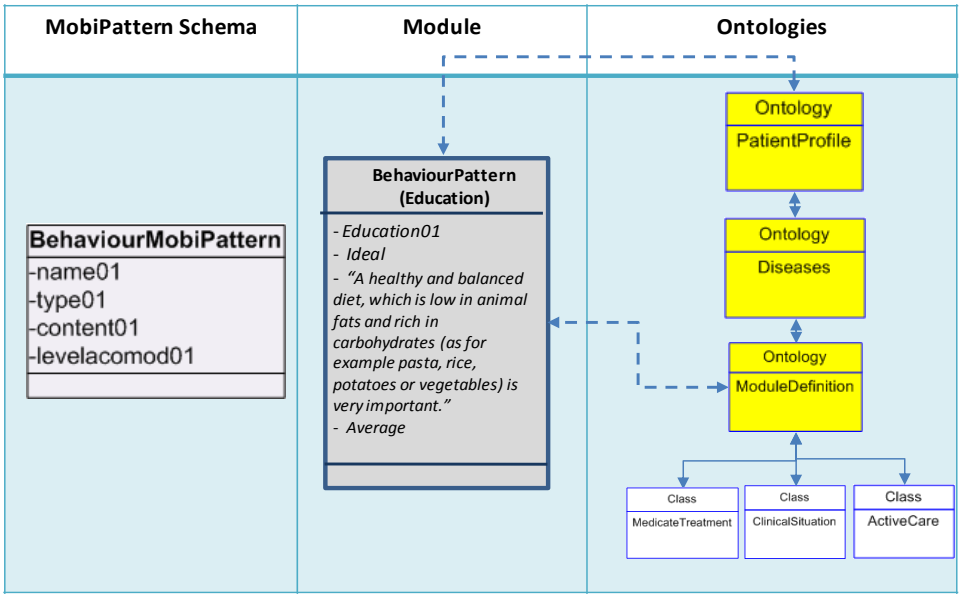


Figura 5-27. *Implementación del BehaviourMobiPattern para la generación del módulo "Education".*

Cada uno de los módulos generados cuentan con la interpretación en rangos (*Low, Normal, High*) de cada medida del paciente, por lo cual cada módulo está asociado a un estado específico del momento en que el paciente se ha tomado esa medida. Es por ello que los módulos funcionales cambiarán según la situación física del paciente.

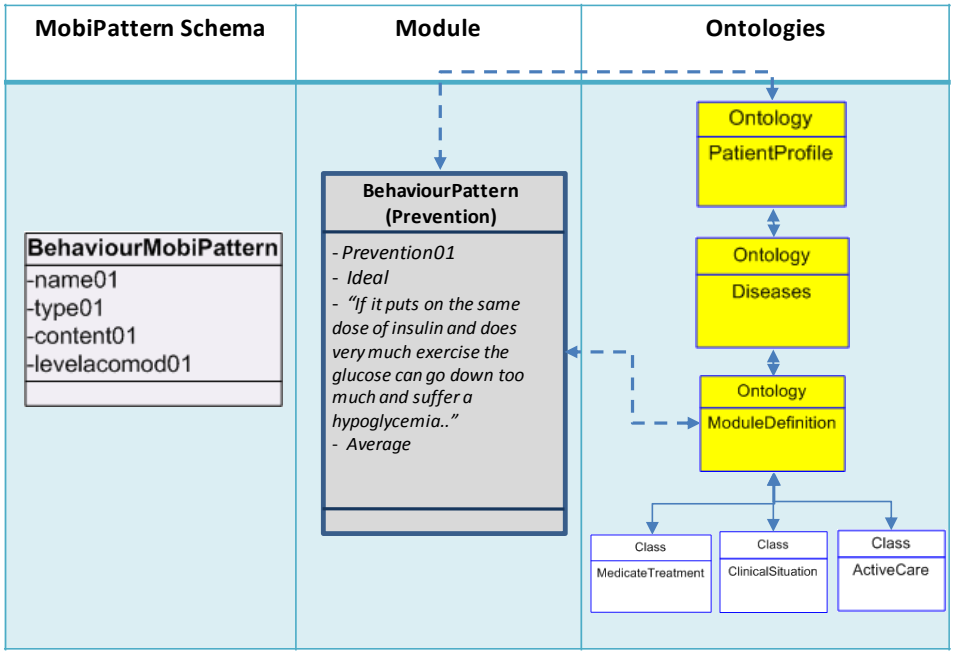


Figura 5-28. *Implementación del BehaviourMobiPattern para la generación del módulo "Prevention".*

5.6.4 Creación del módulo: Autocontrol. (VisualizationMobiPattern)

El módulo de Autocontrol usa el **MobiPattern** de Visualización (**VisualizationMobiPattern**). Los módulos de dieta y alerta también pueden utilizar este patrón con la característica de que cada una de sus ontologías define una especificación diferente para el elemento ***Type***, como se muestra en la tabla 5-8.

Módulo	Elemento del campo *Type*	Descripción
Self-Control	- *Text*	Muestra información (estadísticas) e historiales tipo texto.
	- *Graphic*	Muestra información (estadísticas) e historiales tipo grafico (Tendencias, lineales, etc.)
Diet	- *RestrictedFood*	Define las combinaciones de dietas incluyendo los alimentos restringidos.
	- *ForbiddenFood*	Define las combinaciones de dietas incluyendo los alimentos prohibidos.
	- *AdvisableFood*	Define las combinaciones de dietas incluyendo los alimentos aconsejados.
Alert	- *Information*	Tipo de alerta que se emite al paciente, ante situaciones no muy variadas y poco urgentes. Puede ser enviada también al especialista medico.
	- *Emergency*	Tipo de alerta que se emite al doctor, especialista médico y centro de ambulancias.
	- *Update*	Tipo de alerta que se emite al momento de necesitarse una actualización de la aplicación.

Tabla 5-8. *Especificaciones del elemento Type para los módulos dieta, auto-control y alerta según MobiPattern de Visualización* ***(VisualizationMobiPattern)***

En la tabla 5-9, se muestran los elementos del *framework* que se utilizan para el desarrollo del módulo de autocontrol, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Módulo de autocontrol	VisualizationMobiPattern	-Disease -PatientProfile	**Del Servidor** (comunicación, seguridad, datos) **Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	**- RF07** (CU-14, CU-15)

Tabla 5-9.*Elementos del framework utilizados para la generación del módulo de autocontrol.*

En la figura 5-29, vemos que el módulo de autocontrol se ha generado a partir del **MobiPattern** de Visualización (**VisualizationMobiPattern**), a diferencia que en el campo ***type***, el usuario ver la información en forma de gráfico o texto según le guste.

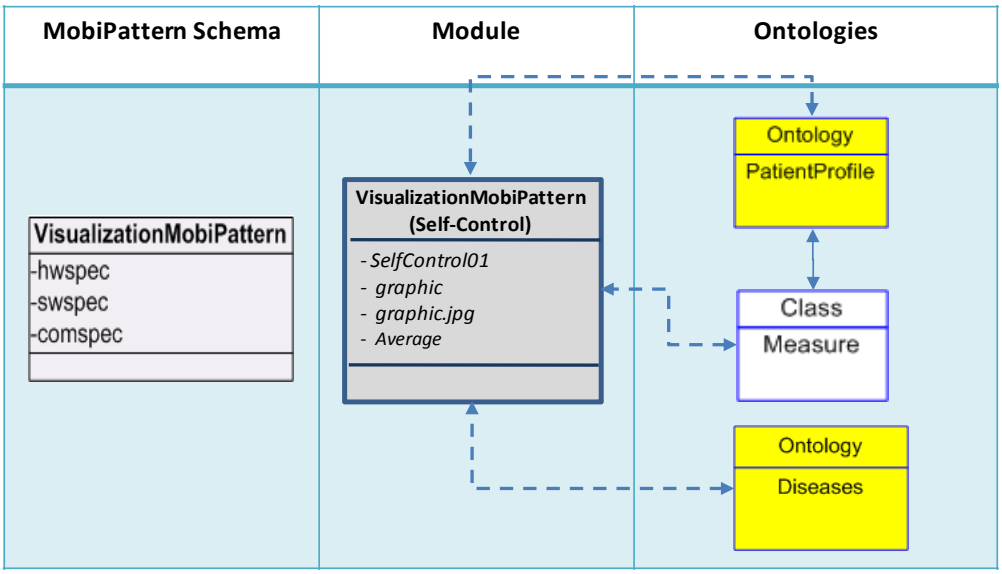


Figura 5-29. *Implementación del VisualizationMobiPattern para la generación del módulo "AutoControl" visto por el usuario de manera gráfica o lista.*

Uso del Patrón y su generalización

La aplicación provee un apartado desde el cual podemos ver en forma de gráfica o en forma de lista las últimas medidas realizadas por el paciente. De esta manera, el usuario podrá llevar un control de enfermedad, así como el médico podrá consultar dichos datos de sus pacientes.

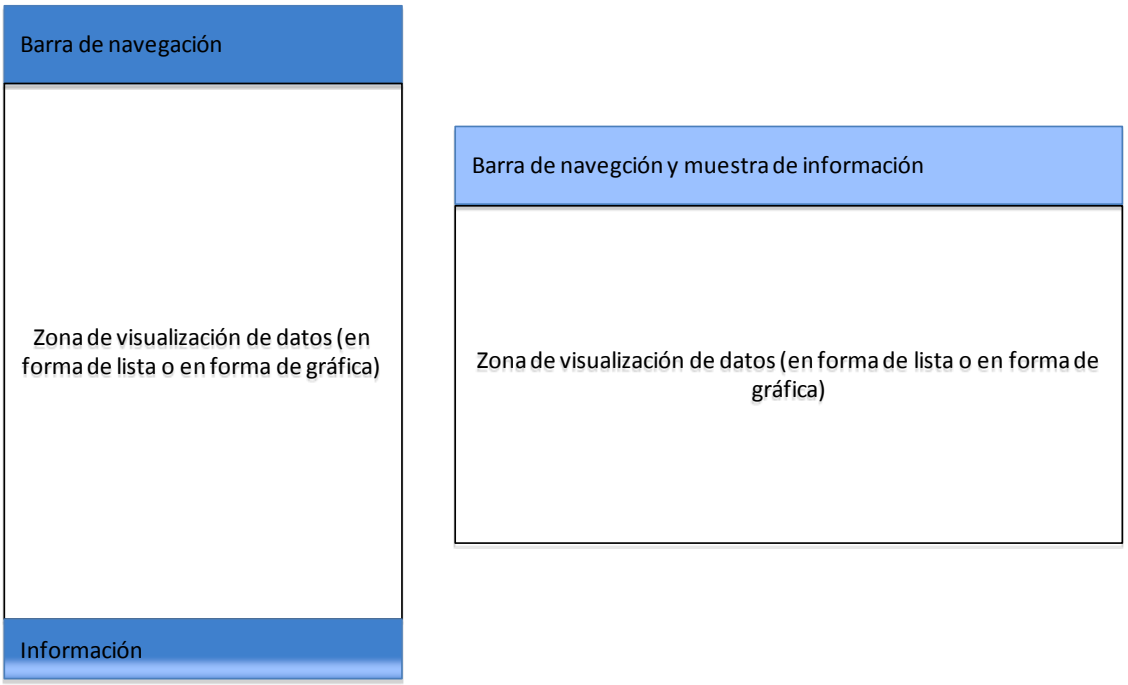


Figura 5-30. *Visualización del patrón de autocontrol: orientación vertical (izquierda) y orientación horizontal (derecha).*

Tal y como se ha mencionado, la interfaz puede mostrar los datos tanto en forma de gráfica, como en forma de lista. La posibilidad de ver los datos usando dos tipos de interfaces nos permite tener un control utilizando un *scroll* vertical (*modo lista*) y la posibilidad de un desplazamiento horizontal (*modo gráfica*).

Nos encontramos también con que la interfaz es distinta según la orientación del dispositivo, para aprovechar los espacios que se producen con la visualización de los datos según en qué posición se encuentre el dispositivo. La estructura de la interfaz con relación al patrón que la define se muestra en la figura 5-30.

Como se puede observar, nos encontramos con dos interfaces distintas, dependiendo de la orientación del dispositivo. Por tanto, si el dispositivo se encuentra en una posición horizontal la barra inferior de información pasa a integrarse en la barra de navegación. De esta manera, el área de visualización de información tiene un espacio mayor y contrarresta la posible falta de espacio impuesta por las características del dispositivo en una posición horizontal.

Modo de Visualización según la orientación del patrón

Para permitir la elección del formato de salida de la interfaz (gráfica o lista), el usuario tendrá en todo momento un botón situado en la esquina superior derecha (incrustado en la barra de navegación) desde el cual podrá elegir la forma de visualización de los datos.

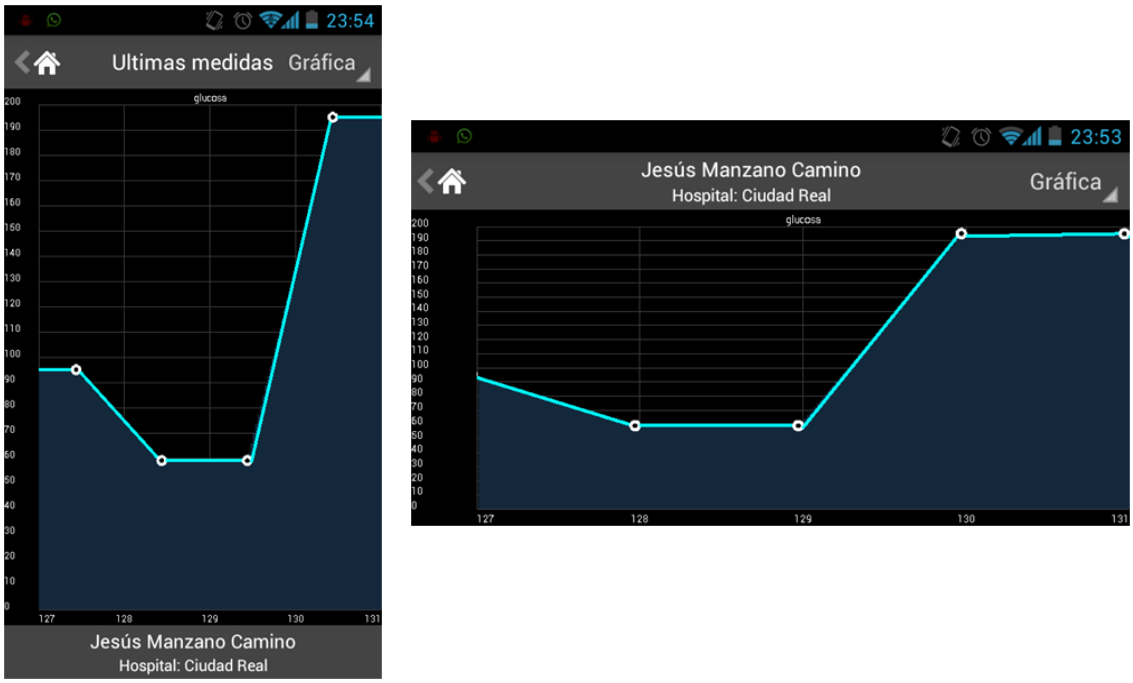


Figura 5-31. *Visualización de la interfaz de autocontrol, según el tipo de visualización gráfico*

En la figura 5-31 y 5-32 se muestran las interfaces generadas según el patrón de orientación vertical y horizontal, para el formato de salida tipo gráfico y tipo lista. Esto le permite al paciente aprovechar las características del dispositivo móvil con que cuenta, teniendo un área de visualización más amplia. Cada interfaz conserva las características propias definidas por el

patrón, reubicando las diferentes barras de navegación según el espacio que tenga cada dispositivo móvil.

Se pueden observar las dos barras de navegación (dos para cada orientación) que se pueden presentar en la interfaz, dependiendo del modo en el que se visualizan los datos y de la orientación del dispositivo. En las barras de navegación que pertenecen a una orientación en horizontal, se observa cómo se incrusta la información en dicha barra, mientras que en las barras de navegación que pertenecen a una orientación en vertical no aparece la información en la barra de navegación.

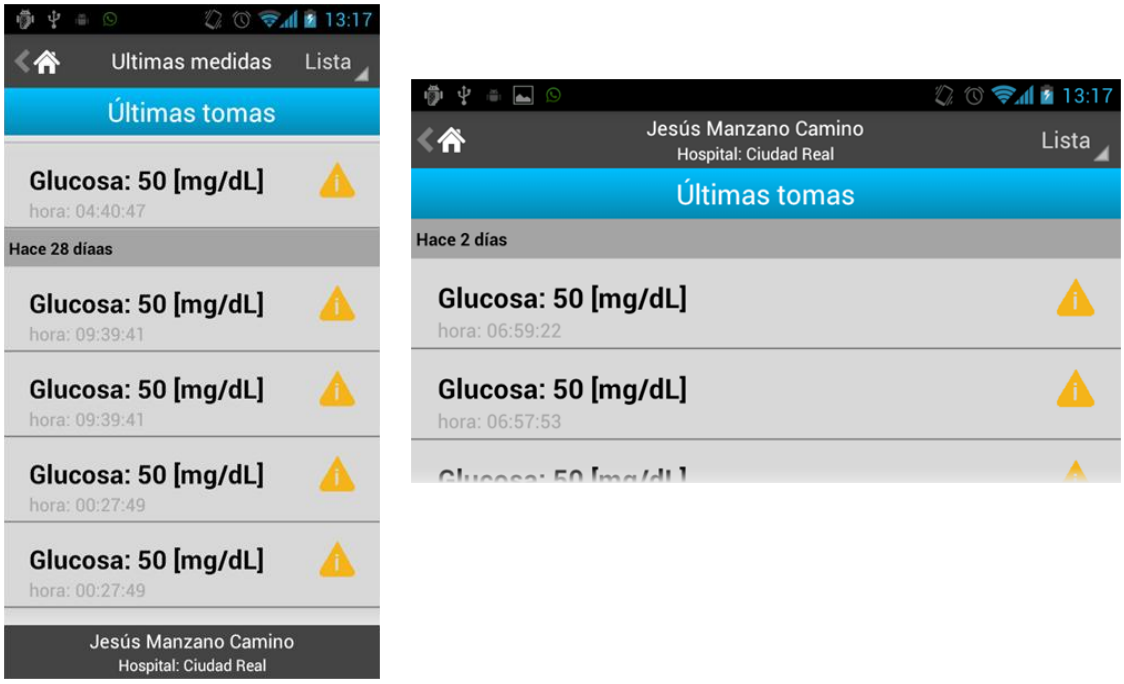


Figura 5-32. *Visualización de la interfaz de autocontrol, según el tipo de visualización: lista*

Las distintas enfermedades se construyen mediante fragmentos, quedando la interfaz que contiene la barra de navegación, la zona inferior de información y la zona contenedora en la que se incrustarán los datos en la Actividad padre. Por lo tanto, la estructura de dichas interfaces quedaría así como se muestra en la figura 5-33.

La variable *mStackLevel* define si la interfaz que contendrá los datos se generará en forma de gráfica o en forma de lista. La clase *CountingFragment* se encuentra empotrada en la clase padre, por lo que ambas se encuentran en el mismo fichero *java*. La clase *CountingFragment* utiliza los mismos datos (se explicará posteriormente el origen de esos datos), pero utiliza formas distintas de mostrarlas.

Para la generación de la lista se utiliza la clase *ListMeasuresAdapter*, al que se le pasan los datos provenientes de la base de datos en red. En el caso de la generación de la gráfica se utiliza la librería *GraphView*, que es importada al proyecto.

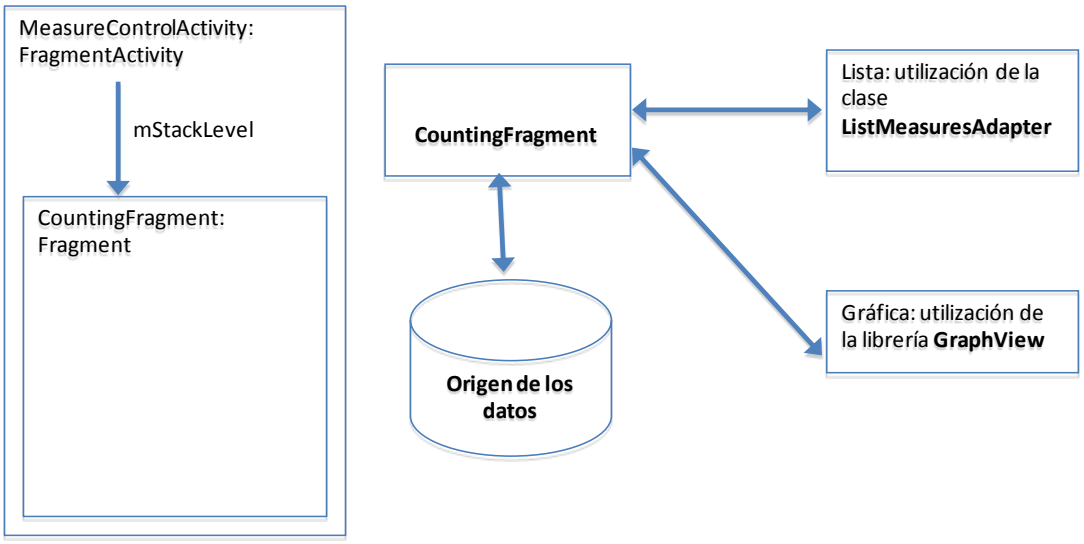


Figura 5-33. *Estructura funcional de los elementos de la interfaz de Auto-Control*

Origen de datos.

Los datos que se utilizan para la generación de las interfaces provienen de la base de datos remota. En ésta ocasión, las distintas medidas que el usuario añadió se descargan a la aplicación y se muestran en el modo seleccionado (gráfica o lista). En la figura 5-34 se muestra la relación entre cada uno de los elementos que definen el origen de datos.

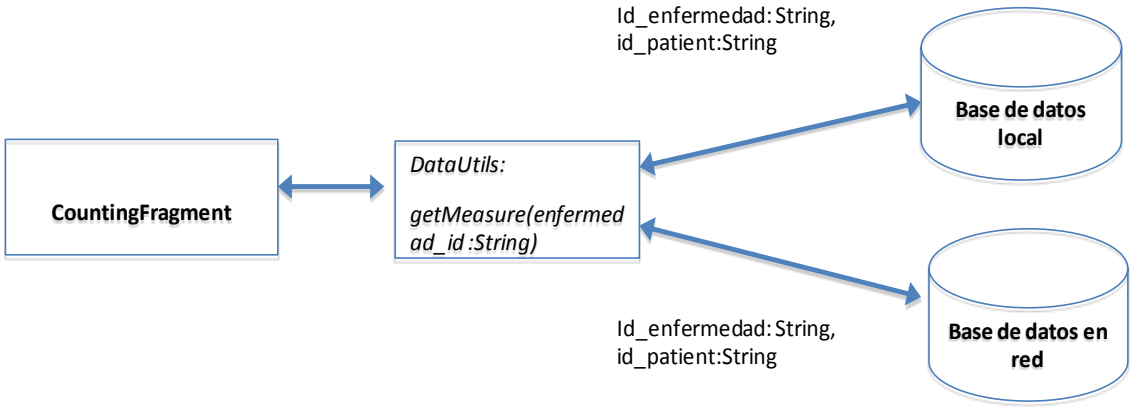


Figura 5-34. *Flujo de datos entre la interfaz de auto-control y la base de datos definida.*

Una de las condiciones contempladas en el aspecto de comunicación, es la posibilidad de que el dispositivo móvil cuente con una base de datos local, que se sincronizará con la base de datos en red. Esto nos permite controlar la situación de conectividad, en el caso de que no haya red disponible, para que la aplicación móvil siga funcionando.

En el listado 5-12, se muestra la relación existente entre el **MobiPattern** de Comportamiento para la generación del módulo gráfico de autocontrol (*Self-Control*) del paciente y la relación con su estructura en **Android** basada en las ontologías del perfil del paciente y sus medidas.

Listado 5-12. *Código de la relación entre el MobiPattern de comportamiento para la generación de módulo gráfico de autocontrol y su Ontología perfil de paciente y medidas.*

```
// Relación de la ontología de medidas y perfil de paciente  según las medidas de una enfermedad a graficar.
        @Override
        public void onClick(View arg0) {
        // TODO Auto-generated method stub
        if (((RadioButton) findViewById(R.id.option1))
                .isChecked()) {
                selected = 0;
                sentenciaSelect = GETGLUCOSE;
        } else if (((RadioButton) findViewById(R.id.option2))
                .isChecked()) {
                selected = 1;
                sentenciaSelect = GETTEMPERATURE;
                } else if (((RadioButton) findViewById(R.id.option3))
                        .isChecked()) {
                        selected = 2;
                        sentenciaSelect = GETPLEASURE;
                        } else {
                                selected = -1;
                                }
                                try {
                                SQliteDB sqlitedb = new SQliteDB(
                                SelectParametersGraphic.this);
                                SQLiteDatabase db = sqlitedb.getReadableDatabase();
                                String[] columns = null;
                                if (selected != -1) {
                                        switch (selected) {
                                        case 0
                        ...
```

5.6.5 Adaptando nuevas funcionalidades e interfaces. (AdaptabilityMobiPattern)

El patrón de adaptabilidad es un patrón de funcionalidad que no define una interfaz gráfica, sino que define las relaciones existentes entre un elemento de algún módulo con otro elemento de otro módulo. En la tabla 5-10, se muestran los elementos del *framework* que se utilizan para la adaptación de nuevas funcionalidades de un módulo, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Funcionalidad de adaptación	AdaptabilityMobiPattern	-ModuleDefinition -PatientProfile	**Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	RF se generan según las funcionalidades deseadas.

Tabla 5-10. *Elementos del framework utilizados para la adaptación de nuevas funcionalidades.*

En la figura 5-35 vemos el esquema general y funcional del **MobiPattern** de Adaptabilidad (**AdaptabilityMobiPattern**) y la relación que existe entre cada uno de sus elementos. Para crear un nuevo módulo con funcionalidades similares a otros módulos, o para modificar la funcionalidad del módulo creado previamente, se establece una relación entre ese módulo

creado previamente y el módulo de medida, el cual recopila toda la información y adapta esa nueva funcionalidad en un nuevo módulo.

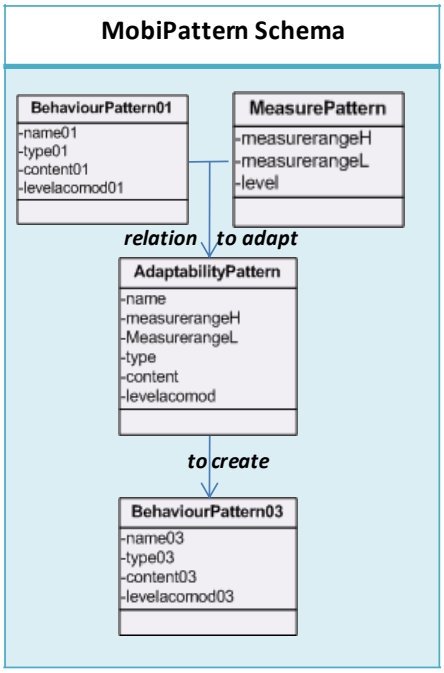


Figura 5-35. *Esquema del AdaptabilityMobiPattern para la adaptación de nuevas funcionalidades*

En la figura 5-36, se muestra cómo funciona este esquema de **MobiPattern** relacionando un módulo de recomendación con el de medidas, para generar un nuevo módulo ***NewSuggestion***, ya que en alguna parte de la aplicación fue solicitada esa adaptación al nuevo módulo.

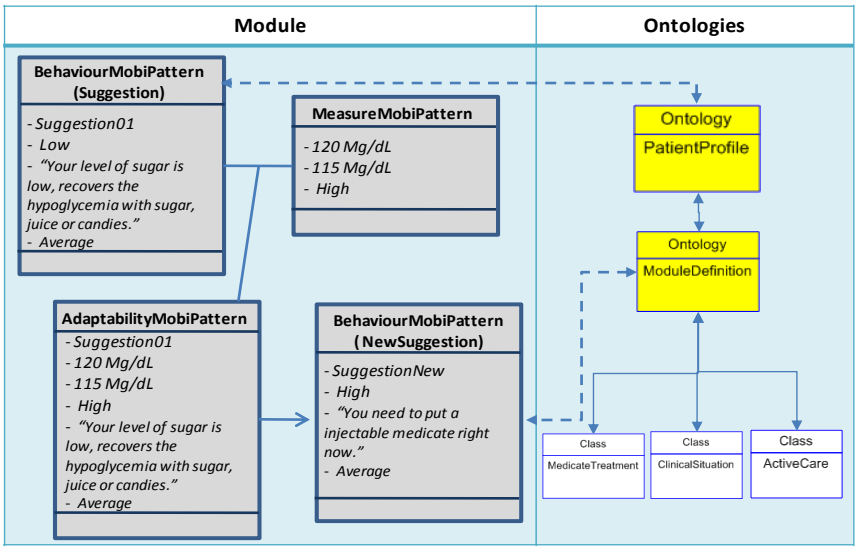


Figura 5-36. *Relación del AdaptabilityMobiPattern con las ontologías perfil de paciente y definición de módulos*

Este patrón permite mediante el esquema facilitar la adición de nuevas funcionalidades a la aplicación, ya sea integrando nuevos elementos de control como también nuevas interfaces de

diseño. Para este capítulo sólo quedará marcado los pasos que se deben seguir y que ayudan a los desarrolladores adaptar nuevas funcionalidades a la aplicación.

Los pasos para adaptar nuevas funcionalidades a diversos módulos, según el patrón de adaptabilidad son:

- **Definir la interfaz que se adaptará**: esta puede ser una nueva interfaz de usuario o cambiar alguna existente. En el caso que se adapte una nueva interfaz, se deben tomar en cuenta los diversos patrones diseñados previamente.
  Si se desea adaptar una nueva se deben seguir con los pasos definidos en este apartado.
- **Seleccionar el tipo de enfermedad**: se debe seleccionar la enfermedad para cual se hará la adaptación. Esto permite definir la funcionalidad de la interfaz de usuario, así como establecer las relaciones entre las ontologías.
- **Definir las ontologías involucradas:** una vez se conozca cual es la enfermedad a adaptar, y las interfaces de usuario requeridas, se seleccionan las ontologías que se utilizarán. Pueden utilizarse las ontologías que propone en *framework* o agregar nuevas ontologías a este.
- **Definir la funcionalidad buscada:** se tiene que definir que funcionalidad tendrá la interfaz de usuario, basándose en los patrones de funcionalidad definidos (FunctionalityMobiPattern).
- **Seguir los pasos definidos en el *framework* para la generación de la arquitectura *software*:** cumplidos todos los pasos anteriores, se siguen todos los pasos para el desarrollo de arquitecturas *software* definidos por el *framework*.

5.6.6 Acomodación de módulos para la generación de Aplicaciones. (AccommodationMobiPattern)

Cómo ubicar cada uno de los módulos generados para nuestra aplicación, es una de las funciones principales del **MobiPattern** de Acomodación (**AccommodationMobiPattern**). Este **MobiPattern** se encarga de definir tres niveles de ubicación (*top, average y low*) para cada uno de sus elementos. Estos tres niveles definen la prioridad con que debe ejecutarse cada uno de los módulos, es decir, se define en que plano estructural de ejecución para cada uno.

En la tabla 5-11, se muestran los elementos del *framework* que se utilizan para la utilización del patrón de acomodación, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Funcionalidad de acomodación	Accommodation MobiPattern	-PatientProfile -ModuleDefinition	**Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	RF se generan según las funcionalidades requerida.

Tabla 5-11. *Elementos del framework utilizados para la adaptación de nuevas funcionalidades.*

Como se muestra en la figura 5-37, los **MobiPattern** de definición (***ProfileMobiPattern*** y el ***MeasureMobiPattern***) se ejecutan en primer plano y ocupan el "*Top Level*" dentro del **MobiPattern** de Acomodación. De igual manera todos los **MobiPattern** de Funcionalidad (***Suggestion, Prevention, Alert, Diet, entre otro***) se ejecutan en segundo plano, es decir ocupan el "*Average Level*". Y en un tercer plano, a nivel más inferior se ejecuta los módulos del ***Modelo Predictivo*** y de ***reajuste*** de las aplicaciones.

La división en niveles de este **MobiPattern** facilita la ubicación de cada uno de sus elementos en la generación de la aplicación final.

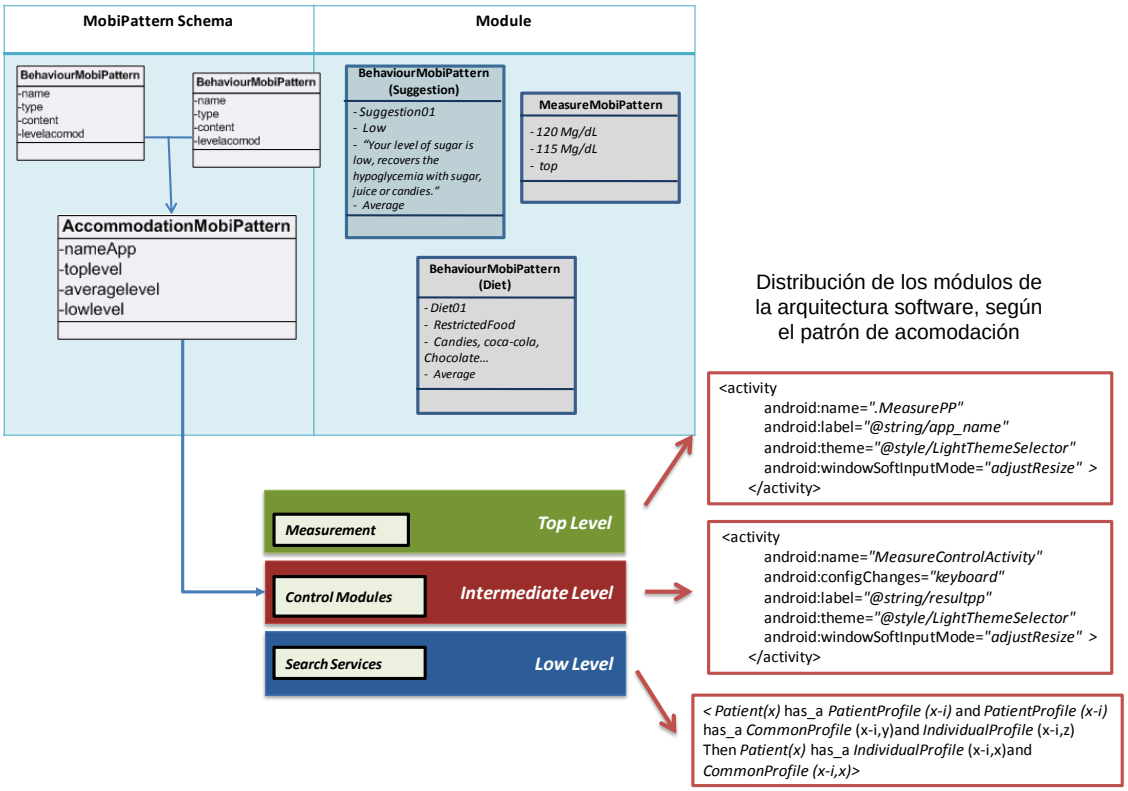


Figura 5-37. *Implementación del AccommodationMobiPattern ubicación de todos los elementos de la arquitectura software.*

Este patrón no define una interfaz gráfica, sino que define la forma en que se organizan todos los módulos dentro de la aplicación final. En la parte inferior de la figura 5-37, se define la codificación que distribuye cada módulo según su funcionalidad entre las tres capas definidas por el patrón.

Si se desea agregar nuevos módulos es necesario ubicarlos en esta distribución, ya que permite la ejecución ordenada de cada elemento y la identificación al momento de darle mantenimiento a la aplicación.

5.6.7 Relación funcional entre los módulos de la aplicación. (ExecutionAppMobiPattern)

En la tabla 5-12, se muestran los elementos del *framework* que se utilizan para la utilización del patrón de ejecución, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Funcionalidad de ejecución	ExecutionApp MobiPattern	-Measure -ModuleDefinition -PatientProfile	**Del Dispositivo Móvil** (aplicación, seguridad, comunicación)	RF se generan según las funcionalidades deseadas.

Tabla 5-12. *Elementos del framework utilizados para la relación funcional entre módulos.*

En la figura 5-38 se puede ver la funcionalidad del **MobiPattern** de Ejecución (**ExecutionAppMobiPattern**) y la forma en que se relacionan cada uno de los módulos creados para cada **MobiPattern** respectivo. Es decir, que luego de generados cada uno de los módulos que se necesitan en la aplicación final, estos módulos deben tener un grado de relación lo que permite la comunicación e intercambio de información entre ellos.

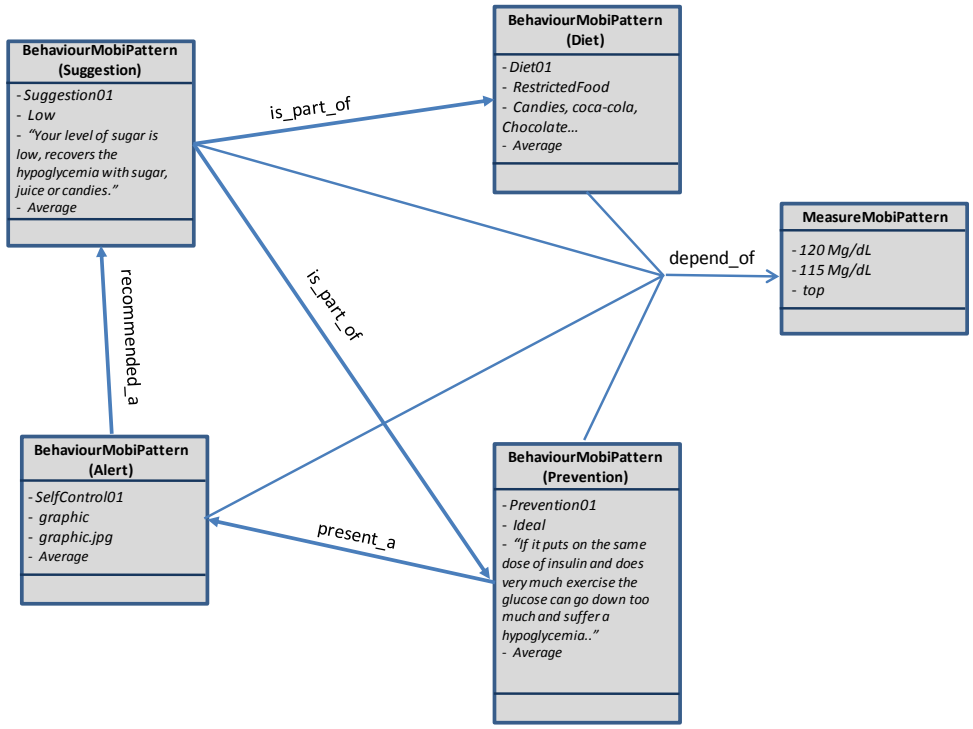


Figura 5-38. *Evaluación del ExecutionAppMobiPattern para la generación de las relaciones entre cada uno de los módulos previamente generados.*

Además de establecer relaciones entre módulos, también se pueden definir relaciones entre otros elementos del *framework* como son las ontologías.

En el listado se muestra un fragmento de la codificación desarrollada en la arquitectura *software*, en donde se definen cada uno de los módulos que han sido desarrollados, según cada patrón explicado previamente. Esto permite conocer la relación funcional entre cada uno de ellos de tal manera que se relacionan en tiempo de ejecución.

Listado 5-13. *Código de la definición de cada módulo principal y la relación existente entre ellos.*

```
    <application
android:name=".ActionBarApplication"
android:icon="@drawable/logo_mini"
android:label="@string/app_name"
android:theme="@style/Theme.GDActionBarExample" >

<!-- android:theme="@style/LightThemeSelector" -->
<!--  -->
<activity
  android:name=".Start"
  android:label="@string/app_name"
  android:screenOrientation="portrait" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
<activity
  android:name=".PatientProfileAc"
  android:label="@string/app_name"
  android:theme="@style/LightThemeSelector"
  android:windowSoftInputMode="adjustPan|stateHidden" >
  <intent-filter>
    <action android:name="android.intent.action.patientprofilely" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
<activity
  android:name=".PatientProfileTour"
  android:label="@string/app_name"
  android:screenOrientation="portrait"
  android:theme="@style/LightThemeSelector"
  android:windowSoftInputMode="adjustPan|stateHidden" >
  <intent-filter>
    <action android:name="android.intent.action.patientprofilely" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
<activity
  android:name=".PMenu"
  android:label="@string/app_name"
  android:windowSoftInputMode="adjustUnspecified" >
</activity>
<activity
  android:name=".MeasurePP"
  android:label="@string/app_name"
  android:theme="@style/LightThemeSelector"
  android:windowSoftInputMode="adjustResize" >
</activity>
```

5.7 IMPLEMENTACIÓN DEL ROL DE MÉDICO

En los puntos anteriores se explicó cómo se habían diseñado las diferentes interfaces, basadas en patrones y analizando el flujo de datos, para la generación de la aplicación para

pacientes. Como ya se ha mencionado previamente, la aplicación puede ser usada, tanto por un paciente como por el médico.

Ahora se procede a realizar los mismos pasos mostrando las pantallas y los flujos de datos cuando se utiliza la aplicación con el rol de doctor. Las pantallas de introducción de datos y de elección del idioma son compartidas por ambos roles, con la diferencia de que en el caso de la selección de un rol a la hora de introducir los datos del usuario, si el usuario introducido tiene un rol de doctor, no se le presenta la pantalla de introducción de enfermedades a monitorizar.

En la tabla 5-13, se muestran los elementos del *framework* que se utilizan desarrollar los módulos del médico, el patrón a utilizar, las ontologías relacionadas, las capas de los elementos que intervienen y los requisitos funcionales y casos de uso que se cubren.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales
Módulo de médico	DoctorMobiPattern	-Measure -ModuleDefinition -PatientProfile	**Del Dispositivo Móvil** (aplicación, seguridad, comunicación) **Del Servidor** (comunicación, seguridad, datos)	**- RF09** (CU-19, CU-20, CU-21) **- RF07** (CU-14, CU-15) **- RF06** (CU-09, CU-10, CU-11, CU-12)

Tabla 5-13. *Elementos del framework utilizados para la generación de la aplicación del médico.*

5.7.1 Ver listado de pacientes

Uso del Patrón y su generalización

Cuando el médico inicia la aplicación, se encuentra con una interfaz en la que aparece un listado con todos los pacientes a cargo de él. En esta lista, se ofrece una primera información sobre sus incidencias más recientes.

En la figura 5-39 a la izquierda, nos encontramos con una pantalla que sigue la estructura general mostrada en esquemas anteriores, con una barra de navegación (en este caso meramente informativa) y una interfaz contenedora que en este caso contendrá una lista (DoctorMobiPattern) en la que aparecerán datos relevantes del usuario, como se puede ver en la figura del centro.

A la parte derecha de la figura podemos ver que la lista contiene el identificador y nombre de los pacientes. El doctor al tocar un elemento de la lista, accederá a otra interfaz (se mostrará posteriormente) en la que podrá ver de forma detallada sus datos y acceder a algunas opciones. No obstante, se pueden acceder a dichas opciones si el doctor toca durante 1 segundo el elemento de la lista sobre el cual desea ver las opciones. En ese momento la aplicación mostrará un menú en el que se indican las opciones, pudiendo seleccionar cualquiera de ellas.

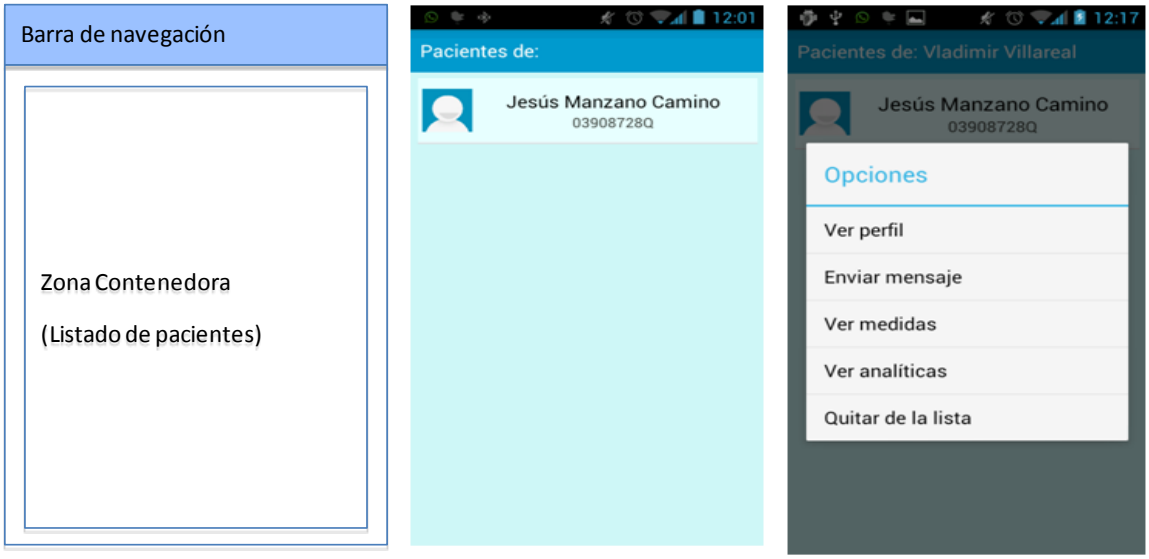


Figura 5-39. *Visualización de las interfaces del paciente, según el DoctorMobiPattern.*

Origen de datos

Una vez mencionado el esquema de la interfaz y su representación, se procede a explicar el origen y flujo de los datos con los que tratará la interfaz. Este flujo estará principalmente orientado a la recolección de los datos de los usuarios de un médico en particular (representado este por su identificador). A continuación se indica un esquema en el que se muestran las clases y rutinas que realizan el flujo en la figura 5-40.

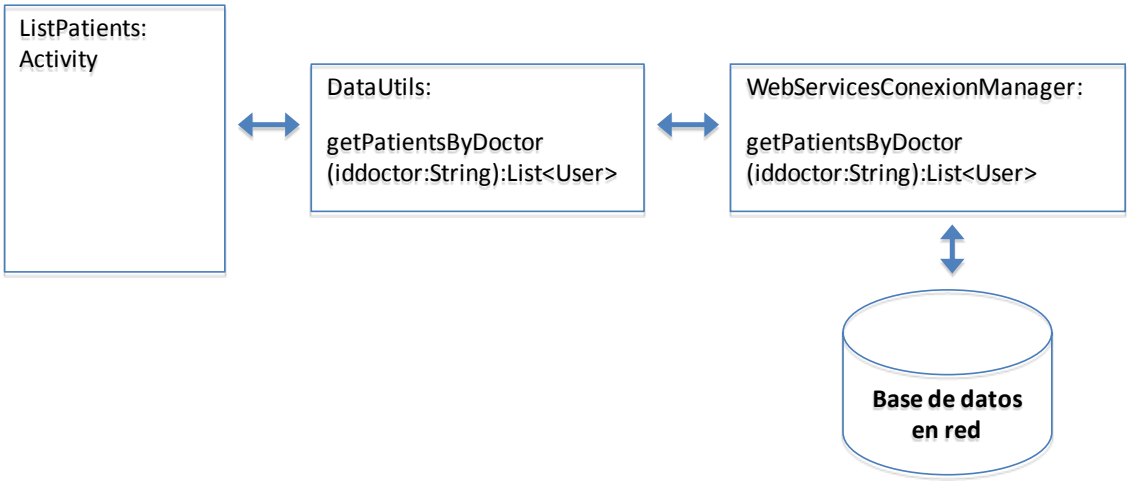


Figura 5-40. *Flujo de datos entre la interfaz de listado de pacientes y la base de datos definida.*

El flujo de los datos es sencillo, se realiza una petición de pacientes cuyo médico sea igual que el indicado por el parámetro *iddoctor*, que será propagada por las capas de la aplicación (a través del servicio web) hasta devolver el listado de pacientes asignados a dicho médico. Estos datos son devueltos a la clase *ListPatients* que rellena una lista con los datos obtenidos. En caso de que no haya pacientes asignados al médico que está utilizando la aplicación, se muestra un mensaje indicando que puede buscarlos.

5.7.2 Información específica de un paciente.

Uso del Patrón y su generalización

Se comentaba al explicar la interfaz anterior, que el médico (usuario de la aplicación), una vez cargada la lista con los pacientes que tiene asignados, podía acceder a información más detallada sobre éstos y realizar ciertas acciones de seguimiento médico

Esta interfaz mostrará de manera detallada información acerca del paciente que fue seleccionado, así como un listado con sus incidencias y opciones que el médico puede realizar sobre él (escribir un mensaje, ver últimas tomas de medidas, etc.).

En esta interfaz, siguiendo con el esquema del DoctorMobiPattern, se ha optado por usar bloques separados visualmente para los distintos datos, por lo que nos encontramos con un bloque para la información detallada, otro para las últimas incidencias y otro para las opciones que el médico puede realizar.

Para utilizar esta interfaz el médico recibe en un primer momento, una cantidad de información apropiada, dichos bloques pueden contraerse y expandirse, de tal manera que los bloques aparecen inicialmente contraídos, y el médico al tocar los bloques los expande, accediendo a información detallada sobre dicho bloque. De igual manera, si el médico quiere contraer la información que ha sido expandida, no tiene más que volver a tocar el bloque, el cual se contraerá mostrando la información básica del bloque. La interfaz consta de tres bloques básicos:

- **Información de paciente**: en dicho bloque se mostrará la información del paciente. En un primer momento se mostrará la información básica, pero si se expande se accede a toda la información del paciente.

- **Incidencias**: En este bloque se mostrará en las últimas incidencias del usuario, pero si es expandido mostrará un número mayor de éstas. Se situará un botón al final del bloque expandido desde el cual se podrán ver más incidencias.

- **Opciones**: En este bloque se verán las opciones que el médico puede realizar sobre el paciente. Estas opciones solo serán visibles una vez que el bloque ha sido expandido.

Las opciones que el doctor puede realizar sobre el paciente del cual se muestra la información son las siguientes:

- **Ver resultados de analíticas**: el médico podrá ver los resultados de las últimas analíticas que se ha hecho el paciente.

- **Envío de mensajes**: el médico podrá enviar un mensaje al paciente.

- **Visualización de últimas medidas**: el médico accederá a la interfaz de muestra de últimas medidas donde se mostrarán las últimas medidas realizadas por el paciente.

- **Eliminación del usuario de su lista de pacientes**: El médico podrá descartar al usuario de su lista de pacientes.

En la figura 5-41, se muestra al lado izquierdo el esquema de la interfaz basado en **DoctorMobiPattern**, en donde se especifica en la zona contenedora la organización del patrón para generar la interfaz gráfica que se muestra al lado derecho de la imagen.

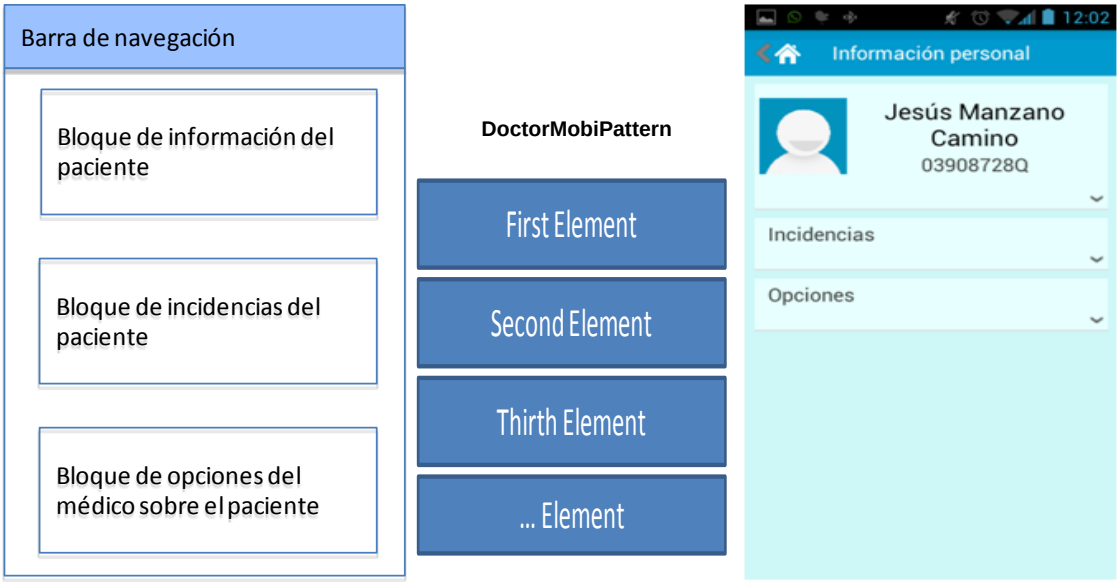


Figura 5-41. *Visualización de la información de un determinado paciente, según el DoctorMobiPattern.*

Se puede observar los elementos principales de la interfaz así como los distintos bloques contraídos (al inicio de la interfaz).

Figura 5-42. *Visualización de todos los elementos de información de un paciente.*

En la figura 5-42 se muestran los elementos faltantes de la información de un paciente seleccionado por el médico. Se pueden ver en más detalle los distintos bloques expandidos. En el segundo bloque, vemos como al final de las incidencias mostradas se posiciona un botón, que al ser presionado cargará toda la lista de incidencias de un paciente (alertas, mensajes generados, etc.). En el bloque de opciones se aprecian las opciones principales que el doctor podrá seleccionar.

Se puede notar en todas las interfaces que se sigue el patrón (**DoctorMobiPattern**) definido para el diseño y desarrollo del rol de médico. El patrón en bloque permite la adición de nuevos elementos sin la necesidad de modificar la estructura de las interfaces.

Origen de datos

Tanto las acciones como la muestra de información hacen uso de datos referentes a un paciente único, o a la relación del paciente de la aplicación con el médico. En la figura 5-43, se muestra el esquema de las rutinas y flujos de datos que se realizan desde los distintos bloques de la interfaz.

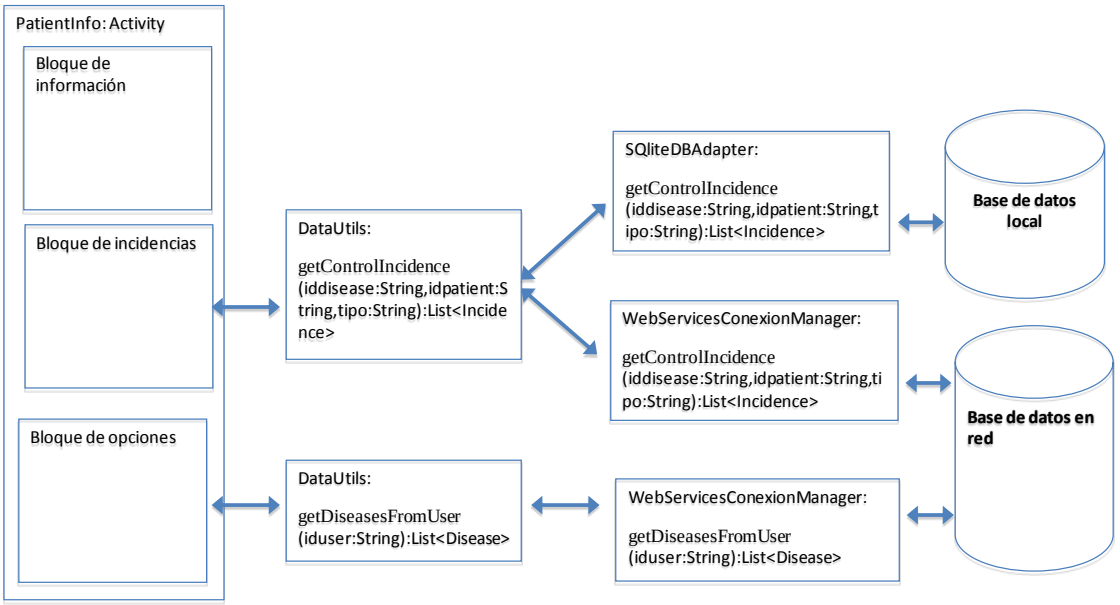


Figura 5-43. *Flujo de datos para la obtención de la información de un determinado paciente, según el DoctorMobiPattern.*

5.8 DEFINICIÓN Y UBICACIÓN DE CAPAS

Las capas definidas en el capítulo cuatro especifican la relación que existe entre cada elemento que interviene en la arquitectura. Es decir, la manera en que se comunican, transfieren información y las almacena. Todos los módulos que se han definido en la aplicación final utilizan de alguna manera una o más capas para la comunicación entre los diversos dispositivos que hemos mencionado. En este apartado definiremos de manera general las principales implementaciones que se han realizado para definir la funcionalidad de cada capa. Estas capas ofrecen servicios a cada uno de los elementos de la aplicación. Mencionaremos las capas que intervienen y algunos elementos que la especifican.

Capas del Servidor

- **Capa de comunicación:**

Esta capa especifica el tipo de comunicación para transmitir/recibir los datos desde y hacia el servidor. En esta capa se ubican los servicios web, alojados en el servidor y que permite el acceso a la base de datos del servidor.

Siguiendo la estructura de la capa de comunicación del servidor, en la figura 5-44, identificamos cada uno de los elementos desarrollados en la arquitectura *software* presentada, según el modelo de distribución en capa.

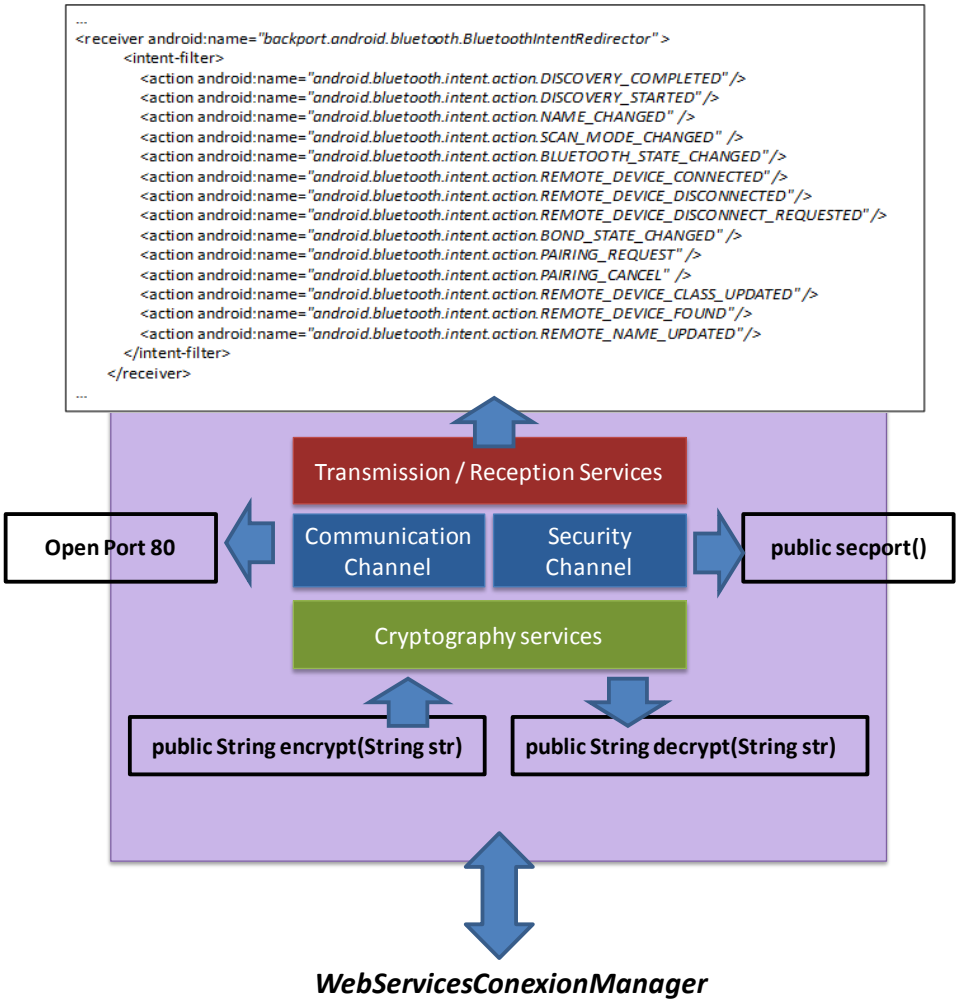


Figura 5-44. *Identificación de los elementos de la capa de comunicación del servidor.*

Un servicio web es una pieza de *software* que utiliza una serie de protocolos para intercambiar datos entre aplicaciones. La ventaja que ofrecen los servicios web es la independencia sobre la plataforma que los use y del lenguaje de programación desde el cual se haga uso. Esta independencia de las plataformas que se conecten se consigue gracias a la adopción de estándares abiertos. Las rutinas que contiene el servicio web alojado en el servidor se muestra la tabla 5-14.

Método 1	`public List<String> getDiabetesRecomendation(String rango)`
	Este método devuelve las recomendaciones asociadas a la diabetes en función del rango o nivel de riesgo.
Método 2	**`public Patient getPatient(string patientId)`**
	Devuelve el usuario (denominado en el servicio web mediante la clase Patient) asociado al identificador *patientId*.
Método 3	**`public List<Patient> getPatientsFromDoctor(string doctorId)`**
	Devuelve los pacientes asociados al doctor cuyo identificador es *doctorId*.
Método 4	**`public List<Disease> getDiseases()`**
	Devuelve una lista con todas las enfermedades que existen en la base de datos en red.
Método 5	**`public void addDisease(Disease disease)`**
	Permite la adición de una enfermedad (establecida por el objeto Disease) a la base de datos.
Método 6	**`public int addPatient(String PatientId,String NamePP,String AddressPP,String MphonePP,String LphonePP,String NcontactPP, String EmailPP, int rol)`**
	Permite la adición de un usuario establecido por los campos que se definen como atributos.
Método 7	**`public int addMeasure(String MeasureId,String IDpatient,String Medida,String Time, String IDCondicion)`**
	Este método permite la adición y almacenamiento de una medida (cuyos elementos vienen definidos por los atributos).
Método 8	**`public int addIncidenceControl(String idPatient, String incidencia, String time,String Tipo)`**
	Define la adición de una incidencia (cuyos elementos vienen definidos por los atributos).
Método 9	**`public List<Incidence> getIncidences(String idPatient,String tipo)`**
	Mediante este método se obtienen todas las incidencias asociadas al un usuario (mediante su identificador) y al tipo de incidencia (para filtrar el contenido).
Método 10	**`public String getRol (String id)`**
	Se obtiene el rol (doctor, paciente) del usuario definido por su identificador.
Método 11	**`public int addPatientToDoctor(String idPatient,String idDoctor)`**
	Permite la asociación de un usuario paciente (definido por *idPatient*) a un doctor (definido por *idDoctor*), existiendo ambos usuarios en la base de datos.
Método 12	**`public List<Measure>getMeasures(String idPatient,String idDisease)`**
	Permite la obtención de todas las medidas introducidas por el usuario (definido por su identificador *idPatient*) de la enfermedad definida por *idDisease*.
Método 13	**`public List<Message> getLastMessagesFromId(String idUser, String lastTime)`**
	Se obtienen los mensajes del usuario (destinatario definido por *idUser*) que fueron enviados después de la fecha establecida por el atributo *lastTime.*
Método 14	**`public List<Disease> getDiseasesFromUser(String idUser)`**
	Se obtienen todas las enfermedades asociadas al usuario definido por el atributo *userId*.
Método 15	**`public int linkUserDisease(String idDisease, String idUsuario)`**
	Se asocia la enfermedad definida por *diseaseId* al usuario definido por su identificador *userId.*

Tabla 5-14. *Especificación de los métodos del servicio web definidos en el capa de comunicación del servidor.*

Las principales razones por las que se adopta la implementación de servicios web son la gran interoperabilidad, el hecho de que los servicios web estén basado en texto y funcionen por el puerto 80 (evitando así el bloqueo por parte de los corta fuegos) y la flexibilidad que nos aporta realizar un servicio independiente de la plataforma que lo consuma.

El servicio web que tratamos en concreto contiene unas rutinas que permitirán a la aplicación **Android** (o cualquier aplicación con conectividad a servicios web) acceder a los datos de la base de datos.

Todos las consultas a las rutinas del servicio web son manejadas por la clase *WebServicesConexionManager*, que se encarga tanto de la conexión y cierre como del tratamiento de los datos para que puedan ser pareados a datos que la aplicación más tarde pueda manejar. En esta clase por tanto nos encontramos con todos los métodos de llamada a las rutinas del servicio web, y es implementada mediante un patrón *Singleton* para evitar problemas con múltiples instancias de dicha clase.

En la figura 5-45 se muestra una aproximación en forma de esquema de la conexión entre la aplicación y el servicio web.

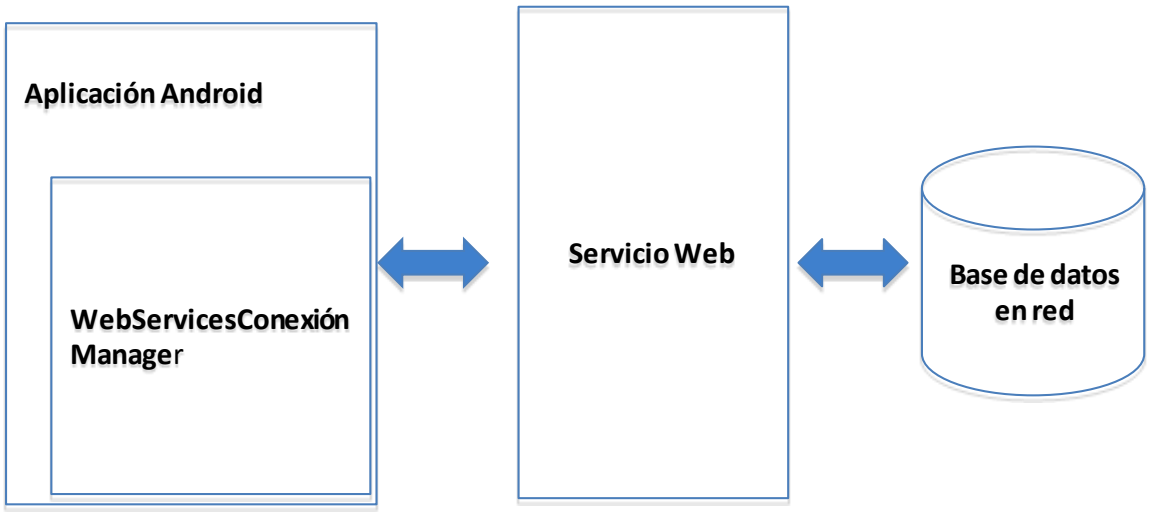


Figura 5-45. *Esquema de conexión entre la aplicación y el servicio web definido en la capa de comunicación del servidor*

- **Capa de Seguridad:**

Siguiendo la estructura de la capa de seguridad del servidor, en la figura 5-46, identificamos cada uno de los elementos desarrollados en la arquitectura *software* presentado, según el modelo de distribución en capa.

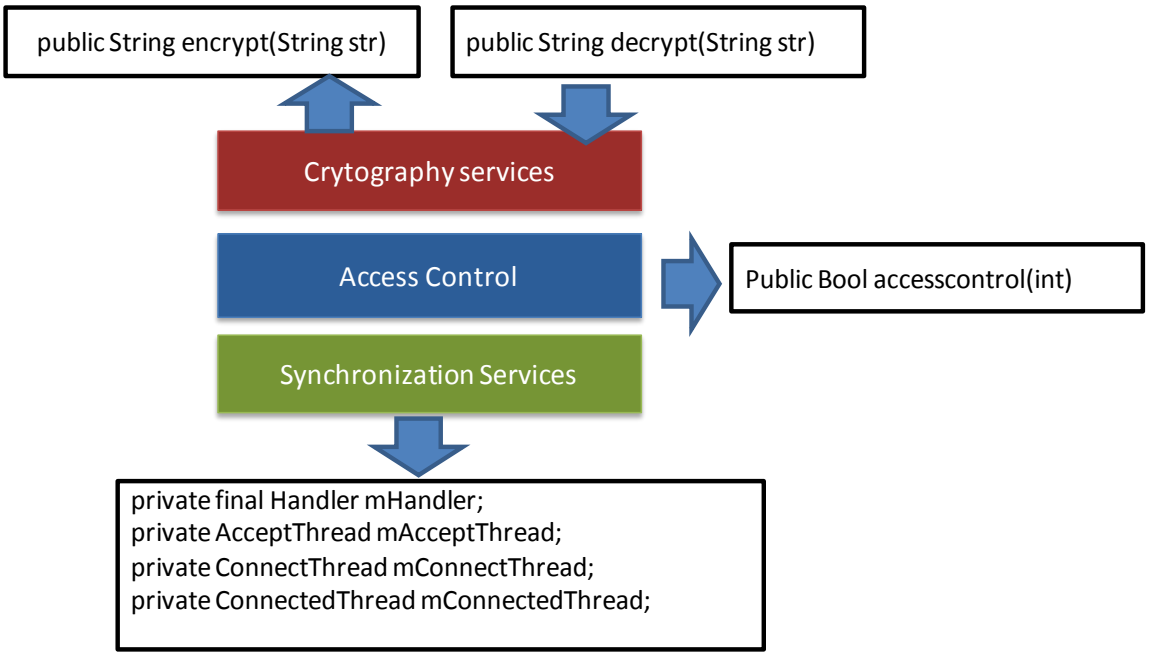


Figura 5-46. *Identificación de los elementos de la capa de seguridad del servidor.*

Esta capa establece los parámetros de seguridad para transferencia de datos entre todos los dispositivos biométricos; se utilizan para ello la librería *javax.crypto*. Con esta librería se especifican las clases para encriptar y desencriptar los datos enviados y recibidos entre los dispositivos móviles y el servidor. El listado 5-14 muestra la codificación de una parte de esta librería.

Listado 5-14. *Codificación que define la capa de seguridad del servidor para la transferencia de datos entre dispositivos móviles.*

```
          …
public class DesEncrypter {
   Cipher ecipher;
    Cipher dcipher;
    DesEncrypter(SecretKey key) {
        try {
            ecipher = Cipher.getInstance("DES");
            dcipher = Cipher.getInstance("DES");
            ecipher.init(Cipher.ENCRYPT_MODE, key);
            dcipher.init(Cipher.DECRYPT_MODE, key);
        } catch (javax.crypto.NoSuchPaddingException e) {
        } catch (java.security.NoSuchAlgorithmException e) {
        } catch (java.security.InvalidKeyException e) { }
    }
    public String encrypt(String str) {
        try {
            // Encode the string into bytes using utf-8
            byte[] utf8 = str.getBytes("UTF8");

            // Encrypt
            byte[] enc = ecipher.doFinal(utf8);
            // Encode bytes to base64 to get a string
            return new sun.misc.BASE64Encoder().encode(enc);
        } catch (javax.crypto.BadPaddingException e) {
```

```
        } catch (IllegalBlockSizeException e) {
        } catch (UnsupportedEncodingException e) {
        } catch (java.io.IOException e) { }
        return null; }
    public String decrypt(String str) {
        try {
            // Decode base64 to get bytes
            byte[] dec = new
            sun.misc.BASE64Decoder().decodeBuffer(str);
            // Decrypt
            byte[] utf8 = dcipher.doFinal(dec);
            // Decode using utf-8
            return new String(utf8, "UTF8");
        } catch (javax.crypto.BadPaddingException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (UnsupportedEncodingException e) {
        } catch (java.io.IOException e) { }
        return null; } }
        …
```

- **Capa de Datos**

La capa de datos del servidor se encarga del tratamiento de datos, es decir, recepción, procesamiento y almacenamiento de estos datos desde los dispositivos móviles.

Siguiendo la estructura de la capa de datos del servidor, en la figura 5-47, identificamos cada uno de los elementos desarrollados en la arquitectura *software* presentada, según el modelo de distribución en capa.

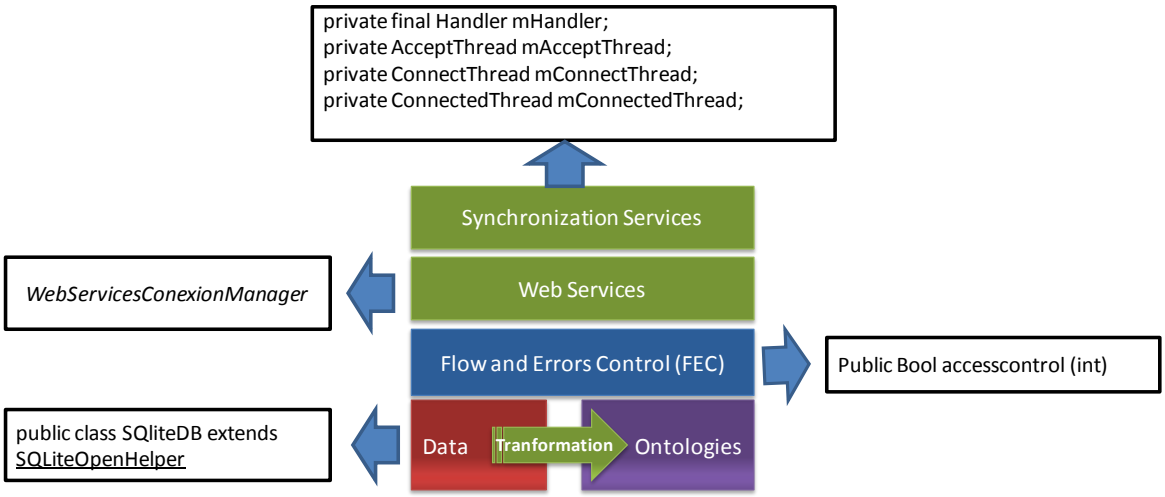


Figura 5-47. *Identificación de los elementos de la capa de datos del servidor.*

La capa de datos está definida por las ontologías que a su vez definen las principales tablas que conforman la base de datos. Además de conformar la base de datos, las ontologías definen los elementos involucrados en la búsqueda de información, que también forma parte de la capa de datos. En la figura 5-48 se muestra el diagrama EER que describe la estructura de la base de datos implementada y que es administrada por la capa de datos del servidor.

Capas del Dispositivo Móvil

- **Capa de Aplicaciones**

Siguiendo la estructura de la capa de aplicaciones del dispositivo móvil, en la figura 5-49, identificamos cada uno de los elementos desarrollados en la arquitectura *software* presentada, según el modelo de distribución en capa.

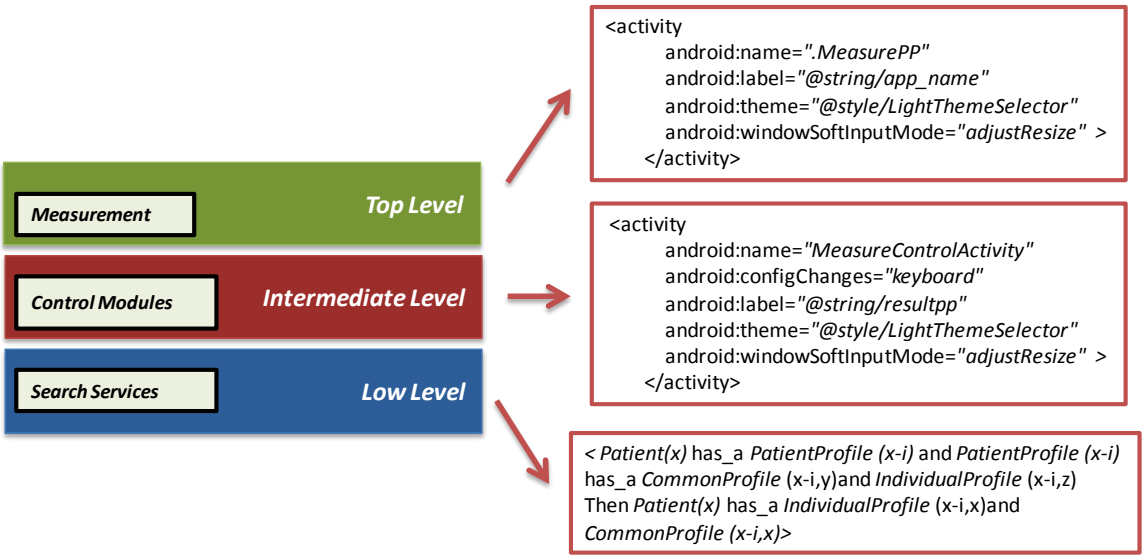


Figura 5-48. *Identificación de los elementos de la capa de aplicaciones del dispositivo móvil.*

El listado 5-15 muestra la organización de la capa de aplicación. Esta capa contiene todas las llamadas a los diversos módulos que componen la aplicación final. Si se necesita modificar algún elemento o módulo de la aplicación, se identifica en la lista de módulos de la capa y se realizan los diferentes cambios.

Listado 5-15. *Ubicación del módulo de perfil de paciente en la capa de aplicación.*

```
…
 <activity>
    android:name=".PatientProfileTour"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@style/LightThemeSelector"
    android:windowSoftInputMode="adjustPan|stateHidden" >
    <intent-filter>
    <action
    android:name="android.intent.action.patientprofilely" />
   <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
 <activity
    android:name=".MeasurePP"
    android:label="@string/app_name"
    android:theme="@style/LightThemeSelector"
    android:windowSoftInputMode="adjustResize" >
</activity>
…
```

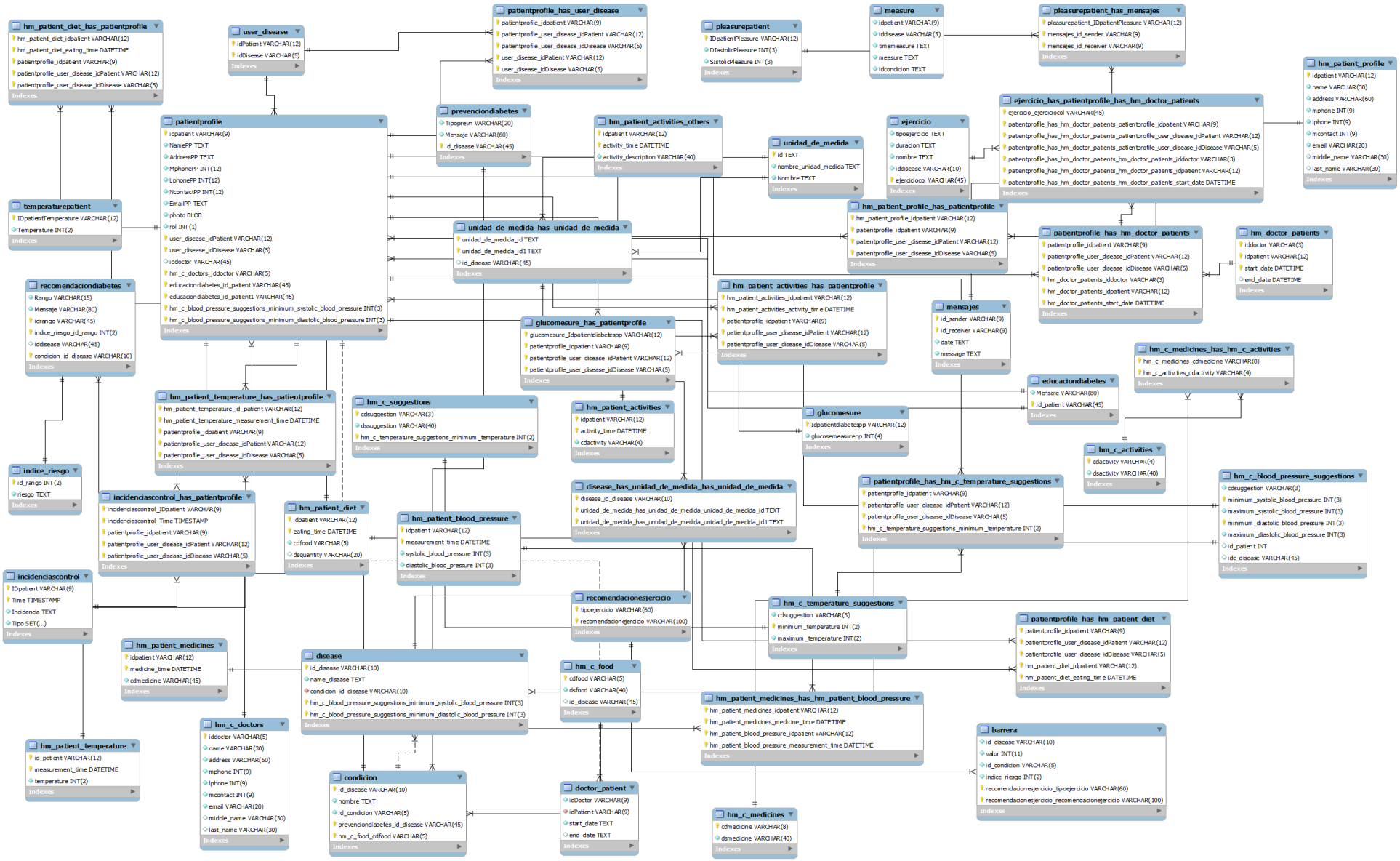


Figura 5-49. *Diagrama EER que define la estructura de la capa de datos según la base de datos del servidor*

- **Capa de Seguridad**

Al igual que la capa de seguridad del servidor, se ha implementado la librería *javax.crypto* para encriptar y desencriptar los datos recibidos desde y hacia el servidor. Estos datos deben ser protegido tanto al momento de ser enviados hacia el dispositivo móvil como al momento de ser recibidos; esto evita la intercepción de los paquetes de datos transmitidos.

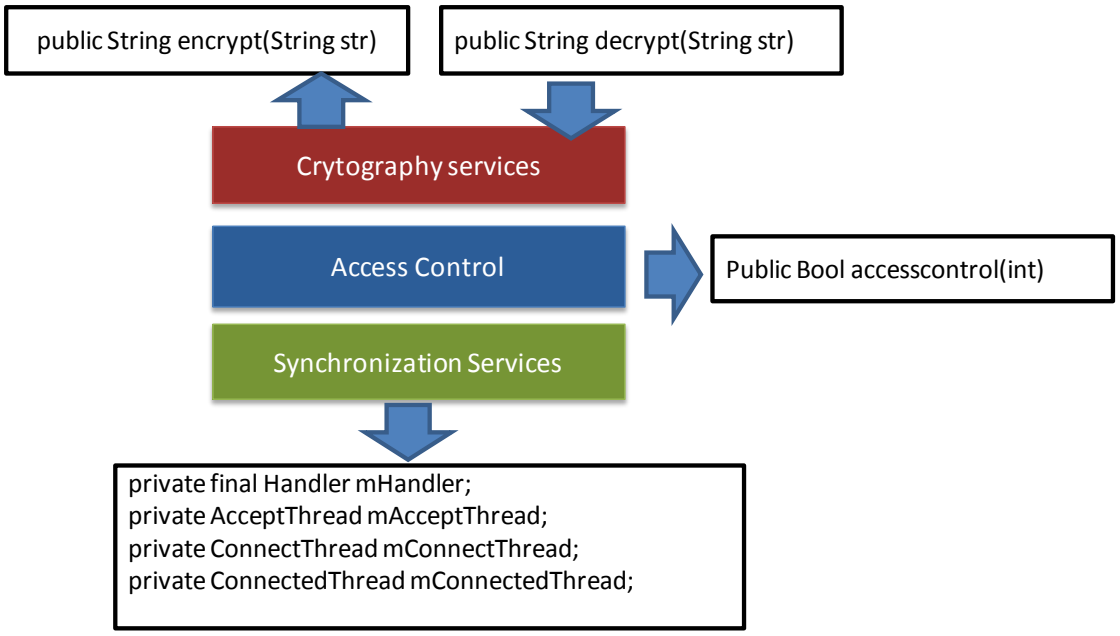


Figura 5-50. *Identificación de los elementos de la capa de seguridad del servidor.*

En la figura 5-50, se muestra los elementos de la capa de seguridad del dispositivo móvil, que tiene los mismos elementos que la capa de seguridad del servidor. Esto es, debido a que ambas capas de ambos dispositivos deben estar distribuidas de forma similar para la recepción/transmisión de datos.

- **Capa de comunicación**

La aplicación tiene un módulo para la comunicación con una base de datos remota (también hace uso de una local) para distintos aspectos tales como la inserción de nuevas enfermedades, de nuevas medidas y datos relacionados con el paciente, etc. Para tener acceso a ésta base de datos situada en un servidor remoto, se ha construido un servicio web en el que se implementa una serie de métodos para la lectura/escritura de los distintos datos que se procesarán.

La capa de comunicación del dispositivo móvil, tiene los mismos elementos que la capa de comunicación del servidor. Esto facilita la interoperabilidad entre las capas de ambos. La figura 5-44 muestra la estructura de estas capas.

Cabe destacar que tanto a la hora de la adición de una enfermedad para monitorizar (en el caso de que seleccionemos añadir una pre-configurada) como a la hora de comprobar si el usuario ya pertenece al sistema la aplicación hace uso de servicios web para conectar con una

base de datos alojada en la red o acceso a bases de datos locales. Para tal fin, se diseña el acceso en tres niveles:

- Al nivel superior pertenece la interfaz, la cual será la encargada de interactuar directamente con el usuario, mostrándole la distinta información procedente de la conexión con los servicios web.
- En el nivel intermedio se encuentran una serie de métodos para la conexión con los servicios web o la base de datos local. Tales métodos se encargan de realizar las peticiones con los datos recogidos en la interfaz y de enviar a la interfaz los datos provenientes de los distintos accesos.
- En el nivel inferior tenemos las clases que se encargan de tratar directamente con los accesos, abrir o cerrar las conexiones (realizando las tareas que eso conlleva) y de enviar las peticiones y tratar los datos para que sean legibles para la aplicación.

Invocación del Método	Método invocado en el Server	Especificación de la llamada
Recomendación.	**Método 1**	`public List<String> getDiabetesRecomendation()`
Perfil de Paciente.	**Método 2**	`public User getUser(String idPatient)` (En la aplicación el objeto Patient viene a ser denominado mediante la clase User)
Pacientes de un doctor.	**Método 3**	`public List<User> getPatientesOfDoctor(String idDoctor)`
Lista de enfermedades.	**Método 4**	`public List<Disease> getDiseases()`
Adición de enfermedad.	**Método 5**	`public void addDisease(Disease d)`
Adición de paciente.	**Método 6**	`public int addUser(User p)`
Adición de una medida.	**Método 7**	`public String addMeasure(Measure m)`
Adición de incidencia.	**Método 8**	`public void addControlIncidence(String idPatient, String incidencia, String tipo)`
Obtención de incidencias.	**Método 9**	`public List<Incidence> getIncidences(String IDpatient, String tipo)`
Obtención de rol.	**Método 10**	`public String getRol(String id)`
Asociación de un medico a un paciente.	**Método 11**	`public int addPatientToDoctor(String idPatient, String idDoctor)`
Obtención de todas las medidas de un paciente.	**Método 12**	`public List<Measure> getMeasures(String idPatient, String idDisease)`
Listar mensajes.	**Método 13**	`public List<Message> getLastMessages(java.util.Date date, String id)`
Enfermedades de un usuario.	**Método 14**	`public List<Disease> getDiseasesFromUser(String userId)`
Asociar una enfermedad a un paciente.	**Método 15**	`public void enlazarUserDisease(String userId, String diseaseId)`

Tabla 5-15.*Invocación de los métodos del servicio web definidos en el capa de comunicación del servidor.*

En la tabla 5-15 se muestra la relación entre cada método definido en la capa de comunicación del servidor y las invocaciones de cada uno de los métodos desde la capa de comunicación del dispositivo móvil.

Capas del Dispositivo Biométrico

Las capas del dispositivo biométrico no se han desarrollado físicamente como las otras capas previamente definidas. Esto se debe a que los dispositivos biométricos no nos ofrecen la posibilidad de modificar mediante código la forma en que ellos trasmiten datos a los dispositivos móviles. Lo que se ha desarrollado es una definición conceptual de la posible distribución de capas dentro de estos dispositivos.

Para ello se ha modificado la estructura de enlace, negociación y seguridad desde el dispositivo biométrico a través de la tecnología **Bluetooth**. Se distribuyen cada uno de los pasos que se llevan a cabo al momento de emparejar el dispositivo móvil con el dispositivo biométrico hasta obtener una medida puntual generada por el dispositivo biométrico.

Cada dispositivo cuenta con capacidad de comunicación, lo que nos permite recibir los datos que envía (siempre que cuente con un protocolo abierto fácil de adaptar en el *framework*). Para los dispositivos biométricos que hemos evaluado con tecnologías Bluetooth, se han identificado las siguientes capas:

- **Capa de enlace**

  Esta capa recibe la solicitud de emparejamiento desde el dispositivo móvil. Este emparejamiento abre la comunicación en el dispositivo móvil para su recepción.

  En la figura 5-51, se muestra la ubicación de la capa de enlace y el procedimiento que inicia el proceso de enlace o emparejamiento entre el dispositivo móvil y biométrico.

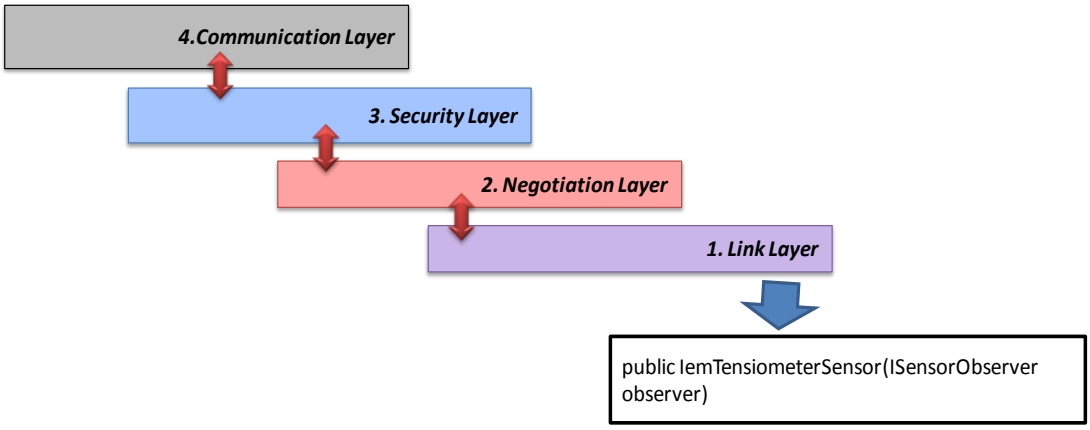


Figura 5-51. *Identificación de los elementos de la capa enlace del dispositivo biométrico.*

Con ese método del listado 5-16, se registra en un *hilo* el servidor **Bluetooth** con un *uuid* característico, de esa manera escuchará lo que envíen los dispositivos sincronizados desde la pantalla de Ajustes **Bluetooth** de Android.

Listado 5-16. *Definición de los elementos ubicados en la capa de enlace para el dispositivo biométrico.*

```
public IemTensiometerSensor(ISensorObserver observer) {
            // DUN
            this.uuid =  UUIDHelper.fromUUID16(0x1103);
            this.bluetoothSocketManager = new
IemTensiometerSocketManager();
            this.sensorThread = new DefaultServerSensorThread(uuid,
observer, bluetoothSocketManager); }
public DefaultServerSensorThread(UUID uuid, ISensorObserver observer,
            IBluetoothSocketManager bluetoothSocketManager) {
            this.observer = observer;
            this.uuid = uuid;
            this.bluetoothSocketManager = bluetoothSocketManager;
            this.listMeasurements = null;
// Use a temporary object that is later assigned to mmServerSocket,
// because mmServerSocket is final
            BluetoothServerSocket tmp = null;
           …
```

- **Capa de Negociación**

La capa de negociación se encarga de establecer los parámetros de comunicación entre los dispositivos biométricos y los dispositivos móviles.

En la figura 5-52, se muestra la ubicación de la capa de negociación y la manera en que se comunica con la demás capas.

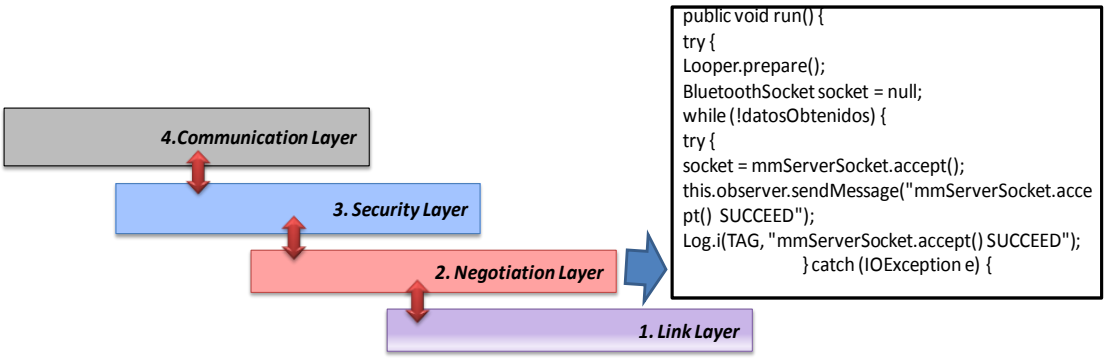


Figura 5-52. *Identificación de los elementos de la capa negociación del dispositivo biométrico.*

En el listado 5-17, se muestra parte de la codificación de la capa de negociación. En esta parte el servidor queda a la escucha y reacciona cuando se acepta un *socket* **Bluetooth** (canal por el que los dispositivos envían los datos). Este método está a la escucha constantemente (aunque descansa medio segundo entre comprobación y comprobación) a la escucha de conexiones con los dispositivos, si encuentra una, realiza la *negociación* y obtención de los datos que transmite.

Listado 5-17. *Definición de los elementos ubicados en la capa de negociación para el dispositivo biométrico.*

```
public void run() {
try {
Looper.prepare();
BluetoothSocket socket = null;
// Keep listening until exception occurs or a socket is returned
boolean datosObtenidos = false;
int totalBytes = 0;

while (!datosObtenidos) {
try {
socket = mmServerSocket.accept();
this.observer.sendMessage("mmServerSocket.accept()
SUCCEED");
Log.i(TAG, "mmServerSocket.accept() SUCCEED");
      } catch (IOException e) {
            …
```

- **Capa de Seguridad**

Cada dispositivo biométrico cuenta con tecnologías de seguridad de encriptación de los datos para que si son interceptados por otro dispositivo no sean entendibles. Con la utilización de *sockets* para la transmisión, no hace falta tener mecanismos de seguridad, ya que estos "*túneles*" hacen que el intercambio de datos sea únicamente entre los dispositivos sincronizados.

Los mecanismos de seguridad de esta capa están definidos por los que posee el dispositivo biométrico por lo cual no se define en este apartado.

- **Capa de Comunicación**

La tecnología de comunicación que posee cada dispositivo biométrico define la funcionalidad de esta capa. Hemos realizado pruebas con dispositivos biométricos con tecnología Bluetooth, para comunicarlos con los dispositivos móviles en donde se ha implementado la arquitectura.

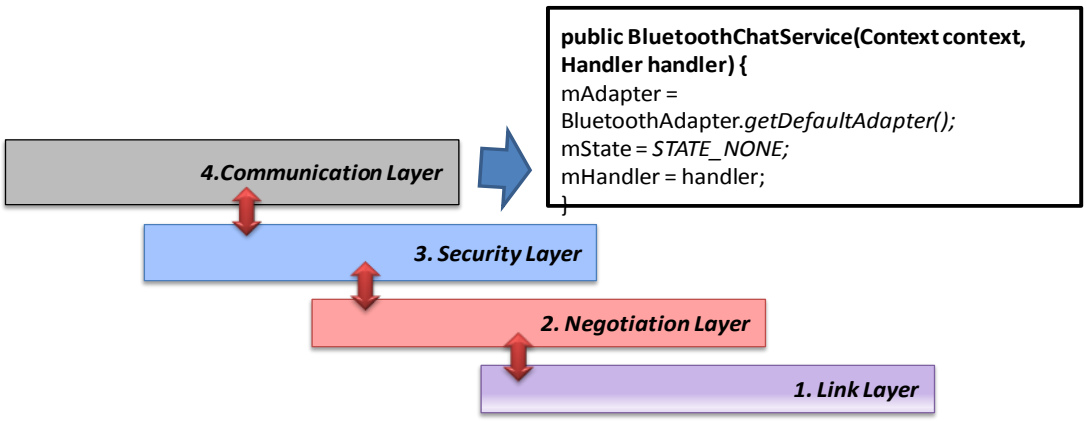


Figura 5-53. *Identificación de los elementos de la capa comunicación del dispositivo biométrico con conectividad Bluetooth.*

En la figura 5-53, se muestra la estructura y ubicación de la capa de comunicación del dispositivo biométrico. Se ha utilizado un dispositivo con conectividad Bluetooth para trasmitir datos al dispositivo móvil.

En la tabla 5-16, se muestra un resumen de los módulos implementados en este capítulo, así como los patrones utilizados, las ontologías relacionadas, las capas que intervienen y los requisitos funcionales cubierto. Esto facilita la comprensión global de la puesta en marcha del *framework* para desarrollar una arquitectura *software* determinada.

5.9 CONCLUSIONES

La implementación de los primeros prototipos de la arquitectura *software*, basada en el *framework* propuesto, ha sido desarrollada siguiente el diseño conceptual definido en capítulos anteriores, de tal manera que se ha podido validar la funcionalidad que ofrece desarrollar aplicaciones basadas en este *framework*.

Se han implementado cada una de las ontologías que por una parte ayudan a la definición del conocimiento necesario para generar la base de datos y por el otro, ofrecen información para la generación de módulos propios de la arquitectura.

También se han podido implementar cada uno de los patrones que hemos diseñado para la generación de las estructuras visuales y funcionales acordes a los requisitos planteados. Estos patrones han sido identificados y diseñados dentro de las diferentes zonas contenedoras, que permite el diseño de interfaces amigables.

La definición y distribución de los elementos del *framework* a través de capas, permiten la identificación y ubicación de los diferentes componentes, lo que facilita el mantenimiento y detección de fallos. Esta distribución ha sido desarrollada además para facilitar la migración entre plataformas para que los desarrolladores que no conozcan la arquitectura se familiaricen fácilmente.

No hemos entrado en detalles para una enfermedad en particular, sino que se definió de manera genérica cada uno de los componentes diseñados en el capítulo 4, para desarrollar los primeros prototipos que validen nuestra propuesta.

Además es muy importante mencionar, que para la implementación de esta primera aproximación, el *framework* desarrollado puede generar aplicaciones para enfermedades que tengan múltiples unidades de medidas. Para estos primeros prototipos se ha monitorizado enfermedades que tienen hasta dos unidades de medidas.

Módulo a implementar	Patrones a utilizar	Ontologías relacionadas	Capas que intervienen	Requisitos Funcionales (RF) y Casos de Uso (CU)
Ajustes Iniciales	• ReadjustmentAppMobiPattern • ProfileMobiPattern	• Disease • PatientProfile	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF02** (CU-03), **RF03** (CU-04, CU-05), **RF04** (CU-07), **RF08** (CU-16, CU-17, CU-18), **RF09** (CU-19, CU-21)
Desarrollo de Menú	• MenuMobiPattern	• Disease • PatientProfile	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF03** (CU-04)
Desarrollo del Perfil de Paciente	• ProfileMobiPattern	• Disease • PatientProfile	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF09** (CU-19, CU-20, CU-21)
Obtención de Medidas	• MeasureMobiPattern	• Disease • PatientProfile	• Del Servidor (todas) • Del Dispositivo Móvil (todas) • Del Dispositivo Biométrico (todas)	**RF01** (CU-01, CU-02) **RF09** (CU-19, CU-20, CU-21)
Definición de ejercicio físico	• MeasureMobiPattern	• Disease • PatientProfile • ModuleDefinition	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF06** (CU-10)
Módulos de Recomendación, Educación y Prevención	• BehaviourMobiPattern	• Disease • PatientProfile • ModuleDefinition	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF05** (CU-08) **RF06** (CU-09, CU-10, CU-11, CU-12)
Módulo de Autocontrol	• VisualizationMobiPattern	• Disease • PatientProfile	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF07** (CU-14, CU-15)
Funcionalidad de adaptación	• AdaptabilityMobiPattern	• PatientProfile • ModuleDefinition	• Del Dispositivo Móvil (todas)	RF se generan según las funcionalidades deseadas.
Funcionalidad de Acomodación	• AccommodationMobiPattern	• PatientProfile • ModuleDefinition	• Del Dispositivo Móvil (todas)	RF se generan según las funcionalidades deseadas.
Funcionalidad de Ejecución	• ExecutionAppMobiPattern	• Measure • ModuleDefinition • PatientProfile	• Del Dispositivo Móvil (todas)	RF se generan según las funcionalidades deseadas.
Módulos Médico	• DoctorMobiPattern	• Measure • ModuleDefinition • PatientProfile	• Del Servidor (todas) • Del Dispositivo Móvil (todas)	**RF09** (CU-19, CU-20, CU-21) **RF07** (CU-14, CU-15) **RF06** (CU-09, CU-10, CU-11, CU-12)

Tabla 5-16. *Resumen de los elementos del framework utilizados para el desarrollo de la arquitectura software.*

CAPÍTULO SEXTO

6 EVALUACIÓN: APLICANDO MÉTODOS Y CASOS DE ESTUDIO

El *framework* desarrollado tiene un carácter general, es decir puede ser especializado para diversas enfermedades que utilicen diversos dispositivos biométricos, pudiéndose adaptar funcionalidades en forma de módulos interconectados para obtener múltiples configuraciones. El diseño general de la arquitectura se ha adaptado de tal manera que pueda reutilizarse gran parte de ella en el momento en que se quieran realizar trabajos futuros basados en el diseño y desarrollo inicial. De esta manera se han desarrollado varias aplicaciones que se apoyan en la arquitectura y que sirven para evaluar su validez para la monitorización móvil de pacientes.

En este capítulo se describen los casos de estudio en los que se ha utilizado e implementado la arquitectura *software* basada en el *framework* para la generación de aplicaciones. Cada caso de estudio verifica la eficiencia y validez de todos los elementos inicialmente propuestos y desarrollados. Como hemos mencionado anteriormente, la arquitectura de monitorización móvil de pacientes permite la interacción entre diversos elementos que le dan mayor robustez y complejidad al proceso de monitorización, integrando diversos dispositivos para facilitar el seguimiento médico de un paciente. Todas estas características son independientes de una enfermedad en particular, lo que le da el sentido de generalización buscado.

Además de los casos de usos, también presentamos una evaluación a nivel de la arquitectura *software*, basándonos en algunos métodos reconocidos por la comunidad

científica y que facilitan la evaluación de los prototipos desarrollados. Cada método, considera un aspecto específico a evaluar, según los requerimientos finales de la aplicación. La evaluación de arquitecturas *software* es un campo abierto y en continua expansión, no existiendo en la actualidad métodos cuantitativos ni objetivos. Los métodos seleccionados, pese a contener un factor subjetivo, ha tenido una reseñable repercusión y aceptación en la comunidad científica y, por ésta razón, han sido seleccionados para evaluar esta arquitectura.

Podemos resumir entonces que este apartado consiste en una *evaluación por parte del usuario final* de la arquitectura, basándonos en la puesta en marcha de la aplicación móvil entre pacientes que necesitan monitorización y una *evaluación metodológica por parte del desarrollador* que hace uso del *framework,* basado en métodos de evaluación de arquitecturas *software*.

6.1 EVALUACIÓN BASADA EN MÉTODOS

Son muchos los aspectos que se pueden evaluar en una arquitectura *software*, no obstante, para nuestra tesis nos interesan los siguientes**: *evaluar la facilidad de modificación***, este aspecto es uno de los principales, ya que la arquitectura debe ir evolucionando con el tiempo para que se cumplan las metas y objetivos para la cual fue desarrollada; ***evaluar el desempeño***, al tratarse de una arquitectura basada en el seguimiento médico de pacientes, ésta debe ofrecer respuestas oportunas y su rendimiento debe ser lo más eficiente posible; y ***evaluar la facilidad de uso de la arquitectura***, otro de los aspectos a tomar en cuenta en el momento en que se inicio el desarrollo de la arquitectura es la facilidad de uso para el programador, esto es, que la arquitectura desarrollada podrá ser utilizada por pacientes de cualquier rango de edad, por lo tanto, debe ofrecer un entorno fácil de usar y fácil de entender.

Todos estos aspectos a evaluar han sido contemplados en tres de los métodos analizados en el capítulo 2, estos métodos son: **ALMA** (evalúa facilidad de modificación), **PASA** (evalúa el desempeño de una aplicación) y **SALUTA** (evalúa la facilidad de uso de la arquitectura final).

En los siguientes puntos vamos a evaluar la arquitectura *software* desarrollada, basándonos en estos métodos, lo que nos permite conocer un poco más al detalle algunos escenarios o situaciones en los cuales la arquitectura responderá según su diseño inicial.

6.1.1 Evaluando la facilidad de modificación (Método ALMA)

Como hemos mencionado en el capítulo 2, el método ***ALMA*** (Architecture-level Modifiability Analysis) permite evaluar la facilidad de modificabilidad de una arquitectura *software*.

Es muy importante evaluar este aspecto de calidad dentro de la arquitectura desarrollada, ya que esto nos permite ir agregando nuevos componentes y funcionalidades al desarrollo presentado en esta tesis. Una de las nuevas metas contempladas es seguir actualizando toda la arquitectura para darle mayor robustez y que cubra nuevos aspectos que vayan surgiendo con el transcurso del tiempo. Esto permite un desarrollo evolutivo que se adapte a los requerimientos futuros planteados en el desarrollo.

Aspectos iniciales

Para desarrollar la evaluación basada en este método seguiremos los cinco pasos que propone ***ALMA***, iniciando con definición de la meta de evaluación presentada, la descripción de la arquitectura *software* desarrollada, la obtención de los escenarios en donde se puede presentar cambios, la evaluación de cada uno de los escenarios ponderados y por último, la interpretación de esos resultados, en donde se identifica la capacidad que tiene la arquitectura de soportar modificaciones en el futuro.

Perfil del Programador: se ha aplicado la evaluación a un programador ajeno al *framework* pero con conocimientos sobre el *framework* y experiencia de 24 meses en el uso de Android y un mes en el uso del *framework.*

Aplicando el método

Como se ha mencionado previamente, el método ALMA, es un método orientado a metas, el cual contempla tres tipos de metas: *predecir el costo de mantenimiento*, *evaluar riesgos* y *seleccionar de un conjunto de arquitecturas para evaluarlas*.

Antes de aplicar el método se debe seleccionar la meta a evaluar. Para nuestra tesis, evaluaremos la primera meta, ya que es una meta propia para una sola arquitectura, mientras que las otras dos son utilizadas para comparar varias arquitecturas candidatas.

a. Definición de las Metas. Aspectos a evaluar por el método

En este paso se debe seleccionar una meta a evaluar. Seleccionando el primer tipo de meta que especifica el método tenemos:

Meta 1: Predicción del Costo de Mantenimiento: nos interesa estimar el esfuerzo que se requiere para modificar parte de la arquitectura ante posibles cambios en el futuro. Se responden preguntas como: *¿Qué tanto esfuerzo se requiere para cambiar una parte o toda la arquitectura en el futuro?*, *¿Qué módulos o componentes necesitan mayor esfuerzo?*, *¿Afectan estos cambios la funcionalidad de la arquitectura?*, *¿Soporta la arquitectura la posibilidad de cambios?*

Técnica de evaluación utilizada: se utilizan en este método los escenarios de cambio. Estos escenarios capturan eventos futuros en los que se requiere que el sistema sea adaptado. Los escenarios de cambio se identifican sobre los componentes de la arquitectura ante situaciones concretas.

b. Descripción de la arquitectura. Identificación de componentes susceptibles a modificación

Recomendamos que la arquitectura de *software* desarrollada se encargue de generar aplicaciones parametrizadas para dispositivo móvil que ***obtienen datos de señales vitales*** a partir de dispositivos biométricos con ***capacidad de comunicación***. Ofrece al usuario ***actividades de control médico*** basándose en un ***perfil*** definido para cada paciente. Además de la lectura automática desde estos dispositivos biométricos, la obtención de señales vitales puede realizar de manera manual, a través del teclado. Con esta funcionalidad definida se

representan los principales componentes que conforman la arquitectura en general. Estos componentes se pueden ver en la figura 6-1. Existen otros componentes que pueden seguir evaluándose, pero consideramos que los que hemos definido, afectan directamente a la capacidad de modificación con que cuenta nuestra arquitectura.

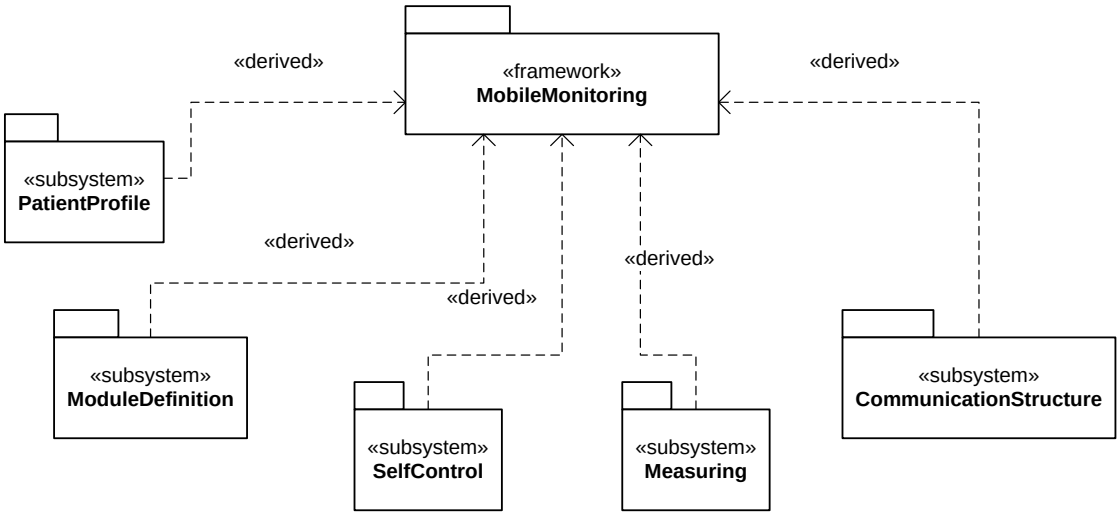


Figura 6-1. *Principales componentes susceptibles a cambios dentro de la arquitectura (Funcionalidad).*

Para continuar con la evaluación de la arquitectura según el criterio de modificabilidad, se identifican los componentes que son susceptibles a cambio.

En la tabla 6-1, se muestran clasificados todos los componentes que vamos a evaluar, la relación que tienen con otros componentes de la arquitectura, la relación con el entorno funcional de la arquitectura y el valor del componente dentro de la propia arquitectura. Este valor está definido sobre la prioridad o importancia que tiene el componente dentro de la funcionalidad de la arquitectura y esta ponderado en un rango de "1" a "5", en donde "1" es un valor bajo y "5" un valor alto. Los componentes con valores altos, son de gran importancia para que la arquitectura funcione, mientras que los componentes con valor bajo no afectan en gran medida sobre la funcionalidad de la arquitectura.

Componente	Relación/ Componentes	Valor del componente en la arquitectura
PatientProfile	Está relacionado con los componentes moduledefinition, selfcontrol y measuring.	3
ModuleDefinition	Está relacionado con los componentes patientprofile y measuring.	4
SelfControl	Está relacionado con los componentes patientprofile, measuring y moduledefinition.	2
Measuring	Está relacionado con los componentes patientprofile, moduledefinition, selfcontrol y communicationstructure	5
Communication Structure	Está relacionado con los componentes measuring y patientprofile.	1

Tabla 6-1. *Clasificación de los componentes funcionales y su relación con otros componentes.*

c. Obtención de los escenarios de cambio

Luego de identificados todos los componentes susceptibles a cambios en el futuro, se procede a clasificar cada uno de los posibles escenarios correspondientes a la meta 1. Cada grupo de escenarios se clasifican en categorías, para nuestra evaluación presentamos la categoría de los cambios funcionales, que es uno de los punto más importantes dentro de la evaluación. Otra categoría que se puede evaluar en el futuro es la categoría de adaptación a nuevas tecnologías, solo la mencionamos, pero no la evaluamos ya que su evaluación se puede realizar dentro de las actividades futuras de esta tesis.

Categoría 1: Cambios Funcionales: esta categoría agrupa todos los posibles escenarios funcionales en los que la arquitectura *software* desarrollada puede cambiar. Estos escenarios definen la relación interna entre módulos y la funcionalidad de cada uno de ellos. Se define un escenario para cada componente identificado en el punto anterior. Dentro de los escenarios de cambio que se encuentran en esta categoría tenemos:

Escenario 1: Se desea agregar dos nuevos campos al *perfil del paciente*, de forma tal que se cuente con nueva información de localización e identificación del paciente. Estos cambios afectan la lógica de la arquitectura y a la estructura de datos. Se desea realizar el cambio en 60 minutos, impactando el mínimo número de componentes funcionales y entidades relacionadas.

Escenario 2: Se desea agregar una nueva lista de diez *recomendaciones* para una enfermedad. Estos cambios afectan a la estructura de datos de la arquitectura. Se desea realiza el cambio en 30 minutos impactando el mínimo número de componentes funcionales.

Escenario 3: Se desea cambiar el formato de salida de la gráfica de *autocontrol*, por un formato en barra. Este cambio afecta todos los componentes que utilicen el módulo de autocontrol. Se desea realizar el cambio en 40 minutos impactando el mínimo número de componentes funcionales y sin afectar a la estructura de datos de la arquitectura.

Escenario 4: Se desea agregar una nueva enfermedad con dos unidades de medidas. Estos cambios afectan a los componentes de obtención de medidas, de configuración de enfermedad y la estructura de datos. Se desea realizar el cambio en 60 minutos impactando el mínimo número de componentes funcionales y sin afectar a la estructura de datos.

Escenario 5: Se desea agregar un nuevo dispositivo biométrico con capacidad de comunicación Bluetooth para la lectura de señales vitales de un paciente. Estos cambios afectan a los componentes de obtención de medidas. Se desea realizar el cambio en 30 minutos impactando el mínimo número de componentes funcionales y sin afectar a la estructura de datos.

Categoría 2: Adaptación a nuevas tecnologías: son componentes externos a la aplicación. Estos componentes permiten modificar la aplicación para ser ejecutada en otros entornos tecnológicos de *software* y *hardware*. El componente identificado en este apartado se define como ***adaptación a plataformas.*** Un ejemplo se muestra en la figura 6-2.

En esta evaluación no se contempla escenarios a corto plazo para esta categoría. La adaptación de la arquitectura a nuevas tecnologías necesita un análisis y cambios a más largo plazo, por lo cual no se puede medir en esta evaluación. Una evaluación posterior en detalle de estos aspectos puede arrojar resultados más exactos.

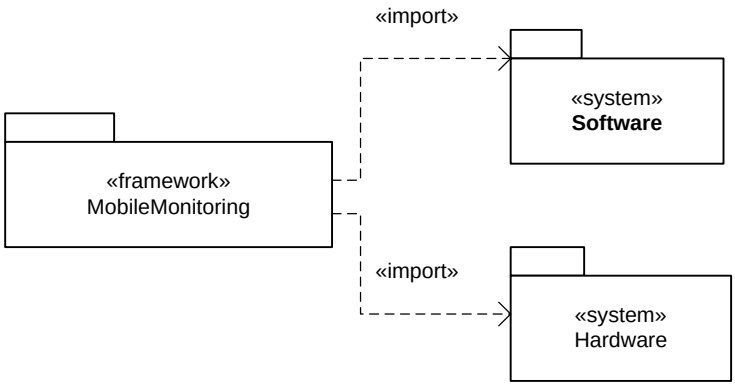


Figura 6-2. *Componente de la categoría de adaptación a nuevas tecnologías.*

d. Evaluación de los escenarios

Se procede a evaluar cada uno de los escenarios clasificados por categorías. En esta evaluación se realiza un análisis cualitativo y cuantitativo que permite identificar el grado final de modificabilidad que tiene cada uno de los componentes definidos previamente, sometiéndolos a los escenarios planteados.

Escenario 1: *Se desea agregar dos nuevos campos al perfil del paciente, de forma tal que se cuente con nueva información de localización e identificación del paciente. Estos cambios afectan la lógica de la arquitectura y a la estructura de datos. Se desea realizar el cambio en 60 minutos, impactando el mínimo número de componentes funcionales y entidades relacionadas.*

En la tabla 6-2 se muestran los elementos de la evaluación de cada escenario. Se evalúan las fuentes sobre los cuales se ha desarrollado el escenario, el estímulo o aspecto a desarrollar en el escenario, los artefactos que intervienen en el escenario, el ambiente en el cual se desarrolla en escenario de cambio y los resultados arrojados basado en la descripción y prueba del escenario.

Fuente	Código de la arquitectura
Estímulo	Agregar un cambio (insertar) dos campos al componente PatientProfile
Artefacto	Las entidades del sistema, el entorno de desarrollo (eclipse), el manejador de base de datos (MySQL) y la interfaz funcional.
Ambiente	Se tiene un ambiente de desarrollo en Eclipse con el proyecto creado y listo para ser modificado.
Clases afectadas	SQliteDB, SQliteAdapter, PatientProfileAC, PatientProfileTour, User.
Resultado	El escenario fue evaluado en 32 minutos, confirmándose la capacidad de modificación que presenta este componente dentro de la arquitectura.

Tabla 6-2. *Evaluación de escenario 1 basado en ALMA.*

Escenario 2: *Se desea agregar una nueva lista de diez recomendaciones para una enfermedad. Estos cambios afectan la estructura de datos de la arquitectura. Se desea realiza el cambio en 30 minutos impactando el mínimo número de componentes funcionales.*

En la tabla 6-3 se muestran los elementos de la evaluación de este escenario. Cada uno de los aspectos ha sido evaluado siguiendo los pasos del método ALMA, según las especificaciones de cada escenario.

Fuente	Código de la arquitectura
Estímulo	Agregar un cambio (insertar) diez campos al componente Measuring.
Artefacto	El manejador de base de datos (MySQL) y la interfaz funcional.
Ambiente	Se cuenta con la estructura de la base de datos creada y lista para ser modificado.
Clases afectadas	SQliteDBAdapter.
Resultado	El escenario fue evaluado en 18 minutos, confirmándose la capacidad de modificación que presenta este componente dentro de la arquitectura.

Tabla 6-3. *Evaluación de escenario 2 basado en ALMA.*

Escenario 3: *Se desea cambiar el formato de salida de la gráfica de autocontrol, por un formato en barra. Este cambio afecta todos los componentes que utilicen el módulo de autocontrol. Se desea realizar el cambio en 40 minutos impactando el mínimo número de componentes funcionales y sin afectar a la estructura de datos de la arquitectura.*

En la tabla 6-4 se muestran los elementos de la evaluación de este escenario. Cada uno de los aspectos ha sido evaluado siguiendo los pasos del método ALMA, según las especificaciones de cada escenario.

Fuente	Código de la arquitectura
Estímulo	Agregar un cambio (modificar) la estructura visual del componente SelfControl.
Artefacto	Las entidades del sistema, el entorno de desarrollo (eclipse), el manejador de base de datos (MySQL) y la interfaz funcional.
Ambiente	Se tiene un ambiente de desarrollo en Eclipse con el proyecto creado y listo para ser modificado.
Clases afectadas	MeasureControlActivity.
Resultado	El escenario fue evaluado en 23 minutos, confirmándose la capacidad de modificación que presenta este componente dentro de la arquitectura.

Tabla 6-4. *Evaluación de escenario 3 basado en ALMA.*

Escenario 4: *Se desea agregar una nueva enfermedad con dos unidades de medidas. Estos cambios afectan a los componentes de obtención de medidas, de configuración de enfermedad y la estructura de datos. Se desea realizar el cambio en 60 minutos impactando el mínimo número de componentes funcionales y sin afectar a la estructura de datos*.

En la tabla 6-5 se muestran los elementos de la evaluación de este escenario. Cada uno de los aspectos ha sido evaluado siguiendo los pasos del método ALMA, según las especificaciones de cada escenario.

Fuente	Código de la arquitectura
Estímulo	Agregar un cambio (insertar/modificar) a la estructura del componente measuring y a los elementos de la estructura de datos.
Artefacto	Las entidades del sistema, el entorno de desarrollo (eclipse), el manejador de base de datos (MySQL) y la interfaz funcional.
Ambiente	Se tiene un ambiente de desarrollo en Eclipse con el proyecto creado y listo para ser modificado.
Clases afectadas	No hay cambios en las clases.
Resultado	El escenario fue evaluado en 30 minutos, confirmándose la capacidad de modificación que presenta este componente dentro de la arquitectura.

Tabla 6-5. *Evaluación de escenario 4 basado en ALMA.*

Escenario 5: *Se desea agregar un nuevo dispositivo biométrico con capacidad de comunicación Bluetooth para la lectura de señales vitales de un paciente. Estos cambios afectan a los componentes de obtención de medidas. Se desea realizar el cambio en 30 minutos impactando el mínimo número de componentes funcionales y sin afectar a la estructura de datos*.

En la tabla 6-6 se muestran los elementos de la evaluación de este escenario. Cada uno de los aspectos ha sido evaluado siguiendo los pasos del método ALMA, según las especificaciones de cada escenario.

Fuente	Código de la arquitectura
Estímulo	Agregar un cambio (insertar) un nuevo dispositivo que captura de señales vitales.
Artefacto	La interfaz funcional debe permitir este cambio, sin tener que cambiar los componentes ni la estructura de datos.
Ambiente	Se tiene la aplicación en funcionamiento y lista para ejecutar las actividades de sincronización de dispositivos.
Clases afectas	No hay clases afectadas pese a requerir un tiempo ligeramente mayor para hacerlo.
Resultado	El escenario fue evaluado en 35 minutos, confirmándose la capacidad de modificación que presenta este componente dentro de la arquitectura.

Tabla 6-6. *Evaluación de escenario 5 basado en ALMA.*

e. Interpretación de resultados.

Una vez evaluado cada escenario por separado, se procede a análisis todos los resultados obtenidos en la evaluación. Estos resultados nos permiten conocer que tan modificable ha sido cada uno de los componentes estudiados en este apartado. Se realizará una interpretación de resultados basada en tiempo. Cabe señalar que existen otros tipos de interpretaciones que se pueden documentar luego de analizados los escenarios como son también número de líneas de código (LoC). Para este caso y como mencionamos previamente, haremos el análisis con base en el tiempo que llevó analizar cada uno de los escenarios y el número de líneas de código finales luego de los cambios.

En la tabla 6-7 se especifican cada uno de los escenarios, el tiempo estimado que se proponía para desarrollar un cambio y el tiempo real en el que se implementó el cambio.

Escenario	Tiempo estimado	Tiempo alcanzado	LoC iniciales	Loc finales	Porcentaje de esfuerzo	Componente evaluado
E1	60 minutos	32 minutos	~ 1861 LoC	~ 1966 LoC	53,3%	PatientProfile
E2	30 minutos	18 minutos	~ 776 LoC	~ 802 LoC	60,0%	ModuleDefinition
E3	40 minutos	23 minutos	~ 545 LoC	~ 562 LoC	57,5%	SelfControl
E4	60 minutos	30 minutos	~ 600 LoC	~ 600 LoC	50,0%	Measuring
E5	30 minutos	35 minutos	~ 400 LoC	~ 400 LoC	116,7%	Communication Structure

Tabla 6-7. *Evaluación de todos los escenarios según ALMA, con los tiempos estimados y alcanzados.*

Se especifica además el porcentaje de esfuerzo para realizar el cambio, con respecto al tiempo máximo estimado para realízalo. Tomando en cuenta que el tiempo estimado (**TE**) corresponde al cien por ciento del esfuerzo estimado (**PEst**), y el tiempo alcanzado (**TA**), corresponde al porcentaje de esfuerzo alcanzado (**PEsf**) calculamos dicho esfuerzo de la siguiente manera:

$$PEsf = (TA * PEst)/(TE) \qquad (1)$$

En el gráfico de la figura 6-3 se muestra la comparativa de tiempo alcanzado para cada uno de los escenarios evaluados en ALMA, con respecto al tiempo estimado propuesto inicialmente.

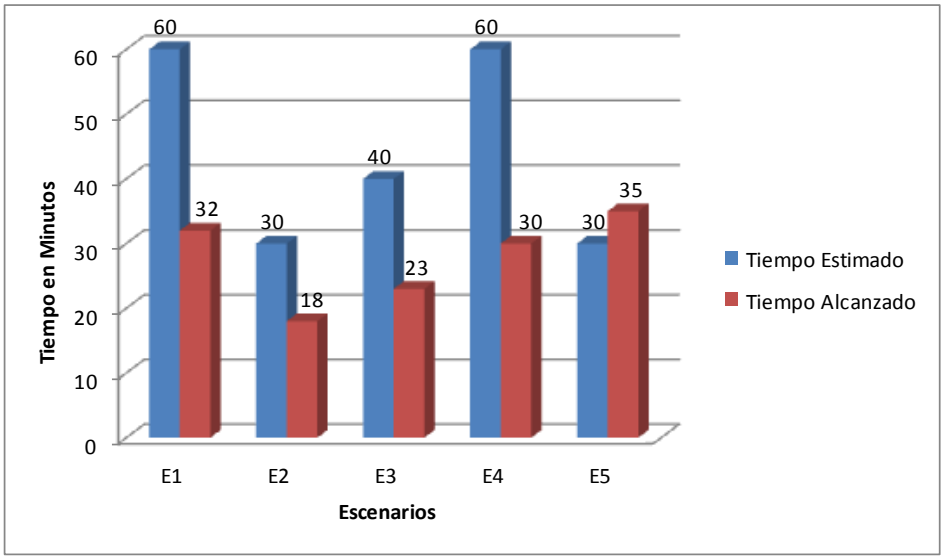


Figura 6-3. *Gráfico de evaluación de escenarios según el tiempo de duración del cambio (ALMA).*

Se puede notar que en los escenarios propuestos que para evaluar la modificabilidad no se ha requerido tanto tiempo como teníamos planteado inicialmente. En el escenario 2 los cambios que hay que hacerle a la aplicación final son mínimos, interviniendo lo menos posible en la estructura final desarrollada. Los escenarios que necesitaron mayor tiempo fueron el 1 y el 3 (perfil de paciente y gráfica de autocontrol respectivamente). El perfil debido a que es el módulo que se encarga de personalizar las características de la aplicación, requiere más detalle y atención a la hora de realizar cualquier cambio. El escenario 5 requirió mayor tiempo que el

propuesto, esto es debido a que los dispositivos biométricos tienen un protocolo propietario de la empresa que lo vende y hay que adaptarse a él.

En general, se puede decir, que la arquitectura cumple con los aspectos de análisis de modificabilidad propuesto por el método de evaluación que hemos utilizado, en este caso, el método ALMA. Además de evaluar el tiempo que tarda un componente de la aplicación en ser modificado, evaluamos también el número de líneas de código (LoC) que son afectadas y agredas para realizar el cambio. En la gráfica de la figura 6-4, se muestran los resultados de la evaluación con respecto al número de LoC iniciales antes de aplicar los escenarios y el número de LoC finales luego de aplicado el escenario.

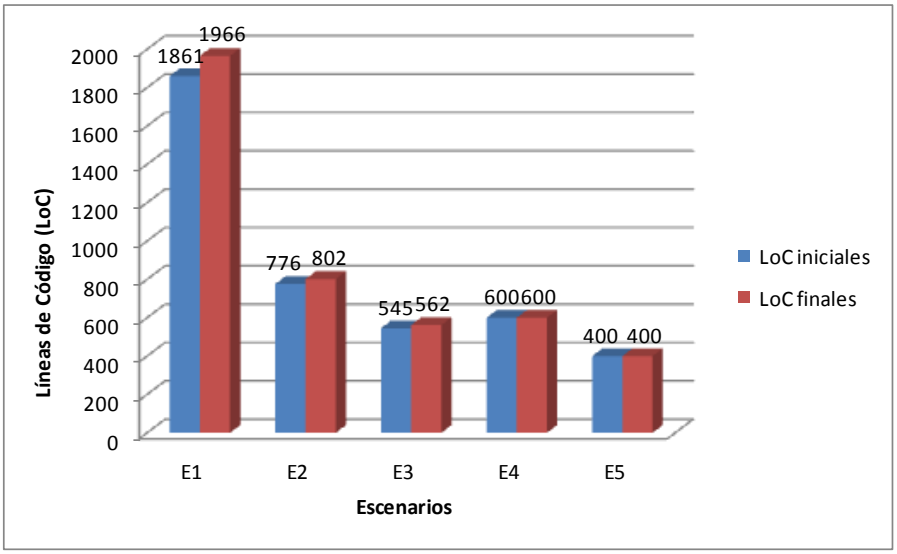


Figura 6-4. *Gráfico de evaluación de escenarios según LoC agregadas (ALMA).*

Podemos notar que en algunos escenarios como el 4 y 5, no se necesitaron realizar cambios a las líneas de código iniciales de la aplicación. Esto debido a que son módulos con más generalidad que pueden funcionar ante cualquier escenario sin tener que modificar la estructura interna de la aplicación. Los demás escenarios presentaron cambios no tan significativos, lo que representa la característica de reutilización de código, mencionada en el capítulo de implementación

Conclusiones

El uso de el método ALMA, nos ha permitido conocer el esfuerzo (valorado en tiempo y líneas de código) que hay que hacer para modificar algunos componentes o elementos de la arquitectura desarrollada. Cada esfuerzo que se tenga que hacer, depende de la complejidad de los cambios a desarrollar, pero en general, la arquitectura soporta cambios o modificaciones que no suponen una transformación completa del diseño inicial. Es cierto, que algunos otros componentes supondrán más cambios y mucho más esfuerzo, pero la generalización de la arquitectura y la característica de reutilización de sus componentes, facilitarán esos cambios. Podemos concluir entonces que la arquitectura soporta cambios que pueden hacerse sin afectar en gran medida la estructura funcional inicialmente desarrollada.

6.1.2 Evaluando el desempeño de la arquitectura (Método PASA)

Otro de los aspectos que se puede evaluar en nuestra arquitectura, es el desempeño funcional. El desempeño nos permite evaluar que tan rápido responde la arquitectura en el momento en que está en ejecución. Para ello nos basaremos en el método PASA (*Performance Assessment of Software Architecture*), para evaluar el desempeño de la arquitectura.

El método pasa está compuesto por nueve pasos, cada uno de ellos explicados en el capítulo 2, iniciando con la presentación del método, presentación de la arquitectura, identificación de los casos de uso críticos, selección de escenarios de desempeño, identificación de los objetivos de desempeño, clarificación de la arquitectura, análisis de la arquitectura, identificación de las alternativas y la presentación de los resultados.

Aspectos iniciales

Los aspectos iniciales a tener en cuenta antes de aplicar el método, permiten al evaluador contar con las herramientas previas a la puesta en marcha de la evaluación.

Casos de uso: puede contener uno o varios escenarios, estos describen la secuencia de acciones requeridas para ejecutar el caso de uso. Se deben especificar los casos de uso en *UML*, como diagramas de secuencia.

Técnica de evaluación utilizada: se utiliza en este método los escenarios. Se identifica los escenarios que son importantes para cada caso de uso. Los escenarios pueden ser aquellos que se ejecutan frecuentemente así como aquellos que son críticos desde la percepción de desempeño del usuario.

Aplicando el método

Los pasos para el proceso de evaluación en donde se analiza el aspecto de desempeño basándose en el método de evaluación PASA son:

a. Presentar el método de evaluación. Conociendo la estructura del método

Para iniciar el proceso de evaluación con el método PASA, se debe identificar algunos aspectos como son: *objetivo de la evaluación*, *en qué consiste el método*, *qué información de la arquitectura* es necesaria para efectuar la evaluación y los *resultados esperados*.

Objetivo de la evaluación: Evaluar el grado de desempeño de toda la arquitectura, según aspectos funcionales, basándose en parámetros en ejecución de la arquitectura.

En qué consiste PASA: este método permite evaluar el desempeño de una arquitectura *software*, mediante la técnica de escenarios. Se presentan algunos casos de uso crítico, evaluando sobre cada uno de ellos los escenarios previamente definidos. Es un método que consta de nueve pasos, desde el planteamiento del método hasta la presentación y resultado de las evaluaciones.

Funcionalidad de la arquitectura: como se ha mencionado en otros apartados, la arquitectura de *software* desarrollada permite la generación de aplicaciones móviles que facilitan el seguimiento y control de pacientes basado en el *framework* desarrollado. Este control y seguimiento se hace a través del dispositivo móvil con que cuenta el paciente, obteniendo datos desde los dispositivos biométricos.

Resultados esperados: se espera obtener el grado de desempeño de cada uno de los elementos importantes dentro de la arquitectura. Al tratarse de una arquitectura para entornos médicos y sanitarios, el tiempo de respuesta desde que el paciente interactúa con el dispositivo móvil hasta que obtiene una respuesta oportuna es de vital importancia para evaluar el desempeño inicialmente establecido en el diseño de la arquitectura.

b. Presentar la arquitectura. Funcionalidad

La funcionalidad principal y general de la arquitectura desarrollada es *generar aplicaciones parametrizadas para dispositivo móvil que obtienen datos de señales vitales a partir de dispositivos biométricos con capacidad de comunicación. Ofrece al usuario actividades de control médico basándose en un perfil definido para cada paciente, Además de la lectura automática desde estos dispositivos biométricos, la obtención de señales vitales puede realizar de manera manual, a través del teclado.*

c. Identificar casos de uso críticos

En este punto se seleccionan cada uno de los posibles casos de uso que son de gran importancia para la operación de la arquitectura. Estos casos de uso serán evaluados con los posibles escenarios de desempeño que se mencionarán en el siguiente punto a través de diagramas de secuencia que lo definen. Cabe destacar que se ha optado por considerar el dispositivo móvil como un actor externo al sistema para una mayor comprensibilidad.

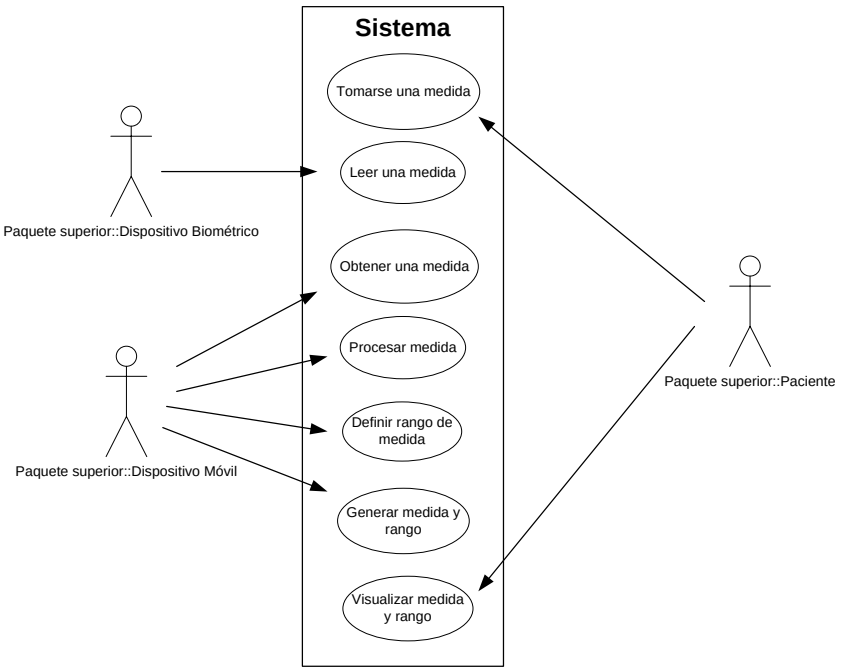


Figura 6-5. *Caso de uso 1 de PASA (obtención de una señal vital - medida).*

Caso de Uso 1: *Obtención de una señal vital (medida) desde dispositivo biométrico en el menor tiempo posible*. El tiempo que tarda la obtención de una medida para luego ser clasificada en un rango, es de vital importancia para el paciente, que espera obtener un resultado sin tener que esperar tanto tiempo. En la figura 6-5 se muestra el caso de uso 1 resumido.

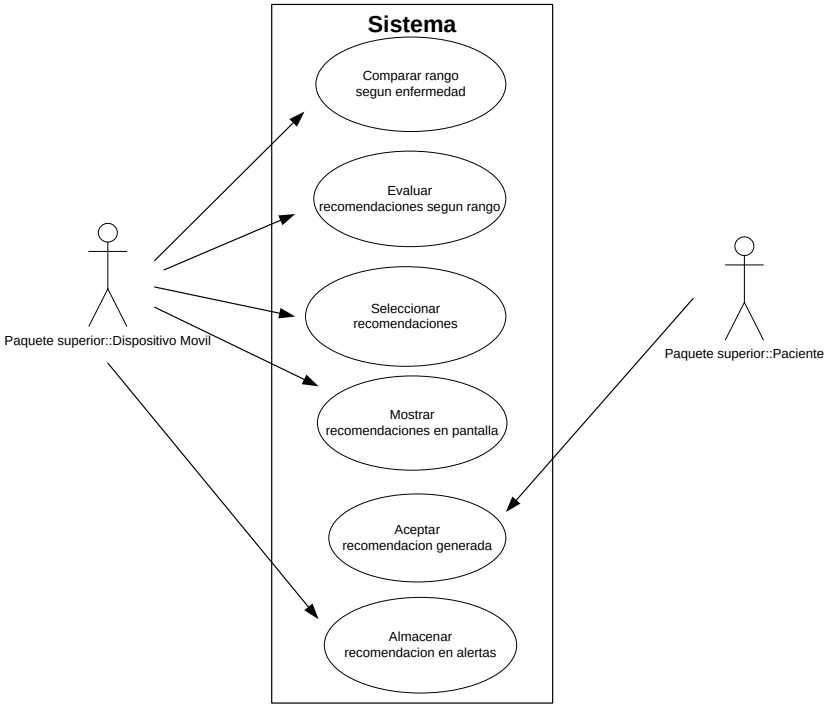


Figura 6-6. *Caso de uso 2 de PASA (generación de una recomendación).*

Caso de Uso 2: Generación de una recomendación a partir de una medida recibida. Nos interesa también conocer el desempeño de la arquitectura en el momento que genera una recomendación basada en una medida obtenida y la manera que se le da seguimiento posteriormente a través de los recordatorios que tiene programada la aplicación final. En la figura 6-6 se muestra el caso de uso 2 resumido.

Evaluaremos estos dos casos de uso según los escenarios presentados ya que son a nuestro criterio las dos situaciones que requieren de respuesta oportunas en el menor tiempo posible. Se pueden seguir analizando nuevos casos de uso para otras situaciones que requieren mayor detalle de explicación. Estas evaluaciones podrán hacer en el futuro.

d. Seleccionar escenarios de desempeño principales

Para cada caso de uso se define uno o más escenarios, en nuestro caso seleccionaremos un escenario para cada caso de uso. En este paso del método sólo se especifican los diagramas de secuencia de cada escenario basándose en los casos de usos planteados en el punto anterior. En los siguientes pasos se analizarán al detalle estas secuencias.

Escenario 1. *Obtención de una medida desde un dispositivo biométrico para el caso de uso 1*. En la figura 6-7 se muestra el diagrama de secuencia que define el escenario 1 del caso de uso 1. Los elementos que intervienen son: *Patient*, *BiometricDevices*, *MobileDevice*, *SQliteDBAdapter*, *DataUtils*, *CountingFragment*, *WebServiceConexionManager*.

Se definen las secuencias de ejecución desde que el usuario se toma una medida de señales vitales, hasta que el paciente ve en su dispositivo móvil el valor de la medida y la asignación de esta medida en su rango (baja, normal, alta). El resultado de la toma de medida se visualiza en la interfaz de medida que se definió en la implementación del capítulo anterior.

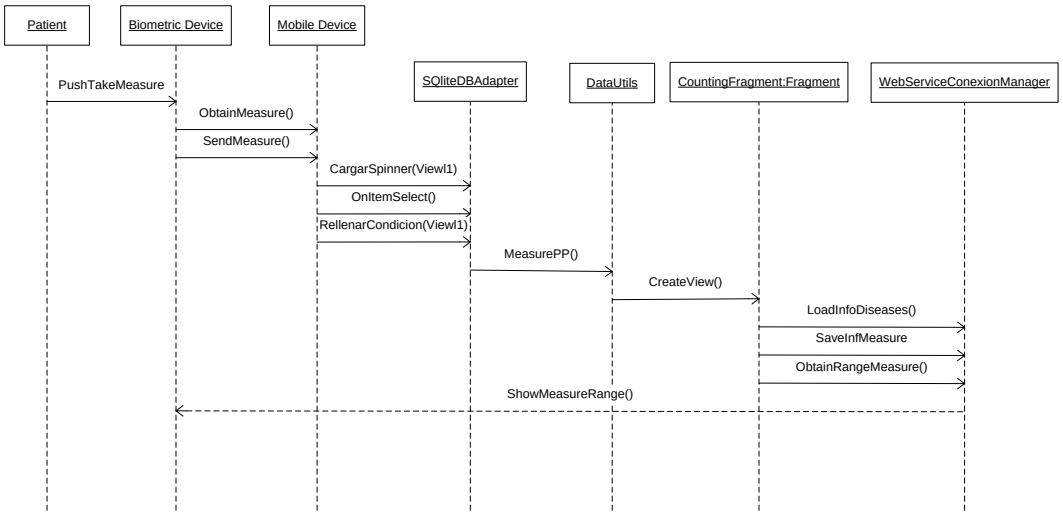


Figura 6-7. *Diagrama de Secuencia para el Caso de uso 1 (obtención de una medida)*

Escenario 2. Generación de una recomendación a partir de una medida recibida para el caso de uso 2. En la figura 6-8 se muestra el diagrama de secuencia que define el escenario 2 del caso de uso 2. Los actores que intervienen son: el dispositivo biométrico, el dispositivo móvil y el paciente.

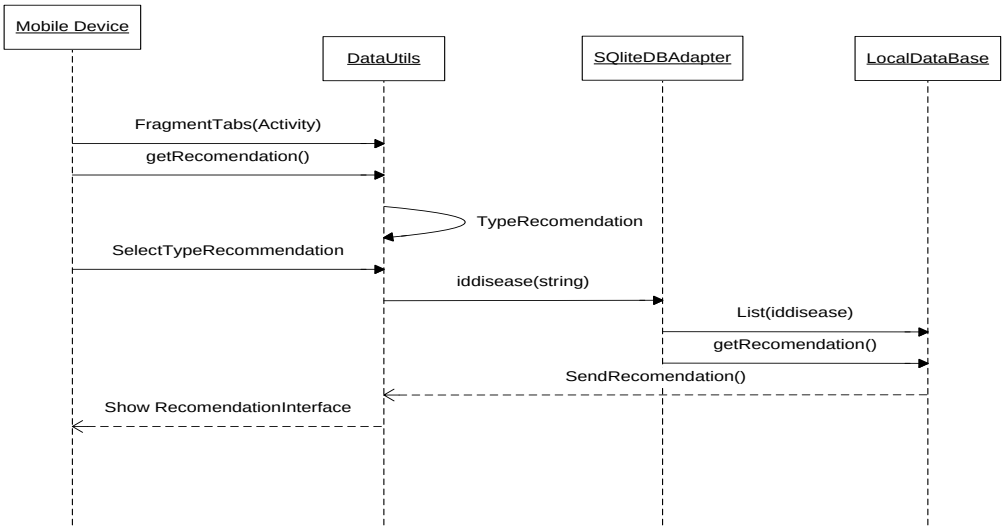


Figura 6-8. *Diagrama de Secuencia para el Caso de uso 2 (obtención de una recomendación basad en la medida o en el tipo de ejercicio a desarrollar)*

e. Identificar objetivos de desempeño

Para cada caso de uso se ha definido un objetivo de desempeño. El objetivo evaluado en este punto es el ***tiempo de respuesta*** que resulta desde que se pone en ejecución el escenario propuesto, sobre la arquitectura desarrollada.

Tiempo de respuesta = tiempo que tarda la aplicación en completar el escenario propuesto

Evaluaremos el tiempo de respuesta en segundos (seg.), realizando 5 iteraciones para un mismo escenario, lo que nos permitirá evaluar el comportamiento de la arquitectura ante varias solicitudes bajo el mismo escenario.

Para ambos escenarios se evaluará el mismo objetivo de desempeño, ya que en esta evaluación nos interesa conocer el tiempo que tarda la aplicación en ofrecer a los pacientes los servicios para la que fue desarrollada.

f. Clarificar la arquitectura y discutirla

En este punto, se ha revisado toda la arquitectura y analizado cada uno de los requerimientos funcionales así como los componentes que la forman. Esto nos permite seleccionar nuevas áreas críticas que no se hayan contemplado en los primeros pasos del método, permitiendo así corregir algunos fallos. Para nuestro caso, y según los escenarios que hemos propuesto, nos basaremos en todas las definiciones que previamente hemos presentado, según los casos de uso y escenarios de evaluación de la arquitectura en general.

Concluimos que el aspecto que nos interesa más evaluar es el tiempo que le toma a la aplicación responder ante una solicitud de medida y obtención de recomendaciones para un paciente. Como mencionamos previamente, estos son los dos puntos más críticos de la arquitectura, ya que necesita la intervención de múltiples dispositivos para obtener la información que permitirá a la aplicación ejecutar todos los demás servicios que ofrecerá al paciente.

Si la lectura de una señal vital tarda mucho en ser obtenida o falla antes de ser capturada por el dispositivo móvil, inhabilitará a la aplicación para ofrecerle al paciente un seguimiento médico constante.

g. Analizar la arquitectura. Definición de anti-patrones

Tras definir cada uno de los escenarios y caso de uso a evaluar, se han seleccionado algunos anti-patrones que afectarán el desempeño de la aplicación. Estos anti-patrones documentarán los problemas que afectarán el desempeño en algún momento determinado. La presencia de estos anti-patrones presenta significantes límites para la escalabilidad. Los anti-patrones de desempeño basados en los casos de uso planteados son:

- *Latencia de conectividad entre dispositivos*: uno de los problemas que puede afectar el rendimiento de desempeño de la aplicación es la latencia de conectividad entre dispositivos. Cuando hablamos de dispositivos hacemos referencia a la comunicación

que se establece entre los dispositivos biométricos y móviles para la obtención de las medidas.

- *Latencia en el almacenamiento de datos*: otro de los problemas que pueden afectar el rendimiento de desempeño es latencia en el proceso de almacenamiento de los datos obtenidos de la medición y las respuestas de las recomendaciones. Todas estas solicitudes necesitan acceder a base de datos, ya sea para leer o escribir. El cambio al acceder entre una base de datos y otra ocasiona una disminución del desempeño que tenemos que minimizar para seguir ofreciéndole al paciente una aplicación segura y que este siempre disponible.

- *Ejecución de otros módulos secundarios*: otros de los aspectos que puede afectar el rendimiento de desempeño de la arquitectura es la ejecución de múltiples módulos en segundo plano. Lo que consume más recurso del dispositivo móvil y ralentiza el procesamiento de otros elementos.

h. Identificar alternativas. Problemas de desempeño, alternativas y soluciones

Luego de seleccionado e identificado los posibles problemas de desempeño, ofrecemos las soluciones que proponemos si se llegase a presentar un problema de desempeño, basado en los anti-patrones enunciados.

- *Latencia de conectividad entre dispositivos*: este problema se agrava cuando se usa más de un dispositivo biométrico, para lo cual se necesitaría estar enlazándolos continuamente. Para ello se recomienda que el paciente use un único dispositivo biométrico para una enfermedad específica.

- *Latencia en el almacenamiento de datos*: para evitar que se presente este problema, se han definido dos ubicaciones de base de datos (una local en el dispositivo móvil y otra en red). La arquitectura se conectará a la base de datos en red siempre que haya conectividad y en caso que no exista conectividad la arquitectura puede seguir trabajando con la base de datos local que tenga. Las tablas contenidas en ambas base de datos se sincronizan una vez se restablezca la conexión. La base de datos en el dispositivo móvil sólo contendrá datos referentes al paciente de dicho dispositivo móvil

  El cambio al acceder entre una base de datos y otra ocasiona una disminución del desempeño que tenemos que minimizar para seguir ofreciéndole al paciente una aplicación segura y que esté siempre disponible.

- *Ejecución de otros módulos secundarios*: para evitar que se presente este problema, como se ha mencionado en capítulos anteriores, se ha distribuido todo los módulos de la aplicación en tres niveles, los cuales ubican en cada uno de ellos los módulos que se ejecutarán. En un nivel superior estará el módulo de lectura de las medidas, mientras que un módulo intermedio, se encuentran todos los demás módulos de autocontrol, que se ejecutan solamente cuando el módulo de medida invoca un servicio. Cuando finaliza el servicio, el módulo se desconecta evitando así consumir recursos del dispositivo.

i. Presentar resultados

Luego de identificados todos los elementos necesarios para realizar la evaluación, se procederá a validar cada uno de los escenarios planteados en cada caso de uso. El objetivo a cumplir es **evaluar el tiempo de respuesta** que resulta desde que se pone en ejecución el escenario propuesto sobre la arquitectura desarrollada hasta que se obtiene un resultado.

Basándonos en la premisa de que: *tiempo de respuesta = tiempo que tarda la aplicación en completar el escenario propuesto.* Hemos realizado cinco iteraciones para cada caso de estudio, es decir, cinco tomas de medidas y cinco generaciones de recomendaciones, con el mismo paciente, el mismo dispositivo móvil y el mismo dispositivo biométrico.

Los resultados se muestran en la tabla 6-8, en donde se evalúan los escenarios propuestos, y el tiempo de respuesta resultante para cada iteración, obteniéndose un promedio de tiempo de respuesta final para cada escenario.

	Iteraciones realizadas (en segundos)					**Promedio**
	I-1	**I-2**	**I-3**	**I-4**	**I-5**	
Escenario 1	7 segundos	6 segundos	7 segundos	7 segundos	6 segundos	6,6 seg.
Escenario 2	2 segundos	2 segundos	3 segundos	2 segundos	2 segundos	2,2 seg.

Tabla 6-8. *Evaluación de todos los escenarios basado en el desempeño.*

En la gráfica de la figura 6-9, se muestra los escenarios y las iteraciones resultantes de cada evaluación.

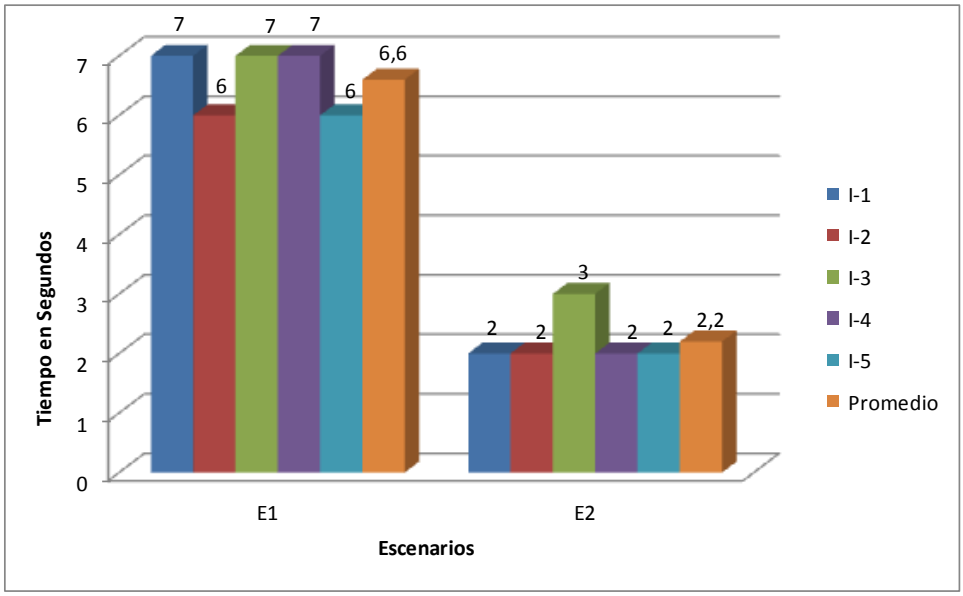


Figura 6-9. *Gráfico del resumen de las iteraciones según el grado de desempeño*

Puede notarse en el gráfico que los tiempos que tarda algunos componentes de la aplicación en ofrecer resultados son muy pequeños. Hemos evaluados los dos aspectos que consideramos más importantes como son la obtención de una medidas a través del dispositivo biométrico y la generación de una recomendación para esa medida.

Los tiempos de obtención de la medida han sido tomados desde el instante en que el dispositivo biométrico (tensiómetro para esta evaluación) muestra en su pantalla la medida y se la envía al dispositivo móvil. No se contempló el tiempo que el paciente inició la obtención de la medida poniéndose el tensiómetro en el brazo ya que el rendimiento de la aplicación se puede evaluar desde que el dispositivo móvil recibe los datos del dispositivo biométrico.

Conclusiones

Podemos concluir que el desempeño de la aplicación es alto, esto es debido a que se han identificado los problemas que en algún momento se podrían presentar y se han propuesto y desarrollado las diferentes soluciones. No obstante, y cuando la aplicación cuente con mayor tiempo en ejecución, podría presentarse otras problemáticas, lo que nos permitirá identificarlas y corregirlas inmediatamente.

Cabe señalar que el grado de desempeño de la aplicación puede también verse afectado por las características técnicas del dispositivo móvil en el cual se esté ejecutando. Aunque se ha desarrollado una aplicación que consume la menor cantidad de recursos de los dispositivos móviles, pueden presentarse problemas de desempeño que en otros dispositivos no se presentan.

6.1.3 Evaluando la facilidad de uso de la arquitectura (Método SALUTA)

Otro de los aspectos que se puede evaluar en nuestra arquitectura es la facilidad de uso. La facilidad de uso es la sencillez con la cual el usuario puede aprender a operar, preparar entradas e interpretar las salidas de la arquitectura. Ya hemos evaluado en otros métodos los criterios de modificabilidad y desempeño, pero además de estos dos aspectos, pretendemos también evaluar la facilidad de uso con este método.

Aspectos iniciales

Para evaluar la facilidad de uso, nos basaremos en el método SALUTA (Scenario based Architecture Level UsabiliTy Analysis), el cual está compuesto de cuatro pasos: *crear perfiles de uso*, *describir la facilidad de uso proporcionada*, *evaluar los escenarios* e *interpretar los resultados*.

Técnica de evaluación utilizada: se utiliza en este método los escenarios de uso. Estos escenarios se agrupan en uno o varios perfiles de uso. El perfil de uso representa la facilidad de uso requerida por la arquitectura desarrollada.

Tipo de resultado a esperar: se desea evaluar el grado de usabilidad de la arquitectura desarrollada, basado en el marco de trabajo de usabilidad propuesto por el método.

Aplicando el método

Como se ha mencionado previamente, este método consta de cuatro pasos importantes, los cuales se explican a continuación:

a. Crear perfiles de uso

En este primer punto se identifican todos los aspectos iniciales que debe tenerse en cuenta antes de realizar la evaluación, como son: *usuarios*, *tareas del sistema*, *contexto de desarrollo de la aplicación* y *atributos a evaluar*.

Usuarios

Para definir los usuarios que realizarán la evaluación de la arquitectura, el método SALUTA, especifica que estos deben organizarse en categorías, lo que permite una clara determinación de los aspectos o atributos que se evaluarán según el tipo de usuario al que va dirigido. Las categorías y usuario identificados para la evaluación son:

Categoría 1: Desarrollador de la arquitectura (P)

Para evaluar de una manera más clara la arquitectura, hemos definido la categoría de desarrollador de la arquitectura. Es muy importante que, como desarrolladores, podamos identificar funcionalmente la facilidad con que la arquitectura desarrollada se ajusta a las exigencias de usabilidad por parte de los usuarios.

Categoría 2: Usuarios finales (UF)

Además de la evaluación realizada por el propio diseñador o programador de la arquitectura, nos interesa evaluar la funcionalidad de uso por parte de los usuarios finales. El usuario final es el que utilizará con mayor frecuencia la arquitectura para la monitorización de señales vitales, es por ello que es muy importante conocer su opinión en el momento en que la arquitectura está en funcionamiento. Este aspecto también puede ser evaluado implementado el prototipado, en casos de estudios específicos, lo que nos dará mayor número de atributos a considerar. En este método evaluaremos los atributos más directos asociados a la facilidad de uso interpretada por un paciente.

Tareas más significativas de la arquitectura

Luego de identificar los usuarios que intervienen en el proceso de evaluación, se procede a enumerar las tareas más significativas que pueden desarrollar estos usuarios. Cada una de estas tareas puede ser evaluada por cada una de las categorías de usuario presentados.

Tarea 1 (T1): Editar un perfil de paciente previamente agregado.

Tarea 2 (T2): Visualizar los gráficos de autocontrol de una enfermedad en particular.

Tarea 3 (T3): Seleccionar un ejercicio a desarrollar por un paciente.

Tarea 4 (T4): Modificar el idioma de la aplicación.

Tarea 5 (T5): Agregar una nueva medida de una enfermedad

Tarea 6 (T6): Revisar las notificaciones y alarmas generadas después de obtenida una medida.

Estas son algunas de las tareas que un usuario puede realizar con la arquitectura desarrollada estas tareas se evaluarán como hemos dicho anteriormente, desde dos tipos de usuarios, los *programadores o diseñador de la arquitectura* (**P**) y por el *usuario final* (**UF**). Esto nos permite evaluar la percepción que tienen cada uno de ellos del grado de facilidad de uso con que cuenta la arquitectura.

Contexto de la arquitectura

Luego de definidas las principales tareas que deseamos evaluar, se define el contexto donde se evaluará la arquitectura. Se recomienda evaluar no más de dos contexto, para nuestro caso evaluaremos los siguientes:

Contexto 1 (C1): Un ambiente de entrenamiento para pacientes con diabetes que necesitan monitorizar su enfermedad, basándose en las tareas establecidas. En este contexto los usuarios ingresaran por teclado el valor de la medición obtenida desde el dispositivo biométrico.

Contexto 2 (C2): Un ambiente de entrenamiento para pacientes con problema de tensión arterial que necesitan monitorizar su enfermedad, basándose en las tareas establecidas. En este contexto la medida se obtendrá desde el dispositivo biométrico hacia el dispositivo móvil, a través de conectividad Bluetooth.

Atributos a evaluar

Este método permite evaluar algunos atributos que afectan al grado de usabilidad de la aplicación. Para ello (Folmer and Bosch, 2003), propusieron un *framework* que definía un conjunto de atributos que pueden ser utilizados a la hora de usar el método **SALUTA**. Para nuestra evaluación nos basaremos en los atributos que ellos proponen para evaluar cada tarea de cada contexto planteado.

En este punto del método, se debe asignar un valor numérico para cada uno de los atributos a evaluar. Se recomienda establecer un rango de 1 a 5 (5 como mayor grado de usabilidad y 1 como menor grado de usabilidad), para cada atributo según la tarea y el contexto en el que se esté evaluando.

Los atributos evaluados son: la *facilidad de aprendizaje* que ofrece la aplicación hacia los usuarios, la *eficiencia de uso*, la *confiabilidad* y la *satisfacción*. Estos atributos serán evaluados en el tercer paso del método.

Selección de escenarios con respecto a los atributos

El conjunto de elementos que se definen inicialmente, componen los escenarios de uso que serán evaluados por parte de los usuarios, asignándole la lista de atributos a cada uno de los usuarios, según las tareas que tengan que desarrollar.

b. Describir la facilidad de uso proporcionada

Cada uno de los aspectos mencionados en los pasos anteriores crean los escenarios de uso que serán evaluados, cada uno de estos escenario de uso evalúan las propiedades de la

arquitectura por parte de los usuario, según las tareas y contextos en los que se presentan. Estos escenarios y casos de uso especifican la facilidad de uso que proporciona la aplicación en tiempo de ejecución.

c. Evaluar escenarios

En la tabla 6-9, se muestran los atributos que evaluaremos con este método estos atributos como mencionamos previamente, han sido propuestos por SALUTA, como un marco de referencia para estandarizar el proceso de evaluación del método. Para cada usuario, se evalúan las tareas propuestas en los contextos definidos.

Usuario	Tareas	Contextos	Facilidad de Aprendizaje	Eficiencia de uso	Confiabilidad	Satisfacción
D	T1	C1				
D	T2	C1				
D	T3	C1				
…	…	…				
UF	T1	C2				
UF	T2	C2				
…	…	…				

Tabla 6-9. *Atributos de usabilidad a evaluar para cada tarea y contexto definido en SALUTA.*

En el anexo 5, se muestra el contenido del documento utilizado en la evaluación para un usuario en particular.

d. *Interpretar resultados*

Luego de aplicada la evaluación de cada una de las tareas a todos los usuarios involucrados, se obtienen las principales valoraciones por parte de cada uno de ellos en la tabla 6-10.

Usuario	Tareas	Contextos	Facilidad de Aprendizaje	Eficiencia de uso	Confiabilidad	Satisfacción
D	T1	C1	4	4	4	4
D	T2	C1	4	4	3	4
D	T3	C1	4	4	4	4
D	T4	C1	5	4	5	5
D	T5	C1	4	4	4	4
D	T6	C1	4	4	5	5
UF	T1	C2	4	5	4	4
UF	T2	C2	4	5	4	4
UF	T3	C2	4	4	5	5
UF	T4	C2	4	4	4	4
UF	T5	C2	4	5	5	4
UF	T6	C2	5	5	4	5

Tabla 6-10. *Resumen de las valoraciones de facilidad de uso por parte de los usuarios.*

La gráfica de la figura 6-10, muestra el grado de facilidad de uso de la arquitectura en general, para cada uno de las tareas analizadas según el contexto 1.

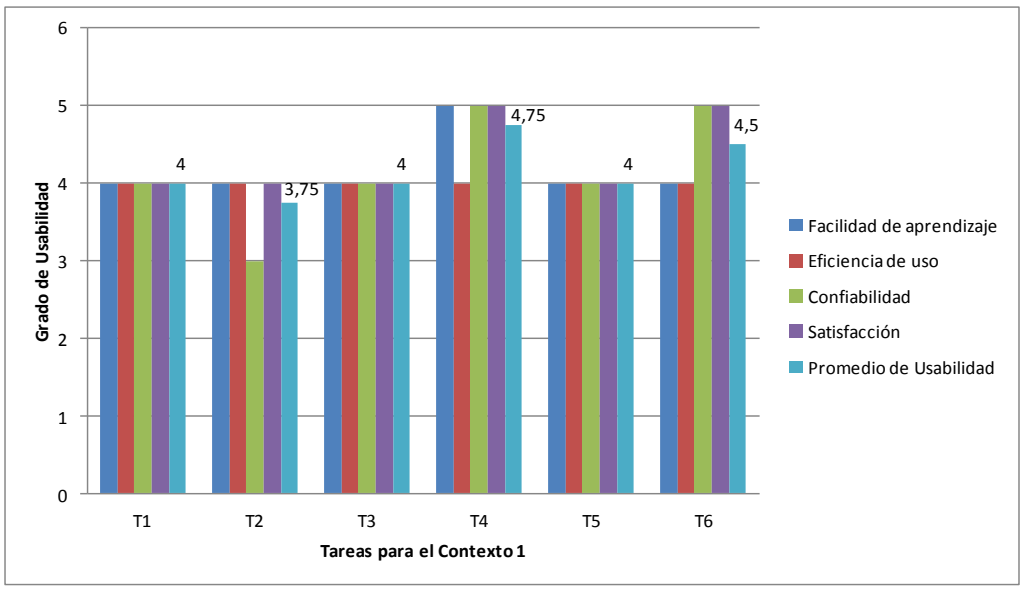


Figura 6-10. *Gráfico de la valoración de facilidad de uso para cada tarea en el contexto 1.*

Puede notarse en el gráfico que la facilidad de uso de la aplicación se encuentra en un rango superior de 4, por lo cual consideramos que la aplicación final es fácil de aprender, fácil de usar y satisface al usuario final. Los aspectos de usabilidad pueden ir mejorando con el tiempo ya que el usuario se familiarizará con la aplicación cada vez que interactúe con ella. El aspecto que se valoró más bajo ha sido el de confiablidad ya que algunos usuarios siguen acostumbrados a llevar anotaciones en papel o por parte del médico referente al seguimiento de su enfermedad. En la figura 6-11, se muestra los resultados de la evaluación realizada a usuarios finales, evaluando cada una de las tareas planteadas para el contexto 2.

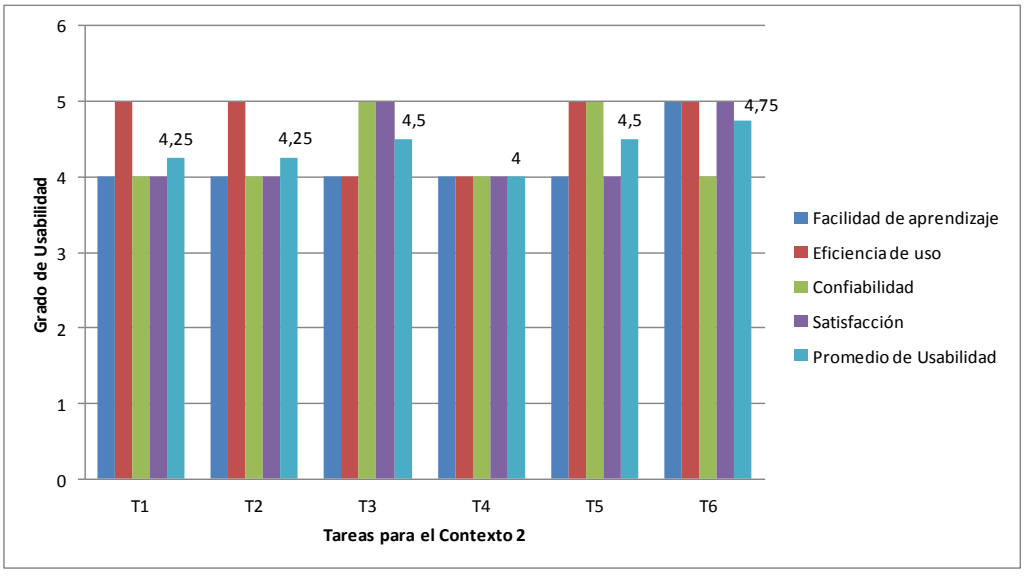


Figura 6-11. *Gráfico de la valoración de facilidad de uso para cada tarea en el contexto 2.*

Al igual que en el contexto anterior, la aplicación para el contexto 2, muestra un alto grado de facilidad de uso.

Conclusiones

En conclusión la arquitectura *software* desarrollada cuenta con un alto grado de facilidad de uso, lo que permite una interacción sencilla y bien descrita. Como se trata de una aplicación para ser utilizada para pacientes de diferentes edades, es evidente que la facilidad de uso para usuarios de rangos de edad menores y rangos mayores puede variar.

Además de la edad, también puede existir el problema para pacientes que utilizan muy poco los dispositivos móviles, por lo cual, sienten mayor restricción y problemas a la hora de utilizar la aplicación. Este problema puede ser corregido ofreciéndoles ayuda a los pacientes en la utilización y familiarización de la aplicación.

Es evidente que uso frecuente de la aplicación para la monitorización de pacientes, irá dándole mayor seguridad y confianza al paciente, lo que le permitirá aprender más acerca de su uso y la importancia que tiene ésta en el seguimiento y autocontrol de su enfermedad.

6.2 EVALUACIÓN DE CASOS DE ESTUDIO (USUARIO FINAL)

Una vez evaluados algunos criterios de calidad de la arquitectura implementada, se ha desarrollado una evaluación adicional, que corresponde a la implementación de prototipos identificando dos dominios en los que la arquitectura puede ayudar.

Estos dominios han sido evaluados por pacientes con diabetes y tensión arterial, que necesitan darle seguimiento a su enfermedad.

Como hemos evaluados los aspectos de *modificabilidad*, *grado de desempeño* y *facilidad de uso* para cada caso de estudio planteado, evaluaremos aspectos de *contenido*, *diseño* y *utilidad* de la arquitectura.

Para evaluar el criterio de *contenido*, responderá a preguntas relacionadas con la organización de los contenidos, la presencia de ayuda durante el uso de la aplicación, facilidad de interpretación, facilidad de identificación de los elementos y grado de valor de los contenidos presentados.

Para el criterio de *diseño*, se responderá a preguntas relacionadas a la distribución de los elementos visuales, especificaciones de las pantallas, interpretación e identificación de menú.

Para el criterio de *utilidad*, se responderá a preguntas relacionadas al grado de utilidad de las recomendaciones generadas, mensajes de prevención y educación y grado de satisfacción de la información mostrada.

6.2.1 Aspectos iniciales a la evaluación

Técnica de evaluación: se aplicará un cuestionario a los participantes en donde responderán a las preguntas concretas luego de utilizada la aplicación.

Aspectos de calidad a evaluar: se evaluarán los aspectos de contenido, diseño y utilidad de la aplicación en usuarios finales.

Contexto a evaluar: el paciente utilizará la aplicación para monitorizar por un determinado tiempo, las señales vitales asociadas a su enfermedad, podrá revisar las actividades de control médico que ofrece (recomendaciones, mensajes e prevención, educación y autocontrol) y agregará actividades físicas a desarrollar basadas en la medida obtenidas.

Población a evaluar: la evaluación se ha aplicado a diez personas (6 hombres y 4 mujeres). La población ha estado formada por dos candidatos a doctor, un doctor, un universitario de pregrado y seis personas no relacionadas a la universidad entre edades de 25 a 60 años. Los usuarios asociados con la universidad están relacionados con la tecnología mientras que los usuarios no relacionados con la universidad tienen poco conocimiento con la tecnología.

Tiempo de la evaluación: el tiempo que el paciente durará usando la aplicación en el contexto definido será de 45 minutos. Luego tomará 15 minutos para responder al cuestionario de evaluación presentado.

Escala a utilizar: se ha establecido una escala Likert de 1 a 5 para evaluar cada preguntar, siendo 1 la valoración más baja para una pregunta (Muy en desacuerdo) y 5 la valoración más alta (Muy de acuerdo.)

6.2.2 Evaluando cada pregunta. Resultados

En este punto evaluaremos cada pregunta, según las respuesta ofrecidas por los participantes, basadas en los criterios de evaluación definidos. En el anexo 6, se muestra el formato del cuestionario que hemos aplicado para evaluar estos criterios.

Criterio 1. Contenido

Pregunta 1. ¿Considera que la organización de los elementos dentro de la aplicación es buena?

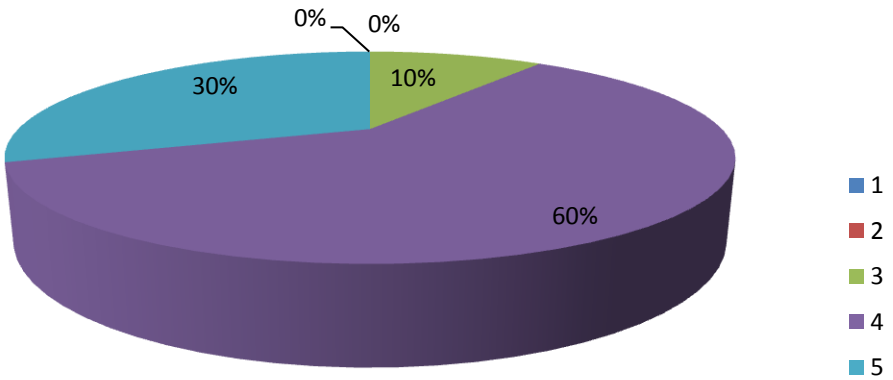


Figura 6-12. *Resultados del cuestionario relacionados con la pregunta 1.*

Pregunta 2. ¿Considera que los elementos gráficos y de texto le han ayudado a entender más claramente la aplicación?

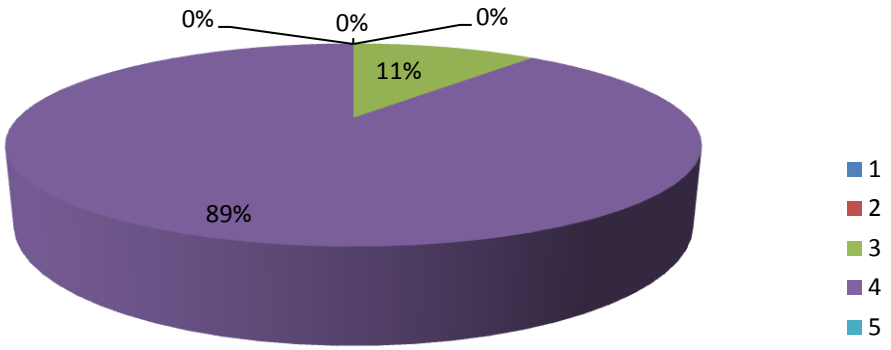


Figura 6-13. *Resultados del cuestionario relacionados con la pregunta 2.*

Pregunta 3. ¿Considera que la aplicación muestra todo la información necesaria para entender el funcionamiento de la interfaz?

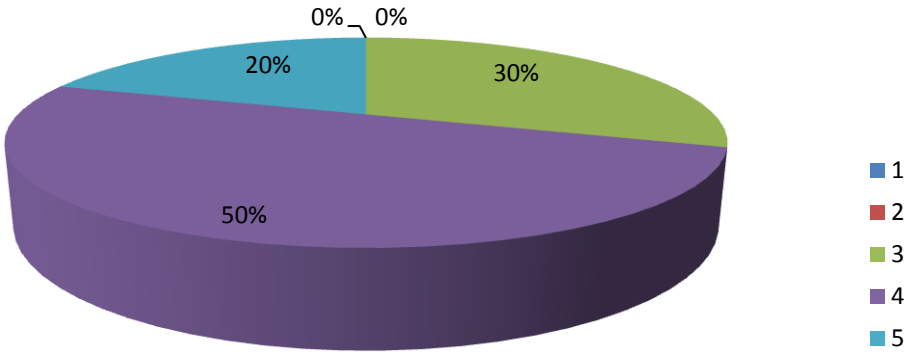


Figura 6-14. *Resultados del cuestionario relacionados con la pregunta 3.*

Pregunta 4. ¿Considera que la ayuda que ofrece la aplicación es suficiente para aclarar algunas dudas?

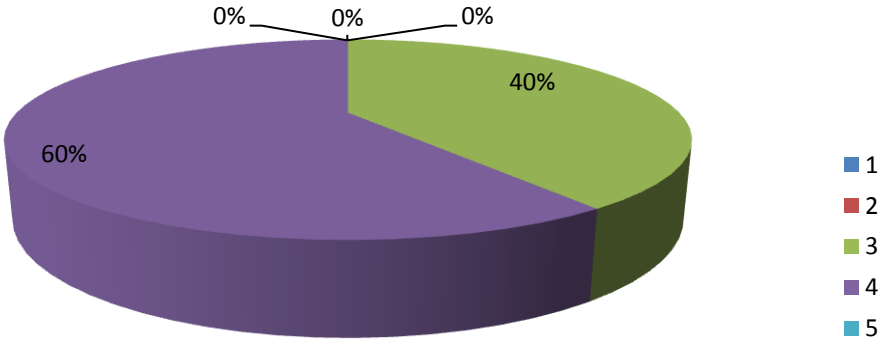


Figura 6-15. *Resultados del cuestionario relacionados con la pregunta 4.*

Pregunta 5. ¿Considera que la aplicación ofrece una facilidad de navegación?

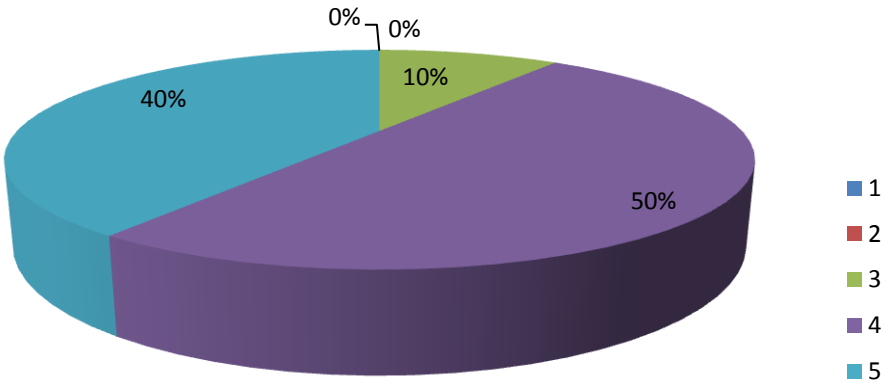


Figura 6-16. *Resultados del cuestionario relacionados con la pregunta 5.*

Pregunta 6. ¿Encuentra fácilmente los mensajes generados dentro de la aplicación?

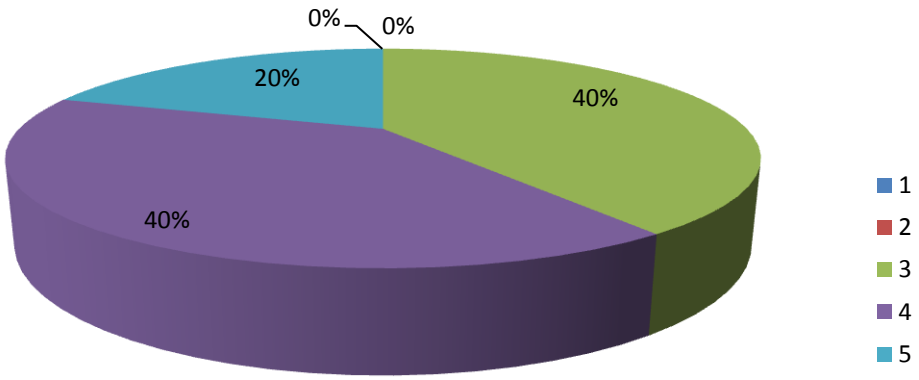


Figura 6-17. *Resultados del cuestionario relacionados con la pregunta 6.*

Pregunta 7. Entre pantallas ¿Sabe en donde se encuentra en cada momento?

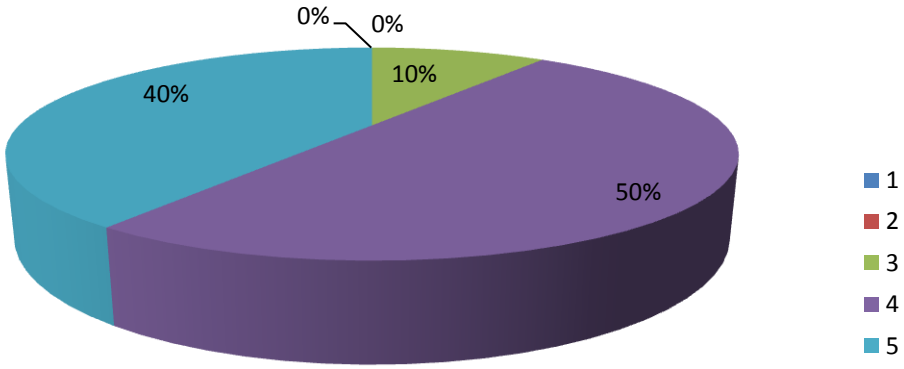


Figura 6-18. *Resultados del cuestionario relacionados con la pregunta 7.*

Pregunta 8. ¿La calidad información mostrada en cada mensaje es adecuada?

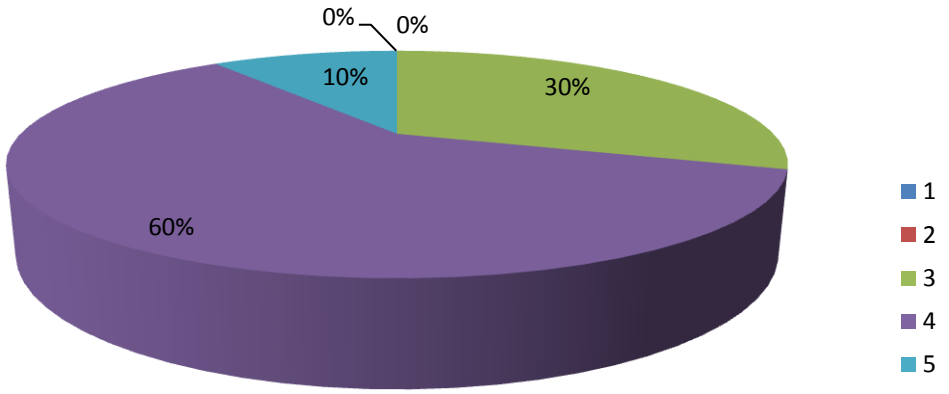


Figura 6-19. *Resultados del cuestionario relacionados con la pregunta 8.*

La mayor parte de los participantes consideran que la aplicación generada tiene un buen contenido, valorando entre 4 y 5 aspectos como facilidad de navegación, entendimiento de los elementos que componen la aplicación, claridad en la información mostrada, entre otros.

El contenido es uno de los aspectos que se depurará con el tiempo de tal manera que se le ofrezca al paciente información más adecuada y detallada según la enfermedad que tiene.

Criterio 2. Diseño

Pregunta 9. ¿Considera que el diseño de las pantallas es entendible?

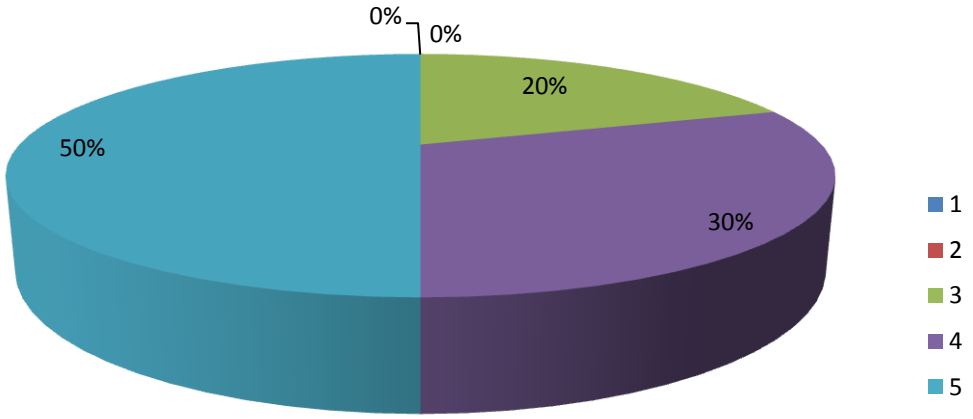


Figura 6-20. *Resultados del cuestionario relacionados con la pregunta 9.*

Pregunta 10. ¿Considera que la organización del menú es entendible?

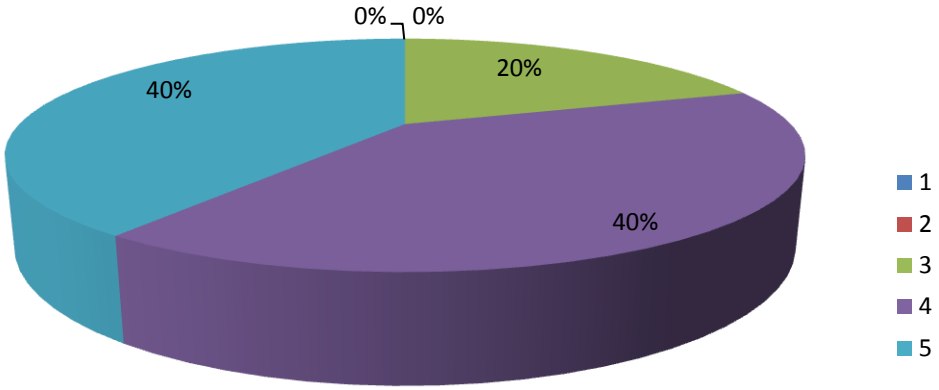


Figura 6-21. *Resultados del cuestionario relacionados con la pregunta 10.*

Pregunta 11. ¿Considera que los mensajes generados por la aplicación son legibles?

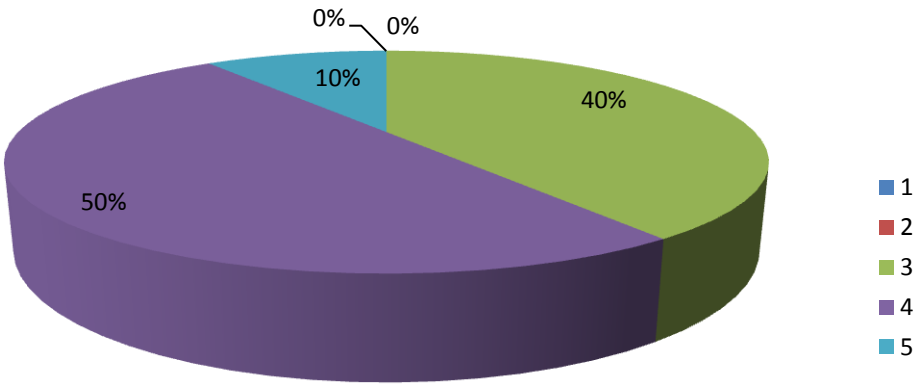


Figura 6-22. *Resultados del cuestionario relacionados con la pregunta 11.*

Pregunta 12. ¿Las funciones de las interfaces te resultaron sencillas de usar?

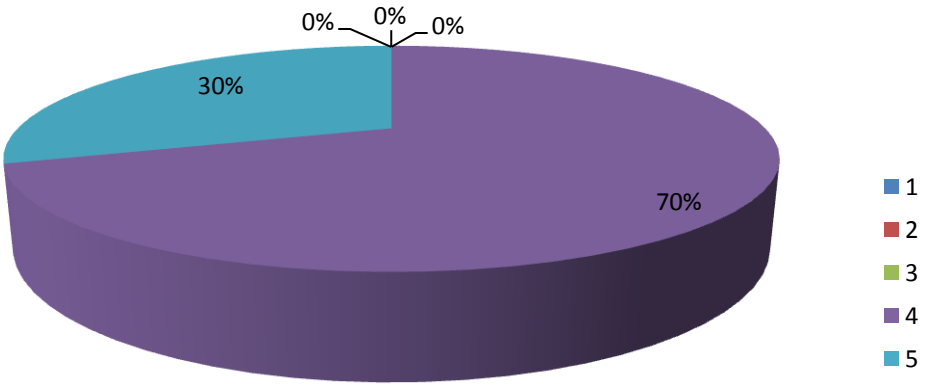


Figura 6-23. *Resultados del cuestionario relacionados con la pregunta 12.*

El criterio de diseño ha sido el aspecto más altamente valorado por los participantes, esto se debe a que hemos sido cuidadosos al momento de establecer los patrones adecuados para el diseño y funcionalidad de interfaces. Se han obtenido ponderaciones entre 4 y 5 según el total de participantes.

Por otro lado la mayoría de los pacientes, sienten que la aplicación ofrece una interfaz o pantallas bien explicativas, comprendiéndose en todo momento lo que está haciendo la aplicación. Esto es debido a que se ha cuidado mucho el diseño para familiarizar al paciente con la aplicación. El uso más cotidiano de la aplicación, hará que el paciente se sienta mejor y más seguro en el seguimiento de la enfermedad que padece.

Criterio 3. Utilidad

Pregunta 13. ¿Considera que las recomendaciones generadas son adecuadas?

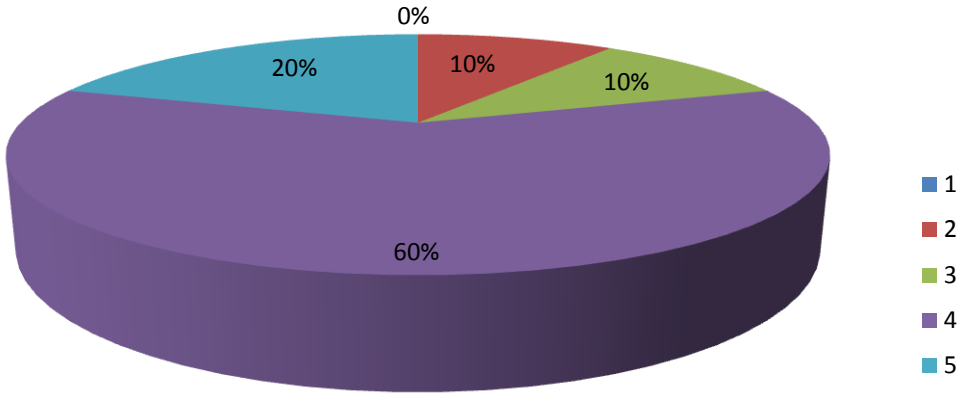


Figura 6-24. *Resultados del cuestionario relacionados con la pregunta 13.*

Pregunta 14. ¿Considera que los mensajes de educación asociados a la enfermedad son correctos?

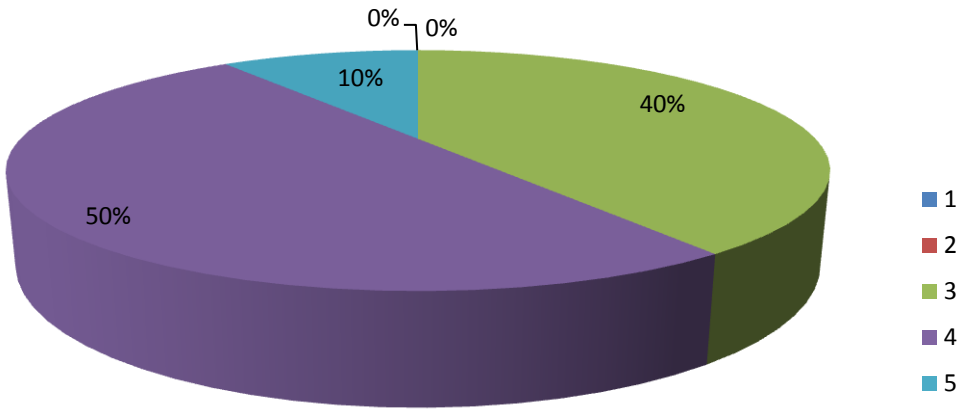


Figura 6-25. *Resultados del cuestionario relacionados con la pregunta 14.*

Pregunta 15. ¿Considera que el sistema le ofrece las respuestas que necesita?

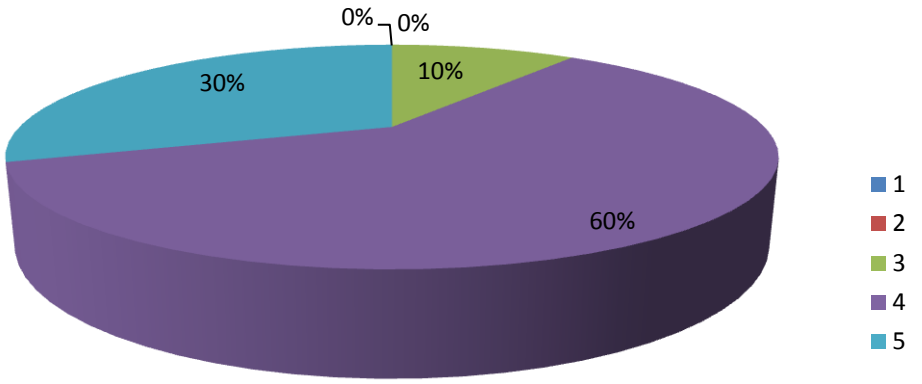


Figura 6-26. *Resultados del cuestionario relacionados con la pregunta 15.*

Pregunta 16. ¿Considera que los mensajes generados son fáciles de entender?

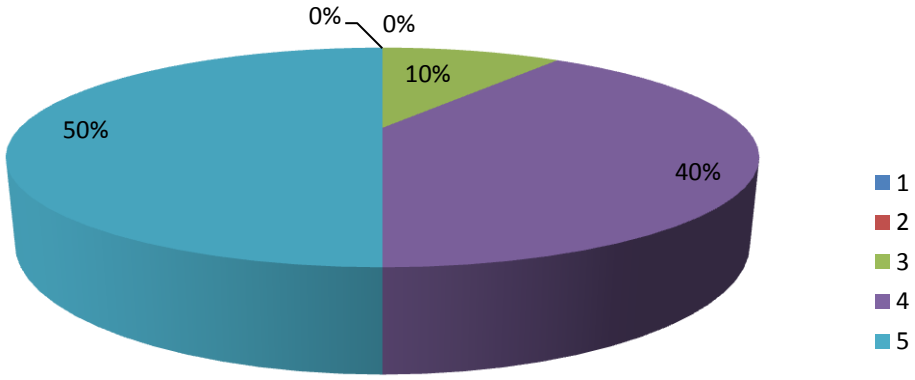


Figura 6-27. *Resultados del cuestionario relacionados con la pregunta 16.*

Pregunta 17. ¿Considera que los mensajes generados son fáciles de recordar?

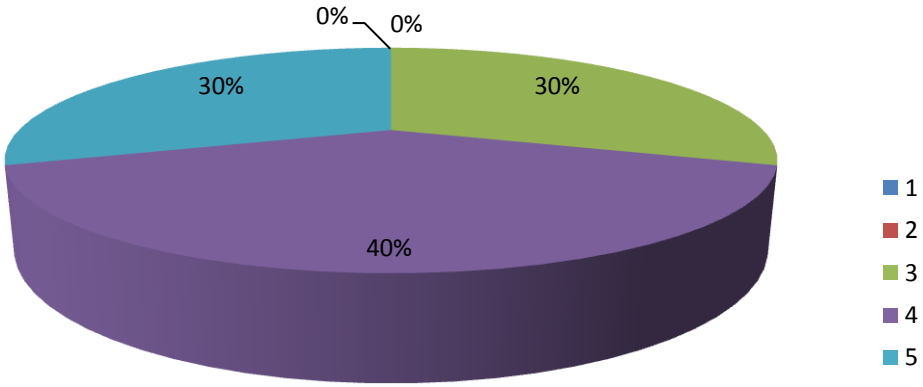


Figura 6-28. *Resultados del cuestionario relacionados con la pregunta 17.*

Pregunta 18. ¿Considera que el sistema es de gran utilidad para el seguimiento de su enfermedad?

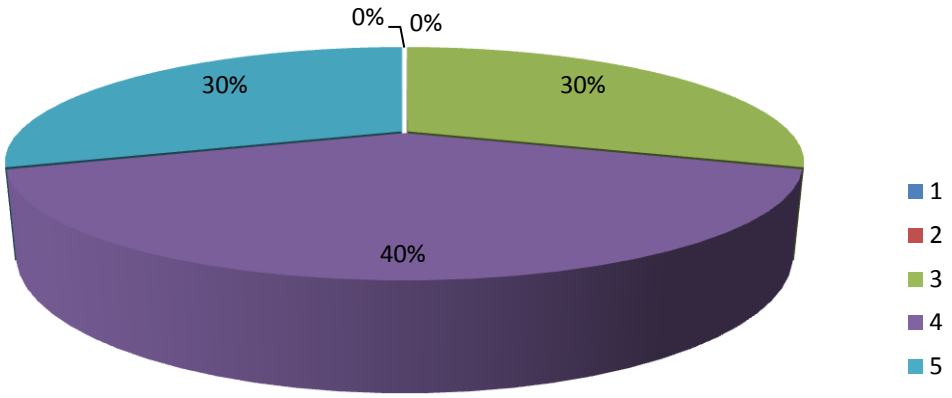


Figura 6-29. *Resultados del cuestionario relacionados con la pregunta 18.*

Pregunta 19. ¿Considera que la aplicación lleno todas sus expectativas funcionales?

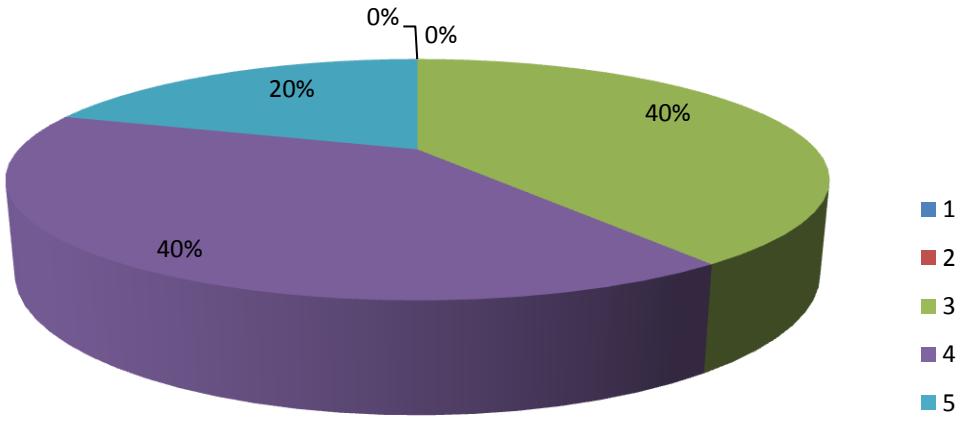


Figura 6-30. *Resultados del cuestionario relacionados con la pregunta 19.*

En términos generales los resultados de la evaluación realizada en esta primera aproximación, han sido bastante satisfactorios. Los aspectos que han tenido una valoración más baja han sido los de utilidad, esto se puede deber a que los pacientes están acostumbrados a realizar sus anotaciones en un cuadernillo, y en otras ocasiones no hacen anotaciones, por lo que no perciben la diferencia al momento de utilizar la aplicación evaluada. Otro motivo puede ser la dificultad que tienen algunos pacientes para utilizar este tipo de tecnologías, en donde ya sea por la edad o la poca experiencia en el uso de dispositivos móviles, sienten cierto rechazo ante ciertas tecnologías que consideran les ocupa tiempo en aprender.

Estos resultados nos ayudan para detectar posibles mejores y corregirlos de tal manera que se ofrezca una arquitectura más robusta y amigable al usuario.

6.3 CONCLUSIONES

El uso de diversos métodos para evaluar la arquitectura tanto en usuarios finales como desarrolladores nos ha ofrecido una panorámica de la validez y robustez de la arquitectura desarrollada. Se han podido evaluar diferentes tareas basadas en los escenarios que hemos detectado como más relevantes para el funcionamiento adecuado de los prototipos desarrollados.

Aspectos como la modificabilidad nos ofrecen información de la capacidad evolutiva inicialmente comparada en el capítulo 3, frente a otras propuestas. De igual manera los aspectos de usabilidad, desempeño, diseño, contenido y utilidad evaluados en este apartado, nos ofrece información importante que debe tomarse en cuenta para los trabajos futuros de esta tesis.

Las evaluaciones realizadas contemplan los principales aspectos de calidad que hemos considerado oportuno presentar, aunque existen otros aspectos de calidad que se pueden evaluar a través de otros métodos, estamos satisfechos por los resultados obtenidos.

Es importante ahora desarrollar una evaluación con mayor número de usuarios y algunos otros escenarios adicionales.

7

CAPÍTULO SÉPTIMO

7 CONCLUSIONES Y LÍNEAS FUTURAS

7.1 CONCLUSIONES

El desarrollo de esta tesis está dirigido por la consecución de los objetivos inicialmente formulados, que derivan de la hipótesis general de trabajo:

> *Es posible desarrollar una solución tecnológica para la generación de módulos y aplicaciones parametrizadas que faciliten la monitorización móvil de pacientes permitiendo el autocontrol sanitario a través de dispositivos móviles y biométricos; mediante un framework conceptual que guie el desarrollo de dicha solución.*

La validación de la tesis puede justificarse con las diferentes evaluaciones realizadas en los entornos médicos en la que fue probada, en las cuales se ha demostrado la validez de nuestra propuesta.

Los pacientes utilizaron la aplicación y evaluaron cada uno de los aspectos de la aplicación final, sin embargo, se recomienda realizar un estudio mucho más amplio en entornos médicos más completos. Esto facilita la interacción más amplia de los pacientes con la aplicación móvil.

La evaluación de los prototipos nos proporciona una validación parcial de los objetivos de esta tesis, a tener en cuenta ya que ha sido realizada por usuarios finales de la aplicación de monitorización móvil. Para cada uno de esos objetivos se definieron metas propuestas para

lograrlos, parcialmente o de manera más completa se analizan en conjunto con cada objetivo planteado.

Objetivo específico 1. Diseñar y desarrollar un modelo ontológico, orientado a la definición del dominio en estudio y a la descripción de cada uno de sus elementos. Además, las ontologías son interpretadas por los módulos para permitir la interoperabilidad entre cada uno de ellos y la aplicación final.

Las aportaciones relacionadas con el objetivo específico 1 son las siguientes:

- Respecto a la meta 1.1: *Realizar una revisión sobre la existencia de ontologías desarrolladas relacionadas a la monitorización de pacientes.*
    - Se han estudiado, analizado y criticado diversas clasificaciones ontológicas presentadas por algunos investigadores referentes al desarrollo de sistemas para facilitar la monitorización móvil de pacientes. Se han identificado las limitaciones o carencias que nos ha llevado a propone un modelo ontológico general que sirva para describir y evaluar el entorno médico en el cual desarrollamos esta tesis.
    - Se han estudiado las principales metodologías y lenguajes para la definición formal de las ontologías propuestas, seleccionándose **METHONTOLOGY** como el enfoque metodológico utilizado para la construcción de las ontologías. Se eligió esta metodología porque permite su utilización en la construcción de ontologías en distintos ámbitos, por la disponibilidad de documentación y por ser la metodología recomendada por la "Fundación para los agentes Físicos Inteligentes", la cual promueve la interoperabilidad entre las aplicaciones.
- Respecto a la meta 1.2: *Diseñar un modelo ontológico con la propuesta de nuevas ontologías, la utilización de algunas existentes o la modificación de una de ellas*.
    - Se ha definido formalmente un modelo ontológico general que relaciona aspectos de tecnologías, dispositivos, alimentos, ejercicios, actividades de control, enfermedades, entre otros aspectos que faciliten la monitorización móvil a través del desarrollo o generación de aplicaciones móviles parametrizadas.
    - Se han utilizado ontologías desarrolladas por otros investigadores, ya sea ajustándolas a las necesidades de nuestra arquitectura o implementando gran parte de ellas. Esto nos ahorra tiempo en centrarnos en los aspectos ontológicos que nos interesan, reutilizando o redefiniendo las ya existentes.

Objetivo específico 2. Diseñar y desarrollar un *framework conceptual* de referencia para la implementación de módulos que permita la monitorización móvil de pacientes, independiente de la tecnología, así como la integración dinámica de nuevos elementos del entorno.

Las aportaciones relacionadas con el objetivo específico 2 son las siguientes:

- Respecto a la meta 2.1: *Evaluar cada una de las propuestas que se han presentado referente a monitorización móvil de pacientes.*
    - Se ha realizado un estudio detallado sobre las diferentes aplicaciones para la monitorización de pacientes, evaluándolas mediante los criterios definidos, lo que facilitó la obtención de resultados comparativos. Dentro de los criterios evaluados están: *diseño* (cohesión, acoplamiento y usabilidad), *adaptabilidad* (capacidad evolutiva y migración tecnológica), *comunicación* (comunicación externa o transmisión de datos), *seguridad* (en el tratamiento de datos y en la transferencia de datos) y *costos* (de implementación y de mantenimiento). El estudio nos ha permitido tener una visión global sobre la cantidad de aplicaciones desarrolladas para la monitorización de pacientes y sobre la calidad de la información. En el estudio comparativo se ha incluido el *framework* MoMo, para poder observar, en función a los criterios definidos, sus carencias y virtudes frente a otras aplicaciones.
    - Se ha realizado una clasificación de estos estudios de la siguiente manera: propuestas de investigadores, que se han convertido o no, en productos *software* final, es decir, propuestas que no han pasado de un primer ciclo de desarrollo; aplicaciones que han sido desarrolladas y que se han utilizado, es decir, aplicaciones evaluadas en entornos reales; y otras aportaciones que no necesariamente son productos finales, sino que ofrecen ayuda parcial al desarrollo de arquitecturas para la monitorización móvil de pacientes.
- Respecto a la meta 2.2: *Evaluar cada una de las tecnologías de comunicación y procesamiento con que cuentan los diferentes dispositivos móviles existentes, para seleccionar los más adecuados para el desarrollo de nuestro framework.*
    - Se ha elegido Android como sistema operativo escalable y que actualmente representa una gran porción del mercado de dispositivos móviles.
    - Se han evaluado gran parte de las tecnologías referentes a dispositivos biométricos existentes, para conocer las prestaciones que ofrecen cada una de ellas y así seleccionar la más adecuada para integrarlas al *framework* desarrollada. Una de esas prestaciones consideradas para su selección, ha sido la capacidad de comunicación o transmisión de los datos obtenidos. Se seleccionó la tecnología *Bluetooth* para la obtención de los datos desde los dispositivos biométricos. Con este estudio se ha podido concluir que aunque existen múltiples dispositivos biométricos con tecnología *Bluetooth*, no todos nos ofrecen protocolos o normativas de acceso para la captura de los datos generados por el dispositivo para su posterior procesamiento por parte del dispositivo móvil.

Objetivo específico 3. Definir y desarrollar una distribución en capas que demuestre la eficiencia en cuanto a comunicación entre cada uno de los elementos del *framework*.

Las aportaciones relacionadas con el objetivo específico 3 son las siguientes:

- Respecto a la meta 3.1: *Realizar una revisión sobre la utilización de la distribución en n-capas de los elementos de un sistema.*
    - Se ha realizado un estudio detallado sobre las diferentes investigaciones que proponen la distribución de elementos de una arquitectura a través de capas. Se ha podido identificar las que se asocian al proceso de desarrollo de aplicaciones, como en otros entornos que no son propios del desarrollo de un producto *software*.
    - Se han extraído los conceptos y elementos más importantes de cada una de las investigaciones, de tal manera que podamos reutilizar y reajustar gran parte de las aportaciones de estas investigaciones. Por otra parte, se han definido nuevos elementos basándonos en los requerimientos de desarrollo y distribución del *framework* presentado.
- Respecto a la meta 3.2: *Definir una arquitectura distribuida en capas, que facilite el mantenimiento y actualización de la framework propuesto.*
    - Se ha definido una estructura basada en capas, que facilita la distribución de todos los elementos que componen la arquitectura desarrollada, facilitando la identificación de cada uno de ellos. Además de facilitar la distribución de los elementos, la estructura en capas, facilita el desarrollo de cada uno de esos elementos del *framework*, ofreciendo un esquema de desarrollo organizado, distribuido e identificable al momento de ser analizado por personal experto en desarrollo de tecnologías *software*. El desarrollo organizado facilita además el mantenimiento posterior de cada uno de los elementos, sin la necesidad de afectar otras partes de la arquitectura desarrollada. Esto permite el desarrollo de tecnologías *software* más estandarizadas y con elementos fácilmente identificables.

Objetivo específico 4. Definir unidades de diseño o patrones propios para la generación de módulos asociados a cada enfermedad y al perfil individual del paciente. Además las unidades de diseño deben permitir definir la ubicación de cada uno de los elementos de la arquitectura *software* generada.

Las aportaciones relacionadas con el objetivo específico 4 son las siguientes:

- Respecto a la meta 4.1: *Realizar una revisión sobre la utilización patrones para la generación de aplicaciones.*
    - Se ha estudiado las conceptualizaciones iniciales de desarrollo de aplicaciones basadas en patrones. Los elementos que componen cada uno de los patrones que se han predefinido y las características de implementación y desarrollo que definen a cada uno de ellos.

- Se han analizado, identificado y evaluado diversas propuestas que contemplen el uso de patrones para el desarrollo de los elementos que componen el desarrollo de aplicaciones. Esta identificación permite seleccionar los elementos más importantes de cada propuesta.

- Respecto a la meta 4.2: *Evaluar esos patrones de aplicaciones fijas para aplicaciones móviles.*

  - Se han identificado algunos patrones que no necesariamente se han utilizado en el desarrollo de aplicaciones móviles, ni en el desarrollo de aplicaciones para la monitorización de pacientes. Esto ha permitido identificar los elementos de interés de cada una de ellas y que podrían ajustarse a los requerimientos de nuestra arquitectura.

  - Se han evaluado aspectos diseño de interfaces para aplicaciones móviles, lo que permite el reajuste de la aplicación final para dispositivos con diferentes características, incluyendo dimensiones y resoluciones de pantalla. Esto es porque existen diversos tipos de dispositivos móviles (tabletas, móviles, PDA, etc.) en los cuales se ejecutará la aplicación generada por el *framework*.

- Respecto a la meta 4.3: *Reajustar y definir patrones para la generación de aplicaciones móviles en entornos médicos.*

  - Se ha identificado la necesidad de guiar el proceso de generación de cada uno de los elementos o módulos de la arquitectura propuesta. Para ello se ha definido patrones para la generación de aplicaciones móviles. Estos patrones se llamarán "*MobiPattern*" y definen aspectos de diseño, funcionalidad y ubicación de cada uno de los módulos que integran la aplicación final.

  - Se han redefinido y reutilizado algunos patrones previamente evaluados y que consideramos como aportaciones a la arquitectura desarrollada, pero que necesitan ser reajustados para nuestro caso de estudio.

Objetivo específico 5. Definir una estructura de relación entre cada uno de los módulos generados por el *framework*.

Las aportaciones relacionadas con el objetivo específico 5 son las siguientes:

- Respecto a la meta 5.1: *Relacionar cada uno de los elementos del framework propuesto, para las respectivas pruebas en tiempo de ejecución.*

  - Se ha identificado la relación funcional entre cada uno de los módulos que permitirán la generación de una aplicación final. Esta relación funcional está determinada por la ubicación de cada módulo en las respectivas capas definidas en la arquitectura. Una capa puede definir una funcionalidad específica de alguno de los módulos.

  - Se ha distribuido cada uno de los módulos en tres niveles, de acuerdo a la prioridad o necesidad de ejecución de cada módulo. Esta distribución está

definida en la capa de aplicaciones del dispositivo móvil. El primer nivel (nivel alto), define la ubicación de los módulos que serán ejecutados con mayor frecuencia, dentro de estos está el módulo de medidas. El segundo nivel (nivel intermedio), en este nivel se ubican todos los demás módulos de control de pacientes. En el tercer nivel (nivel bajo), están las funcionalidades relacionadas con la toma de decisiones, que se ha hecho de forma programática.

- Se ha definido la relación de cada elemento de la arquitectura desarrollada con las representaciones ontológicas previamente definidas. Se definió la ubicación de los datos en una capa datos, lo que facilita el tratamiento y procesamiento de los datos adquiridos. Este aspecto potencia la interoperabilidad de las aplicaciones generadas a partir del *framework* propuesto con aplicaciones externas, así como la adaptabilidad ante nuevas enfermedades o, en general, nuevos requisitos.

- Se han integrado todos los elementos definidos en una arquitectura completa, integrando todos los elementos de comunicación, procesamiento y almacenamiento de los datos que interactúan con la aplicación final generada.

Objetivo específico 6. Definir y desarrollar un modelo de comunicación entre dispositivos móviles y biométricos.

Las aportaciones relacionadas con el objetivo específico 6 son las siguientes:

- Respecto a la meta 6.1: *Probar la arquitectura software a través de conexiones remotas, ya sea a través de la red de telefonía móvil o red inalámbrica.*

  - Se han definido capas adicionales, para evaluar los aspectos de comunicación, seguridad, transmisión, enlace y negociación entre los elementos que interactúan con el *framework* desarrollado. Estas capas permiten la interoperabilidad de la aplicación en diversas redes, sin perder calidad de transferencia.

  - Se ha probado la funcionalidad de la arquitectura a través de la red de telefonía móvil e inalámbrica. Estas pruebas se han desarrollado para evaluar el tiempo de respuesta en el momento en que se está ejecutando la aplicación desde el dispositivo móvil. Esto nos asegura la funcionalidad de la aplicación en todo momento.

- Respecto a la meta 6.2: *Realizar pruebas de comunicación entre los dispositivos móviles elegidos y los dispositivos biométricos existentes.*

  - Se ha desarrollado un mecanismo de comunicación entre los dispositivos móviles y biométricos, basados en la tecnología Bluetooth. Se ha probado con dispositivos como tensiómetros, obteniendo los valores de las mediciones generadas.

- Se ha definido un método adicional de obtención de las mediciones desde los dispositivos biométricos que no cuenten con tecnología Bluetooth o alguna tecnología de comunicación. Este método permite la introducción manual del valor obtenido, para su posterior procesamiento. Se ha probado la funcionalidad en ambos métodos, confirmando que ambos casos son funcionales.

Objetivo específico 7. Validar la propuesta para la generación de módulos accesibles desde dispositivos móviles.

Las aportaciones relacionadas con el objetivo específico 7 son las siguientes:

- Respecto a la meta 7.1: *Realizar validaciones en diferentes entornos, asociados cada uno de ellos al uso de la arquitectura software generada mediante el uso por parte de pacientes y médicos.*

  - Se ha evaluado y validado la arquitectura *software* a través de dos prototipos desarrollados que hacen uso de todas las funcionalidades del *framework* MoMo, concretamente el primer prototipo para monitorizar pacientes con *diabetes* y el segundo prototipo para monitorizar pacientes con problemas de *tensión arterial*. Estos prototipos funcionales nos confirman el carácter genérico que tiene la arquitectura desarrollada, permitiendo su especialización en diferentes entornos de monitorización de enfermedades.

  - Mediante la evaluación de experiencias reales con los usuarios (pacientes) se ha obtenido información acerca de la adaptación de la arquitectura ante enfermedades en particular. Cada uno de los usuarios, después de utilizar la aplicación, responde ante situaciones de funcionalidad, interfaz de usuario, tiempo de respuesta, efectividad de las respuestas obtenidas, entro otros aspectos.

  - Se ha podido comprobar cómo la arquitectura en general, facilita el control médico de pacientes a través de todo el proceso de monitorización, ofreciendo un valor agregado para los pacientes. Toda propuesta que ofrezca en alguna u otra manera un apoyo a los pacientes y personas con padecimiento de alguna enfermedad monitorizable, es considerada como un paliativo para el mejoramiento del estilo de vida de estos pacientes.

Con las aportaciones y conclusiones relacionadas con cada una de las metas alcanzadas, se pretende haber validado el conjunto de objetivos específicos y en consecuencia el objetivo general de la tesis:

Objetivo general: *Diseñar y desarrollar un framework conceptual para la generación de módulos y aplicaciones parametrizadas que permita la monitorización móvil de pacientes, en la que intervienen diferentes dispositivos y servicios, los cuales actúan dependiendo del tipo de enfermedad, el perfil y las señales vitales de cada paciente.*

El *framework* desarrollado, basado en la definición de ontologías, la generación a partir de patrones funcionales y de diseño y la distribución de sus elementos en capas lógicas y físicas, da el soporte necesario que se puedan generar aplicaciones parametrizadas adaptadas a entornos médicos, en donde el proceso de monitorización sea necesario. Este es un aporte más a la problemática de seguimiento médico que tiene un paciente en general. Esta es la aportación principal de la tesis y pretende validar la hipótesis de partida.

7.2 DISCUSIÓN

Hay aspectos que quedan abiertos a la discusión y debate, de los que no se ha podido obtener conclusiones definitivas en esta tesis:

- **Generación de aplicaciones basada en el *framework***: una de las principales interrogantes que pueden surgir es cuándo usar el *framework* para la generación de aplicaciones móviles y por qué no utilizar o desarrollar una aplicación específica. La principal fortaleza del *framework* desarrollado, es ofrecer al usuario una aplicación genérica y adaptable que pueda ser utilizada en múltiples entornos médicos y con diversos dispositivos que interactúen entre sí para enviar y recibir las señales de un paciente. Esto es lo que hace diferente a nuestro desarrollo con respecto a una aplicación específica. El *framework* facilita el desarrollo de aplicaciones de distinto grado de complejidad, desde una aplicación para monitorizar una enfermedad en particular hasta una que monitoriza diversas enfermedades al mismo tiempo, pero siempre manteniendo la generalidad para ofrecer esta característica. Pese a todo, sigue quedando abierto el debate de si, en casos muy sencillos donde no se prevé un cambio en las necesidades/características del usuario, ni se prevé un cambio en su contexto, conviene obviar la arquitectura propuesta y desarrollar la aplicación desde cero.

- **Implementación del *framework* en entornos puntuales**: otra de las interrogantes que surgen luego de desarrollado el *framework*, es saber si la funcionalidad general de la cual hemos hablado se mantendrá siempre ante nuevos cambios en el desarrollo. Esto se debe a que a pesar de que hemos hablado de multimonitorización móvil, la arquitectura ha sido probado en muy pocos entornos médicos. Se hará necesaria la implementación para un mayor número de escenarios inicialmente desarrollados. Luego de hacer una implementación más completa irán surgiendo nuevas interrogantes en cuanto a diseño y funcionalidad. Estamos convencidos que el *framework* seguirá ofreciendo las características y funcionalidades para las cuales fue diseñado, pero se tendrán que hacer algunos cambios y ajustes a la hora de seguir implementándolo en otros entornos. La capacidad evolutiva de nuestro desarrollo y la reutilización de sus componentes hará posible estos cambios.

- **Funcionalidad de las ontologías desarrolladas**: el uso de ontologías en el desarrollo facilita la correcta definición de los datos almacenados en la aplicación y además ofrece servicios de búsqueda de información dentro de la arquitectura. Esta ha sido la idea inicial a la hora de implementar ontologías en el desarrollo del

framework, pero no se han aprovechado al máximo todas las facilidades que ofrecen las ontologías en el desarrollo de arquitecturas. Se hace necesario la explotación más al detalle de las ontologías planteadas, la interoperabilidad con otras existentes y la adaptación de nuevas propuestas a nuestro desarrollo. Aunque estamos seguros que las ontologías implementadas cubren los requisitos iniciales para las cuales fueron desarrolladas, quedan aún más funcionalidades que pueden ser aprovechadas en nuestra arquitectura.

7.3 LÍNEAS FUTURAS

El presente trabajo, pese a ofrecer un marco teórico, conceptual y práctico completo para la generación de aplicaciones móviles parametrizadas, ha dejado abierta líneas que pueden ampliar y mejorar esta propuesta. Estas líneas, que marcan la continuidad de la tesis, se pueden resumir en:

- **Adecuación de la arquitectura a otras tecnologías de comunicación**: la arquitectura desarrollada se ha planteado de forma genérica para garantizar la independencia respecto a los requerimientos *software* y *hardware* para la generación de aplicaciones. Sin embargo, la arquitectura actual solo es funcional con dispositivos basado en tecnología Bluetooth, por tal razón se hace necesario la integración de otras tecnologías, ya que existen en el mercado más dispositivos biométricos con diversas tecnologías de comunicación. Para facilitar los trabajos futuros en este aspecto que se ha definido capas (comunicación trasmisión, enlace, negociación, seguridad) que ubican cada uno de esos elementos de comunicación, de tal manera que si se necesitan reajustar sean fácilmente identificable.

- **Adecuación de la arquitectura a otras plataformas *software* (SO)**: como en el caso anterior, la arquitectura ha sido desarrollada para una plataforma (sistema operativo) específica, en nuestro caso Android. Por lo tanto, y considerando la diversidad de plataformas tecnológicas existentes, se podría migrar hacia todas ellas. Esto le daría mayor interoperabilidad a la arquitectura en general. Para facilitar la migración, la distribución de cada patrón basado en estándares (XML) pueden ser reajustados para nuevas plataformas. Además del desarrollo y clasificación basada en patrones, se ha definido la capa de aplicación, que contiene todos los módulos de la aplicación, facilitan la identificación y reajuste de la aplicación inicial.

- **Mejoramiento y depuración de la estructura ontológica definida**: la multi-monitorización, es una de las características que define al *framework* desarrollado. Cuando hablamos de multi-monitorización, nos referimos a la integración de múltiples personas (*aspecto que ya está contemplado en la arquitectura*), múltiples dispositivo (*delimitado en esta primera aproximación a dispositivos con tecnología Bluetooth*) y múltiples enfermedades. Este último aspecto necesita ser considerado con mayor detalle en los trabajos futuros relacionadas a esta tesis, es decir, es necesario el mejoramiento y depuración de

las estructura ontológicas definidas, de tal manera que siempre se mantenga actualizada para múltiples enfermedades. La organización y estructura de la arquitectura desarrollada, permite la integración de más enfermedades, siguiendo los patrones de diseño y funcionalidad desarrollados. En esta primera aproximación se han desarrollado prototipos completos para diabetes y tensión arterial, pero se hace necesario realizar pruebas para otras enfermedades existentes.

- **Motor de inferencia**: a pesar de que se han hecho las primeras aproximaciones definiendo un mecanismo de aprendizaje a través de las ontologías definidas, no se ha especificado un mecanismo completo que defina este aspecto. Es necesario integrar nuevos elementos que den mayor robustez a la arquitectura, de tal manera, que se desarrolle e implemente de forma completa un motor de inferencia que ofrezca mecanismos de aprendizaje automático basándose en el comportamiento y medidas generadas por un paciente. Esto le daría mayor autonomía al dispositivo móvil para el autocontrol médico.

- **Integración de nuevos elementos a la arquitectura**: la generalización del *framework*, ofrece servicios necesarios para el control médico de pacientes. Facilita su autocontrol, la obtención de sus señales vitales y la adecuación de información adicional, facilitando la generación de recomendaciones, alarmas, mensajes, etc. A pesar de que esta arquitectura ofrece múltiples servicios, podría integrarse más funcionalidades en un futuro. Funcionalidades como la comunicación con los datos médicos de algún centro especializado, adición de nuevos módulos de control, entre otros.

- **Realizar un estudio más amplio en entornos médicos más completos:** a pesar de que se han realizado valoraciones de la arquitectura en escenarios concretos, es decir enfermedades concretas, se recomienda realizar un estudio mucho más amplio en entornos médicos más completos. Esto facilita la interacción más amplia de los pacientes con la aplicación móvil y facilita la definición de nuevos requerimientos adicionales a los inicialmente planteados.

7.4 PUBLICACIONES

Durante el desarrollo de esta tesis doctoral se han ido extrayendo contribuciones parciales y se han enviado a diversas revistas y congresos con el fin de validar los avances conforme se iban produciendo, recibiendo además las consideraciones al respecto de parte de la comunidad científica. Expresándolo en números se ha conseguido 2 publicaciones en revistas, 4 capítulos de libro y 8 contribuciones a congresos:

- **Presentación a Journals (Indexados JCR)**

  A Proposal of an Ubiquitous Monitoring Architecture Patterns-Enabled: A case study on Diabetes (pendiente). Vladimir Villarreal, Jesús Fontecha, Ramón Hervas, José Bravo. Journal of Science of Computer Programming. 2012. x(x): pp. xxx-xxx, DOI:xxxxxxxx

Using a communication model to collect measurement data through mobile devices (pendiente). José Bravo, Vladimir Villarreal, Ramon Hervás, Gabriel Urzaiz. Journal of Sensors. 2012. X(x): pp. xxx-xxx, DOI:xxxxxx

- **Contribuciones a Congresos Nacionales e Internacionales**

Respecto a la propuesta inicial

Towards a Ubiquitous Mobile Monitoring for Health-care and Ambient Assisted Living. Vladimir Villarreal, José Bravo, Ramon Hervás. In II International Workshop on Ambient Assisted Living, IWAAL 2010 with CEDI 2010. Valencia, Spain. 2011

Aplicabilidad de la tecnología móvil en Residencias y Centros Especializados. Jesus Fontecha, Francisco Navarro, Luis Sánchez, Vladimir Villarreal, Ramón Hervás & José Bravo. SEMER. Almería, España. 2011

MoMo: A Framework Proposal for Patient Mobile Monitoring. Vladimir Villarreal, José Bravo, Ramon Hervás. In 5th Conference of the Euro-American Association on Telematics and Information Systems (EATIS-2010). Panamá, Panamá. 2010

Una propuesta de multimonitorización móvil para el hogar. José Bravo Rodríguez, Vladimir Villarreal Contreras, Ramón Hervás Lucas, Jesús Fontecha Diez. In IX Congreso Nacional de la Sociedad Española de Electro medicina e Ingeniería Clínica, (SEEIC). Santa Cruz de Tenerife, España. 2010

Respecto a la propuesta en entornos médicos específicos

Ambient Intelligence: technological solutions for wellness and supporting to daily activities. Vladimir Villarreal, Jesús Fontecha, Ramón Hervás, José Bravo. In 10th Latin American and Caribbean Conference – LACCEI 2012. Panamá, Rep. de Panamá. 2012

A proposal for Mobile Diabetes Self-Control: Towards a patient Monitoring Framework. Vladimir Villarreal, Javier Laguna, Silvia López, Jesus Fontecha & Carmen Fuentes. In Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing and Ambient Assisted Living. Salamanca, Spain. 2009

Diabetes Patients' care based on Mobile Monitoring. Vladimir Villarreal, Ramón Hervás, Javier Laguna, Ana Diez, Carlos Sánchez, Silvia López. In IADIS International Conference, Applied Computing 2009. Rome, Italy. 2009

Respecto a la definición y generación ontológica

Applying Ontologies in the Development of Patient Mobile Monitoring Framework. Vladimir Villarreal, Ramón Hervás, Ana Diez Fernández & José Bravo. Advancements of Medical Bioengineering and Informatics. Constanta, Romania. 2009

Ontology Development for a Patient Monitoring Framework. José Bravo, Vladimir Villarreal, Ramon Hervás, Carlos Sánchez, Silvia Lopez, Ana Diez, Carmen Fuentes,

Javier Laguna. In II WSEAS International Conference on BIOMEDICAL ELECTRONIC and BIOMEDICAL. Moscow, Russia. 2009

Respecto a la definición de Patrones

Using and Applying MobiPattern to Design MoMo Framework Modules. Vladimir Villarreal, Jesús Fontecha, Ramón Hervás & José Bravo. Ambient Assisted Living, volumen 6693. Málaga, Spain. 2012

Respecto a la definición de los elementos distribuidos en capas

Monitoring Architecture to collect measurement data and medical patient control through mobile devices. Vladimir Villarreal, Gabriel Urzaiz, Ramon Hervas and Jose Bravo. In V International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI2011). Riviera Maya, Mexico. 2011

Respecto a la implementación y desarrollo del framework

An Architecture to development an Ambient Assisted Living applications: A study case in Diabetes. Vladimir Villarreal, Jesus Fontecha, Ramon Hervas & Jose Bravo. In V International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI2011). Riviera Maya, Mexico. 2011

7.5 CONSIDERACIONES FINALES

Esta memoria pretende ofrecerle al lector una visión detallada y uniforme del trabajo realizado en esta tesis. Se incluye en este capítulo un análisis de cada uno de los objetivos inicialmente planteados y la consecución de cada uno de ellos a través de las metas propuestas, lo que nos llevó a la comprobación del la hipótesis inicial. Además se dan a conocer los trabajos futuros que se podrán abordar a partir de la finalización de la investigación, y que darán mayor robustez al tema de tesis desarrollado. Se presenta además, todas aquellas aportaciones a congresos asociados al área de estudio, en donde se ha podido intercambiar conocimientos, situación ha permitido dar a conocer lo que en un principio proponíamos.

El desarrollo de un *framework* genérico que facilite el desarrollo de aplicaciones móviles para el control de pacientes, viene a ofrecer una solución a la necesidad de seguimiento médico que tienen los pacientes. No se ha pretendido desarrollar una aplicación concreta, sino definir y desarrollar una arquitectura que pueda ser integrada en otros entornos y que facilite el desarrollo organizado de todos los elementos que hemos mencionado.

Somos conscientes de que ésta no es la única solución que se ofrece, pero si estamos completamente seguros que ésta propuesta cumple con el objetivo inicialmente planteando, como una solución para integrar tecnologías que ayuden y fortalezcan las actividades diarias de las personas que padecen de alguna enfermedad.

REFERENCIAS BIBLIOGRÁFICAS

(W3C), W. W. W. C. (2008). Guia Breve de la Web Semántica. from http://www.w3c.es/Divulgacion/Guiasbreves/WebSemantica

AirStrip, T. (2011). *AirStrip Patient Monitoring*. Last access 2011, from http://www.airstriptech.com/.

Alexander, C. (1979). *The timeless way of building*. New York, Oxford University Press.

Ambler, S. W., J. Nalbone, et al. (2005). *Extending the RUP with the Zachman Framework*, Pearson Education.

Android Developer. (2011). *The Developer's Guide*. Last access 2012, from http://developer.android.com/index.html.

Appleton, B. (2000). *Patterns and Software: Essential Concepts and Terminology*, Brad Appleton.

Arpírez, J. C., A. Gomez-Perez, et al. (1998). *(ONTO)2Agent: an ontology-based WWW broker to select ontologies*. In Workshop on applications of ontologies and problem-solving methods (ECAI98), Brighton, UK.

Atkinson, C. and T. Kuhne (2003). *Model-driven development: a metamodeling foundation*. Software, IEEE 20(5): 36-41.

Barbacci, M. R. (1995). *Quality attributes*, Carnegie Mellon University, Software Engineering Institute.

Barbarán, J., M. Díaz, et al. (2007). *RadMote: a mobile framework for radiation monitoring in nuclear power plants.* In XXI International Conference on Computer, Electrical, and Systems Science and Engineering CESSE 2007, Vienna, Austria.

Bass, L., M. Klein, et al. (2000). *Quality Attribute Design Primitives*. Software Engineering Institute, Carnegie Mellon University.

Baz Alonso, a., I. Ferreira Artime, et al. (2008). *Dispositivos móviles*. Oviedo, Universidad de Oviedo.

Bengtsson, P., N. Lassing, et al. (2004). *Architecture-Level Modifiability Analysis (ALMA)*. The Journal of Systems and Software 69: 129-147.

Bézivin, J. (2004). *In search of a basic principle for Model Driven Engineering* UPGRATE European Journal 5(2): 21-24.

Bhargava, A. and M. Zoltowiski (2003). *Sensor and Wireless Communications for Medical Care*. In 14th International Workshop on Database and Expert Systems Applications (DEXA 03).

Blázquez, M., M. Fernandez-Lopez, et al. (1998). *Building ontologies at the knowledge level using the ontology design environment*. In Knowledge Acquisition of Knowledge-Based System Workshop (KAW) SHARE 4-1-SHARE 4-15, Banff, Alberta, Canada.

Blum, J. M. and E. H. Magill (2010). *The Design and Evaluation of Personalised Ambient Mental Health Monitors*. In Consumer Communications and Networking Conference (CCNC) - 7th IEEE.

Bodart, F. and J. Vanderdonckt (1995). *Towards a Systematic Building of Software Architecture: the TRIDENT Methodological Guide*. In Design, Specification and Verification of Interactive Systems, DSV-IS95, Springer-Velarg.

BOSCH. (2011). *Health Buddy System*. Last access 2012, from http://www.bosch-telehealth.com.

Braun, A. (2010). *Asistencia sanitaria: tecnologías claves para Europa*. Informes de Tecnologías Claves de la Comisión Europea. C. Europea. Madrid, Círculo de Innovación en Biotecnología**:** 89.

Brey, G. A., G. Escobar, et al. (2005). *Arquitecturas de Proyectos de IT. Evaluación de Arquitecturas*. Facultad Regional de Buenos Aires - Departamento de Sistemas. Buenos Aires, Argentina, Universidad Tecnológica Nacional.

Brhargava, A. and M. Zoltowiski (2003). *Sensor and Wireless Communications for Medical Care*. In 14th International Workshop on Database and Expert Systems Applications (DEXA 03).

Broens, T., A. V. Halteren, et al. (2007). *Towards an application framework for context-aware m-health applications*. International Journal of Internet Protocol Technology 2(2): 109-116.

Brown, B., M. Kohls, et al. (2002). *FDA XML Data Format Design Specification*. Food and Drug Administration.

BTnodes. (2010). *BTnodes*. Last access 2011, from http://www.btnode.ethz.ch/.

Cantais, J., D. Dominguez, et al. (2005). *An example of food Ontology for Diabetes Control*. In Working notes of the ISWC 2005 Workshop on Ontology Patterns for the Semantic Web, Galway, Ireland.

CEN. (1999). *Health Informatics - Vital Signs Representation*. Last access 2011, from http://www.cen.eu/cenorm/homepage.htm.

Clements, P., F. Bachman, et al. (2002). *Documenting Software Architecture: Views and Beyond*, Addison Wesley.

Crossbow-Technology-Inc. (2010). Last access 2011, from http://www.xbow.com.

Cheun, D. W., H. M. Lee, et al. (2010). *An Effective Framework for Monitoring Service-Based Mobile Applications*. In IEEE 7th International Conference on e-Business Engineering, IEEE.

Chu, W. C., C. Juez-Nan, et al. (2001). *Implementing an agent system using N-tier pattern-based framework*. In Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International.

D'Souza, D. F. and A. C. Wills (1998). *Objects, Components, and Frameworks with UML*. Massachusetts, Addison Wesley Longman, Inc.

Dagtas, S., Y. Natchetoi, et al. (2007). *An integrated wireless sensing and mobile processing architecture for assisted living and healthcare applications*. In 1st International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (ACM SIGMOBILE), San Juan, Puerto Rico, ACM.

Daniel, S. (2007). *Interaction design and implementation for multimodal mobile semantic web interfaces*. In Conference on Human Interface, Beijing, China, Springer-Verlag.

Daria, C. n., M. John, et al. (2000). *Rapid application development using agent itinerary patterns*.

DICOM. (2004). *DICOM Part 5: Data Structures and Encoding*. Digital Imaging and Communications in Medicine, Last access 2012, from http://www.dclunie.com/dicom-status/status.html#DiffsStandard20032004.

Ellison, R. J., R. C. Linger, et al. (1999). *Survivable Network Analysis Method: A case study*. Software, IEEE 16(4): 70-77.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall.

Fensel, D., I. Horrocks, et al. (2000). *OIL in a Nutshell*. In Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, London, UK, Springer-Verlag.

Fernandez-Lopez, M. (1999). *Overview of Methodology for Building Ontologies*. In Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends (IJCAI99).

Fernandez-Lopez, M. and A. Gomez-Perez (2002). *Overview and analysis of methodologies for building ontologies*, Cambridge University Press. 17**:** 129-156.

Fernández-Luque, L., J. L. Sevillano, et al. (2006). *eDiab: A System for Monitoring, Assisting and Educating People with Diabetes*. In 10th International Conference, ICCHP 2006, Linz, Austria, Springer.

FIPA. (2000). *FIPA Ontology Service Specification*. Last access 2011, from http://www.fipa.org/specs/fipa00086/XC00086C.html.

Folmer, E. and J. Bosch (2003). *Usability patterns in Software Architecture*. In HCI International - 2003.

Folmer, E., J. v. Gurp, et al. (2003). *Investigating the Relationship between Usability and Software Architecture, Software Process Imporvement and Practice*, Wiley.

Folmer, E., J. v. Gurp, et al. (2003). *Scenario-Based Assessment of Software Architecture Usability*. In Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction - ICSE, Portland.

Gabriel, R. P. (1998). *Patterns of Software: tales from the software community*. New York, Oxford University Press, USA.

Gamma, R., R. Johnson, et al. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley.

Gao, T., D. Greenspan, et al. (2005). *Vital Signs Monitoring and Patient Tracking Over a Wireless Network*. In 27th Annual International Conference of yhe IEEE EMBS, Shanghai.

Georga, E., V. Protopappas, et al. (2009). *Data mining for blood glucose prediction and knowledge discovery in diabetic patients: The METABO diabetes modeling and management system*. In Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE.

Gómez-Perez, A. (1998). *Knowledge Sharing and Reuse*. The Haanbook of Applied Expert Systems. L. CRC.

González V.,Jose A. (2009). *Aplicación para la monitorización remota de pacientes.*Universidad de Valladolid. Dep. de Ingeniería Técnica de Informática de Sistemas. PFC, pp. 139.

Guarino, N. (1998). *Formal Ontology in Information Systems*. In 1st International Conference, Trento, Italy, ACM.

Heijst, G. V., A. T. Schreiber, et al. (1997). *Using explicit ontologies in KBS development*. International Journal of Human-Computer Studies 46(2-3): 183-292.

Hellbruck, H., X. Hua, et al. (2009). *Effective Movement Classification for Context Awareness in Medical Applications Networking*. In Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on.

Hellín, P. D., C. Fuentes, et al. (2010). *A mobile proposal for Pediatric Obesity Treatment*. Journal Child: Care, Health and Development 36(1): 48.

Hellín, P. D., C. Sánchez, et al. (2009). *Propuesta Ontológica para el control de la Obesidad Infantil*. XXVII Congreso Anual de la Sociedad Española de Ingeniería Biomédica - CASEIB. Cádiz, España.

Hervás, R. (2009). *Modelado de Contexto para la Visualización de Información en Ambientes Inteligentes*. Departamento de Tecnologías y Sistemas de Información. España, Universidad de Castilla-La Mancha. Doctor**:** 330.

HL7. (2005). *Health Level Seven Standards versión 3*. Last, from http://www.hl7.org.

Horrocks, I., D. Fensel, et al. (2000). *The Ontology Inference Layer OIL*. Last access 2011, from http://www.ontoknowledge.org/oil/.

Hung, K. and Y.-T. Zhang (2003). *Implementation of a WAP-Based Telemedicine System for Patient Monitoring*. IEEE Trans. Information Technology in Biomedicine 7(2): 101-107.

In, H., R. Kazman, et al. (2001). *From Requeriments Negotiation to Software Architectural Decisions*, Software Engineering Institute, Carnegie Mellon University. Obtenido el 15-8-2002.

INE. (2011). *Instituto Nacional de Estadística.* Last access 2011, from http://www.ine.es/prensa/np587.pdf.

Kaisler, H. (2005). *Software Paradigms*. London, UK, John Wiley & Sons, Inc.

Karlof, C., N. Sastry, et al. (2004). *TinySec: A Link Layer Security Architecture for Wireless Sensor Network*. In Second ACM Conference on Embedded Networked Sensor Systems, SenSys 2004.

Kazman, R., P. Clements, et al. (2001). *Evaluating Software Architectures. Methods and case studies*, Addison-Wesley Professional.

Kebler, C. (2007). *Similarity measurement in context*. In Sixth International and Interdisciplinary Conference on Modeling and Using Context, Springer - Verlag.

Kellogg, M. (2008). *WordReference. Diccionario de la Lengua Española*. Last access 2011, from http://www.wordreference.com/definicion/monitorizaci%F3n

Kent, S. (2002). *Model Driven Engineering*. In Third International Conference on Integrated Formal Methods (IFM2002), London, UK, Springer-Velarg.

Kitchenham, B. (2004). *Procedures for Performing Systematic Review*.

Kong, K., C. Ng, et al. (2000). *Web-Based Monitoring of Real-Time ECG Data*. Computers in Cardiology 27: 189-192.

Koutkias, V. G., I. Chouvarda, et al. (2010). *A Personalized Framework for Medication Treatment Management in Chronic Care*. Information Technology in Biomedicine, IEEE Transactions on 14(2): 464-472.

Krco, S. and V. Delic (2003). *Personal Wireless Sensor Network for Mobile Health Care Monitoring*. In 6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service, Serbia y Montenegro, Nis.

Larman, C. (2004). *UML y Patrones: Introducción al análisis y diseño orientado a objetos*, Prentice-Hall Hispanoamericana.

Lee, H. S., S. H. Park, et al. (1997). *Remote patient monitoring service through World-WideWeb*. In 19th International Conference of IEEE/EMBS.

Lenat, D. B. and R. V. Guha (1990). *Building Large Knowledge Based Systems: Representation and Inference in the CyC Project*. Reading, Massachusetts, Addison-Wesley Publishing.

Lewis, P., S. Madden, et al. (2004). *The Emergence of NetworkingAbstractions and Techniques in TinyOS, NSDI04*. In First Symposium on Networked Systems Design and Implementation, San Francisco, California.

Lin, Y.-H., I.-C. Jan, et al. (2004). *A wireless PDA-Based Physiological Monitoring System for Patient Transport*. IEEE Trans. Information Technology in Biomedicine 8(4): 439-447.

Lorenz, A., D. Mielke, et al. (2007). *Personalized mobile health monitoring for elderly*. In 9th International Conference on Human Computer Interaction with Mobile Devices and Services, Singapore, ACM.

Magrabi, F., N. H. Lovell, et al. (1999). *A web-based approach for electrocardiogram monitoring in the home*. International Journal of Medical Informatics 54: 145-153.

Mamykina, L., E. D. Mynatt, et al. (2006). *Investigating health management practices of individuals with diabetes*. In SIGCHI Conference on Human Factors in Computing Systems, Montreal, Quebec, Canada, ACM.

Martin, J. (1991). *Rapid Application Development*. new York, Macmillan Coll Div.

McGuinness, D. L. and F. v. Harmelen. (2004). *OWL Web Ontology Language Overview*. Last access 2012, from http://www.w3.org/TR/owl-features/.

Mei, H., I. Widya, et al. (2006). *A Flexible Vital Sign Representation Framework for Mobile Healthcare*. In Pervasive Healthcare Conference and Workshops 2006, Innsbruck, Austria.

Memon, Q. A. (2009). *Implementing Role Based Access in Healthcare Ad-Hoc Networks*. Journal of Network 4(3): 192-199.

Miller, J. and J. Mukerji. (2003). *MDA Guide Version 1.0.1 Retrieve: 08-07-2007*. Last access 2012, from http://www.omg.org/docs/omg/03-06-01.pdf.

MobiHealth. (2003). *MobiHealth Project*. Last access 2011, from http://www.mobihealth.org/.

Mohanalakshmi, S. and J. Jacob (2007). *Design and implementation of an intelligent instrument for apnea patients*. In Information and Communication Technology in Electrical Sciences (ICTES 2007), 2007. ICTES. IET-UK International Conference on.

Nelwan, S., T. van Dam, et al. (2002). *Ubiquitous Mobile Access to Real-time Patient Monitoring Data*. Computers in Cardiology 29: 557-560.

Nirmalya, R., P. Gautham, et al. (2007). *A Middleware Framework for Ambiguous Context Mediation in Smart Healthcare Application*. In Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Washington, DC, IEEE Computer Society.

OMG. (1999). *OMG: Unifed Modeling Language 1.3, Standard.* . Last access 2011, from http://www.omg.org/mda/.

ORMSCWhitePaper. (2001). *A Proposal for an MDA Foundation Model, OMG*. Last.

Paradiso, R., A. Alonso, et al. (2008). *Remote health monitoring with wearable non-invasive mobile system: The Healthwear project*. In Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE.

Park, S. H., J.-H. Park, et al. (1998). *Real-Time Monitoring of Patients on Remote Sites*. In 20th Annual Int'l Conference of IEEE/EMBS.

Patel, A., M. McRoberts, et al. (2009). *Identity propagation in N-tier systems*. In Military Communications Conference, 2009. MILCOM 2009. IEEE.

Pattichis, C. S., E. Kyriacou, et al. (2002). *Wireless Telemedicine Systems: An Overview*. IEEE Antennas and Propagation Magazine. 44**:** 143-153.

Peeters, J. P., M. F. Bense, et al. (2010). *Gentag Inc. Corporate*. Last access 2011, from http://www.gentag.com/index.html.

Peis-Redondo, E., E. Herrera-Viedma, et al. (2009). *Ontologías, taxonomías y agentes: recuperación "semántica" de la información*. In JOTRI 2003: II Jornadas de Tratamiento y Recuperación de Información.

Preuveneers, D. and Y. Berbers (2008). *Mobile phones assisting with health self-care: a diabetes case study*. In 10th International Conference on Human Computer Interaction with Mobile Devices and Services, Amsterdam, The Netherlands, ACM.

Ren, K., W. Lou, et al. (2007). *Multi-user broadcast authentication in wireless sensor network*. In Fourth Annual IEEE Communications Society Conference on Sensor , Mesh and ad-hoc Communications and networks, SECON 2007, San Diego, California.

Rogers, R., J. Lombardo, et al. (2009). *Android Application Development Programming with the Google SDK*. Sebastopol, CA, O'Reilly Media, Inc.

Roudsari, A., S. Zhao, et al. (2000). *Web-based decision support and telemonitoring system for the management of diabetes*. In Engineering in Medicine and Biology Society. 22nd Annual International Conference of the IEEE.

Ryder, J., B. Longstaff, et al. (2009). *Ambulation: A Tool for Monitoring Mobility Patterns over Time Using Mobile Phones*. In Computational Science and Engineering, 2009. CSE '09. International Conference on.

Shannon, C. (1948). *A mathematical theory of communication* The Bell System Technical Journal 27: 379–423 and 623–656.

Sing-Hui, T., L. Seung-Chul, et al. (2008). *WSN Based Personal Mobile Physiological Monitoring and Management System for Chronic Disease*. In Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on.

Staab, S., H. P. Schnurr, et al. (2001). *Knowledge Processes and Ontologies*. IEEE Intelligence Systems. 16(1).

Steve, G., A. Gangemi, et al. (1998). *Integrating Medical Terminologies with ONIONS Methodology*. Information Modelling and Knowledge Bases VIII (IOS), Press.

Swartout, B., R. Patil, et al. (1997). *Toward distributed use of large-scale ontologies*. In AAAI-97 Spring Symposium series on ontological engineering.

Tadj, C. and G. Ngantchaha (2006). *Context handling in a pervasive computing system framework*. In 3rd International Conference on Mobile Technology, Applications and Systems, Bangkok, Thailand, ACM.

Terrenghi, L., A. Quigley, et al. (2009). *A taxonomy for and analysis of multi-person-display ecosystems*. Personal and Ubiquitous Computing, Springer-Velarg. 13**:** 583-598.

Uschold, M. and M. King (1995). *Towards a Methodology for Bulding Ontologies.* In Workshop on basic Ontological Issues in Knowledge Sharing.

Varady, P., Z. Benyo, et al. (2002). *An Open Architecture Patient Monitoring System Using Standard Technologies*. IEEE Trans. Information Technology in Biomedicine 6(1): 95-98.

Vazquez, J. I., D. López-de-Ipiña, et al. (2006). *SoaM: A Web-powered Architecture for Designing and Deploying Pervasive Semantic Devices*. International Journal of Web Information Systems 2(3-4): 212-224.

Villarreal, V., J. Bravo, et al. (2010). *MoMo: A Framework Proposal for Patient Mobile Monitoring*. In 5th Conference of the Euro-American Association on Telematics and Information Systems. EATIS 2010, Panama, ACM.

Villarreal, V., J. Bravo, et al. (2010). *Towards Ubiquitous Mobile Monitoring for Health-care and Ambient Assisted Living*. In International Workshop on Ambient Assisted Living, 2010, Valencia, Spain, Springer-Velarg.

Villarreal, V., J. Fontecha, et al. (2011). *Using and Applying MobiPattern to Design MoMo Framework Modules*. Ambient Assisted Living. Springer-Velarg. Málaga, Springer-Velarg. 6693.

Villarreal, V., R. H. Lucas, et al. (2009). *Applying ontologies in the development of patient mobile monitoring framework*. In 2nd International Conference on e-Health and Bioengineering - EHB 2009, Constata, Romania, IEEE.

Villarreal, V., G. Urzaiz, et al. (2011). *Monitoring Architecture to collect measurement data and medical patient control through mobile devices*. In 5th International Symposium on Ubiquitous Computing and Ambient Intelligence-UCAmI, Riviera Maya, Mexico.

W3C. (2000). *Resource Description Framework (RDF) Especificación del esquema 1.0*. Last access 2012, from http://www.sidar.org/recur/desdi/traduc/es/rdf/rdfsch.htm.

Wan-Young, C., L. Seung-Chul, et al. (2008). *WSN based mobile u-healthcare system with ECG, blood pressure measurement function*. In Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE.

WellDoc. (2011). *WellDoc Health Platform*. Last access 2012, from http://www.welldoc.com/Products-and-Services/Our-Products.aspx.

Williams, L. G. and C. U. Smith (2002). *PASA: A Method for the Performance Assessment of Software Architecture*. In 3th Workshop on Software Performance, Rome, Italy.

Yoneki, E. and J. Bacon (2003). *Gateway:A message hub with store-and-forward messaging in mobile networks*. In Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on.

Zhang, Y., L. Zhu, et al. (2009). *Key Management and Authentication in Ad-Hoc Network based on Mobile Agent*. Journal of Networks 4(6): 487-494.

ANEXO I

Criptografía. Cifrado de cadenas con DSE. Paquete javax.crypto

Las bases fundamentales de la criptografía, fueron establecidas por Claude Shannon (Shannon, 1948) en un artículo titulado "*A mathematical theory of communication* " donde introduce las bases de la teoría de la información.

Este ejemplo pone en práctica una clase para cifrar y descifrar cadenas que usan DES. La clase es creada con una llave y puede ser usada repetidamente para cifrar y descifrar cadenas que usan aquella llave.

Este paquete proporciona las clases e interfaces para usos criptográficos que ponen en práctica algoritmos para el cifrado, el desciframiento, o el acuerdo clave. Las cifras de corriente son apoyadas así como asimétricas, simétricas y cifras de bloque. Las puestas en práctica de cifra de proveedores diferentes pueden ser integradas usando las clases abstractas del SPI (Interfaz de Proveedor de servicio). Con la clase *SealedObject* un programador puede asegurar un objeto cifrándolo con una cifra. La autenticación puede estar basada en el MAC (Código de Autenticación de Mensaje) como HMAC (Hash MAC, por ejemplo con una función hash en SHA-1).

Un ejemplo que usa esta clase.

```
try {
  // Generate a temporary key. In practice, you would save this key.
   // See also Encrypting with DES Using a Pass Phrase.
   SecretKey key = KeyGenerator.getInstance("DES").generateKey();
   // Create encrypter/decrypter class
   DesEncrypter encrypter = new DesEncrypter(key);
   // Encrypt
   String encrypted = encrypter.encrypt("Don't tell anybody!");
   // Decrypt
   String decrypted = encrypter.decrypt(encrypted);
}
catch (Exception e) {
}
```

Librería javax.crypto

En esta arquitectura se utilizará el paquete para encriptar y desencriptar los datos enviados a través de los servicios web, basándonos en la librería *javax.crypto* para encriptar y desencriptar cadenas basadas en *DSE*. Su implementación es la siguiente:

```
package net.sf.andhsli.hotspotlogin;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
```

```
 import javax.crypto.SecretKey;
 import javax.crypto.spec.SecretKeySpec;
 public class SimpleCrypto {
  public static String encrypt(String seed, String cleartext) throws
Exception {
        byte[] rawKey = getRawKey(seed.getBytes());
                byte[] result = encrypt(rawKey, cleartext.getBytes());
                return toHex(result); }
        public static String decrypt(String seed, String encrypted)
     throws Exception {
                byte[] rawKey = getRawKey(seed.getBytes());
                byte[] enc = toByte(encrypted);
                byte[] result = decrypt(rawKey, enc);
                return new String(result); }
        private static byte[] getRawKey(byte[] seed) throws Exception {
                KeyGenerator kgen = KeyGenerator.getInstance("AES");
                SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
                sr.setSeed(seed);
            kgen.init(128, sr);
            SecretKey skey = kgen.generateKey();
            byte[] raw = skey.getEncoded();
            return raw; }
        private static byte[] encrypt(byte[] raw, byte[] clear) throws
      Exception {
            SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
                Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
            byte[] encrypted = cipher.doFinal(clear);
                return encrypted; }
        private static byte[] decrypt(byte[] raw, byte[] encrypted)
     throws Exception {
            SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
                Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.DECRYPT_MODE, skeySpec);
            byte[] decrypted = cipher.doFinal(encrypted);
                return decrypted; }
        public static String toHex(String txt) {
                return toHex(txt.getBytes()); }
        public static String fromHex(String hex) {
                return new String(toByte(hex)); }
        public static byte[] toByte(String hexString) {
                int len = hexString.length()/2;
                byte[] result = new byte[len];
                for (int i = 0; i < len; i++)
                        result[i] =
   Integer.valueOf(hexString.substring(2*i, 2*i+2), 16).byteValue();
                return result; }
        public static String toHex(byte[] buf) {
                if (buf == null)
                        return "";
                StringBuffer result = new StringBuffer(2*buf.length);
                for (int i = 0; i < buf.length; i++) {
                        appendHex(result, buf[i]); }
                return result.toString(); }
        private final static String HEX = "0123456789ABCDEF";
        private static void appendHex(StringBuffer sb, byte b) {
        sb.append(HEX.charAt((b>>4)&0x0f)).append(HEX.charAt(b&0x0f));
        } }
```

ANEXO II

Codificación para la obtención de medidas de un paciente

Primeramente se procederá a explicar la carga de condiciones en el *Spinner* para su posterior selección por parte del usuario. Dichos datos son cargados de la siguiente manera:

```
private void cargarSpinner(View l1) {
            // TODO Auto-generated method stub
      name_condition = (Spinner) l1.findViewById(R.id.spinnerConditions);
      name_conditions_data = rellenarCondiciones(l1);
      aa = new ArrayAdapter<String>(mContext,
                  android.R.layout.simple_spinner_item,
                  name_conditions_data);
      aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
name_condition.setAdapter(aa);
name_condition.setOnItemSelectedListener(new OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
            int arg2, long arg3) {
                  // TODO Auto-generated method stub
                  id_condicion = name_conditions_data[arg2];
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                  // TODO Auto-generated method stub
            }     }); }
```

Se puede observar que el primer paso en la generación de la interfaz es obtener los datos de la enfermedad a la que se le añadirá la medida. Como se observa se rellena un Spinner con los datos procedentes del método rellenarCondiciones, cuyo código es este:

```
private String[] rellenarCondiciones(View l1) {
            // TODO Auto-generated method stub
      String[] condiciones = null;
      TextView t = ((TextView) l1.findViewById(R.id.unidad_medida));
      t.setText(mDisease.getMedida().get(0));
      condiciones = new String[mDisease.getCondiciones().size()];

      for (int j = 0; j < mDisease.getCondiciones().size(); j++) {
                  condiciones[j] =
                  mDisease.getCondiciones().get(j).getNombre();
      }
      return condiciones; }
```

Viendo el código, podemos observar que las condiciones se rellenan usando las que la variable mDisease contiene, siendo ésta variable de Tipo Disease, que alberga una estructura de datos representando una enfermedad. Para poder tener una mejor idea de la estructura de datos nombra, este es su código:

```
public class Disease {
      private String IDDisease;
      private String Disease_name;
      private List<Condicion> Condiciones;
      private List<String> Medidas;
```

```
	private List<String> dispositivo_biometrico;
	public Disease(String nombre,String IDDisease, boolean activa,
List<String> medida) {
		this.Disease_name = nombre;
		this.IDDisease = IDDisease;
		this.Medidas = medida;
		this.Condiciones = new ArrayList<Condicion>();
		this.dispositivo_biometrico = new ArrayList<String>(); }
	public Disease(String nombre, boolean activa) {
		this.Disease_name = nombre;
		this.Condiciones =  new ArrayList<Condicion>(); }
	public Disease() {
		// TODO Auto-generated constructor stub }
	public void setNombre(String nombre) {
		this.Disease_name = nombre; }
	public String getNombre() {
		return Disease_name; }
	public void setMedida(List<String> medida) {
		this.Medidas = medida; }
	public List<String> getMedida() {
		return Medidas; }
	public void setCondiciones(
			List<Condicion> condiciones) {
		this.Condiciones = condiciones; }
	public List<Condicion> getCondiciones() {
		return Condiciones; }
	public String getId() {
		return IDDisease; }
	public void setId(String id) {
		this.IDDisease = id; }
	public void addCondicion(Condicion condicion) {
		// TODO Auto-generated method stub
		this.Condiciones.add(condicion); }
	public List<String> getDispositivo_biometrico() {
		return dispositivo_biometrico; }
	public void setDispositivo_biometrico(List<String>
dispositivo_biometrico) {
		this.dispositivo_biometrico = dispositivo_biometrico; }
	public boolean equals(Object o){
		Disease d = (Disease) o;
	if(d.getNombre().equals(this.getNombre())&&d.getId().equals(this.getId
())){
		return true;
		}else{
			return false; }
	}
```

Hemos visto cómo se produce la obtención de datos a la hora de añadir alguna medida y cómo se genera la interfaz, veamos ahora cual es el proceso de envío de medidas y cómo afecta a la interfaz específica. A la hora de general la interfaz específica para la adición de medidas, se define un *listener* que estará a la escucha de ciertas interacciones del usuario con la interfaz específica. Una de ellas es la encargada de escuchar las adiciones que se puedan realizar. El código del *listener* definido para tal propósito es el siguiente:

```
private void setListener(final View l1) {
	// TODO Auto-generated method stub
	l1.findViewById(R.id.listo).setOnClickListener(new OnClickListener() {
		@Override
		public void onClick(View arg0) {
```

```
                List<String> resultados = new ArrayList<String>();
                boolean seguir = true;
                for (int i = 0; i < mEditText.size(); i++) {
                      if
(mEditText.get(i).getText().toString().equals("")) {
                            seguir = false;
                            break;
                      } else {
                       resultados.add(mEditText.get(i).getText().toString(
                       )); } }
                visualizarAlarmas(resultados);
                if (seguir == true) {
                       measureAddListener.onMeasureAdd(resultados);
                } } }
```

El código muestra cómo se realiza la lectura de los datos introducidos, y su posterior tratamiento. El pedazo de código *measureAddListener.onMeasure(*resultados) de lo que se encarga es de comunicar al Activity padre que se procede a añadir una medida. El método *visualizarAlarmas*(resultados) viene definido así:

```
private void visualizarAlarmas(List<String> medidas) {
            // TODO Auto-generated method stub
            List<Barrera> l = null;
            for (int i = 0; i < mDisease.getCondiciones().size(); i++) {
            if (mDisease.getCondiciones().get(i).getNombre()
            .equals(id_condicion)) {
          l = mDisease.getCondiciones().get(i).getBarreras();
                        break; } }
            List<Barrera> l2 = new ArrayList<Barrera>();
            List<String> barrera = new ArrayList<String>();
            if (l != null) {
                  if (l.size() > 0) {
                        if (l2.size() < 1) {
                              l2 = ordenarLista(l); }
                        for (int j = 0; j < medidas.size(); j++) {
                              for (int i = 0; i < l2.size(); i++) {
                                    if (Integer.parseInt(medidas.get(0)) <
                                    l2.get(i).getMedida()) {
                                          barrera.add(l2.get(i).getNombre
                                          ());
                                          break;
                                    }  }  }  }   }
                  gestionarMedida(barrera, medidas); }
```

En este fragmento se contemplan las barreras que son alcanzadas por las distintas medidas introducidas, y se procede a su gestión que consistirá en la visualización de mensajes de alarma e introducción en la base de datos vía Internet. Este es el código para tal propósito:

```
private void gestionarMedida(List<String> barrera,List<String> medida) {
      // TODO Auto-generated method stub
      List<String> nivel_advertencia = new ArrayList<String>();
      //if (enfermedad_id.equals("12")) {
            for (int i = 0; i < barrera.size(); i++) {
                  if (barrera.get(i).equals("Medio")) {
                        nivel_advertencia.add("medio");
                  } else {
                        if (barrera.get(i).equals("Bajo") ||
                  barrera.get(i).equals("Alto")) {
                              if (barrera.get(i).equals("Bajo")) {
```

```
                                        nivel_advertencia.add("bajo");
                                  } else {
                                        // time = 30000000;
                                        nivel_advertencia.add("alto"); }
                                   } else if (barrera.equals("Por encima de lo
                              normal")) {
                                        nivel_advertencia.add("muy alto"); } }
                        }
                  Intent i = new Intent(mContext, ShowAlarmActivity.class);
                        i.putExtra("enfermedad_id", enfermedad_id);
                        i.putExtra("id_condicion", id_condicion);
                        i.putExtra("n_nivel_advertencia",barrera.size() );
                        for(int j = 0 ; j < barrera.size();j++){
                              i.putExtra("medida_"+j, medida.get(j));
                              i.putExtra("barrera_"+j, barrera.get(j));
                              i.putExtra("nivel_advertencia_"+j,
nivel_advertencia.get(j)); }
                        i.putExtra("nombre enfermedad", mDisease.getNombre());
                        startActivityForResult(i, ADD_MEASURE); }
```

Lo que hace este fragmento de código es establecer qué barreras son las que cumple las medidas introducidas y adecuarlas para su correcto uso en otra Activity, ShowAlarmActivity, que se muestra a continuación:

```
public class ShowAlarmActivity extends Activity {
      private Measure m;
      String enfermedad_id, medida, id_condicion , nivel_advertencia,
      barrera,nombre_enfermedad;
      List<String> list_barreras;
      List<String> list_medidas;
      List<String> list_nivel_advertencia;
      int n_barreras;
      Date d;
      long time = 0;
      Disease enfermedad;
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setContentView(R.layout.alarm_dialog);
            list_medidas = new ArrayList<String>();
            list_barreras = new ArrayList<String>();
            list_nivel_advertencia = new ArrayList<String>();
            Bundle bundle = getIntent().getExtras();
            enfermedad_id = bundle.getString("enfermedad_id");
            n_barreras = bundle.getInt("n_nivel_advertencia");
            for (int i = 0; i < n_barreras; i++) {
                  list_barreras.add(bundle.getString("barrera_" + i));
                  list_medidas.add(bundle.getString("medida_" + i));
      list_nivel_advertencia.add(bundle.getString("nivel_advertencia_"
                              + i));
                  if (i == 0) {
                        medida = list_medidas.get(i);
                  } else {
                        medida = medida + "-" + list_medidas.get(i);
                  }
            }
            // medida = bundle.getString("medida");
            // barrera = bundle.getString("barrera");
```

```
            id_condicion = bundle.getString("id_condicion");
            nombre_enfermedad = bundle.getString("nombre enfermedad");
            // nivel_advertencia = bundle.getString("nivel advertencia");
            Calendar c = Calendar.getInstance();
            Date d = c.getTime();
            enfermedad =
            DataUtils.getInstance(getApplicationContext()).getDisease(
                        enfermedad_id);
            m = new Measure(enfermedad_id,
            DataUtils.getInstance(this).mPatient.getIdPatient(),
            medida,this.id_condicion);
            DataUtils.getInstance(this).addLastMeasure(m);
            ((TextView) findViewById(R.id.textView2)).setText(getResources()
                        .getString(R.string.measure)
                        + medida
                        + " "
                        + enfermedad.getMedida().get(0));
            mostrarImagenesSegunNivelAdvertencia();
            findViewById(R.id.button1).setOnClickListener(new
OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        // TODO Auto-generated method stub
                        int res =
Integer.parseInt(WebServicesConexionManager
                                    .getInstance().addMeasure(m));
      DataUtils.getInstance(getApplicationContext()).addMeasure(m);
                        if (res != -1) {
                              boolean riesgo = false;
                              for (int i = 0; i <
                              list_nivel_advertencia.size(); i++) {
                                    if
                                    (!list_nivel_advertencia.get(i).equals
                                    ("medio")) {
                                          riesgo = true;
                                          break; } }
                              if (riesgo == true) {
                                    mostrarDialogoSegunNivelAdvertencia();
                              } else {
                                    setResult(RESULT_OK);
                                    finish(); }
                        } else {
                              Toast.makeText(getApplicationContext(),
                              getResources().getString(R.string.addValorEr
                              ror),
                              Toast.LENGTH_LONG).show();
                              setResult(RESULT_CANCELED);
                              finish(); } } }
      protected void mostrarDialogoSegunNivelAdvertencia() {
            // TODO Auto-generated method stub
            for( int i = 0 ; i < list_nivel_advertencia.size();i++){
                  final int j = i;
                  if(!list_nivel_advertencia.get(i).equals("medio")){
                        AlertDialog.Builder builder = new
                        AlertDialog.Builder(this);
                        builder.setMessage(
                                    "Se mostrará una alarma-recordatorio
                                    para que se vuelva a tomar una nueva
                                    medida de "
```

```
                                                + nombre_enfermedad
                                                + " en "
                                                + (time / 1000)
                                                / 60
                                                + " minutos")
                                        .setCancelable(false)
                                        setPositiveButton(getResources().getSt
                                        ring(R.string.ok),new
                                        DialogInterface.OnClickListener() {
                              public void onClick(DialogInterface dialog,int id)
                        {
                                    Alarm a = new Alarm("Glucosa: " +
                                    list_barreras.get(j), medida, time,
                                    ShowAlarmActivity.this);
                                    DataUtils.list_alarm.add(a);
                                    Toast.makeText(
                                          ShowAlarmActivity.this,
                                          getResources().getString(
                                                R.string.addValorOk),
                                                Toast.LENGTH_LONG).show();
                                                SmsManager sms =
                                                SmsManager.getDefault();
                                    ShowAlarmActivity.this.setResult(RESULT_OK);
                                    ShowAlarmActivity.this.finish();
                                                      }});
                              AlertDialog alert = builder.create();
                              alert.show();
                        } } }
private void mostrarImagenesSegunNivelAdvertencia() {
            // TODO Auto-generated method stub
                  for (int i = 0; i < this.list_nivel_advertencia.size();
                  i++) {
                  if (list_nivel_advertencia.get(i).equals("medio")) {
                        ((ImageView) findViewById(R.id.imageView1))
                              .setImageResource(R.drawable.ok_icon_md);
                        ((TextView) findViewById(R.id.textView1))
                                    .setText("Está a punto de añadir una
                                    nueva medida, ¿Está seguro?");
                  } else {
            if (list_nivel_advertencia.get(i).equals("alto")
            || list_nivel_advertencia.get(i).equals("bajo")) {
                  time = 900000;
                  ((ImageView) findViewById(R.id.imageView1))
                               .setImageResource(R.drawable.warning_icon);
                              ((TextView) findViewById(R.id.textView1))
                                          .setText("Está a punto de
av±adir una medida considerada de riesgo , ¬øEstv° seguro?");
                        } else {
                              ((ImageView) findViewById(R.id.imageView1))
                              .setImageResource(R.drawable.danger_icon);
                              time = 1800000;
                              ((TextView) findViewById(R.id.textView1))
                                          .setText("Está a punto de
av±adir una medida considerada de alto riesgo, ¬øEstv° seguro?");
                        } }  }  }  }
```

Este código lo que hace es mostrar al usuario una advertencia si las medidas introducidas superan algún umbral de riesgo, y proceder al envío de las medidas mediante el método

addMeasure(m : Measure) de la clase DataUtils. Dicho método realiza la inserción tanto de manera local como a la base de datos alojada en la red de la siguiente manera:

```
public int addMeasure(Measure m) {
            int resul = 0;
            if (ConnectivityUtils.isOnline(mContext)) {
                  resul =
Integer.parseInt(WebServicesConexionManager.getInstance()
                              .addMeasure(m));
            }
            SQliteDBAdapter.getInstance().open(mContext);
            SQliteDBAdapter.getInstance().insertMeasure(m);
            SQliteDBAdapter.getInstance().close();
      return resul;
}
```

Los métodos addMeasure de la clase WebServiceConexionManager y el método insertMeasure de la clase SQliteDBAdapter se definen a continuación:

```
// Inserción de medidas a través de la red (base de datos remota)
public String addMeasure(Measure m) {
            METHOD = "addMeasure";
            SoapObject request = new SoapObject(NAMESPACE, METHOD);
            PropertyInfo pi1 = new PropertyInfo();
            pi1.setName("MeasureId");
            pi1.setValue(m.getIDMeasure());
            request.addProperty(pi1);
            pi1 = new PropertyInfo();
            pi1.setName("IDpatient");
            pi1.setValue(m.getIDpatient());
            request.addProperty(pi1);
            pi1 = new PropertyInfo();
            pi1.setName("Medida");
            pi1.setValue(m.getMedida());
            request.addProperty(pi1);
            pi1 = new PropertyInfo();
            pi1.setName("Time");
            pi1.setValue(m.getTime());
            request.addProperty(pi1);
            pi1 = new PropertyInfo();
            pi1.setName("IDCondicion");
            pi1.setValue(m.getCondicion());
            request.addProperty(pi1);
            SoapSerializationEnvelope envelope = new
            SoapSerializationEnvelope(
            SoapEnvelope.VER12);
            envelope.dotNet = true;
            envelope.setOutputSoapObject(request);
            HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
            androidHttpTransport.debug = true;
            SOAPACTION = NAMESPACE + METHOD;
            try {
                  androidHttpTransport.call(SOAPACTION, envelope);
                  Object response = (Object) envelope.getResponse();
            return  response.toString();
            } catch (Exception e) {
                  Log.d("error", e.getMessage().toString());
                  String a = e.getMessage().toString();
                  return "-1"; }  }
```

```
// Inserción de medidas en el dispositivo móvil (base de datos local)
public int insertMeasure(Measure m) {
            ContentValues cv = new ContentValues();
            cv.put(MEASURE, m.getIDMeasure());
            cv.put(IDPATIENT, m.getIDpatient());
            cv.put(MEDIDA, m.getMedida());
            cv.put(TIME, m.getTime());
            cv.put(SQliteDB.IDCONDICION, m.getCondicion());
            try {
            database = dbHelper.getWritableDatabase();
            return (int) database.insert(MEASURES_TABLE, null, cv);
            } catch (Exception e) {
            return -1; } }
```

El primero de ellos supone la inserción de la medida a través de Internet hasta la base de datos alojada en la red. En el segundo, se procede a la introducción en la base de datos local. En ambos casos la tabla que define la introducción de medidas es la siguiente:

Nombre	Iddisease	Idpatient	Medida	Time	IDCondicion
Tipo	Varchar(5)	Varchar(12)	text	text	text

ANEXO III

Codificación para la selección de un tipo de ejercicio a desarrollar por un paciente

Como resumen, el procedimiento que se sigue seleccionar los distintos tipos de ejercicios que se encuentran en la base de datos local y mostrarlos para una posterior elección. Una vez explicado y añadido el código de la obtención de los tipos de actividades posibles, debemos explicar un detalle que se encuentra en el primer fragmento de código cuando explicábamos la obtención de éstos datos.

En este fragmento podemos observar cómo se implementa un *listener* para el *Spinner* que se carga, el cual hace que se rellene una pequeña lista con las actividades que pertenecen al grupo escogido. Dicha selección hace uso de la base de datos local, por lo que explicaremos insertando el código los pasos que realiza:

```
String tipo_ejercicio_name = tipos_ejercicio[arg2];
    ListView listView = (ListView) l1.findViewById(R.id.list1);
    lv_arr = DataUtils.getInstance(mContext)
            .getActivitiesByType(tipo_ejercicio_name);
SelectedAdapter s = new SelectedAdapter(mContext, 0, lv_arr);
         listView.setAdapter(s);
```

El código de la clase *DataUtils* de ésta parte es:

```
public List<String> getActivitiesByType(String tipo_ejercicio_name) {
      List<String> array;
     SQliteDBAdapter.getInstance().open(mContext);
     array = SQliteDBAdapter.getInstance().obtainActivityNameByType(
     tipo_ejercicio_name);
     SQliteDBAdapter.getInstance().close();
     return array;
         }
```

Y el código de la clase SQliteDBAdapter para proporcionar éstos datos es:

```
public List<String> obtainActivityNameByType(String tipo_ejercicio_name) {
         // TODO Auto-generated method stub
         database = dbHelper.getReadableDatabase();
         Cursor c = database.rawQuery(
         "Select nombre from ejercicio where tipoejercicio = ? ;",new
         String[] { tipo_ejercicio_name });
        return parserToListActivityName(c);
         }
private List<String> parserToListActivityName(Cursor c) {
         // TODO Auto-generated method stub
         List<String> res = new ArrayList<String>();
         while (c.moveToNext()) {
             if (!res.contains(c.getString(c
                         .getColumnIndex("nombre")))) {
                 res.add(c.getString(c.getColumnIndex("nombre")));
             }  }
         return res; }
```

Nos encontramos por tanto ante accesos a los mismos datos de la base de datos, con la diferencia que en este último acceso seleccionamos el nombre de los ejercicios cuyo tipo coincide con el que se busca.

Podemos observar en el código que representa el acceso a más bajo nivel de los datos que necesitamos, como nos encontramos con una tabla denominada ejercicio que tendría la siguiente estructura:

Nombredato	tipoejercicio	duración	nombre	iddisease
Tipo dato	TEXT	TEXT	TEXT	VARCHAR(10)

ANEXO IV

Elementos de la cuadrícula del patrón de menú.

Todos los elementos del menú están definidos por la siguiente estructura *XML*, lo que permite una generalización al momento de querer implementar nuevos elementos, bien definidos estructuralmente

```
<LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_below="@id/label_header"
        android:layout_weight="1"
        android:layout_above="@id/label_bottom"
        android:orientation="horizontal" >
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:orientation="vertical" >
         <RelativeLayout
                android:id="@+id/new_measure"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1"
            android:background="@drawable/background_selector" >
                <LinearLayout
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:layout_centerInParent="true"
                    android:orientation="vertical" >
                    <ImageView
                        android:id="@+id/ImageAddMeasure"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:background="@null"
                        android:src="@drawable/new_measure" />
                    <TextView
                        android:layout_width="fill_parent"
                        android:layout_height="wrap_content"
                        android:layout_below="@id/ImageB1"
                        android:layout_centerHorizontal="true"
                        android:layout_marginTop="8dip"
                        android:gravity="center"
                        android:text="Nueva medida"
                        android:textColor="@color/black"
                        android:textSize="18dip" />
                </LinearLayout>
            </RelativeLayout>
            <RelativeLayout
                android:id="@+id/button1"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1"
```

```
        android:background="@drawable/background_selector"
           ><LinearLayout
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_centerInParent="true"
              android:orientation="vertical" >
              <ImageView
                android:id="@+id/ImageB1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:background="@null"
                android:src="@drawable/ic_contact_picture"
                  />
               <TextView
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_below="@id/ImageB1"
                  android:layout_centerHorizontal="true"
                  android:layout_marginTop="8dip"
                  android:gravity="center"
                  android:text="@string/patientprofile"
                  android:textColor="@color/black"
                  android:textSize="18dip" />
           </LinearLayout>
      </RelativeLayout>
      <RelativeLayout
          android:id="@+id/button3"
          android:layout_width="fill_parent"
          android:layout_height="fill_parent"
          android:layout_weight="1" >
          <LinearLayout
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_centerInParent="true"
              android:orientation="vertical" >
              <ImageView
                  android:id="@+id/ImageB3"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_gravity="center"
                  android:background="@null"
                  android:src="@drawable/grafica_blue" />
              <TextView
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_below="@id/ImageB3"
                  android:layout_centerHorizontal="true"
                  android:layout_centerInParent="true"
                  android:layout_marginTop="8dip"
                  android:gravity="center"
                  android:text="@string/autocontrol"
                  android:textColor="@color/black"
                  android:textSize="18dip" />
          </LinearLayout>
      </RelativeLayout>
  </LinearLayout>
  <LinearLayout
      android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
android:layout_weight="1"
android:orientation="vertical" >
<RelativeLayout
 android:id="@+id/new_activity"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:layout_weight="1"
 android:background="@drawable/background_selector"
 ><LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:orientation="vertical" >
        <ImageView
            android:id="@+id/ImageB5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:background="@null"
           android:src="@drawable/fisical_activity"
           /> <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/ImageB5"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="8dip"
            android:gravity="center"
            android:text="Nueva actividad física"
            android:textColor="@color/black"
            android:textSize="18dip" />
    </LinearLayout>
</RelativeLayout>
<RelativeLayout
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
  android:background="@drawable/background_selector"
    > <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:orientation="vertical" >
        <ImageView
            android:id="@+id/ImageB5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:background="@null"
            android:src="@drawable/nueva_enfermedad"
            /> <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/ImageB5"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="8dip"
            android:gravity="center"
            android:text="@string/addenfermedad"
```

```
                    android:textColor="@color/black"
                    android:textSize="18dip" />
            </LinearLayout>
        </RelativeLayout>
        <RelativeLayout
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1" >
            <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_centerInParent="true"
                android:orientation="vertical" >
                <ImageView
                    android:id="@+id/ImageB2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center"
                    android:background="@null"
                    android:src="@drawable/sync" />
                <TextView
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:layout_below="@id/ImageB2"
                    android:layout_gravity="center"
                    android:layout_marginTop="8dip"
                    android:gravity="center"
                    android:text="@string/sync"
                    android:textColor="@color/black"
                    android:textSize="18dip" />
            </LinearLayout>
        </RelativeLayout>
    </LinearLayout>
</LinearLayout>
```

ANEXO V

Modelo de evaluación de la facilidad de uso según el método SALUTA.

Arquitectura: *Framework* para la Monitorización Móvil de Pacientes, a través de dispositivos móviles y biométricos.

Usuario: *Usuario Final (UF)*

Tareas a evaluar

Tarea 1 (T1): Editar un perfil de paciente previamente agregado.
Tarea 2 (T2): Visualizar los gráficos de autocontrol de una enfermedad en particular.
Tarea 3 (T3): Seleccionar un ejercicio a desarrollar por un paciente.
Tarea 4 (T4): Modificar el idioma de la aplicación.
Tarea 5 (T5): Agregar una nueva medida de una enfermedad
Tarea 6 (T6): Revisar las notificaciones y alarmas generadas después de obtenida una medida.

Contextos

Contexto 1 (C1): Un ambiente de entrenamiento para pacientes con diabetes que necesitan monitorizar su enfermedad, basándose en las tareas establecidas.
Contexto 2 (C2): Un ambiente de entrenamiento para pacientes con problema de tensión arterial que necesitan monitorizar su enfermedad, basándose en las tareas establecidas.

Atributos

T1	C1	Facilidad de Aprendizaje	Eficiencia de uso	Confiabilidad	Satisfacción
T1	C1				
T2	C1				
T3	C1				
T4	C1				
T5	C1				
T6	C1				
T1	C2				
T2	C2				
T3	C2				
T4	C2				
T5	C2				
T6	C2				

Para cada atributo a evaluar, se ponderará la facilidad de uso con una escala de 1 a 5, en donde **5**= *alta facilidad de uso*, **4**=*buena facilidad de uso*, **3**=*Mediana facilidad de uso*, **2**=*baja facilidad de uso* y **1**=*pobre facilidad de uso*.

ANEXO VI

Modelo de evaluación de la aplicación por parte de los usuarios.

Fecha: _____ **Edad**: _____
Enfermedad a evaluar: _____

Le agradezco su disposición para participar en esta evaluación que nos ayudará a detecta problemas de la arquitectura que hemos desarrollado, si es que los tuviera. Pretendemos evaluar la funcionalidad de la arquitectura para la monitorización móvil de paciente, a través del dispositivo móvil (en este caso el móvil) el cual obtendrá la lectura de señales vitales desde dispositivos biométricos (de medición). El tiempo establecido para contestar el cuestionario será de 30 minutos. El rango de respuesta estará definido en una escala de 1 a 5, siendo 1 el valor más bajo (Muy en desacuerdo) para un criterio y 5 el más alto (muy de acuerdo).

Preguntas

Preguntas clasificadas por categorías	**Valoración**				
Categoría 1. Contenido	**1**	**2**	**3**	**4**	**5**
1. ¿Considera que la organización de los elementos dentro de la aplicación es buena?					
2. ¿Considera que los elementos gráficos y de texto le han ayudado a entender más claramente la aplicación?					
3. ¿Considera que la aplicación muestra todo la información necesaria para entender el funcionamiento de la interfaz?					
4. ¿Considera que la ayuda que ofrece la aplicación es suficiente para aclarar algunas dudas?					
5. ¿Considera que la aplicación ofrece una facilidad de navegación?					
6. ¿Encuentra fácilmente los mensajes generados dentro de la aplicación?					
7. Entre pantallas ¿Sabe en donde se encuentra en cada momento?					
8. ¿La calidad información mostrada en cada mensaje es adecuada?					
Categoría 2. Diseño	**1**	**2**	**3**	**4**	**5**
9. ¿Considera que el diseño de las pantallas son entendibles?					
10. ¿Considera que la organización del menú es entendible?					
11. ¿Considera que los mensajes generados por la aplicación son legibles?					
12. ¿Las funciones de las interfaces te resultaron sencillas de usar?					
Categoría 3. Utilidad	**1**	**2**	**3**	**4**	**5**
13. ¿Considera que las recomendaciones generadas son adecuadas?					
14. ¿Considera que los mensajes de educación asociados a la enfermedad son correctos?					
15. ¿Considera que el sistema le ofrece las respuestas que necesita?					
16. ¿Considera que los mensajes generados son fáciles de entender?					
17. ¿Considera que los mensajes generados son fáciles de recordar?					
18. ¿Considera que el sistema es de gran utilidad para el seguimiento de su enfermedad?					
19. ¿Considera que la aplicación llenó todas sus expectativas funcionales?					

ACERCA DEL AUTOR

De nacionalidad panameña, nacido el 9 de noviembre de 1979. Obtiene el título de Licenciado en Tecnologías de Programación y Análisis de Sistemas de la Universidad Tecnológica de Panamá, Centro Regional de Chiriquí, en el 2002.

En el 2004, obtiene la idoneidad por parte de la Junta Técnica de Ingenieros y Arquitectos de Panamá. Más adelante en el 2005, obtiene el Título de Posgrado en Docencia Superior de la Universidad Autónoma de Chiriquí. En el 2005 y 2006 cursa estudios de inglés, en el St. Clair College, en Windsor Ontario, Canadá, donde obtiene la certificación de **English Language Institute: Immersion I and Immersion II**. En el 2006 inicia el Posgrado en Informática Educativa, en la Universidad Tecnológica de Panamá.

En el 2007 forma parte del Grupo Modelling Ambient Intelligence (MAmI) de la UCLM, participando como investigador en diferentes áreas en torno a temas en ambientes inteligentes, lo que le lleva a desarrollar su tesis doctoral titulada: "***Framework* para la Monitorización Móvil de Pacientes**". En el 2009 obtiene el **Máster en Tecnologías Informáticas Avanzadas** en la Universidad Castilla-La Mancha, España, en el área de Computación Ubicua, Ambientes Inteligentes, Generación de Aplicaciones y entornos médicos dentro del grupo MAmI.

En el 2000, inicia labores en la Universidad Tecnológica de Panamá, Centro Regional de Chiriquí, como Instructor de Seminarios de actualización y capacitación continua a estudiantes y profesionales del área privada y gubernamental. En el periodo entre 1999 y 2002, funge como Especialista en Computadoras de los Laboratorios de Informática de la **Facultad de Sistemas de Ingeniería de Sistemas Computacionales,** en el Centro Regional de Chiriquí (UTP). Del año 2004 al 2006, fue Coordinador del primer Laboratorio adscrito a la **Vicerrectoría de Investigación, Posgrado y Extensión**, instalado en el Centro Regional de Chiriquí (UTP). En el 2005, inicia sus labores como profesor de la **Universidad del Istmo**, sede de David en la carrera de ***Administración de Sistemas***. Para el 2006, continúa sus labores docentes a nivel superior como Profesor Tiempo Parcial de la FISC del Centro Regional de Chiriquí (UTP). Del 2006 al 2007, es asignado como **Coordinador del Sistema de Ingreso Universitario** del **Centro Regional de Chiriquí** (UTP). Actualmente es ***Profesor a Tiempo Completo*** de la **Facultad de Ingeniería de Sistemas Computacionales** en el **Centro Regional de Chiriquí** (UTP).

El autor ha sido financiado por la **Fundación Carolina**, **Secretaría Nacional de Ciencia y Tecnología (SENACYT)** y el **Instituto para la Formación y Aprovechamiento de los Recursos Humanos (IFARHU)** para desarrollar su tesis doctoral en estos últimos años.

vladimir.villarreal@utp.ac.pa
http://vladvill.vvcdesigntech.com