Desarrollo de una aplicación educativa basada en Android para dispositivos móviles

Development of an Android-based educational application for mobile devices

Gisela T. de Clunie
Fac. de Ingeniería de Sistemas Computacionales
Universidad Tecnológica de Panamá
gisela.clunie@utp.ac.pa

Sérgio Crespo
Programa de Maestría en Computación Aplicada- PIPCA
UNISINOS, Brasil
crespo@unisinos.br

Lucas Monteiro Braz
Programa de Maestría en Computación Aplicada- PIPCA
UNISINOS, Brasil
lmonteirobraz@gmail.com

Tássia Serrão
Programa de Maestría en Computación Aplicada- PIPCA
UNISINOS, Brasil
tassiacomp@gmail.com

Norman Rangel
CIDITIC
Universidad Tecnológica de Panamá
norman.rangel@utp.ac.pa

Aris Castillo
Fac. de Ingeniería de Sistemas Computacionales
Universidad Tecnológica de Panamá
aris.castillo@utp.ac.pa

Boris Gómez
CIDITIC
Universidad Tecnológica de Panamá
boris.gomez@utp.ac.pa

Kexy Rodríguez
CIDITIC
Universidad Tecnológica de Panamá
kexy.rodriguez@utp.ac.pa

Olinda de Barraza
Fac. de Ingeniería de Sistemas Computacionales
Universidad Tecnológica de Panamá
olinda.velarde@utp.ac.pa

Jeannette Riley
Fac. de Ingeniería de Sistemas Computacionales
Universidad Tecnológica de Panamá
jeanette.riley@utp.ac.pa

RESUMEN

Este artículo presenta el desarrollo de MLEA, una plataforma que facilita, a través del uso de celulares y *tablets* con sistema operativo Android, la movilidad de usuarios de ambientes virtuales de aprendizaje. Es una aplicación que utiliza técnicas computacionales, como "*web services*", patrones de diseño, ontologías y tecnologías de computación móvil, para permitir la comunicación entre los dispositivos móviles y el sistema administrador de contenidos - Moodle. Está basada en una arquitectura cliente servidor orientada a servicios en que se combinan el protocolo REST y el formato JSON para el intercambio de datos. El cliente contará con facilidades tales como alertas, bajar archivos, participar en foros y chats, ver calificaciones, realizar ejercicios cortos y ver el calendario, entre otras.

ABSTRACT

This paper is about the development of MLEA, a platform that assists, through Android cellphones and tablets, the mobility of users of learning virtual environments. MLEA is an application that implements computational techniques such as web services, design patterns, ontologies, and mobile computational techniques in order to allow the communication between mobile devices and the content management system – Moodle. It´s based on a service oriented, client server architecture that combines the REST protocol and JSON format for data interchange. The client will be provided with features such as alerts, file downloads, chats and forums, grade books, quizzes, and calendar, among others.

Categorías y Temas

3. [**Computación Móvil y Ubicua**]: *Acceso a Internet y Aplicaciones, Aprendizaje móvil* 4. [**Tecnologías Web y Sociedad**]: *Accesibilidad de la Web, Aprendizaje Electrónico* .

General Terms
Desempeño, Diseño, Experimentación

Palabras Clave
Aprendizaje móvil, m-learning, REST, JSON, web services, computación móvil, interoperabilidad, patrones de diseño, SOA.

Keywords
Mobile learning, m-learning, REST, JSON, web services, mobile computing, interoperability, design patterns, SOA.

1. INTRODUCCIÓN
Cada día aumenta más el número teléfonos celulares adquiridos por los ciudadanos comunes. Esto lleva a que todos queramos tener a mano, en un dispositivo conectado a Internet, desde una agenda con nuestros contactos hasta el acceso a nuestras actividades diarias y, entre ellas, las formativas. Hoy podemos decir que los dispositivos móviles inteligentes están cambiando la forma como trabajamos, estudiamos y actuamos [1].

Tanto el e-learning o "aprendizaje electrónico" [2], como el "m-learning o "aprendizaje móvil" [3], [4], se refieren a la utilización de dispositivos móviles y aplicaciones que permitan el uso de los mismos en entornos educativos a distancia o bien para facilitar el acceso de los estudiantes a recursos educativos sin necesidad de estar físicamente en un mismo lugar. Ambos involucran la integración de diversas tecnologías; por un lado, están los protocolos asociados con la plataforma de educación a distancia y, por el otro, los de los dispositivos móviles correspondientes.

El proyecto MLEA (*Mobile Learning Environment Adapter*), desarrollado conjuntamente entre la Universidad Tecnológica de Panamá y la UNISINOS-Brasil, tiene el objetivo de brindar a los estudiantes y profesores los beneficios del aprendizaje móvil, a partir de la construcción de una arquitectura modular, móvil, basada en *Service Oriented Architecture* (SOA), que integra el ambiente educativo Moodle a celulares de tipo *smartphone* y *tablets* con sistema operativo Android. El proyecto desarrolla toda la infraestructura tanto del lado del servidor como de la aplicación cliente que será utilizada por los estudiantes.

MLEA brinda respuesta a las necesidades de formación de aquellas personas que no pueden asistir regularmente a cursos en aulas presenciales. Esta solución constituye un apoyo a la educación a distancia y, a la vez, un soporte a la educación presencial [5], siempre y cuando se utilicen herramientas de educación a distancia.

MLEA resulta un aporte valioso, ya que se trata de una aplicación gratuita para los usuarios de dispositivos Android, dado que a la fecha no existe una herramienta personalizada móvil, totalmente funcional, que les permita acceder a Moodle de una forma transparente y eficiente. Con las herramientas alternativas actuales que se ofrecen para dispositivos móviles, el usuario puede acceder al servidor, pero lo hace con formatos (tamaño y forma de la pantalla), que no están adaptados para el tipo de dispositivo; razón por la cual, en algunos casos, la experiencia puede resultar perturbadora.

Un aspecto importante a destacar es que la arquitectura desarrollada hace posible que, en el futuro, se puedan desarrollar aplicaciones clientes para diferentes sistemas operativos móviles, como por ejemplo iPhone, aprovechando gran parte de la infraestructura construida y, de esta forma, se pueda alcanzar un número mayor de usuarios.

Iniciamos el artículo con una breve discusión de los trabajos relacionados encontrados en la literatura, que guardan mayor afinidad con el proyecto presentado. En la sección 3 se discute la arquitectura de MLEA, presentando aspectos propios de su funcionamiento, tales como el flujo de ejecución, la vista del lado cliente y la vista del lado servidor. Seguidamente, se discute la implementación de los *web services*, se presenta su especificación y, a manera de ejemplo, una porción de código representativa. La sección 5, presenta el modelo de integración de los *web services*. A continuación, la sección 6, presenta los resultados esperados a partir del desarrollo del proyecto y su puesta en ejecución. La sección 7, presenta las conclusiones y trabajos futuros. Finalmente, presentamos nuestros agradecimientos y literatura referenciada en el trabajo.

2. TRABAJOS RELACIONADOS
La literatura técnica reporta algunos trabajos relativos a comunicaciones móviles y distribución de servicios. Ibrahim y Zhao [6], proponen un modelo conceptual de "framework" para comunicación en dispositivos móviles que da soporte a la movilidad y distribución de servicios sobre múltiples dispositivos. Muchos de los principios de la computación móvil, entre ellos las bases de la comunicación cliente-servidor, resultan un referente para nuestro proyecto; pero a diferencia de MLEA, aquel modelo está diseñado para Java Virtual Machines (JVM). La primera propuesta formal por parte de Moodle (*Module Object-Oriented Dynamic Learning Environment*) presenta algunas capturas de pantalla para el desarrollo de una aplicación de Moodle 2.0 en teléfonos iPhone y Android. Incluye ocho características: la carga de archivos, visualización de la información del curso para estudiantes y profesores, la asistencia a cursos, el envío y recepción de mensajes, calendario de actividades y encuesta. Esta resulta la primera propuesta formal por parte de Moodle para los desarrolladores. Como se mencionó anteriormente, este proyecto se encuentra aún en fase de propuesta, y sólo hay algunos diseños de pantalla [7].

Mientras tanto, los miembros de Moodbile y Moodle4iPhone han anunciado la colaboración en un proyecto denominado Moodle para móviles (M2M) que tiene como objetivo crear un plugin para Moodle con el fin de permitir cualquier tipo de dispositivo móvil para acceder a los servicios. Sin embargo, no se ha publicado progreso hasta ahora. No se trata de una aplicación para el dispositivo móvil, sino para el servidor de Moodle [8]. Existen algunos otros proyectos dirigidos a atender el mismo asunto. Softwarelivre es un proyecto en Portugal con miras a crear una aplicación que permitirá que los dispositivos Android puedan acceder a los recursos en un servidor de Moodle. Sin embargo, debido al uso del Protocolo *Simple Access Oriented Protocol* (SOAP), la aplicación resulta más pesada en comparación con *Representational State Transfer* (REST), el protocolo utilizado en MLEA, para los *web services* [9].

MBOT es una aplicación gratuita para Android originales. Le permite al usuario almacenar los datos de sesión y recordar las actividades como: las clases visitadas, las tareas realizadas y foros.

También permite al usuario añadir sus compañeros de clase, contactos de Google y, finalmente, abrir y guardar archivos de Word, Power Point y PDF, si el visor está instalado. El mayor inconveniente es que cuando el usuario hace clic en cualquier aplicación o actividad, se abre a una página web que no tiene el formato de un dispositivo móvil; en otras palabras, como si el usuario hubiera accedido a ellos a través del navegador web de Android [10]. MLEA es una aplicación dedicada que no existe para la plataforma Android, guarda los datos de acceso, y el acceso a los servicios es expedito. Mientras que, con un navegador, los datos se deben introducir cada vez que se ingresa al ambiente.

Mobile Learning Engine (MLE) Moodle es una aplicación libre, que inicialmente fue desarrollada para correr en dispositivos bajo plataforma Java 2, Micro Edition (J2ME), Sin embargo, existe una versión para Android versión 1.0, que no funciona en los teléfonos actuales. Este sistema integrado para el aprendizaje desde teléfonos móviles, está diseñado como un plug-in para el LMS-Moodle (Learning Management System). Transfiere el estudio automático basado en aprendizaje multimedia (e-learning) a un ambiente móvil (a través de un teléfono móvil). El usuario también puede acceder a la plataforma desde el teléfono móvil, puede usar su propio teléfono para acceder a una versión formateada MLE-Moodle. Para que esta aplicación funcione correctamente, el usuario debe configurar el servidor de Moodle [11].

MoodleTouch (mTouch) se ha desarrollado para el sistema operativo del iPhone (IOS), pero el equipo de desarrollo está trabajando en una versión alternativa de Android. La interfaz de la aplicación está bien desarrollada y es muy intuitiva. Todos los recursos y las actividades están formateados para la visualización en un dispositivo móvil. Esta aplicación no requiere ninguna configuración en el servidor de Moodle para trabajar, por lo que es compatible con la versión 2.0 de Moodle. Mientras tanto, ésta no es una aplicación gratuita [12]. MLEA es una solución gratuita para los usuarios con teléfonos celulares de tipo *smartphone* y *tablet*, con sistema operativo Android.

En el caso de ontologías aplicadas a la computación ubicua [13], discuten algunas, entre ellas CoBrA (SOUPA), Gaia, GLOSS, ASC (CoOL), SOCAM (CONON), GAS y CoDAMoS que nos sirven de referencia. A pesar que en su estudio no incluyen SOA, la ontología a ser aplicada en nuestro proyecto, podemos identificar algunas características y relaciones comunes que permiten la construcción y rápido prototipaje de servicios de alertas al contexto (*context-aware services*) en ambientes de computación móvil. SOA [14] es una arquitectura orientada a servicios que favorece el desarrollo rápido de procesos en ambientes complejos, apoya la autonomía, incluye semánticas de contexto abierto, comparte y reutiliza patrones de procesos de negocios.

Como hemos visto, hay mucho trabajo en este campo; sin embargo, muchos de ellos se encuentran en la etapa de propuesta anticipada y los que más se parecen a nuestro proyecto no son aplicaciones libres. Por otro lado, ninguno de estos proyectos incluyen alertas de notificación de la actividad, que es una contribución importante de nuestra propuesta.

3. MODELO DE ARQUITECTURA

La Figura No.1 presenta la arquitectura de MLEA. El lado cliente representa la aplicación desarrollada para los dispositivos móviles con sistema operativo Android, mientras que del lado servidor se visualiza la infraestructura que realiza la integración con el ambiente Moodle.

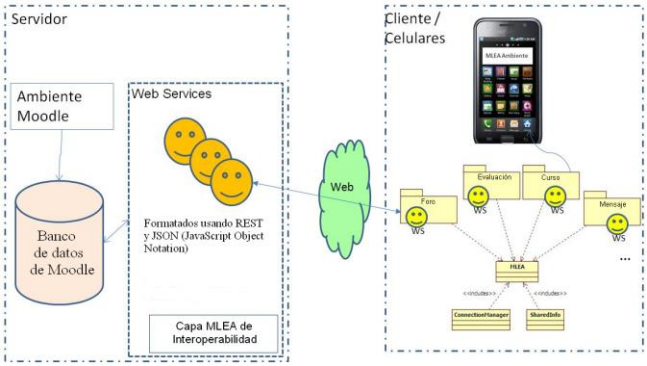

Figura No.1 Arquitectura de MLEA

En el lado servidor, un conjunto de *web services* actúa como interfaz de comunicación entre los clientes y el servidor, resultando responsables por ofrecer a los clientes las funcionalidades integradas. Por medio de los *web services*, los clientes tienen acceso a los principales recursos de Moodle, como foro, evaluación, mensajes, chat, descarga de archivos, localización, alertas, anuncios, calificación y curso, entre otros. Al recibir una solicitud, un *web service* accesa la base de datos de Moodle para recuperar y/o manipular las informaciones necesarias para responder adecuadamente la solicitud.

En el lado cliente, la implementación está basada en el patrón de desarrollo de Android: para cada pantalla de la aplicación, existe una clase Java responsable por controlar las acciones de esa pantalla. De esta forma, para cada funcionalidad provista (por ejemplo foro, evaluación, mensajes, chat, …) existe un conjunto de pantallas y, por lo tanto, un conjunto de clases Java, representadas en la arquitectura mediante el uso de paquetes.

3.1 Flujo de ejecución

La Figura No.2 representa el flujo de ejecución de la aplicación. Dado que para cada pantalla en la aplicación Android existe una clase responsable de controlarla, los datos que la clase exhibe, crea o modifica, provienen del servidor a partir de la invocación de diversos *web services*. Para realizar alguna llamada a esos servicios, la clase controladora invoca a la clase MLEA que, a la vez, realiza los ajustes necesarios y transfiere el control de la ejecución para la clase ConnectionManager, la cual tiene la función de invocar el *web service* apropiado para aquella funcionalidad. Seguidamente, el *web service* invocado transfiere la llamada para el DAO adecuado, el cual es el responsable de realizar la acción solicitada. Para terminar, la ejecución sigue el flujo inverso hasta la pantalla de control, la cual presentará los datos en la pantalla del dispositivo móvil.

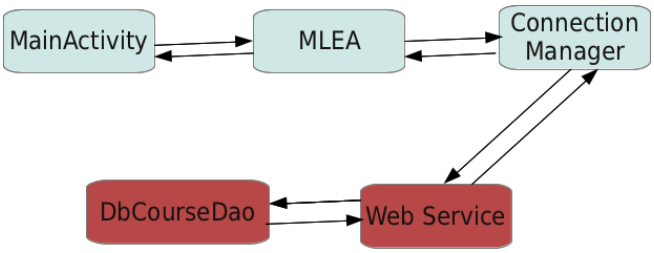

Figura No.2 Flujo de ejecución

En la figura se observa el flujo de ejecución iniciado por la clase MainActivity. Esta clase es la responsable de controlar la pantalla inicial de la aplicación, la cual presenta la lista de cursos en los cuales el usuario está inscrito. Inicialmente la clase MainActivity solicita a la clase MLEA todos los cursos, luego la clase MLEA recupera, de la base de datos del celular, el usuario conectado y transfiere la ejecución para la clase ConnectionManager, la cual invoca un web service en la parte del servidor; el *web service* invoca la clase DAO correspondiente, en este caso DbCourseDao, que es responsable por manipular las informaciones relativas a los cursos. Para finalizar, DbCourseDao realiza la consulta en la base de datos de Moodle, de los cursos deseados, y retorna la información realizando el recorrido inverso hasta llegar a la MainActivity, que mostrará los cursos para que el usuario seleccione el curso deseado.

3.2 Cliente

El lado cliente, es decir, la aplicación Android, utiliza el patrón de proyecto Facade [15] para realizar la comunicación entre las pantallas y la clase responsable por invocar los *web services*. Como mencionamos anteriormente, cada pantalla en la aplicación Android posee una clase asociada. La clase MLEA actúa en la aplicación como una fachada para realizar la comunicación entre las clases controladoras y ConnectionManager. Cuando una clase controladora realiza alguna solicitud, la clase MLEA realiza los ajustes necesarios para que la ConnectionManager comprenda la solicitud. Por ejemplo, si ConnectionManager requiere como parámetro el usuario conectado, la clase MLEA utiliza la clase SharedInfo para recuperar la información de la base de datos del celular. Lo mismo ocurre cuando ConnectionManager responde a la solicitud.

La Figura No.3 ilustra el uso de la fachada, la cual es utilizada por todos los requisitos desarrollados. En ésta se puede observar que los módulos creados, (por ejemplo, Curso, Foro, Chat) son clientes de la clase MLEA, la cual define una interfaz entre las clases ConnectionManager y SharedInfo, aislándolas del resto de la aplicación. La figura solamente presenta algunos de los módulos desarrollados, que son: login, foro, evaluación, mensajes, chat, descarga de archivos, localización, alertas, anuncios, calificación, escoger curso, encuesta, anuncio, evalúa foro, estadísticas del foro y visualizar usuarios conectados.

La clase SharedInfo administra la base de datos del dispositivo móvil, donde son almacenadas las informaciones de los usuarios, tales como: datos de autenticación, identificadores de usuario conectado y de curso escogido, así como datos de caché.

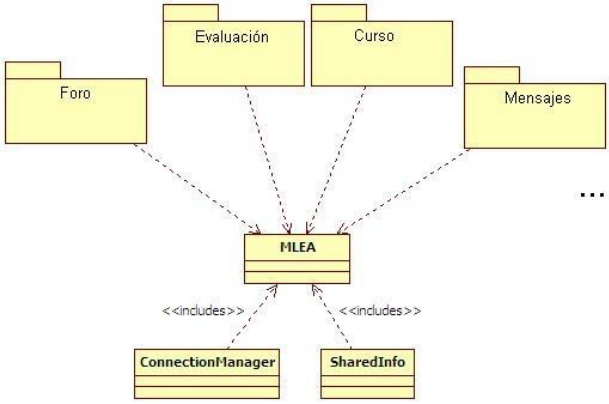

Figura No.3 Vista del Cliente

3.3 Servidor

El lado servidor utiliza dos patrones de diseño para responder a las solicitudes de los clientes. Primeramente, los *web services* utilizan el patrón de proyecto DAO (*Data Access Object*) [16], para accesar y manipular las informaciones en la base de datos de Moodle. Este patrón de proyecto brinda una capa de abstracción que separa la aplicación del mecanismo de persistencia, proporcionando la flexibilidad de modificar el local donde están almacenados los datos, sin la necesidad de alterar la lógica de programación. De esta forma es posible utilizar diferentes bases de datos, tener las informaciones almacenadas en bases de datos locales y/o remotas, inclusive utilizar archivos (práctica no recomendada).

Para cada tipo de dato que será utilizado, existe una interfase DAO que indica las operaciones que se pueden realizar con este tipo. La capa del modelo de la aplicación posee los siguientes tipos de datos, cada uno con una interfase DAO específica: login, foro, evaluación, mensajes, chat, descarga de archivos, localización, alertas, anuncios, calificación, escoger curso, encuesta, anuncio, evalúa foro, estadísticas del foro y visualizar usuarios conectados.

Para aumentar la flexibilidad de la aplicación, las clases DAO no son instanciadas directamente por los *web services*, en vez de esto es utilizada una fábrica para la construcción de las clases DAO. Esta práctica corresponde al uso del patrón de proyecto FacthoryMethod, el cual garantiza que toda la aplicación utilice las DAO apropiadas para la configuración escogida. La siguiente figura presenta el uso de los patrones de diseño, donde el conjunto de *web services* utiliza la clase DaoFactory para crear instancias de las interfaces, las cuales son utilizadas para accesar y manipular las informaciones en la base de datos de Moodle.

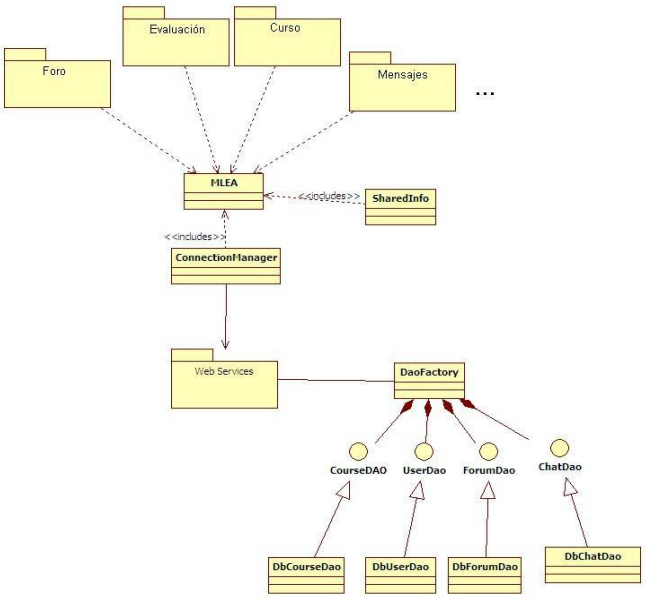

Figura No.4 Vista del Servidor

4. IMPLEMENTACION DE WEBSERVICES

Para cada funcionalidad de la aplicación, existe una clase responsable por proveer el conjunto de *web services* correspondientes al requisito específico.

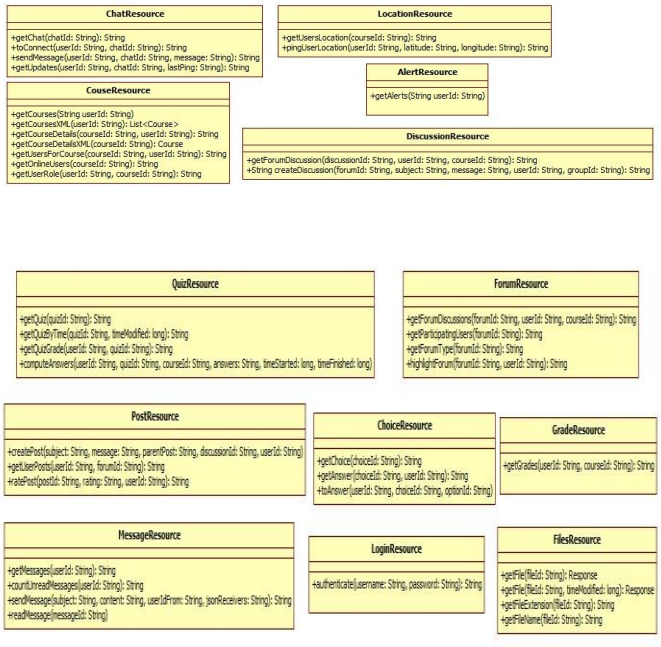

Figura No.5 Especificación de los *web services*

Estos *web services* fueron desarrollados utilizando el protocolo REST (*Representational State Transfer*) para *web services* [17], pues Android no posee soporte nativo para otro tipo de protocolo. De esta forma, el desarrollo resultó más sencillo con el uso de las bibliotecas existentes en Android para la invocación de los servicios. Además del HTTP, protocolo patrón utilizado por REST para la comunicación, se utilizó JSON (*JavaScript Object Notation*) para definir el formato de los datos, por resultar un formato liviano para el intercambio de datos computacionales, lo cual resulta en ventajas en el procesamiento de aplicaciones móviles. En la Figura No. 5, se presentan las clases implementadas, con sus respectivos *web services*. Por ejemplo, la clase ForumResource, contiene 4 *web services*: el primero que es listado, es el getForumDiscussions, este servicio recibe como parámetro los identificadores del foro, del usuario y del curso, retornando todas las discusiones que componen un foro específico.

Tabla No.1 Servicios de Moodle

Web Service	Descripción
moodle_course_create_courses	Crea nuevos cursos
moodle_course_get_courses	Recupera los cursos existentes
moodle_enrol_get_enrolled_users	Recupera la lista de participantes del curso
moodle_enrol_get_users_courses	Recupera la lista de cursos en que un usuario participa
moodle_enrol_manual_enrol_users	Registra un usuario para participar en un curso
moodle_file_get_files	Navegar por los archivos de un curso
moodle_file_upload	Realizar el upload de un archivo
moodle_group_add_groupmembers	Agregar miembros a un grupo
moodle_group_create_groups	Crear nuevos cursos
moodle_group_delete_groupmembers	Excluir miembros de un grupo
moodle_group_delete_groups	Excluir determinado grupo
moodle_group_get_course_groups	Recupera todos os grupos definidos en un curso
moodle_group_get_groupmembers	Recupera los participantes de determinado grupo
moodle_group_get_groups	Recupera los detalles de determinado curso
moodle_message_send_instantmessages	Enviar mensajes
moodle_notes_create_notes	Crear notas
moodle_role_assign	Atribuir roles a un usuario
moodle_role_unassign	Remover rol de usuario
moodle_user_create_users	Crear usuarios
moodle_user_delete_users	Excluir usuarios
moodle_user_get_course_participants_by_id	Recupera el perfil de los participantes de un curso
moodle_user_get_users_by_courseid	Recupera los participantes de determinado curso
moodle_user_get_users_by_id	Recupera un conjunto de participantes
moodle_user_update_users	Actualizar los usuarios
moodle_webservice_get_siteinfo	Recupera algunas informaciones sobre o sitio

Inicialmente, pensamos utilizar los *web services* de Moodle en sus versiones más recientes; sin embargo, identificamos que la cantidad de servicios disponibles por Moodle es muy poca y, de los mismos, apenas una pequeña parte sería aprovechable para los propósitos del proyecto. Frente a esta situación, optamos por desarrollar todos los *web services* necesarios para el funcionamiento de la aplicación.

La Tabla No.1 presenta los servicios provistos por Moodle actualmente, destacando aquellos que fueron aprovechados para el desarrollo de la aplicación. Los *web services* desarrollados fueron presentados en la Figura No.5, en donde cada método constituye un *web service*. Como se puede observar en la Tabla, los servicios provistos por Moodle no resultan suficientes para atender las necesidades de MLEA; por tal razón, todos los *web services* utilizados por MLEA fueron implementados de manera independiente de Moodle; también fueron implementados los pocos servicios aprovechables de Moodle.

4.1 Código de los web services

Considerando la gran cantidad de *web services* implementados, y el hecho que todos poseen una estructura de código bastante semejante (fueron implementados utilizando los mismos patrones de diseño), a continuación presentamos el código de un *web service*, el cual es responsable de recuperar la lista de todos los cursos en que participa un usuario específico.

CourseResource

```
/**
   * Web Service responsavel por buscar todos os cursos em que um usuario
   * participa.
   *
   * @param userId O identificador do usuario
   * @return Os cursos em que o usuario participa
   */
  @POST
  @Produces({MediaType.APPLICATION_JSON})
  public List<Course> getCourses(@FormParam("userId") String userId) {

    logger.info("Web service invocado. Tentando recuperar os cursos nos "
        + "quais o usuario participa");

    try {
      // Recupera os cursos nos quais o usuario participa
      List<Course> courses = courseDao.fetchCoursesByUser(userId);

      // Atualiza a informacao de quando o usuario acessou o Moodle pela ultima vez
      userDao.updateUserLastAccess(userId);

      // Registra no log do Moodle que o usuario visualizou a tela principal
      logDao.viewHome(userId);

      // Retorna os cursos encontrados
      return courses;

    } catch (DaoException ex) {
      logger.log(Level.SEVERE, ex.getMessage(), ex);
      throw new WebApplicationException(Response.Status.INTERNAL_SERVER_ERROR);
    } catch (InvalidArgumentException ex) {
      logger.log(Level.SEVERE, ex.getMessage(), ex);
      throw new WebApplicationException(Response.Status.BAD_REQUEST);
    }
  }
```

Figura No.6 Código de CourseResources

La Figura No.6 presenta el código del *web service* responsable de recuperar todos los cursos en que un usuario participa. Por limitaciones de espacio, las clases utilizadas por el *web service* han sido omitidas.

5. MODELO DE INTEGRACIÓN

A continuación presentamos la arquitectura completa del lado servidor. A partir de los patrones de diseño escogidos, es posible realizar la integración entre diferentes módulos de la aplicación. Por ejemplo, el Módulo de Notas (grade) necesita interactuar con los demás módulos (ejm. foro, evaluación) para proveer su funcionalidad de manera adecuada. A partir de la arquitectura propuesta, el *web service* responsable por esta integración puede utilizar la fábrica para instanciar las diferentes DAOs y aprovechar las funcionalidades de los diferentes módulos involucrados.

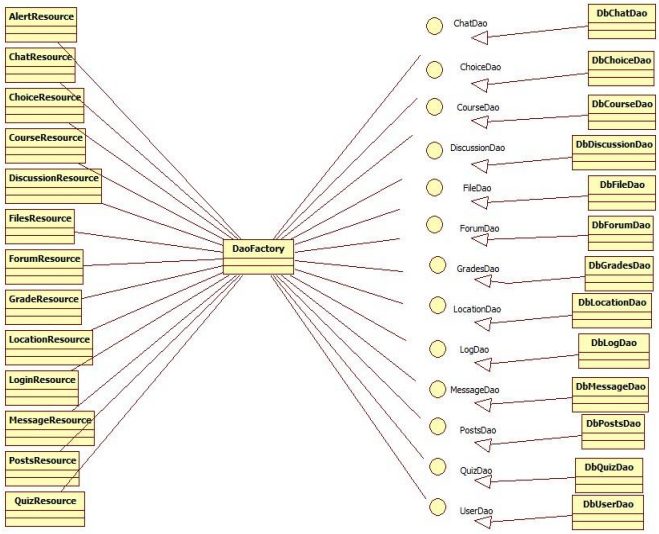

Figura No.7 Modelo de integración de los *web services*

6. RESULTADOS ESPERADOS

El resultado esperado del proyecto es que los clientes desarrollados, basados en la arquitectura propuesta, puedan intercambiar datos e informaciones por medio de agentes de software basados en *web services* de forma transparente al usuario, estableciéndose de esta forma una capa de adaptación entre el ambiente de educación a distancia de la Universidad Tecnológica de Panamá y la infraestructura robusta que dará soporte a la ejecución y simulación de software (*Grid* computacional).
Los usuarios, estudiantes y profesores, recibirán los beneficios del aprendizaje móvil, a partir de MLEA.

7. CONCLUSIONES Y TRABAJOS FUTUROS

El Programa de la Universidad Virtual; viene contribuyendo a que,

cada día más, un mayor número de personas puedan tener acceso a una formación de nivel superior o puedan concluir sus estudios universitarios, sin que esto dependa de la ubicación física de los interesados. La conclusión de estudios ha permitido que nuevos profesionales incursionen en nuevas y mejores plazas del mercado de trabajo, a nivel nacional e internacional. Ha permitido, también, que los egresados accedan a estudios de especialización que complementan su formación profesional.

A partir de la puesta en marcha de MLEA, se recibirán los primeros beneficios del proyecto en los diversos escenarios y sedes de la Universidad Tecnológica de Panamá, que participaron en el desarrollo del mismo. La participación de todos los actores, también nos permitirá identificar situaciones y casos que orienten hacia el mejoramiento y adecuación continua de la aplicación.
Un aspecto importante a destacar es que la arquitectura desarrollada hace posible que, en el futuro, se puedan desarrollar aplicaciones clientes para diferentes sistemas operativos móviles, como por ejemplo iPhone, aprovechando gran parte de la infraestructura construida y, de esta forma, se pueda alcanzar un número mayor de usuarios.

8. AGRADECIMIENTOS

Este trabajo es financiado por la Secretaría Nacional de Ciencia, Tecnología e Innovación - SENACYT, en el marco de la Convocatoria Pública de Fomento a la Colaboración Internacional en I+D. Los autores expresan sus agradecimientos a SENACYT y a la Universidad Tecnológica de Panamá, en Panamá, así como a la Universidade do Vale do Rio dos Sinos, en Porto Alegre, Brasil, por el apoyo para el desarrollo del presente proyecto. El agradecimiento se extiende a los estudiantes Ariel Jaramillo, Josué Manzzo y Stephen Krol; así como a los voluntarios Clifton Clunie Jr., Javier Sánchez, Víctor Shum y Guilherme Sesterhei.

9. REFERENCIAS

[1] A. Mohamed, Mobile Learning: Transforming the Delivery of Education and Training. AU Press, Athabasca University, 2009
[2] C. Clunie, G. T. Clunie, Educación Virtual: Lecciones aprendidas, Panamá, 2010
[3] ¿Qué es el Aprendizaje Móvil? http://a01306702.blogspot.com/2010/03/que-es-el-aprendizaje-movil.html Accesado el 8 de mayo 2011.
[4] Mlearning: http://en.wikipedia.org/wiki/MLearning Accesado el 8 de mayo de 2011
[5] Proyecto MLEA: http://mle.utp.ac.pa/mlea/ Accesado 25 de mayo de 2011
[6] A. Ibrahim, L. Zhao, Supporting the OSGi Service Platform with Mobility and Service Distribution in Ubiquitous Home Environments. The Computer Journal. London:Mar 2009. Vol. 52, Iss. 2, p. 210-239 (30 pp.)
[7] Mobile App, disponible en línea en: http://docs.moodle.org/en/Development:Mobile_app accesado el 3 de noviembre de 2011.
[8] Moodle2 Mobile Project (M2M), disponible en línea en: http://moodle.org/mod/forum/discuss.php?d=152927 accesado el 3 de noviembre de 2011.
[9] Softwarelivre Project, disponible en línea en: http://softwarelivre.sapo.pt/projects/androidmoodle/ accesado el 6 de febrero de 2011.
[10] Mbot, the #1 Android App for Moodle? disponible en línea en: http://codeguild.com/app/mbot/ accesado el 3 de noviembre de 2011.
[11] MLE-Moodle, disponible en línea en: http://mle.sourceforge.net/mlemoodle/index.php?lang=en accesado el 3 de noviembre de 2011.
[12] Moodle Touch (Moodle on Mobiles), disponible en línea en: http://www.pragmasql.com/moodletouch/home.aspx accesado el 3 de noviembre de 2011.
[13] Ye, J., Coyle, L., Dobson, S., Nixon, P., Ontology-based models in pervasive computing systems. The Knowledge Engineering Review. Cambridge:Dec 2007. Vol. 22, Iss. 4, p. 315-347 (33 pp.)
[14] D.K. Barry, Service-Oriented Architecture (SOA) Definition, available online at:http://www.service-architecture.com/web-services/articles/service_oriented_architecture_soa_definition.html accesado el 3 de noviembre de 2011.
[15] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995.
[16] D. Alur, D. Malks, J. Crupi. Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition). Prentice Hall, 2003.
[17] S. Allamaraju, RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity. O`REILLY, 2010