Universidad de Valladolid

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN INGENIERÍA DE SISTEMAS DE TELECOMUNICACIÓN

Aplicación Android de Acceso a los Contenidos de Moodle

Autor:

D.Javier Antolín Blanco

Tutor:

Dña.Miriam Antón Rodríguez

Valladolid, 27 de Abril de 2015

TÍTULO: **APLICACIÓN ANDROID DE ACCESO A LOS CONTENIDOS DE MOODLE**

AUTOR: JAVIER ANTOLÍN BLANCO

TUTORA: MÍRIAM ANTÓN RODRÍGUEZ

DEPARTAMENTO: TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA

Miembros del tribunal

PRESIDENTE: MÍRIAM ANTÓN RODRÍGUEZ

SECRETARIO: DAVID GONZÁLEZ ORTEGA

VOCAL: MARIO MARTÍNEZ ZARZUELA

SUPLENTE: MARÍA ÁNGELES PÉREZ JUÁREZ

FECHA DE LECTURA: Abril de 2015

CALIFICACIÓN:

RESUMEN DEL PROYECTO

La aplicación de las Tecnologías de la Información y la Comunicación (TIC) cada vez están más instauradas en la vida cotidiana. Este conjunto de tecnologías permiten el acceso, tratamiento, comunicación y producción de información presentada en diferentes formatos (imagen, texto, sonido…). Una de las principales ventajas de las TIC es el poder brindar de grandes beneficios y adelantos a campos como la salud y la educación.

La aplicación de las TIC al sistema educativo, también conocido como *e-Learning,* se desarrolla gracias al empleo de múltiples plataformas. Estas plataformas permiten la formación de los alumnos más allá de las horas lectivas, así como la comunicación entre alumnos y profesores.

Una de las plataformas más utilizadas es *Moodle*, la cual se puede definir como una plataforma *e-Learning* de código abierto y libre, que permite crear y añadir módulos y aplicaciones que el personal docente considere de interés.

Una consecuencia del *e-Learning*, es el conocido como *m-Learning*, que al igual que el primero trata de fomentar el aprendizaje más allá de las horas lectivas pero en este caso utilizando como plataforma un dispositivo móvil. De esta manera, se permite acceder a las ventajas del *e-Learning* sin tener que disponer de equipos pesados como los ordenadores.

Este es el objetivo de este proyecto. Realizar una aplicación móvil *Android* para los usuarios de *Moodle*, que les permita por un lado acceder a los cursos en los que estén matriculados, visualizar sus diferentes secciones y recursos de cada sección, para finalmente poder descargar dichos recursos directamente a sus dispositivos móviles. Por otro lado, se implementará el servicio de mensajería interna de *Moodle*, para que los usuarios puedan comunicarse entre sí y con el personal docente a través de la aplicación. Por último se habilitará un registro de usuarios para aquellos que usen por primera vez la plataforma *Moodle*.

PALABRAS CLAVE

Android, base de datos, JSON, Moodle, PHP, REST.

ABSTRACT

The application of Information Technology and Communications (ICTs) are increasingly more introduced in everyday life. This set of technologies makes accessing, processing, communication and production of information in different formats (images, text, sound…). One of the main advantages of ICTs is to provide great benefits and advances in different areas such as health and education.

The application of ICTs in the educational system, also known as "e-Learning", is developed through the use of multiple platforms. These platforms allow the formation of students in addition to school hours, as well as communication between students and teachers.

One of the most popular platforms is Moodle, which can be defined as an e-Learning platform of free and open code, which lets you to create and to add modules and applications that the teaching staff considered relevant.

A consequence of e-Learning is the know as m-Learning, which like the first one want to promote learning in addition to school hours but in this case used as a mobile device platform. Thus, it is allowed the access to the benefits of e-Learning without having heavy equipment such as computers.

This is the goal of this project. To carry out an Android mobile application for Moodle users, enabling them to access the courses in which they are enrolled, view the different sections and resources directly to their mobile devices. On the other hand, internal messaging service for Moodle will be implemented, so that users can be communicated with each other and with the teaching staff through the application. Finally, a user registration for those who use the Moodle platform for the first time will be enabled.

KEYWORDS

Android, Data Base, JSON, Moodle, PHP, REST.

Agradecer en primer lugar a Antonio y Lola, mis padres, el haber estado encima de mí siempre. Todo lo que consiga de provecho en esta vida es gracias a vosotros y a la educación y cariño que me habéis dado.

A mi hermano Pablo, por ser la persona que mejor me conoce y a la que menos le cuesta ponerse en mi lugar siempre.

A Leti, por aguantarme los malos ratos y darme paz durante mis agobios y malos humos.

A todos mis amigos con los que he compartido esta etapa de mi vida: Argucieros, mis Onetazo y KNP Posse.

Y por último agradecer a Míriam el ser primero una muy buena profesora y segundo una mejor tutora.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I.

INTRODUCCIÓN

1. INTRODUCCIÓN GENERAL

Como se puede comprobar de manera cotidiana, las Tecnologías de la Información y la Comunicación (TIC) cada vez están más instauradas en la vida cotidiana. Este conjunto de tecnologías permiten el acceso, tratamiento, comunicación y producción de información presentada en diferentes formatos (imagen, texto, sonido…).

Una de las principales ventajas de las TIC es el poder brindar de grandes beneficios y adelantos a campos como la salud y la educación. En este punto es donde entra en valor el tema abordado en el presente documento: La aplicación de las TIC al sistema educativo, también conocido como *e-Learning.*

La educación constituye un pilar fundamental en la sociedad contemporánea y todo aquello que contribuya a aumentar su eficiencia y calidad será siempre positivo. Con el *e-Learning* lo que se pretende es tener una herramienta más para el proceso educativo. Muchas veces se asocia a la educación a distancia y esto es un error, ya que como mejor desempeño tiene es mezclado con las técnicas de enseñanza tradicionales suponiendo un apoyo y no un sustituto de las mismas (E-abclearning, 2011a).

Lo que se busca por un lado es elaborar mecanismos y herramientas virtuales que permitan apoyar los cursos que actualmente se están dictando, y por otro lado mejorar la comunicación docente-alumno y alumno-alumno, con el fin de desarrollar parte de las enseñanzas desde el lugar de estudio o el hogar de manera que se optimice el tiempo, logrando mayor eficacia durante las reuniones presenciales. Una herramienta consecuencia de lo explicado anteriormente es la plataforma *Moodle.*

Moodle es un software diseñado para ayudar a los docentes a crear cursos en línea y entornos de aprendizaje virtual de alta calidad. Una de las características principales de esta plataforma es la de estar desarrollada en base a la pedagogía social constructivista, donde la comunicación tiene un espacio relevante en el camino de la construcción del conocimiento, siendo el objetivo final crear una experiencia de aprendizaje enriquecedora (Moodle, 2015c).

El objetivo principal del *e-Learning* como ya se ha comentado es permitir la formación de los alumnos más allá de las horas lectivas, como por ejemplo permitiendo el acceso a un recurso académico en cualquier momento a través de un ordenador con internet. Aquí es donde entra en juego la idea principal en torno a la cual está desarrollado este trabajo, esta idea no es otra que la de fomentar el aprendizaje a través de un dispositivo móvil (*Smartphones, Tablets…*), evitando la necesidad de disponer de un ordenador para poder aprovechar las ventajas del *e-Learning,* en nuestro caso poder utilizar una serie de prestaciones de la plataforma *Moodle*, las cuales serán explicadas de manera detallada en el presente documento.

El aprendizaje a través de un dispositivo móvil se conoce como *m-Learning*. El objetivo es que el alumno pueda acceder a cualquier contenido de una plataforma virtual de apoyo educativo aparte de en cualquier momento, también en cualquier lugar sin la necesidad de un ordenador.

2. MOTIVACIÓN

A diario se puede observar como el uso de los dispositivos móviles está completamente instaurado en nuestra sociedad, concretamente el uso de *smarthpones* entre la población joven es algo totalmente extendido.

De manera sencilla, echando un vistazo a plataformas de distribución de aplicaciones móviles como *Play Store* y *App Store* se puede comprobar cómo muchas empresas que utilizan un software online de cara al público, tienen sus versiones móviles del mismo, ya que son conscientes de la importancia de las aplicaciones móviles y el uso que se hace de ellas, cada vez más extendido. Gestiones que antes se hacían desde un ordenador a día de hoy se pueden hacer desde un dispositivo móvil de manera rápida y sencilla.

La motivación principal que tiene este proyecto, es la de dar respuesta a una necesidad promovida por lo citado en el párrafo anterior. Esta no es otra que la de crear una aplicación móvil de acceso a las prestaciones de la plataforma *Moodle*.

El uso extendido de *smartphones* entre los jóvenes sumado a que *Moodle* es un software muy utilizado dentro de la comunidad educativa lleva a pensar que el desarrollo de una aplicación móvil que permita disponer de *Moodle* en cualquier momento puede resultar muy beneficioso de cara al alumnado, permitiendo disfrutar de las ventajas de *Moodle* únicamente con un dispositivo móvil, sin la necesidad de un ordenador.

Esta idea se empezó a desarrollar en Septiembre de 2012 por Fernando Herrero Herrero, como parte de su proyecto fin de carrera como ingeniero de telecomunicaciones en la universidad de Valladolid. En ese momento no existía ninguna aplicación que implementara *Moodle* como aplicación móvil (Herrero Herrero, 2013).

Actualmente en Play Store se pueden encontrar varias aplicaciones Moodle de uso privado como la disponible para la Liverpool Hope University, dicha aplicación no es de uso libre y no se puede implementar en cualquier Moodle, es de uso exclusivo de dicha universidad.

De igual manera se puede encontrar un par de aplicaciones Android, que intentan satisfacer esta necesidad, intentando ser compatible para cualquier Moodle. Una de ellas se encuentra obsoleta (Mdroid Legacy), y luego se encuentra Moodle Mobile, la cual atendiendo a las valoraciones y comentarios de los usuarios se cae en la cuenta de que no es una aplicación completa ni resolutiva siendo varias personas las que coinciden en

que es más un acceso directo que una aplicación, ya que para muchas funcionalidades se redirige al usuario a la web de *Moodle*, sin dar la posibilidad de realizar una determinada gestión desde la misma aplicación. Efectivamente tras instalar y probar dicha aplicación el usuario se ve redirigido a la web en cuanto intenta acceder por ejemplo a la descarga de recursos o cuando quiere acceder a un cuestionario online.

Esto es algo que en este proyecto se evitará, todas las funcionalidades implementadas se desarrollarán exclusivamente desde la aplicación móvil, en ningún caso se accederá a la versión web de *Moodle* desde el navegador de nuestro *smartphone* o *tablet*.

Lo que se busca con la aplicación móvil es evitar cargar la versión de PC en un dispositivo móvil, optimizando así el tiempo de carga.

3. OBJETIVOS

Este proyecto tiene como objetivo principal realizar una aplicación móvil que permita a los usuarios de la plataforma *Moodle* descargar y visualizar los recursos que se encuentren disponibles en los cursos en los que el alumno esté matriculado.

Por otro lado se implementará el servicio de mensajería interna de *Moodle*, gracias al cual los alumnos se podrán comunicar con otros compañeros o con el docente de manera sencilla mediante mensajes de texto, desde la aplicación móvil sin la necesidad de utilizar un servicio de correo alternativo.

Esta aplicación parte de un primer proyecto fin de carrera ya desarrollado el cual consistía en una aplicación móvil que permitía el acceso a una serie de funcionalidades de *Moodle* (Herrero Herrero, 2013). En dicha aplicación primó el análisis de las comunicaciones sobre la creación de nuevas funcionalidades y el diseño de interfaz. También cabe resaltar que era útil para la versión 1.9 y anteriores de *Moodle*, pero se encuentra obsoleta para la última versión estable (2.7.1), debido a cambios sustanciales que se explicarán con detalle en los posteriores capítulos de este documento.

La aplicación de manera resumida se desarrollará de la siguiente manera:

- En primer lugar, se realizará un sistema de registro para alumnos que utilicen por primera vez la plataforma *Moodle*. Estos usuarios podrán acceder a la aplicación una vez sean confirmados por el administrador.
- En segundo lugar, se implementará un sistema de *login*, mediante el cual solo podrán identificarse de manera positiva los usuarios registrados y confirmados por el administrador en la plataforma *Moodle*.
- En tercer lugar, y tras ser autentificados de manera positiva en el paso anterior, los usuarios se encontrarán con un menú, que dará las dos opciones mencionadas

anteriormente. Por un lado el acceso a los servicios de mensajería y por otro lado la visualización de los cursos en los que está matriculado el alumno.

- En cuarto lugar y tras seleccionar el servicio de mensajería, el usuario accederá a otro menú, el cual contendrá las opciones de visualizar los mensajes no leídos, leídos, enviados y la opción de escribir un nuevo mensaje
- En quinto lugar y tras seleccionar la opción de "asignaturas" en el menú principal, el usuario accederá a una lista de las mismas, únicamente en las que esté matriculado. Tras seleccionar cualquiera de estas asignaturas, se accederá a otra lista en la que se podrán visualizar las diferentes secciones del curso, y por último tras elegir una de estas secciones el usuario de la aplicación accederá al conjunto de recursos para descargar. Pulsando sobre cualquiera de estos recursos se procederá a la descarga del mismo, quedando almacenado este en la memoria interna del dispositivo del usuario, en el apartado *downloads*.

4. INTRODUCCIÓN AL DOCUMENTO

Este documento se estructura en una serie de seis capítulos como se explica a continuación:

- El primer capítulo sirve de introducción al proyecto y al presente documento, explicando los objetivos y la motivación del mismo.
- En el segundo capítulo se realizará un análisis de las tecnologías existentes para la realización del proyecto y se expondrán los motivos de elección de cada una.
- El tercer capítulo trata acerca de las características de *Moodle*, plataforma sobre la que gira el desarrollo de este proyecto.
- En el cuarto capítulo se hablará de los servicios Web, gracias a los cuales se implementan las funcionalidades que se abordan en este proyecto.
- El quinto capítulo recoge el desarrollo del proyecto, tanto desde el lado del cliente como del servidor y de la comunicación de los mismos. También se mostrará la interfaz gráfica de la aplicación con explicaciones detalladas de cada pantalla y proceso.
- El sexto y último capítulo recogerá las conclusiones y las posibles líneas futuras a seguir en una próxima evolución del proyecto.

- Finalmente se presentará la bibliografía empleada en el desarrollo del proyecto y se adjuntarán una serie de anexos complementarios para la comprensión del mismo.

CAPÍTULO II.

JUSTIFICACIÓN DE LAS TECNOLOGÍAS EMPLEADAS

1.INTRODUCCIÓN

En este capítulo se hará un repaso de las tecnologías disponibles para realizar la aplicación móvil planteada, dando por último una justificación de por qué se ha seleccionado una u otra. Por un lado se expondrán los diferentes sistemas operativos móviles, y por otro se repasarán las diferentes plataformas de *e-Learning* disponibles.

2.SISTEMAS OPERATIVOS MÓVILES

Un sistema operativo (SO) es el software de una computadora que provee una interfaz entre el resto de programas del ordenador, los dispositivos hardware y el usuario.

Las funciones básicas del Sistema Operativo son administrar los recursos de la máquina, coordinar el hardware y organizar archivos y directorios en dispositivos de almacenamiento.

Actualmente los dispositivos móviles, con el fin de mejorar la interacción con el usuario, y las nuevas necesidades requeridas por los mismos, van equipados con un sistema operativo, con el mismo objetivo que el descrito para los PCs.

En el mercado mundial de los sistemas operativos móviles se encuentran varios tipos. A la cabeza de todos se encuentra Android, el sistema operativo de Google (ver Tabla 1).

Su competencia principal es iOS el sistema operativo de Apple, aunque lo sigue a larga distancia. Hay otros sistemas con mucha menor aceptación en el mercado como son Windows Phone, Firefox OS, BlackBerry, Ubuntu Touch, Tizen, WebOS…

Este apartado se centrará en dar una breve explicación de todos los anteriores, focalizándose principalmente en las características de Android e iOS ya que son los que ocupan entre ambos el 96.5% de la cuota del mercado. Este apartado finalizará con una justificación del sistema operativo elegido para el desarrollo de esta aplicación (Amate, 2014).

Global Smartphone Operating System Shipments (Millions of Units)	Q2 '13	Q2 '14
Android	186.8	249.6
Apple iOS	31.2	35.2
Microsoft	8.9	8.0
BlackBerry	5.7	1.9
Others	0.5	0.5
Total	233.0	295.2

Global Smartphone Operating System Marketshare %	Q2 '13	Q2 '14
Android	80.2%	84.6%
Apple iOS	13.4%	11.9%
Microsoft	3.8%	2.7%
BlackBerry	2.4%	0.6%
Others	0.2%	0.2%
Total	100.0%	100.0%

Total Growth Year-over-Year %	48.9%	26.7%

Source: Strategy Analytics

Tabla 1 – Distribución del mercado de sistemas operativos móviles años 2013 y 2014

2.1 ANDROID

Es el sistema operativo número uno en cuanto a popularidad. Con una cuota de mercado cercana al 85% el sistema operativo de *Google* se caracteriza por ser abierto y disponible para cualquier fabricante interesando en utilizarlo para sus dispositivos móviles. Esta característica hace que su cuota de mercado sea muy amplia, ya que no necesita una marca de dispositivos concreta para poder ser implementado (Marín, 2014).

El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de *XML*, 2,8 millones de líneas de lenguaje *C*, 2,1 millones de líneas de *Java* y 1,75 millones de líneas de *C++*. Se puede observar la arquitectura de este sistema operativo de manera gráfica en la Figura 1:

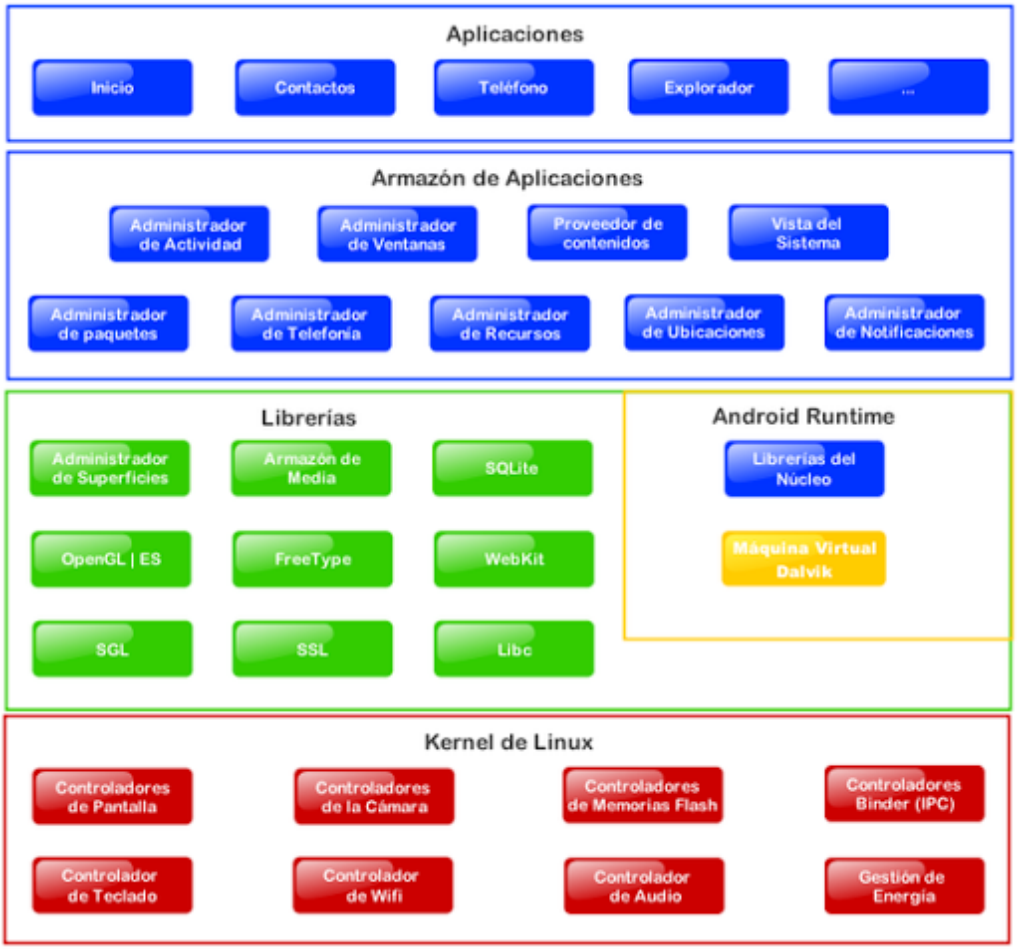

Figura 1 – Arquitectura de *Android*

- Aplicaciones: Este nivel contiene, tanto las incluidas por defecto de *Android* como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

- Framework de Aplicaciones: Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para *Android*, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo *framework* representado por este nivel.

- Librerías: La siguiente capa se corresponde con las librerías utilizadas por *Android*. Éstas han sido escritas utilizando *C/C++* y proporcionan a *Android* la mayor parte de sus capacidades más características. Junto al núcleo basado en *Linux*, estas librerías constituyen el corazón de *Android*.

- Tiempo de ejecución de *Android*: Al mismo nivel que las librerías de *Android* se sitúa el entorno de ejecución. Éste lo constituyen las *Core Libraries*, que son librerías con multitud de clases Java y la máquina virtual Dalvik.

- Núcleo Linux: *Android* utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde *Android* es crear las librerías de control o drivers necesarios dentro de este *kernel* de Linux incorporado en el propio *Android* (Zamora, 2014).

2.2 iOS

Este sistema operativo supone el máximo competidor para *Android,* tal y como se advertía en la Figura 1. Este sistema operativo se encuentra únicamente en los dispositivos *Apple*, ya que son los únicos que lo pueden implementar. Esto limita de manera notable su cuota de mercado (Picurelli, 2014).

Apple diseñó este sistema para *iPhone* (teléfono de la marca *Apple*), pero más tarde también lo han usado en el *iPod Touch* y en el *iPad*. Es un sistema derivado de *Mac OS X,* el cual está basado a su vez en *Darwin BSD.* Es un sistema basado en capas, como se observa en la Figura 2.

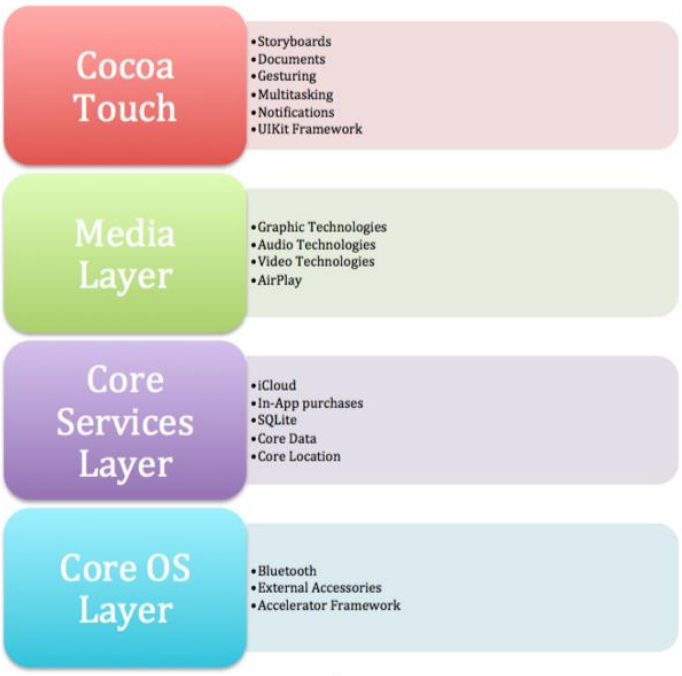

Figura 2 – Arquitectura de *iOS*

- Cocoa Touch: Se compone de un conjunto de *frameworks* para desarrollar aplicaciones. El lenguaje utilizado para programar esta librería, y por tanto las aplicaciones de *iOS,* es *Objetive-C*.

- Media: Esta capa se encarga de los servicios multimedia que pueden usar los usuarios en sus aplicaciones.

- Core Service: provee una abstracción sobre los servicios que proporciona el sistema.

- Core OS: Es la capa base "Core OS" supone el núcleo del sistema operativo, es la que se encarga de la administración de memoria, el sistema de ficheros, las comunicaciones, entre otras tareas. Esta capa interactúa directamente con el hardware.

2.3 OTROS SISTEMAS OPERATIVOS MÓVILES

2.3.1WINDOWS PHONE

Es el sistema operativo móvil desarrollado por *Microsoft*. Esta compañía está realizando un gran esfuerzo financiero para posicionar *Windows Phone* como una tercera opción interesante para los consumidores. Su alianza con *Nokia* y su posterior compra le ha ayudado a darse a conocer mejor e ir arañando cuota de mercado a los dos líderes. Los últimos datos hablan de un 2,5% a nivel mundial.

Con un diseño radicalmente distinto a las dos opciones ya comentadas, *Windows Phone* destaca por su pantalla de inicio personalizable que ofrece las notificaciones de las aplicaciones de una manera sencilla y limpia. Además ofrece una experiencia de usuario muy buena independientemente del tipo y gama de terminal en que se esté usando.

Aunque con menos aplicaciones disponibles que en *Android* e *iOS*, *Windows Phone 8.1*, cuenta ya con más de 300.000 *Apps* en su tienda, además de ofrecer aplicaciones propias de la compañía como *Skype*, *OneDrive* o *Xbox Live (*Amate, 2014).

2.3.2 FIREFOX OS

Un sistema operativo basado en *HTML5* con núcleo *Linux*, de código abierto. Desarrollado por *Mozilla Corporation* con apoyo de empresas como Telefónica. El sistema operativo está basado en *Linux* y usa la tecnología de *Mozilla*, *Gecko*. Se basa en estándares abiertos como por ejemplo *HML5*, *CSS3* y *JavaScript*.

Pensado para ser un sistema operativo realmente abierto, a diferencia de *Android*, donde *Google* controla ciertos aspectos del sistema. Esta característica, permite a *Firefox OS* llegar a cubrir el nicho de mercado de la gama baja con mayor facilidad que A*ndroid*. El anuncio hecho en febrero de este año de lanzar un *smartphone* por 25 euros va completamente en esa línea.

Entre las interesantes características de este sistema operativo abierto están las aplicaciones web y pueden ser de dos tipos diferentes: aplicaciones de servidor o empaquetadas. A diferencia de los SO ya comentados, en este caso, las *Apps* de servidor, corren vía web, es decir son páginas webs con la apariencia de aplicaciones y sin conexión a internet no es posible acceder a estas. Las aplicaciones empaquetadas necesitan la descarga de un paquete comprimido y se cargan desde la fuente local cada vez que se accede a la aplicación (Amate, 2014).

2.3.3 BLACKBERRY

Blackberry, anteriormente conocida como *RIM,* no está pasando por sus mejores momentos. Al igual que le pasó a *Nokia*, el cambio de paradigma en los *smartphones* no le fue beneficioso. Acostumbrado a ofrecer terminales con teclado físico, el paso a las pantallas táctiles supuso un problema.

Sin embargo, los esfuerzos realizados por la compañía canadiense para recuperar el terreno perdido fueron grandes y en el año 2012 lanzaron su órdago con un renovado sistema operativo el *Blackberry 10*. Aun así, los últimos estudios sobre cuota de mercardo lo dejan en tan solo un 0,5% mundial.

Blackberry 10 tiene una interfaz más fluida, un teclado inteligente y táctil más depurado y otra serie de opciones que lo acercan a las de la competencia. Al igual que con *iOS*, el SO es software propietario y solamente los teléfonos de la compañía llevan su sistema instalado.

2.3.4 UBUNTU TOUCH

Otro sistema operativo basado en Linux pero en esta ocasión bajo la famosa firma *Ubuntu*. Presentado en el 2013, se trata de un proyecto de Canonical.

Ubuntu Touch utiliza las mismas tecnologías de la versión de escritorio, por lo que ambas comparten *Apps* sin problemas de compatibilidad. Dispone también de algunas de las aplicaciones más populares como *Facebook* y *Youtube*.

2.3.5 TIZEN

Sistema operativo móvil, también basado en *Linux*, patrocinado por *Linux Fundation* y Fundación LiMo. Se ha desarrollado a partir de la plataforma *Linux* de *Samsung*. Aunque en un principio fue presentado como un SO de código abierto, *Tizen 2* funciona con un sistema de licencias no abiertas. El SDK completo fue publicado bajo licencia de *Samsung* de código no abierto.

Aunque pueda parecer que *Tizen* forma parte de la estrategia de *Samsung* a largo plazo, su apuesta errática por este sistema operativo hace que no se sepa muy bien qué pasará con él. De momento algunos de sus dispositivos ya lo incorporan como el caso del famoso *smartwatch Samsung Gear S*.

2.4 ELECCIÓN

Como ya se explicó en la introducción, la diferencia del uso en terminales móviles de *Android* e *iOS* es infinitamente superior a la de sus competidores. En la introducción de este documento ya se comentó que una de sus motivaciones principales de esta aplicación era la de permitir el uso de la misma al mayor número de alumnos y docentes. Parece lógico pensar que tendrá que ser desarrollada para un sistema operativo que abarque la mayor cuota de mercado, por lo tanto parece claro que nuestra elección para desarrollar esta aplicación *Moodle* será *Android*.

La motivación principal que hace que decanta el desarrollo en de esta aplicación con *Android* es la cuota de mercado, para poder abarcar el mayor número de usuarios. Pero hay otros motivos que hacen que la balanza caiga por el lado de *Android,* en detrimento de *iOS* su inmediato rival:

- Para desarrollar en *iOS* los desarrolladores deben abonar una cuota anual de 99$, algo que en *Android* resulta mucho más barato, ya que para subir aplicaciones a *Google Play*, únicamente habrá que hacer un único pago de 25$ para obtener la licencia (Picurelli, 2014).

- La segunda razón es que para desarrollar en *iOS* es necesario disponer de un terminal *Mac* (PCs de la marca *Apple*) y en consecuencia disponer de un terminal móvil *Apple*, ambos de precio elevado. Mientras que para desarrollar Android únicamente debemos disponer de un PC normal y un móvil *Android*, los cuales se pueden encontrar a un precio mucho más asequible en el mercado.

3.INTRODUCCIÓN PLATAFORMAS E-LEARNING

Como ya se expuso en la introducción de este documento, este proyecto gira en torno a la idea del *e-Learning*. Este nuevo concepto educativo es una revolucionaria modalidad de capacitación que posibilitó Internet, y que hoy se posiciona como la forma de capacitación predominante en el futuro. Este sistema ha transformado la educación, abriendo puertas al aprendizaje individual y organizacional.

La plataforma *e-Learning* por excelencia es *Moodle*, es la más desarrollada y utilizada, pero existen otras alternativas que permiten la educación y capacitación a través de Internet. En este apartado repasaremos brevemente las características básicas de *Moodle* por un lado y las del resto de plataformas por otro.

3.1 MOODLE

Una de sus principales características es que es un software Libre. Esto implica que tiene derechos de autor, pero el usuario puede modificar y usar *Moodle* de manera libre siempre que se acepte proporcionar el código fuente a terceros, no modificar en ningún caso la licencia original y/o los derechos de autor, y se deberá aplicar esta licencia a cualquier trabajo derivado de él.

Moodle, es adaptable a las necesidades de cada institución o empresa. También es fácilmente integrable a infraestructuras existentes y tiene una enorme capacidad de crecimiento.

Una de las diferencias más importantes es la disponibilidad de opciones. *Moodle* tiene una enorme variedad de herramientas para la creación de cursos, y todas están disponibles en forma libre. No hay una "Licencia Básica" con opciones recortadas y una "Licencia Plus" que incluya todos los módulos, de hecho no hay licencia: *Moodle* incluye todo desde el momento en que lo instalamos. Su arquitectura y herramientas fueron diseñadas para clases en línea, así como también para complementar el aprendizaje presencial (Moodle, 2015c).

La instalación requiere una plataforma que soporte *PHP* y la disponibilidad de una base de datos. *Moodle* tiene una capa de abstracción de bases de datos por lo que soporta los principales sistemas gestores de bases de datos. Aunque se recomienda ser implementado con *MySQL*.

Se ha puesto énfasis en una seguridad sólida en toda la plataforma. Todos los formularios son revisados, las cookies cifradas, etc. La mayoría de las áreas de introducción de texto (materiales, mensajes de los foros, entradas de los diarios, etc.) pueden ser editadas usando el editor *HTML*, tan sencillo como cualquier editor de texto.

3.2 OTRAS PLATAFORMAS DISPONIBLES

Con mucho menos nivel de desarrollo y uso, en el campo del *e-Learning* nos podemos encontrar con otra serie de plataformas, aquí mostramos una breve lista de las mismas con sus principales características (Fontela Sánchez, 2013):

- *ATutor*: Es otra plataforma de *e-Learning* o *LMS* desarrollada en *PHP* con base de datos *MySQL* y completamente *opensource* (de código abierto). Es una solución bastante efectiva y completamente compatible con paquetes *SCORM* o *IMS*.

- *Claroline*: Otra alternativa desarrollada en *PHP* con base de datos *MySQL*, completamente *opensource* y tan efectiva como *ATutor* y *Moodle*. Dispone de bastantes herramientas colaborativas.

- *eFront*: Un *LMS* que está disponible en 2 versiones, una de pago y otra gratuita. También está desarrollada en *PHP* con base de datos *MySQL*. También ofrece un servicio autohosteado por los desarrolladores.

- *ILIAS*: Este es otro sistema para gestionar cursos y grupos. Permite crear todo tipo de recursos educativos y algunos módulos para realizar funciones avanzadas como integración *WebDav* y *LDAP*.

- *Dokeos*: Este es otro software parecido a *Moodle*, está disponible en dos versiones, una gratuita y otra de pago, pero la versión gratuita tiene algunas limitaciones importantes. Esta desarrollado en *PHP* con base de datos *MySQL*. Dispone de unas capacidades impresionantes, hasta en la versión gratuita.

- *Sakai*: Esta es una herramienta colaborativa con posibilidades de ser utilizada como plataforma e-Learning para cursos online. Es un sistema desarrollado para funcionar sobre *Apache Tomcat* y con muy buen rendimiento.

Ninguna de estas plataformas de *e-Learning* está a la altura de *Moodle*, ya que todos los sistemas *e-Learning* anteriormente citados tienen las funcionalidades menos desarrolladas y con menos características.

3.3 ELECCIÓN DE LA PLATAFORMA E-LEARNING

Moodle es la plataforma *e-Learning* más desarrollada. Se seleccionará esta plataforma para implementarla en versión móvil atendiendo a las tres razones siguientes:

- Es la plataforma utilizada tanto por el Grupo de Telemática e Imagen de la Universidad de Valladolid, entidad para la que se desarrolla este proyecto, como por la propia Universidad de Valladolid
- Es la plataforma *e-Learning* más completa.
- Disponemos de mucha información y documentación acerca de la misma.

Moodle está implementado con *PHP-MySQL* por lo que la parte del servidor de nuestro proyecto, será desarrollada de igual manera.

CAPÍTULO III.

MOODLE.

1. INTRODUCCIÓN

Moodle es una aplicación web de tipo Entorno de Aprendizaje Virtual (*Virtual Learning Enviroment, VLE*), un Sistema de Gestión de Cursos de Código Abierto (*Open Source Course Management System, CMS*), de distribución libre, también conocido como Sistema de Gestión del Aprendizaje Virtual (*Virtual Learning Enviroment, VLE*) que ayuda a los educadores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conoce como LCMS (*Learning Content Management System*).

El diseño y el desarrollo de Moodle se basan en una determinada filosofía del aprendizaje, una forma de pensar que a menudo se denomina pedagogía construccionista social". Este punto de vista mantiene que la gente construye activamente nuevos conocimientos a medida que interactúa con su entorno. La palabra Moodle, en inglés, es un acrónimo para Entorno de Aprendizaje Dinámico Modular, Orientado a Objetos (Modular Object-Oriented Dynamic Learning Environment), lo que resulta fundamentalmente útil para los desarrolladores y teóricos de la educación. También es un verbo anglosajón que describe el proceso ocioso de dar vueltas sobre algo, haciendo las cosas como se vienen a la mente... una actividad amena que muchas veces conllevan al proceso de comprensión y, finalmente, a la creatividad (Moodle, 2015a).

Moodle es un software que permite crear comunidades educativas basadas en Internet. Constituye una potente herramienta de apoyo al docente que le permite impartir sus conocimientos utilizando las nuevas tecnologías.

De una manera más coloquial, podemos decir que *Moodle* es una aplicación para crear y gestionar plataformas educativas, es decir, espacios donde un centro educativo, institución o empresa, gestiona recursos educativos proporcionados por unos docentes y organiza el acceso a esos recursos por los estudiantes, y además permite la comunicación entre todos los implicados (alumnos y profesores).

Moodle aunque es una aplicación de código libre tiiene derechos de autor, pero el usuario puede copiar, modificar y usar *Moodle* de manera libre siempre que se acepte proporcionar el código fuente a terceros, no modificar en ningún caso la licencia original y/o los derechos de autor, y se deberá aplicar esta licencia a cualquier trabajo derivado de él.

Por último y para resumir lo anterior diremos que *Moodle* es un sistema para el Manejo del Aprendizaje en línea gratuito, que les permite a los educadores la creación de sus propios sitios web privados, llenos de cursos dinámicos que extienden el aprendizaje, en cualquier momento, en cualquier sitio.

2. CARACTERÍSTICAS

Atendiendo a la documentación oficial de *Moodle* (Moodle, 2015c), hablaremos de las características generales de la plataforma, más tarde de las características administrativas y las características para desarrollo y gestión de los cursos.

2.1 CARACTERÍSTICAS GENERALES

- Modularidad: Cada uno de los recursos que componen la plataforma se define como una pieza de software independiente que puede ser modificada o eliminada según las necesidades de la institución de enseñanza que la adopte.

- Interfaz moderna y fácil de usar: Diseñada para ser responsiva y accesible.

- Tablero personalizado: La página inicial dispone de un "tablero" que permite organizar y mostrar los cursos de múltiples formas y permite ver rápidamente el conjunto de mensajes y tareas actuales.

- Actividades y herramientas colaborativas: Se dispone de módulos como los foros, las wikis y los glosarios, los cuáles permiten trabajar y aprender de manera conjunta.

- Calendario todo-en-uno: Este calendario te ayuda a mantener al día su calendario académico o el de la compañía, fechas de entrega dentro del curso, reuniones grupales y otros eventos personales.

- Gestión conveniente de archivos: Se permite la colocación de archivos desde servicios de almacenamiento en la nube, incluyendo MS Skydrive, Dropbox y Google Drive.

- Editor de texto simple e intuitivo: Posibilita dar formato convenientemente al texto y añadir multimedia e imágenes con un editor que funciona con todos los navegadores de Internet.

- Notificaciones: Cuando se habilitan, los usuarios pueden recibir alertas automáticas acerca de nuevas tareas y fechas para entregarlas, publicaciones en foros y también de los mensajes privados mandados/recibidos entre ellos.

- Monitoreo del progreso: Los docentes y los alumnos pueden monitorear el progreso y el grado de finalización con un conjunto de opciones para monitoreo de actividades individuales o recursos, y también a nivel del curso.

2.2 CARACTERÍSTICAS ADMINISTRATIVAS

- Diseño personalizable del sitio: fácilmente permite personalizar un tema de *Moodle* con su logo, esquema de colores y mucho más.

- Autenticación e inscripciones (matriculaciones) masivas seguras: Más de 50 opciones para autenticación e inscripción, para añadir e inscribir usuarios a su sitio y cursos *Moodle*.

- Capacidad multilingüe: Se posibilita que los usuarios vean el contenido del curso y aprendan en su propio idioma, o también se da la opción de configurar el sitio para organizaciones y usuarios multilingües.

- Creación masiva de cursos: Permite añadir cursos en lotes.

- Fácil gestión de permisos y roles de usuario: Se atajan las preocupaciones sobre seguridad al definir roles para especificar y gestionar el acceso de los usuarios.

- Soporta estándares abiertos: Se da la posibilidad de importar y exportar fácilmente cursos IMS-LTI, SCORM y más, hacia y desde *Moodle*.

- Alta interoperabilidad: Está habilitada la posibilidad de integrar libremente aplicaciones externas y contenidos, o cree su propio *plugin* para integraciones personalizadas.

- Gestión simple de *plugins* y complementos: Permite instalar y deshabilitar complementos y *plugins* desde adentro de una sola interfaz administrativa.

- Actualizaciones regulares de seguridad: *Moodle* es actualizado regularmente con los últimos parches de seguridad, para ayudar a asegurar que su sitio *Moodle* sea seguro.

- Reportes y bitácoras detalladas: Permite generar y ver reportes sobre actividad y participación a nivel de curso y de sitio.

2.3 CARACTERÍSTICAS PARA EL DESARROLLO Y GESTIÓN DEL CURSO

- Rutas directas de aprendizaje: Se da la posibilidad de diseñar y gestionar cursos para cumplir con diversos requisitos. Las clases pueden ser dirigidas por el instructor, auto-reguladas, mixtas o completamente en-línea.

- Fomento de la colaboración: Las características incluidas para la publicación colaborativa fomentan que el alumno se comprometa y realice colaboración impulsada por el contenido.

- Utilización de recursos externos: Permite enseñar con materiales e incluir tareas provenientes de otros sitios y conectándolos al libro de calificaciones en *Moodle*.

- Integración multimedia: El soporte incluido en *Moodle* para multimedia le permite buscar fácilmente e insertar archivos de audio y video en sus cursos.

- Gestión de grupo: se recomienda agrupar los alumnos para compartir los cursos y diferenciar actividades, para favorecer así el trabajo en equipo.

- Flujo grama de puntuación: Permite por un lado asignar convenientemente a diferentes personas para que califiquen tareas y gestionar la moderación de calificaciones. Y por otro lado permite controlar cuando se liberan las calificaciones a los alumnos individuales.

- Calificación en línea: Se da la posibilidad de revisar con facilidad y proporcionar retroalimentación en-línea, al hacer anotaciones directamente sobre archivos PDF dentro del navegador de Internet.

- Evaluación propia y por pares: Actividades incluidas, tales como talleres y encuestas, estimulan a los alumnos para que vean, califiquen y evalúen el trabajo de ellos mismos y el de otros participantes del curso como un grupo.

- Insignias integradas: Totalmente compatible con las Insignias Abiertas de *Mozilla* (*Mozilla Open Badges*), con las que se busca motivar a los estudiantes y recompensa la participación y los logros con insignias personalizadas.

- Competencias y rúbricas: Permite seleccionar entre métodos avanzados de calificación para personalizar el libro de calificaciones del curso y de acuerdo a sus criterios de exámenes.

- Seguridad y privacidad: Posibilita enseñar y compartir conocimientos y recursos dentro de un espacio privado, al que solamente pueden acceder el docente y su grupo de alumnos.

3. ARQUITECTURA

3.1 INTRODUCCIÓN

Desde la perspectiva de un administrador de sistemas *Moodle* ha sido diseñado de acuerdo con los siguientes criterios (Moodle, 2015b):

***Moodle* debe poder ejecutarse en la más amplia posible variedad de plataformas:**

Uno de los lenguajes de programación más utilizados en aplicaciones Web y que funciona en la mayoría de las plataformas es *PHP* combinado con *MySQL*, y este es el entorno en el que *Moodle* ha sido desarrollado (sobre *Linux, Windows,* y *Mac OS X*). *Moodle* también usa la librería *ADOdb* para la abstracción de bases de datos, lo que significa que *Moodle* puede usar más de diez marcas diferentes de bases de datos.

***Moodle* debe de ser fácil de instalar, aprender y modificar:**

Los primeros prototipos de *Moodle* (1999) se construyeron usando Zope, un avanzado servidor de aplicaciones Web orientado a objetos. Desafortunadamente resultó que aunque la tecnología era bastante buena, tenía una curva de aprendizaje muy elevada y no era muy flexible en términos de administración del sistema. El lenguaje *PHP*, por otro lado, es muy fácil de aprender (especialmente si el desarrollador ha realizado algo de programación usando cualquier otro lenguaje de script).

La reutilización del código se archiva en librerías con funciones claramente tituladas y con una disposición de los archivos de script, consistente. PHP es también fácil de instalar (existen versiones ejecutables para todas las plataformas) y está ampliamente disponible, pues la mayoría de los servicios de alojamiento lo proporcionan como un estándar.

Debe ser fácil de actualizar desde una versión a la siguiente:

Moodle sabe cuál es su versión (así como las versiones de todos los módulos) y se ha construido un mecanismo interno para que Moodle pueda actualizarse a sí mismo de forma apropiada a las nuevas versiones (por ejemplo, puede renombrar las tablas de las bases de datos o añadir nuevos campos). Usando CVS en Unix, por ejemplo, uno tan sólo tiene que hacer un "cvs update -d" y luego visitar la página principal del sitio para completar la actualización.

Debe ser modular para permitir el crecimiento:

Moodle tiene una serie de características modulares, incluyendo temas, actividades, interfaces de idioma, esquemas de base de datos y formatos de cursos. Esto le permite a cualquiera añadir características al código básico principal o incluso distribuirlas por separado. Se abordará este tema de manera profunda en la siguiente sección.

Debe poder usarse junto a otros sistemas:

Una de las cosas que hace *Moodle* es mantener todos los archivos para un curso en un único directorio en el servidor. Esto podría permitir que el administrador de un sistema proporcione similares formas de acceso a un nivel de archivo para cada profesor, tal como *Appletalk, SMB, NFS, FTP, WebDAV* y demás. Los módulos de autenticación le permiten a *Moodle* usar *LDAP, IMAP, POP3, NNTP* y otras bases de datos como fuentes de información de los usuarios.

3.2 ESTRUCTURA

El directorio principal *Moodle* del servidor tiene almacenados todos los contenidos de la plataforma. A continuación, se detalla la estructura de este directorio y sus elementos principales (Herrero Herrero, 2013):

- **config.php**: Contiene la configuración fundamental. Este archivo no viene con *Moodle*, se crea durante la instalación.
- **install.php**: El script que se ejecuta para crear el archivo config.php. Se lanza cuando se instala el paquete de *Moodle*.
- **version.php**: Define la versión actual del código de *Moodle*.
- **index.php**: La página principal del sitio donde hay que autentificarse.
- **help.php**: Muestra la página de ayuda cuando se pulsa sobre el icono de ayuda.
- **admin/**: Código para administrar todo el servidor.
- **auth/**: Módulos para la autenticación de usuarios.
- **blocks/**: Módulos para los pequeños bloques laterales contenidos en la página principal.

- **backup/**: Código para la generación de copias de seguridad.
- **blog/**: Código para la administración de blogs.
- **calendar/**: Código para manejar y mostrar eventos de calendario.
- **course/**: Código para presentar y gestionar los cursos.
- **enrol/**: Código para los módulos de inscripción.
- **error/**: Código sobre los errores de la plataforma.
- **files/**: Código para presentar y gestionar los archivos cargados.
- **filter/**: Código para gestionar los filtros en los módulos.
- **grade/**: Código para calificar los exámenes y trabajos de los alumnos de los cursos.
- **group/**: Código para establecer grupos de alumnos.
- **install/**: Código para la instalación de *Moodle*.
- **ipllokup/**: Muestra información sobre la dirección IP.
- **lang/**: Textos en diferentes idiomas, un directorio por idioma.
- **lib/**: Librerías del código fundamental de *Moodle*.
- **login/**: Código para manejar las entradas y creación de cuentas.
- **message/**: Código para gestionar los distintos mensajes que van apareciendo durante todo el uso de *Moodle*.
- **mod/**: Todos los módulos de los cursos de *Moodle*.
- **my/**: Configuración de la página personal de *Moodle*.
- **pix/**: Gráficos genéricos del sitio (íconos).
- **question/**: Código para gestionar el módulo de preguntas.

- **rss/**: Configuración de los canales RSS.
- **search/**: Configuración de las búsquedas internas.
- **tag/**: Etiquetas asignadas a los blog de usuarios y a sus intereses.
- **theme/**: Paquetes de temas para cambiar la apariencia del sitio.
- **user/**: Código para mostrar y gestionar los usuarios.

3.3 MÓDULOS DE ACTIVIDADES

Estos son los módulos más importantes, y se encuentran en el directorio mod. Por defecto hay siete módulos: Tarea, Consulta, Foro, Glosario, Cuestionario, Recurso, y Encuesta. Cada módulo está en un subdirectorio separado y consiste en los siguientes elementos obligatorios (más los scripts extra que son únicos para cada módulo):

- mod_form.php: un formulario para establecer o actualizar una instancia de este módulo.
- version.php: define alguna meta-información y proporciona código de actualización
- icon.gif: se trata de un icono de 16x16 para el módulo
- db/install.xml: define la estructura de las tablas para todos los tipos de bases de datos. Se utiliza cuando se instala el módulo.
- db/upgrade.php: define los cambios en la estructura de las tablas. Se utiliza cuando se actualiza el módulo
- db/access.php: define los permisos.
- index.php: es una página para presentar la lista de todas las instancias en un curso
- view.php: se trata de una página para ver una instancia en particular

- lib.php: cualquiera/todas las funciones definidas para el módulo deben estar aquí. Si el módulo se llama "mod", entonces las funciones requeridas incluyen:
    - mod_add_instance() - código para añadir una nueva instancia.
    - mod_update_instance() - código para actualizar una instancia existente.
    - mod_delete_instance() - código para borrar una instancia.
    - mod_user_complete() - dada una instancia, imprime detalles sobre la contribución de un usuario.
    - mod_user_outline() - dada una instancia, devuelve un resumen de una contribución de un usuario.
- Finalmente, cada módulo tendrá algunos archivos de idioma que contienen cadenas para ese módulo.

3.4 TEMAS

Los temas definen la apariencia de un sitio. Con la distribución básica se proporciona una serie de temas simples, pero el usuario puede querer crear su propio tema, con sus propios colores, logo, estilos y gráficos a través del uso de XHTML y CSS.

- Los temas pueden ser configurados a nivel de sitio, curso y/o usuario.
- Cada página se maneja en forma individual por CSS.
- Las clases CSS que nombra el sistema utiliza inglés simple, es consistente y fácilmente comprensible
- Los nuevos módulos pueden decir a Moodle qué estilos necesitan e incluyen automáticamente éstos en el stylesheet.

Los temas se pueden basar en el tema estándar, que es muy simple pero funcional. Se pueden también basar en cualquier otro tema. Esto permite que se creen fácilmente familias de temas, o las variaciones de un tema.

3.5 IDIOMAS

Moodle ha sido diseñado para ser internacional. Cada cadena o página de texto que se presenta como parte de la interfaz surge de una serie de archivos de idioma. Cada idioma es un subdirectorio del directorio lang. La estructura del directorio lang es la que sigue:

- lang/en - directorio que contiene todos los archivos para un idioma.
- moodle.php - cadenas para la interfaz principal.
- assignment.php - cadenas para el módulo de tareas.
- choice.php - cadenas para el módulo consulta.
- forum.php - cadenas para el módulo del foro.
- journal.php - cadenas para el módulo del diario.
- quiz.php - cadenas para el módulo del cuestionario.
- resource.php - cadenas para el módulo de materiales.
- survey.php - cadenas para el módulo de encuesta.

Se llama a las cadenas desde los archivos usando las funciones: get_string() o print_string().

- lang/en/help - contiene todas las páginas de ayuda (para las ayudas emergentes sensibles al contexto)

Las páginas principales de ayuda están situadas aquí, mientras que las páginas específicas de cada módulo están localizadas en subdirectorios con el nombre del módulo. Con la función helpbutton, se puede insertar un botón de ayuda en una página.

No obstante también es posible editar los idiomas en línea, usando las herramientas web de Administración bajo "Idioma". Esto hace que sea fácil no sólo crear nuevos idiomas sino también refinar los existentes (Moodle, 2015f).

3.6 ESQUEMA DE BASES DE DATOS

Dada una base de datos funcionando con tablas definidas, el intencionalmente simple SQL usado en Moodle debe funcionar bien con una amplia variedad de marcas de bases de datos.

Existe un problema con la creación automática de nuevas tablas en una base de datos, que es lo que Moodle intenta hacer tras la instalación inicial. Debido a que cada base de datos es muy diferente de las otras, aún no existe una manera de hacer esto de manera independiente del tipo de plataforma. Para ayudar a la automatización en cada base de datos, pueden crearse esquemas que enumeren el SQL requerido para crear tablas en Moodle en una base de datos determinada. Estos son los archivos que hay en lib/db y dentro del subdirectorio db de cada módulo.

4. OTRAS CARACTERÍSTICAS DE MOODLE

ESCALABILIDAD

La infraestructura debe poder ampliarse o escalar para resolver el futuro crecimiento, tanto en términos de volumen de contenidos educativos como del número de estudiantes.

Moodle funciona con una amplia variedad de tecnologías de servidores web y bases de datos. Al igual que sucede con cualquier instalación de sistemas de software basados en servidor y con los sistemas de bases de datos, resulta crucial elegir muy cuidadosamente los equipos, el sistema operativo y el sistema de bases de datos, a fin de asegurar que el sistema puede afrontar un gran rendimiento.

Sería posible lograr un equilibrio de cargas en una instalación *Moodle* usando, por ejemplo, más de un servidor web si fuera necesario. Los servidores web separados consultarían la misma base de datos y apuntarían a la misma área de almacenamiento de archivos pero, en otro caso, la separación de las capas de la aplicación resulta del todo suficiente para hacer viable esta clase de *clustering*. De igual modo, la base de datos podría ser un conglomerado (*cluster*) de servidores (Moodle, 2015c).

Todo lo anterior implica que la arquitectura de *Moodle* facilita responder a futuras demandas, adaptando las tecnologías bajo las que se ejecuta. Esto sería posible incluso en un entorno vivo, a fin de mejorar el servicio sin interrupciones importantes.

FACILIDAD DE USO

Moodle es una aplicación sencilla y potente. Tiene un interfaz familiar e intuitivo para la navegación que hace fácil crear cursos, y gracias a su diseño modular hace fácil agregar contenidos que motiven al estudiante.

ALTA DISPONIBILIDAD

El *LMS* debe ser lo suficientemente robusto como para satisfacer las diversas necesidades de miles de estudiantes, administradores, creadores de contenidos y profesores simultáneamente.

Los patrones de uso variarán considerablemente dependiendo del contexto específico de la implementación, pero en términos generales *Moodle* presenta una interfaz basada en web de alta disponibilidad, permitiendo a los alumnos, tutores y administradores iniciar sesión de manera permanente y ejecutar sus tareas diarias.

COMUNIDAD ACTIVA

Moodle tiene una comunidad activa muy grande de gente que usa el sistema y que desarrolla nuevas funcionalidades y mejoras. Son varios cientos los desarrolladores con acceso a los archivos de la aplicación y muy frecuentados los foros de sugerencias, comentarios y evaluación de cada uno de los módulos.

La comunidad *Moodle* ha sido indispensable para el éxito del sistema. Con tantos usuarios globales, siempre hay alguien que puede contestar a una pregunta o dar un consejo. Al mismo tiempo, los desarrolladores y usuarios de *Moodle* trabajan juntos para asegurar la calidad, añadir nuevos módulos y funcionalidades y proponer nuevas ideas para desarrollar. Como los usuarios son libres de experimentar, muchas personas usan y prueban nuevas características, actuando como un gran departamento de control de calidad.

Las tres ventajas ya mencionadas hacen que *Moodle* sea único dentro del universo de los LMS, pero además cumple con los requisitos inherentes a la definición de un buen gestor de contenidos educativos. Se detallarán a continuación.

INTEROPERABILIDAD

Para admitir contenido de diferentes fuentes, y soluciones de equipos de cómputo o programas de diversos proveedores, el LMS debería intercambiar información utilizando estándares abiertos de la industria para implementaciones web.

En cuanto a la autenticación, *Moodle* admite autenticación contra *LDAP*, el protocolo estándar más utilizado con este propósito.

En lo referente al contenido, existen otros aspectos:

- *Moodle* admite la importación/exportación de Objetos Reutilizables de Aprendizaje empaquetados de acuerdo a los estándares *IMS Content Packaging* y *SCORM*.

- Las preguntas de los cuestionarios pueden ser exportadas en el formato estándar internacional *IMS QTI 2*.

- En *Moodle*, los canales de noticias RSS pueden integrarse en un sitio web completo o un curso.

- Se puede acceder a las discusiones de los foros como noticias RSS, y por lo tanto integrarse en otros sistemas o sitios web con funcionalidad RSS.

En *Moodle*, el uso de XML para importar/exportar información es un procedimiento estándar. El método servicios WEB de intercambio de información con otros sistemas está en continuo desarrollo. De hecho, será el empleado para la realización de este proyecto, como se verá más adelante.

ESTABILIDAD

La infraestructura del LMS puede soportar de manera confiable y efectiva una implementación productiva a gran escala las 24 horas del día, los 7 días de la semana.

SEGURIDAD

Al igual que sucede con cualquier solución colaborativa, el LMS puede limitar y controlar selectivamente el acceso de su diversa comunidad de usuarios a los contenidos en línea, recursos y funciones del servidor tanto interna como externamente.

5. VERSIONES DE MOODLE

Moodle como ya se ha explicado, se encuentra en continuo desarrollo, por lo que lo normal es que haya una nueva actualización de la última versión, cada 2 o 3 meses.

Para nombrar a sus versiones *Moodle* hace lo siguiente, no presenta una nueva versión hasta que no considera que los cambios respecto a la anterior sean sustanciales. Esto quiere decir, que si por ejemplo alguien presenta una nueva funcionalidad pero que no implica un cambio importante, dicha funcionalidad se presentará junto con otras en una actualización de la versión actual. Es por esto que si atendemos a la historia de las versiones, vemos como por ejemplo entre las versiones estables 2.6 y 2.7, tenemos hasta 8 actualizaciones de la versión 2.6, pero todas ellas (2.6.1, 2.6.2…) no suponen versiones

en sí mismas sino actualizaciones de la versión inmediatamente anterior (Moodle, 2015).

Este proyecto se ha desarrollado sobre *Moodle* 2.7.2 ya que era la última actualización estable disponible cuando se empezó a desarrollar el mismo (Julio 2014). Coincidiendo también con el que había instalado en ese momento en la escuela.

El proyecto del que partimos, se había desarrollado sobre *Moodle* 1.9 por lo que actualmente se encontraba obsoleto debido a que hay una serie de cambios muy importantes en la configuración de *Moodle.*

Principalmente estos cambios influyen de manera notable en la configuración de la base de datos, debido a cambios en el cifrado de la información y la supresión de muchas tablas, teniendo en cuenta también que la interacción entre dichas tablas ha cambiado en casi todos los casos.

Estos cambios se conocerán en detalle en el capítulo que se explica el desarrollo de este proyecto.

6. MOODLE MÓVIL

Como ya se comentó en el capítulo 1, cuando Fernando Herrero comenzó a desarrollar esta aplicación, no existía ninguna aplicación *Moodle* para dispositivos móviles. Actualmente podemos encontrar varias en Google Play. Las más destacables son las siguientes (atendiendo a sus valoraciones):

- LHU Moodle Mobile: Es una aplicación privada, para estudiantes de la Liverpool Hope University. No se puede disponer de ella para cualquier moodle.
- MDroid Legacy - Moodle mobile: Es una aplicación que empezó a desarrollarse, pero actualmente se encuentra obsoleta como podemos observar en la descripción de la misma en Google Play.
- Moodle Mobile: Es la aplicación más completa de todas las disponibles. Es la oficial de *Moodle*, en la descripción podemos ver como se ofertan casi todos los servicios de la versión para PC, pero atendiendo a los comentarios de los usuarios vemos como esto todavía está lejos de la realidad.
  Son muchos los que hacen referencia a que la aplicación está mal programada, y que tarda demasiado en cargar, o en otros casos directamente no funciona. Podemos ver en la Figura 3, los comentarios más recientes. En ellos podemos apreciar la disconformidad de los usuarios con esta App, sobre todo referidos a la lenta navegación debida a la mala optimización de la misma (Google Play, 2015).

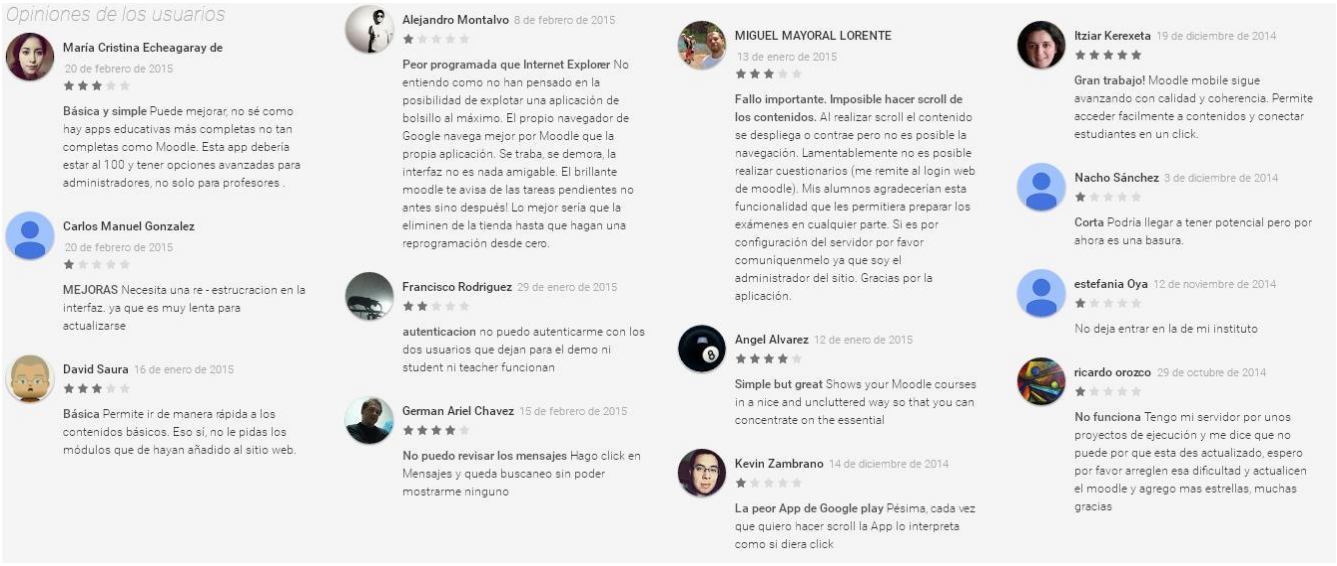

Figura 3 – Opiniones Moodle Mobile

La aplicación desarrollada lo que va a ofrecer, sin poder ofertar el total de funcionalidades que tiene *Moodle,* es el acceso a los recursos y la mensajería, de una manera rápida, sencilla y optimizada. Evitando así los problemas de los que hablan los usuarios de la app ya existente. Como desventaja no tendremos todas las posibilidades que nos da *Moodle* desarrolladas en nuestra App pero las que tendremos estarán 100% integradas funcionando de manera correcta. Dispondremos una interfaz simple pero efectiva, y realizaremos una optimización de las funciones para lograr una navegación rápida.

CAPÍTULO IV.

SERVICIOS WEB

1. INTRODUCCIÓN A LOS SERVICIOS WEB

Los Servicios Web (Web Service, o Web Services), atendiendo a la definición que da el W3C se pueden definir como *"Un conjunto de aplicaciones o de tecnologías con capacidad de interoperar en la web. Estas tecnologías intercambian datos entre sí con el fin de ofrecer un servicio. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la red"* (World Wide Web Consortium, 2013).

Como resumen podríamos decir que un Servicio Web es un tipo de *middleware* que permite la comunicación entre aplicaciones remotas que interactúan entre sí para presentar información dinámica al usuario.

Es necesaria una arquitectura de referencia estándar para poder proporcionar interoperabilidad entre estas aplicaciones y que al mismo tiempo sea posible su combinación para realizar operaciones complejas.

Las características deseables para poner el Servicio Web a disposición para su uso son las siguientes:

- Es necesario definir en primer lugar el Servicio Web. Para favorecer la interacción con otros se debe utilizar un lenguaje común que permita estructurar los datos que componen el servicio. Para ello se suele hacer uso de XML (*eXtensible Markup Language*).

- Un servicio debe poder ser accesible a través de la Web. Para ello debe utilizar protocolos de transporte estándares como HTTP, y codificar los mensajes en un lenguaje estándar que pueda conocer cualquier cliente que quiera utilizar el servicio.

- Un servicio debe contener una descripción de sí mismo. De esta forma, una aplicación podrá saber cuál es la función de un determinado Servicio Web, y cuál es su interfaz, de manera que pueda ser utilizado de forma automática por cualquier aplicación, sin la intervención del usuario. El lenguaje que se utiliza para la descripción del servicio es WSDL *(Web Service Description Language)*, y la publicación se del mismo se realiza mediante UDDI (*Universal Description, Discovery and Integration*), indiferentemente del carácter del servidor (público o privado).

- Debe poder ser localizado. Deberemos tener algún mecanismo que nos permita encontrar un Servicio Web que realice una determinada función. De esta forma

tendremos la posibilidad de que una aplicación localice el servicio que necesite de forma automática, sin tener que conocerlo previamente el usuario.

Desde el punto de vista del usuario, cuando se quiera solicitar un Servicio Web, se debe de disponer de un directorio que tenga las referencias a los servicios disponibles. Para intercambiar información entre el proveedor y el solicitante se debe de hacer uso de un protocolo de comunicaciones, como SOAP (*Simple Object Access Protocol*), que transmitirá los datos sobre HTTP, FTP o SMTP en formato XML, o REST, que intercambiará la información sobre HTTP en formato XML o JSON.

2. XML: eXtensible Markup Language

XML (*eXtensible Markup Language*, Lenguaje de Marcado eXtensible) es un lenguaje desarrollado por el *World Wide Web Consortium* basado en SGML (*Standard Generalized Markup Language*, Lenguaje de Marcado Generalizado Estándar). XML es utilizado para el almacenamiento e intercambio de datos estructurados entre distintas plataformas. Es un metalenguaje, es decir, puede ser empleado para definir otros lenguajes, llamados dialectos XML, como por ejemplo GML, MathML, RSS…

Al igual que HTML, XML es un lenguaje de marcado, pero a diferencia del primero que fue diseñado para mostrar datos, XML fue diseñado para transportar y almacenar esos datos, centrándose en el contenido.

Normalmente diferentes sistemas independientes trabajan con formatos incompatibles, XML se encarga de simplificar el intercambio de estos datos, ya que almacena los mismos en formato de texto plano, proporcionando un software independiente de la forma de almacenamiento de los datos.

La unidad fundamental de un documento XML es el elemento, o lo que es lo mismo: Un par de etiquetas de inicio y fin coincidentes y el contenido que se encuentra en el medio de las dos. En un documento XML debe de haber un único elemento raíz que contenga al resto de elementos, y los elementos deberán estar correctamente anidados.

Un ejemplo de una correcta anidación es el siguiente:

```
<etiqueta1><etiqueta2></etiqueta2></etiqueta1>
```

Un ejemplo incorrecto por el contrario sería este otro:

```
<etiqueta1><etiqueta2></etiqueta1></etiqueta2>
```

En un documento XML, todos los nombres de los elementos son *case sensitive*, es decir, sensibles a letras minúsculas y mayúsculas, teniendo que cumplir las siguientes normas:

- Pueden contener letras minúsculas, letras mayúsculas, números, puntos, guiones medios y guiones bajos.
- Asimismo, pueden contener el carácter dos puntos ":". No obstante, su uso se reserva para cuando se definan espacios de nombres.
- El primer carácter tiene que ser una letra o un guión bajo "_".

Las etiquetas en XML no están definidas sino que cada uno debe definir sus propias etiquetas.

Por otra parte, hay que tener en cuenta que, detrás del nombre de una etiqueta se permite escribir un espacio en blanco o un salto de línea. Por ejemplo, sintácticamente es correcto escribir:

<etiqueta1 >Nombre</etiqueta1>

Otro concepto importante es el de atributo. Estos pueden aparecer dentro de las etiquetas de inicio como pares “nombre=valor”. Además, un mismo atributo solo puede aparecer una vez en una etiqueta.

Las letras no inglesas (á, Á, ñ, Ñ...) están permitidas. Sin embargo, es recomendable no utilizarlas para reducir posibles incompatibilidades con programas que puedan no reconocerlas.

En cuanto al carácter guión medio "-" y al punto ".", aunque también están permitidos para nombrar etiquetas, igualmente se aconseja evitar su uso; el guión medio porque podría confundirse con el signo menos, y el punto porque, por ejemplo al escribir nombre.completo, podría interpretarse que completo es una propiedad del objeto nombre (Pes, 2014).

3. WSDL

WSDL (Web Services Description Language) es un lenguaje basado en XML utilizado para describir la funcionalidad que proporciona un servicio Web. Una descripción WSDL (fichero WSDL) de un servicio web proporciona una descripción entendible por la máquina (*machine readable*) de la interfaz del servicio Web, indicando cómo se debe llamar al servicio, qué parámetros espera, y qué estructuras de datos devuelve (Dept. Ciencia de la Computación, 2014).

WSDL describe un servicio utilizando varios elementos (etiquetas xml). Dichos elementos podemos clasificarlos como abstractos o concretos. La parte WSDL abstracta describe las operaciones y mensajes con detalle. En otras palabras, la parte abstracta de un WSDL especifica qué hace el servicio:

- Qué operaciones están disponibles
- Qué entradas, salidas, y mensajes de error tienen las operaciones
- Cuáles son las definiciones de los tipos para los mensajes de entrada, salida y error

En el mundo Java, podemos pensar en la parte abstracta de un WSDL como en la definición de una interfaz o una clase abstracta, con la definición de sus métodos, pero no sus implementaciones. La parte abstracta de un WSDL contiene dos componentes principales:

- Las operaciones que forman la definición de la interfaz
- Los tipos de datos para los parámetros de entrada, salida y error, de las operaciones

La parte WSDL concreta describe el cómo y dónde del servicio:

- Cómo tiene que llamar un cliente al servicio
- Qué protocolo debería usar
- Dónde está disponible el servicio

En el mundo Java podemos pensar en la parte concreta de un WSDL como en la implementación de la parte abstracta, aunque en términos de servicios Web, solamente describe dónde se encuentra dicha implementación para utilizarse. La parte concreta de un WSDL contiene dos componentes principales:

- Información de enlazado (binding) sobre el protocolo a utilizar
- La dirección en donde localizar el servicio

4. UDDI

UDDI nos permite localizar Servicios Web. Para ello define la especificación para construir un directorio distribuido de Servicios Web, donde los datos se almacenan en XML. En este registro no sólo se almacena información sobre servicios, sino también sobre las organizaciones que los proporcionan, la categoría en la que se encuentran, y sus instrucciones de uso (normalmente WSDL). Tenemos por lo tanto 3 tipos de información relacionados entre sí:

- Páginas blancas: Datos de las organizaciones (dirección, información de contacto, etc).

- Páginas amarillas: Clasificación de las organizaciones (según tipo de industria, zona geográfica, etc).

- Páginas verdes: Información técnica sobre los servicios que se ofrecen. Aquí se dan las instrucciones para utilizar los servicios. Es recomendable que estas instrucciones se especifiquen de forma estándar mediante un documento WSDL.

5. RELACIÓN DE LOS SERVICIOS WEB CON OTRAS TECNOLOGÍAS

Los servicios web son, principalmente, una tecnología de integración. Sin embargo, son independientes de la forma propiamente dicha. Las tecnologías de componentes para servicios web son definidas en forma común e interactúan en XML, según se mencionó anteriormente. Sin embargo, ya que el propio XML es independiente de lenguaje, los servicios web también lo son. Por tanto, los servicios web se pueden desarrollar en varios lenguajes de programación, como Java, Python, Android, Perl, C#, Basic y otros.

El caso que a nosotros nos interesa es la compatibilidad con Android, vamos a conocer las características más básicas de los dos protocolos de comunicación más usados a la hora de crear un servicio web y finalizaremos este apartado eligiendo uno de ellos, añadiendo una explicación de por qué nos resulta más útil en detrimento de los protocolos rechazados.

6. SOAP (Simple Object Access Protocol):

Se trata de un protocolo derivado de XML que nos sirve para intercambiar información entre aplicaciones.

Normalmente utilizaremos SOAP para conectarnos a un servicio e invocar métodos remotos, aunque puede ser utilizado de forma más genérica para enviar cualquier tipo de contenido (Dept. Ciencia de la Computación, 2014). Podemos distinguir dos tipos de mensajes según su contenido:

- Mensajes orientados al documento: Contienen cualquier tipo de contenido que queramos enviar entre aplicaciones.

- Mensajes orientados a RPC: Este tipo de mensajes servirá para invocar procedimientos de forma remota (Remote Procedure Calls). Podemos verlo como un tipo más concreto dentro del tipo anterior, ya que en este caso como contenido del mensaje especificaremos el método que queremos invocar junto a

los parámetros que le pasamos, y el servidor nos deberá devolver como respuesta un mensaje SOAP con el resultado de invocar el método.

Puede ser utilizado sobre varios protocolos de transporte, aunque está especialmente diseñado para trabajar sobre HTTP.

En primer lugar hay que empezar diciendo que Android no incluye en su SDK ningún tipo de soporte para el acceso a servicios web de tipo SOAP. Es por esto por lo que se dispone de una librería externa para hacer más fácil esta tarea. Entre la oferta actual, la opción más popular y más utilizada es la librería ksoap2-android. Esta librería es un fork, especialmente adaptado para Android, de la antigua librería kSOAP2. Este framework permitirá de forma relativamente fácil y cómoda utilizar servicios web que utilicen el estándar SOAP (Paszniuk, 2014).

7. RESTful

Los servicios del tipo REST son los más utilizados en la actualidad a la hora de crear un servicio Web.

Se trata de un protocolo cliente/servidor sin estado. Esto quiere decir que cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST). Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

Este protocolo tiene una serie de operaciones bien definidas como son, que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Las cuales veremos con más detenimiento en el capítulo de desarrollo del proyecto. Para poder realizar peticiones http en Android deberemos de usar la libería http-request (Amatellanes, 2014).

8. ELECCIÓN

REST y SOAP pueden utilizarse para implementar una funcionalidad similar, pero en general SOAP debe utilizarse cuando se necesita una característica concreta de SOAP, y REST generalmente es la mejor opción en caso contrario.

REST y SOAP muchas veces se utilizan una en lugar de la otra, pero son totalmente diferentes enfoques. REST es un estilo de arquitectura para generar aplicaciones de cliente-servidor. SOAP es una especificación de protocolo para intercambiar datos entre dos extremos.

Para el desarrollo de este proyecto se ha decidido realizar la implementación utilizando RESTful, atendiendo a los siguientes motivos:

- Como ya hemos explicado en su apartado, se trata de un tipo de implementación basada totalmente en especificaciones del protocolo HTTP, donde todo actúa como un recurso. Todos los mensajes se tratan de igual manera, sin diferenciar el contenido, esto nos reporta una sencillez evidente.

- Dichos mensajes son ligeros, lo cual nos hace ayuda a que la escalabilidad y el rendimiento no nos suponga un problema. Esta característica nos resulta especialmente útil, ya que hay que recordar que nuestra aplicación va a ser una aplicación para dispositivos móviles, y aunque los procesadores de los mismos cada vez son más potentes, todavía se encuentran lejos de los procesadores de los ordenadores convencionales o servidores.

- SOAP es una implementación mucho más estricta, se basa en un estándar oficial, mientras que RESTful tiene recomendaciones de implementación, pero no resultan tan rigurosas.

- RESTful a la hora de la representación de los datos, permite XML o JSON, mientras que SOAP especifica que deben de ser representados en XML.

- Por último cabe recordar que este proyecto parte de otro ya empezado, en el que RESTful era el protocolo elegido y por lo tanto atendiendo a las anteriores razones y a esta última, creemos que continuar el proyecto utilizando las mismas tecnologías que el proyecto del que se parte suponen la mejor opción.

9. REPRESENTACIÓN DE LOS DATOS

Como se ha explicado en el apartado anterior a este, RESTful permite representar los datos en formato XML o JSON, lo cual nos lleva a otra discusión, teniendo que elegir entre una de estas dos opciones. XML se explicó al comienzo de este capítulo como parte fundamental para poder entender bien los Servicios Web, JSON en cambio no ha sido explicada hasta ahora.

JSON (JavaScript Object Notation) es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML, el fácil uso en javascript ha generado un gran número de seguidores de esta alternativa.

Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

La principal ventaja de JSON frente a XML es que escribir un analizador sintáctico mediante JSON es mucho más sencillo que utilizando XML, y su procesado es más rápido en cualquier ordenador (Esquiva Rodríguez, 2013).

En entornos donde el flujo de datos es alto entre cliente y servidor y donde los tiempos de procesado son más bien pequeños, hace que sea más utilizado JSON, ya que los tiempos de respuesta se optimizan debido a su ligero peso.

La representación de datos elegida será JSON. Al igual que se ha explicado en la elección de RESTful, el proyecto del que partimos utiliza JSON como representación de datos, lo que ya nos supone una razón de peso para utilizarlo de nuevo. A parte de por esta razón, hemos elegido este tipo de representación por los siguientes motivos:

- Es mucho más simple que XML.
- JSON es la mejor herramienta para compartir datos. Esto es porque los datos están almacenados en vectores y registros mientras que XML almacena los datos en árboles. Ambos tienen sus ventajas, pero las transferencias de datos son mucho más fáciles cuando los datos se almacenan en una estructura que está familiarizada a los lenguajes orientados a objetos. Esto hace que sea muy sencillo importar datos desde un fichero JSON a Android, Perl, Ruby, Javascript, Python, y otros muchos lenguajes. Para hacer lo mismo con XML, necesitaría primero transformar los datos antes de que puedan ser importados. Por este motivo, JSON es un formato de fichero superior para las APIs web.
- Ambos ficheros XML y JSON son legibles para el ser humano. Al menos, son entendibles para los programadores que trabajan con estos formatos de

ficheros. Sin embargo, los ficheros JSON son más restrictivos y por lo tanto ligeramente más legibles. Esto se debe a que el número de formatos de datos permitidos por JSON es mucho menor que XML. Además, la estructura de los datos está más estandarizada con los ficheros JSON debido al hecho de que existen menos opciones cuando se compara con el formato XML.

CAPÍTULO V.

DESARROLLO DEL PROYECTO

Todas las funcionalidades de la aplicación móvil que se abordan en este proyecto, tienen algo en común: La necesidad de acceder a la base de datos de *Moodle*. Ya sea para insertar algún contenido o para acceder a información alojada en la misma.

Moodle soporta muchas bases de datos, pero lo más común y recomendable, siendo la opción que adoptan la mayoría de sistemas es *MySQL*. Como ya se explicó anteriormente para el desarrollo de este proyecto vamos a emplear *MySQL*, accediendo a la base de datos mediante ficheros *PHP* alojados en el servidor.

1. CONDICIONES PREVIAS

La implementación de este proyecto se ha llevado a cabo siempre teniendo presente que se trata de un complemento de la plataforma *Moodle*. El objetivo final ya mencionado en capítulos anteriores es:

1. Dar la posibilidad al usuario de registrarse si no lo ha hecho.
2. Previa identificación positiva:
    a. Acceder a los recursos disponibles en los cursos en los que el alumno esté matriculado
    b. Acceder al servicio de mensajería de *Moodle* desde la aplicación móvil

Por ello, para esta versión de la aplicación no se podrá realizar la subida de recursos a los cursos, dado que la aplicación está diseñada para poder descargar los recursos que se hayan subido a la plataforma vía web. Además, es necesario señalar que desde la aplicación únicamente se visualizarán recursos como ficheros de texto, imágenes, pdf, .docx…

Se podrá acceder de manera completa al servicio de mensajería de *Moodle* pudiendo ver por parte del usuario, los mensajes leídos, no leídos, los enviados y permitirá también enviar nuevos mensajes. Pero en esta versión no se tendrá acceso a foros, encuestas, chats u otras funcionalidades de la plataforma *Moodle*.

2. APLICACIÓN MÓVIL. DISEÑO DE LA INTERFAZ

2.1 INTRODUCCIÓN

En una aplicación Android la interfaz de usuario es construida usando objetos *View* y *ViewGroup*, u objetos que sean herederos de estos. De esta forma, *View* sirve de base para una serie de subclases llamadas *widgets*, que no son otra cosa que objetos de la interfaz de usuario completamente desarrollados, como campos de texto o botones. Por su parte, *ViewGroup* hace otro tanto de lo mismo sirviendo de base a las subclases llamadas *layouts*, las cuales ofrecen arquitecturas de disposición de elementos tipo *Linear, Tabular, Relative* y un largo etcétera.

Un objeto View es una estructura de datos cuyas propiedades almacenan los parámetros de disposición y contenido de un área específica de la pantalla del dispositivo. Además, este objeto es capaz de gestionar sus propias medidas, su disposición, el *scroll* si es que lo hay y prácticamente cualquier interacción por parte del usuario en el área en el que está definido. Los objetos *ViewGroup* son utilizados como puntos de ramificación del árbol de jerarquía de vistas definido en un archivo *XML layout*, y de ellos colgarán distintos elementos *View* o *ViewGroup*, tal y como se puede observar en la Figura 4:

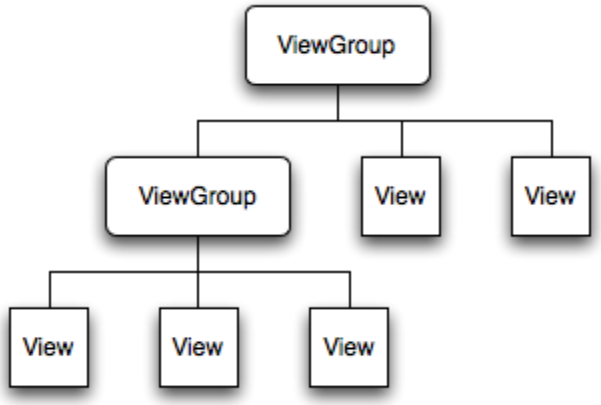

Figura 4 – Jerarquía de Vistas en Android

La jerarquía puede ser tan simple o compleja como el desarrollador desee y puede ser construida a partir de *layouts* y *widgets* predefinidos por *Android* o implementando objetos *View* personalizados. Para unir el árbol definido en la jerarquía de vistas a la pantalla del dispositivo para el renderizado es necesario que la *Activity* a la que se quiere adjuntar la jerarquía de vistas llame al método *setContentView()* pasando una referencia al nodo del objeto raíz. El sistema *Android* utilizará dicha referencia para dibujar el árbol de vistas en la pantalla.

Un *widget* es un objeto que extiende la clase *View* y que sirve como interfaz para la interacción con el usuario. *Android* adjunta un conjunto de *widgets* completamente implementados, como botones, *checkboxes* y campos de entrada de texto entre muchos otros que permiten construir de una forma rápida y eficaz la interfaz de usuario de cualquier aplicación.

En cuanto a los *layouts*, es importante destacar que esta palabra tiene dos acepciones en el entorno de *Android*. Por una parte, los *layouts* son los archivos XML utilizados para definir la jerarquía de vistas asociada a una *Activity*. Cada elemento del archivo XML será un objeto *View* o *ViewGroup* (o un descendiente de los mismos). La segunda acepción de *layout* hace referencia a todas las subclases de *ViewGroup* que pueden ser añadidas a la jerarquía de vistas.

El nombre de los elementos del archivo XML representa la clase de Java a la que hace referencia el nodo, por lo que un elemento con el nombre <*TextView*> hará referencia a un objeto heredero de *View TextView*, mientras que un nodo <*LinearLayout*> representará a la clase heredera de *ViewGroup LinearLayout (Herrero Herrero, 2013)*.

2.2 INTERFAZ DE LA APLICACIÓN MÓVIL

A la hora de desarrollar la interfaz de esta aplicación móvil, se ha optado por seguir uno de los principios de diseño de *Moodle*: la simplicidad. Por lo tanto se va a utilizar el mínimo de interfaz necesario para obtener la funcionalidad que se desea implementar.

En la gran mayoría a excepción de los menús, el *layout* de la aplicación estará formado por *ListViews*, con las que se podrá visualizar de manera fácil e intuitiva por un lado los cursos, secciones y recursos de cada curso, y por otro lado los mensajes, recibidos, enviados y no leídos. Así como la lista de usuarios a los que podemos enviar un nuevo mensaje.

En este apartado, se irán explicando las diferentes pantallas que el usuario se va a encontrar al navegar por la aplicación, todas aquellas acompañadas de una imagen que ayude a entender lo que el usuario va a ver al ejecutar la aplicación.

Los menús que permitirán el acceso a las diferentes secciones están desarrollados con buttons con imágenes indicativas de cada sección, las cuales permitan una navegación lo más intuitiva posible.

PANTALLA DE INICIO

Cuando se inicia la aplicación, en primer lugar aparece una pantalla de autenticación de usuarios, tal y como podemos observar en la Figura 5. Esta pantalla consta por un lado de dos campos a rellenar: el nombre de usuario y la contraseña, más un botón de *login* el cual el usuario pulsará cuando crea que ha introducido los datos correctamente. Por otro lado debajo de este botón hay un rótulo que indica "Nuevo usuario". Al pulsar sobre este último rótulo se accederá a una pantalla que permite el registro de nuevos usuarios.

Los campos a introducir en los campos antes mencionados se tratan de los datos que cada usuario utilice para acceder a su cuenta de *Moodle* vía web. Para acceder a la aplicación es necesario introducir el valor correcto de estos dos campos. En caso de introducir mal alguno de los dos campos, saltará un cuadro de diálogo de aviso indicando al usuario que debe introducir correctamente los datos. Si el usuario está registrado en la aplicación pero no confirmado por el administrador, de igual manera aparecerá otro cuadro de diálogo *toast* avisando de la incidencia.

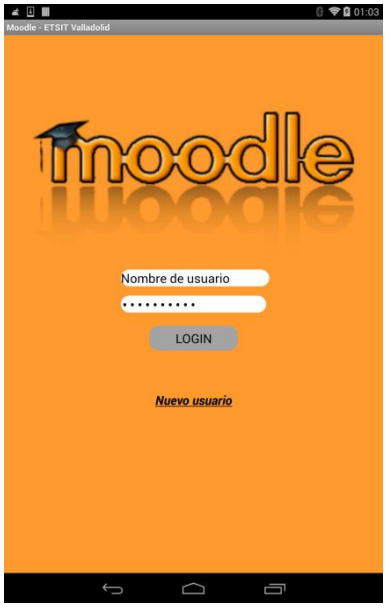

Figura 5 – Pantalla de inicio

PANTALLA MENÚ PRINCIPAL

El menú principal, el cual se puede ver en la Figura 6, consta de dos Buttons, uno que permite acceder a la mensajería representado por un sobre y otro que permite el acceso a los cursos o asignaturas en los que el usuario está matriculado, representado por el icono de una libreta y rotulado con la palabra asignaturas. Pulsando cualquiera de los dos iconos se accederá a una sección u otra.

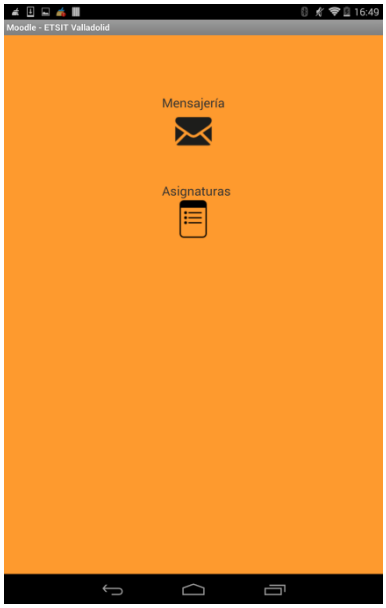

Figura 6 – Pantalla de Menú Principal

PANTALLA DE REGISTRO DE NUEVO USUARIO

Si el usuario selecciona en la pantalla principal el rótulo de nuevo usuario accederá a esta pantalla (Figura 7). Dicha pantalla no es más que un formulario de registro, el cual una vez completado y tras pulsar el botón de "Registrar nuevo usuario" se lanzará un mensaje toast indicando que el nuevo usuario queda registrado a la espera de ser validado por el administrador de Moodle.

Figura 7 – Pantalla Registro Nuevo Usuario

PANTALLA DE CURSOS

Una vez autenticado en la aplicación, y tras pulsar el icono asignaturas en la pantalla de menú principal aparecerá un listado de los cursos en que el usuario esté matriculado. Simplemente seleccionando uno de ellos se podrá acceder a las secciones en que está dividido el curso, de la manera que se observa en la Figura 8.

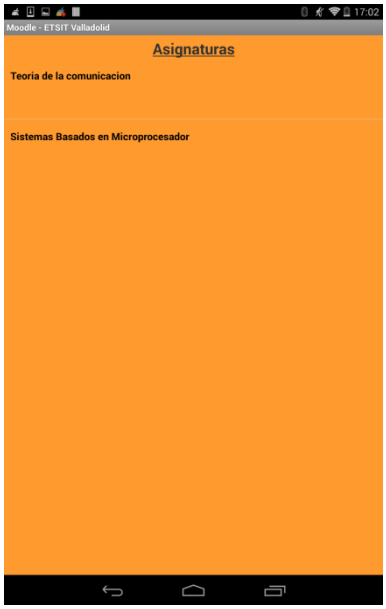

Figura 8 – Pantalla de Cursos

PANTALLA SECCIONES

De la misma manera que en la pantalla de cursos, en esta pantalla se mostrará un listado de las secciones en que esté dividido el curso que se ha seleccionado en la pantalla anterior. E igual que antes, al seleccionar una sección se accederá a los recursos que contenga dicha sección.

En este caso no siempre tiene por qué haber recursos disponibles en una sección. En realidad, toda sección tendrá contenidos, pero se vuelve a hacer referencia a los objetivos del proyecto en este punto: los foros, encuestas u otros contenidos cuya acción por parte del usuario sea diferente a la de descarga no serán tratados. Así, en caso de que no haya recursos, se observará en pantalla. Atendiendo a la Figura 9, se observa cómo hay disponibles tres temas, si estos no existieran, donde ahora se observa "Tema 1", estaría en su lugar escrito, "No hay recursos disponibles para esta asignatura".

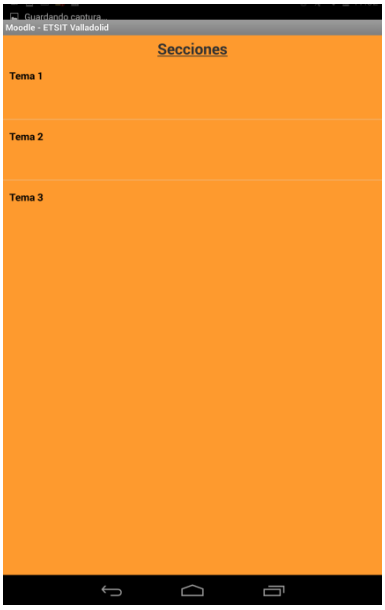

Figura 9 – Pantalla de secciones

PANTALLA RECURSOS

Similar a las anteriores (Figura 10), se verá un listado de los recursos que están disponibles en la sección que se haya seleccionado. Cuando se seleccione un recurso se procederá automáticamente a su descarga, se observará en la parte superior que un icono representativo de una descarga reciente, correspondiente al recurso descargado. Tal y como observamos en la Figura 11. Pulsando sobre dicho indicador se procederá a abrir dicho recurso.

Figura 10 – Pantalla de Recursos

Figura 11 – Pantalla de Descarga

PANTALLA MENÚ MENSAJES

Al igual que en la pantalla menú principal, en este caso se muestran una serie de buttons que conforman un menú referido al apartado de mensajería de Moodle, dicho menú está capturado en la Figura 12. En dicho menú se observan 5 opciones:

1. Mensajes no leídos: representado por un sobre cerrado, si el usuario pulsa sobre este botón accederá a una lista de mensajes no leídos.
2. Mensajes leídos: representado por un sobre abierto, si el usuario pulsa sobre este botón se accederá a una lista de mensajes previamente leídos. Tanto por a través de la aplicación móvil o a través de la plataforma moodle del PC.
3. Enviados: representados por un sobre con una flecha, permite al usuario consultar una lista completa de los mensajes que ha enviado.
4. Nuevo mensaje: representado por una hoja en blanco, permite al usuario acceder a una lista de destinatarios a los cuales puede enviar un nuevo mensaje.
5. Menú principal: permite al usuario volver a la pantalla de menú principal.

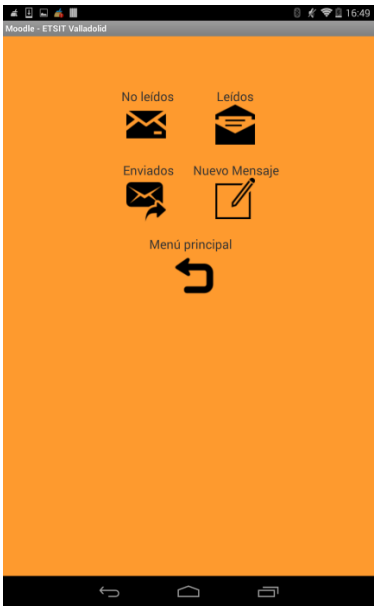

Figura 12 – Menú Mensajes

PANTALLA MENSAJES NO LEÍDOS

Al seleccionar en el menú de mensajes la opción de mensajes no leídos, el usuario visualizará únicamente aquellos mensajes que no se hayan leído, o bien a través de la aplicación móvil o a través de la plataforma para PC. Si el usuario no dispone de ningún mensaje no leído, en esta pantalla aparecerá un mensaje del tipo "No tiene mensajes sin leer".

En la parte inferior el usuario dispone de un botón para vaciar la bandeja de entrada como se puede observar en la Figura 13, al seleccionarlo se accederá a la pantalla de la Figura 14, y los mensajes que aparecían como no leídos al volver a entrar en esta pantalla ya no aparecerán y se encontrarán en el apartado de mensajes leídos.

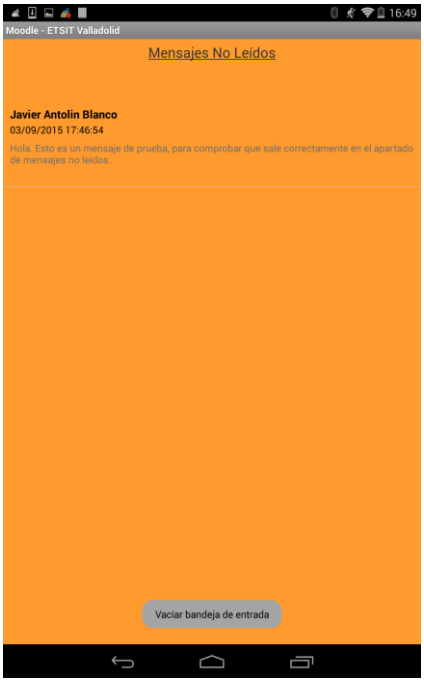

Figura 13 – Mensajes No Leídos

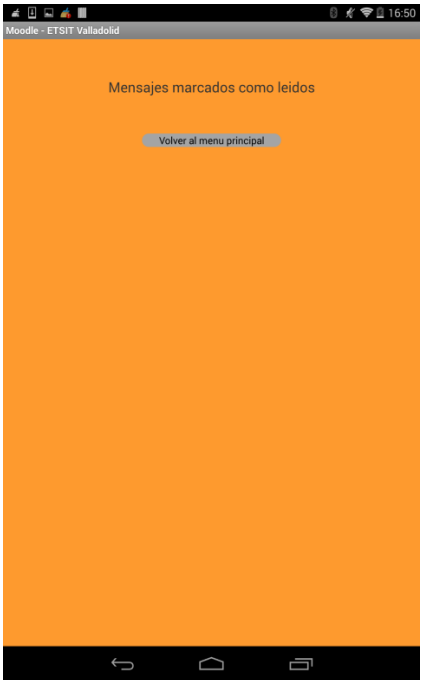

Figura 14 – Mensajes Marcados Como Leídos

PANTALLA MENSAJES LEÍDOS

Es igual que la pantalla de mensajes no leídos que se observaba en la Figura 13, pero en este caso aparecerán los mensajes en los que el usuario sea destinatario pero que ya hayan sido leídos por el mismo. En ambos casos cabe destacar que si el usuario selecciona cualquier mensaje (tanto en la pantalla de leídos como no leídos), se accederá a una vista individual como se observa en la Figura 15 a través de la que se podrá contestar a dicho mensaje. Si seleccionamos la opción de responder accederemos a la pantalla escribir nuevo mensaje. En la que el receptor por defecto del mensaje que va a escribir el usuario será el emisor del mensaje que está seleccionado.

Figura 15 – Mensaje Recibido Individual

PANTALLA ENVIAR NUEVO MENSAJE

Al seleccionar esta opción en el menú de mensajería, el usuario verá una lista formada por aquellas personas a la que tiene permiso para enviar un mensaje (ver Figura 16).

Esta lista de nombres la forman, alumnos y profesores que coincidan con el usuario en alguna asignatura.

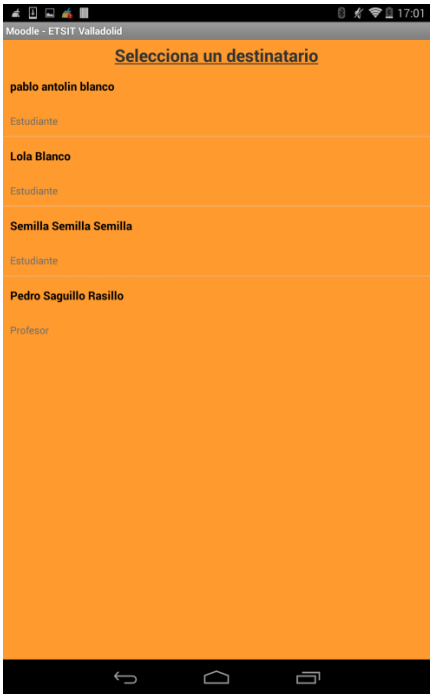

Figura 16 – Lista Destinatarios

Una vez seleccionado un destinatario, accederemos a otra pantalla, la cual se puede observar en la siguiente figura.

Figura 17 – Nuevo Mensaje

En esta pantalla observamos un cuadro de entrada de texto en el que se introducirá el mensaje que el usuario quiera enviar. Por otro lado hay dos botones, uno para enviar el mensaje de manera efectiva y otro para cambiar el destinatario, que al ser pulsado manda al usuario al paso anterior.

Una vez enviado el mensaje, el usuario verá por pantalla un mensaje del tipo "Mensaje enviado correctamente" y será redirigido al menú principal.

Este mensaje recién enviado, se podrá comprobar en el apartado de mensajes enviados acto seguido de haber sido enviado.

PANTALLA MENSAJES ENVIADOS

Es una pantalla del mismo tipo que la de recibidos y no leídos, está formada por un *listview* y permite ver los mensajes que ha enviado el usuario, junto a información como la fecha y el receptor, pulsando sobre un determinado mensaje el usuario tendrá acceso a una vista individualizada del mismo.

3. ACCESO A LA BASE DE DATOS DE MOODLE

3.1 ESTRUCTURA DE LA BASE DE DATOS DE MOODLE

La base de datos de *Moodle* tiene alrededor de 350 tablas, dato que de primeras nos puede resultar abrumador y del cual podríamos deducir una gran complejidad. Dicha complejidad está presente, ya que cada funcionalidad de *Moodle* tiene de manera intrínseca una interacción con un gran número de estas tablas.

Todas las tablas de la base de datos de *Moodle* comienzan con el prefijo "mdl_" continuado de una palabra clave que nos ayuda a ubicar dicha tabla. Por ejemplo para el caso de los usuarios, la tabla principal será "mdl_user".

Refiriéndonos a este proyecto en concreto, es buen momento para recordar que funcionalidades tienen relación con la base de datos del servidor *Moodle*:

- Registro de nuevos usuarios.
- Logeo de usuarios registrados.
- Visualización de cursos en los que estamos matriculados y descarga de recursos asociados a esos cursos.
- Servicio de mensajería interna de *Moodle*.

De manera muy resumida, acabamos de enumerar las funcionalidades que desarrollaremos en este proyecto, cada una de ellas está relacionada con una serie de

tablas de la base de datos. Por lo tanto lo que vamos a hacer no es explicar las 350 tablas, sino únicamente, explicaremos aquellas tablas con las que hemos tenido que interactuar para desarrollar nuestra aplicación. Y más concretamente explicaremos los campos de cada tabla que nos han resultado de interés. No es necesario conocer el resto de tablas para poder entender perfectamente el desarrollo de nuestro trabajo.

Mdl_user

Esta tabla contiene información de los usuarios registrados en el sistema, es posiblemente la tabla más importante para nuestro proyecto, ya que de manera directa o indirecta interviene en todas nuestras funcionalidades.

Los campos *username* y *password* recogen el nombre y la contraseña respectivamente de cada usuario registrado. Estos campos son los que utiliza cualquier persona para realizar la autentificación en nuestra aplicación.

El campo *password* está cifrado con la función *password_hash().* Esta función lo que hace es crear un nuevo hash de contraseña usando un algoritmo de hash fuerte de único sentido. Para lograr un cifrado más complejo y aumentar la seguridad a mayores utiliza el algoritmo *PASSWORD_BCRYPT* (Php, 2013). Usar el algoritmo *CRYPT_BLOWFISH* para crear el hash. Producirá un hash estándar compatible con *crypt()* utilizando el identificador "$2y$". El resultado siempre será un *string* de 60 caracteres, o FALSE en caso de error. Por eso si observamos en la base de datos de *Moodle* este campo concretamente, no veremos la contraseña tal y como la introduce el usuario a la hora de registrarse, sino que veremos algo del tipo:

$2y$ + 54 caracteres aleatorios producto del cifrado anterior

Esta forma de cifrado se instauró a partir de la versión 2 de *Moodle* y posteriores. Esto supuso un gran cambio para nosotros, ya que partíamos de una funcionalidad que ya estaba desarrollada por Fernando Herrero, como era el "Logeo" pero que había quedado obsoleta ya que para versiones anteriores a la 2, el cifrado del campo *password* era únicamente un cifrado md5, lo cual simplificaba mucho el recurso PHP. Esta nueva forma de cifrado que hay actualmente, ha obligado a rehacer la parte de autentificación ya que se encontraba obsoleta para versiones 2 y posteriores de *Moodle*.

Otro campo importante de la tabla *mdl_user* es el campo *id.* Este campo es un entero autoincremental con cada entrada en la tabla y que al ser diferente para cada usuario lo identifica de manera inequívoca e individual.

Por último el campo *confirmed* nos indicará con un "0" o con un "1" si el usuario está confirmado por el administrador. Ya que el usuario puede estar registrado, pero no confirmado, motivo por el que se le impediría entrar en el sistema. En la figura, podemos ver algunos de los campos que componen esta tabla. Mostramos únicamente los que han influído en el desarrollo de este proyecto.

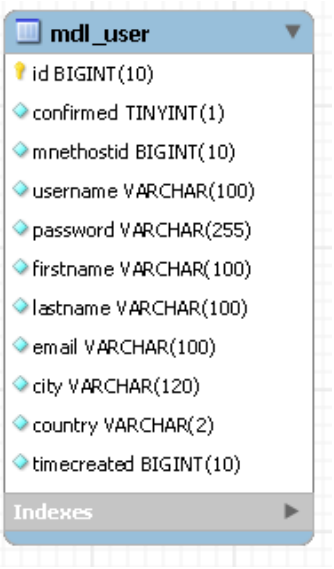

Figura 18 – Campos de Mdl_user

Tablas relacionadas con el módulo de mensajería de Moodle:

Mdl_message

Esta tabla recoge todos los mensajes que todavía no se han leído, únicamente esos mensajes. Es una tabla importante para nosotros ya que interaccionaremos cuando queramos mostrar los mensajes no leídos. Los campos *useridto* y *useridfrom* identifican el receptor y el emisor del mensaje respectivamente. Ambos campos están relacionados con el campo id antes mencionado de la tabla *mdl_user*, ya que lo que estamos haciendo con ellos es identificar que usuarios están intercambiando los mensajes.

El campo *subject* nos indica con el nombre completo real y no el identificador con el que usuario está enviando el mensaje. Y por último el campo *fullmessage* recoge el mensaje completo que se visualizará al abrir el mensaje.

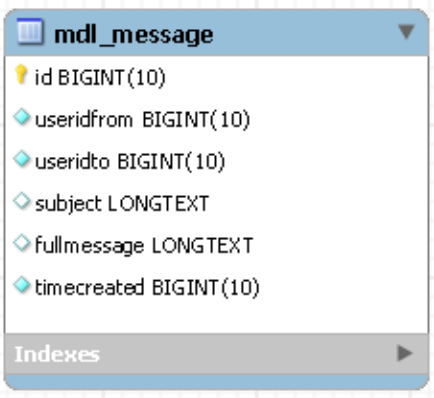

Figura 19 – Campos de Mdl_message

Mdl_message_read

Es una tabla muy similar al anterior, con la salvedad de que esta recoge únicamente los mensajes ya leídos. Un mensaje que esté guardado en esta tabla no puede estar en la tabla *mdl_message* y viceversa.

Los campos anteriormente descritos son comunes para esta tabla y tiene otro campo añadido, que no se encontraba en la tabla de mensajes no leídos. Este campo es el de *timeread* este campo hace referencia al momento en el que se leyó el mensaje, es una fecha en formato *timestamp* de *unix.*

Más tarde al explicar cada funcionalidad implementada veremos como al marcar los mensajes como leídos, se mueven de una tabla a otra los mismos, o como por ejemplo a la hora de ver todos los mensajes enviados se muestran los mensajes de ambas tablas en los que el *id* del solicitante sea el *useridfrom* de ambas tablas.

Figura 20 – Campos de Mdl_message_read

Tablas relacionadas con los cursos y la descarga de contenidos.

Como ya se ha explicado el proyecto del que se parte a la hora de realizar esta aplicación (Herrero Herrero, 2013), se encontraba obsoleto, principalmente en cuanto a las consultas de los recursos PHP, ya que a partir de la versión 2. De *Moodle* toda la configuración de la base de datos referida a recursos y cursos también cambió al igual que, como antes se comentaba, cambió el cifrado de las contraseñas.

En las versión de *Moodle* 1.9 y anteriores todo lo relativo a los cursos se desarrollaba entre las tablas *Mdl_course_display*, *Mdl_course*, *Mdl_course_sections, Mdl_course_modules, Mdl_course_label y Mdl_resource*. Actualmente la tabla Mdl_course_display ha desaparecido y la tabla *Mdl_course_modules* ha cambiado de configuración respecto a sus campos. La tabla *Mdl_course_display* era la que contenía la relación entre los usuarios y

los cursos, nos permitía de una manera muy sencilla saber en qué cursos estaba matriculado cada usuario (Pueyo, 2009).

Toda la información que antes se recogía en dicha tabla ahora se reparte entre las tablas *mdl_role_assignments, mdl_context* y *mdl_course* que explicaremos a continuación.

La supresión de esta tabla hace que el trabajo desarrollado por Fernando Herrero del cual partíamos se encontre obsoleto para una versión actual de Moodle, ya que como estamos indicando la configuración de la base de datos cambia de forma significativa en este caso también.

Mdl_role_assignments

Esta tabla es muy importante ya que nos relaciona cada usuario, con el conjunto de cursos, foros y demás módulos de *Moodle*. Esto lo hace de la manera siguiente: El parámetro *userid* es equivalente al parámetro *id* de la tabla mdl_user, por lo tanto identifica de manera inequívoca a cada usuario.

Por otra parte otro campo importante es el de *contextid* el cual hace referencia a un determinado módulo. Con la información que tenemos en la esta tabla no podemos saber si se refiere por ejemplo a un curso, o a un foro por ejemplo, pero para esto está la tabla *mdl_context* que explicaremos a continuación, gracias a esta tabla se hará distinción de un tipo de módulo u otro. El campo *contextid* lo que nos hace es decir el usuario "X" tiene acceso al foro/wiki/curso con *contextid* "Y" donde X e Y son dos números enteros.

El campo *roleid* nos indica que rol tiene el usuario al que se refiere el *userid* dentro del módulo que indica el *contextid*. Los dos *roleid* más importantes son el "3" que identifica a los profesores y el "5" que identifica a los estudiantes (Pueyo, 2009).

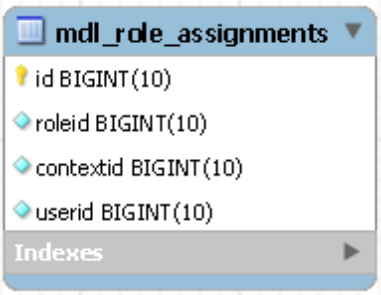

Figura 21 – Campos de Mdl_role_assignments

Mdl_context

Es una tabla relacionada estrechamente con la anterior, ya que sin esta, la tabla mdl_role_assignments no se puede entender.

El campo *id* de esta tabla es el mismo que el campo *contextid* de la tabla anterior. El campo *contextlevel* nos dice de qué tipo de módulo se trata, si es un "50" nos indica que es un curso. Solo nos vamos a centrar en este tipo ya que es el que nos interesa para desarrollar nuestra aplicación.

Por último el campo *instanceid* nos proporciona la relación con el campo *id* de la tabla mdl_course, identificando cada curso en concreto (Pueyo, 2009).

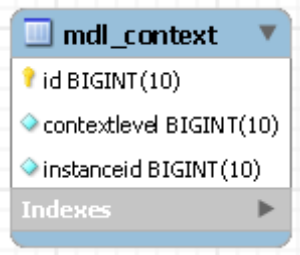

Figura 22 – Campos de Mdl_context

Mdl_course

Esta tabla recoge la información principal de cada curso, un campo *id* (entero incremental) al igual que el campo *id* de la tabla mdl_user identifica de manera única a cada curso. El nombre completo del curso se recoge en el campo *fullname.* Está relacionado con las tablas anteriores con el campo *instanceid* de la tabla *mdl_context*.

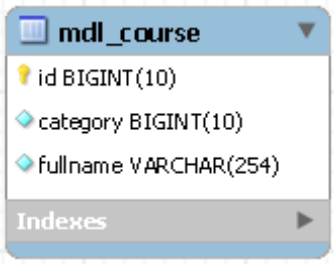

Figura 23 – Campos de Mdl_course

Mdl_course_modules

Esta tabla contiene información acerca de los módulos que están visibles en cada curso. Contiene un campo id que, como siempre, identifica inequívocamente una instancia de un módulo dentro de un curso. Además, contiene un campo course que señala la relación

con la tabla mdl_course, indicando que un determinado módulo pertenece a un determinado curso. Existen otros dos campos de interés en esta tabla: module e instance.

El campo module es un entero que va a indicar el tipo de módulo que se está tratando, si es igual a 17 se tratará de un recurso y si es igual a 12 será una etiqueta. Este campo está relacionado con el campo id de la tabla mdl_modules, pero al trabajar únicamente con estos dos tipos, nos evitamos una nueva consulta, y de manera interna en el recurso php identificamos cada tipo sin tener que recurrir a una nueva consulta que relacione a la tabla mdl_modules.

El campo instance también es un entero y va a indicar la relación con una instancia determinada de cada módulo. Pero al contrario que el resto de relaciones, en este caso no se va a hacer referencia al campo id de una tabla única, sino que la tabla en la que se ha de buscar la instancia variará en función del campo module de esta misma tabla (Herrero Herrero, 2013).

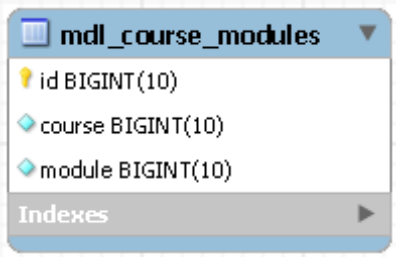

Figura 24 – Campos de Mdl_course_modules

Mdl_label

Esta tabla contiene la información de los rótulos de las secciones en que se divide cada curso. Los principales campos son un id autoincremental que identifica unívocamente cada rótulo con un curso, el campo name que es el propio rótulo y el campo course que señala la relación con la tabla mdl_course, indicando a qué curso pertenece ese rótulo (Herrero Herrero, 2013).

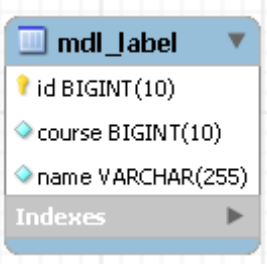

Figura 25 – Campos de Mdl_label

Mdl_resource

Esta tabla también ha cambiado significativamente, sigue conteniendo información acerca de los recursos que se encuentran disponibles en cada sección de cada curso. Los más destacados son el id autoincremental, el campo course que indica a qué curso pertenece ese recurso y el campo name que es el nombre del recurso.

En las versiones 1.9 y anteriores había un campo type que indicaba el tipo de recurso (html, file o text, entre otros) y el campo reference que indicaba la ruta relativa de cada recurso. Todo esto ya no existe, debido a que el cifrado de los recursos disponibles en *Moodle* ha cambiado. Como explicaremos más adelante la ruta relativa para poder acceder a estos recursos también se ha visto modificada. En este punto cobra importancia la tabla *Mdl_files* que ocupa el vacío producido al modificar los campos de la tabla Mdl_resource a partir de la versión 1.9.

Figura 26 – Campos de Mdl_resource

Mdl_files

Esta tabla (ver Figura 27) es la que definitivamente nos va a permitir construir la ruta relativa para poder descargar el recurso que hayamos seleccionado. Nos interesan los campos *contenthash* y *source,* los cuales guardan información de la ruta relativa y el nombre completo del recurso respectivamente. Más adelante en el apartado acceso a los recursos se explicará de manera detallada.

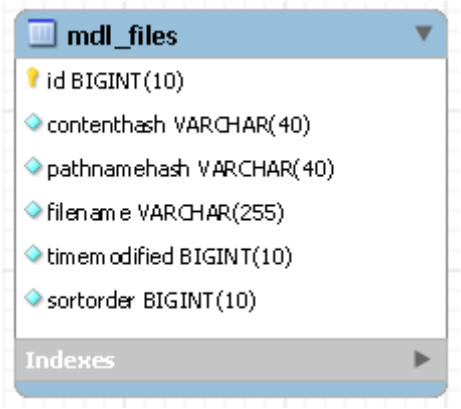

Figura 27 – Campos de Mdl_files

3.2 ACCESO A LOS RECURSOS

El directorio en el que se almacenan todos los recursos que se suben a Moodle es la carpeta moodledata. La ubicación de esta carpeta se decide a la hora de instalar la plataforma en el servidor. Por defecto, suele estar un nivel por encima del directorio web del servidor. Esto implica que los recursos no van a estar visibles de cara al exterior, por lo que no se podrían descargar directamente a la aplicación. Este hecho se debe a temas de seguridad, ya que es mucho más seguro que los recursos no estén accesibles dado que de esta manera no se podrán realizar ataques tratando de modificar su contenido, borrarlos o incluir nuevos recursos con contenido dañino para los equipos. Por lo tanto, aunque se modificase en la instalación la ubicación del directorio moodledata, sería bastante peligroso incluirlo en el directorio web del servidor.

La alternativa que se ha tomado para poder acceder a los recursos es realizar un duplicado de los recursos, de manera que la carpeta moodledata quede fuera del directorio web del servidor y en el mismo se incluya un directorio en el que se grabarán los ficheros paralelamente al moodledata. Este nuevo directorio será una copia exacta de moodledata, por lo que a partir de la ruta relativa del recurso se podrá construir una ruta absoluta que sí nos llevará al recurso que se haya seleccionado para poder descargarlo.

Para versiones anteriores a la 1.9 se descargaban los archivos con una dirección del tipo:

"http://dominio_del_servidor/moodle/nombre_de_nuestra_carpeta/id_curso/ruta_relativa"

Entonces de manera sencilla una vez que se había conseguido hallar el campo reference de la tabla mdl_resource se tenía la ruta relativa para llegar a un determinado recurso. Era bastante directo pensar que construyendo la ruta absoluta a partir de la ruta relativa

se podía acceder a ese recurso sin problemas. Sin embargo, todo lo anterior cambió para versiones posteriores a la 1.9, se suprimió el campo reference que nos daba la ruta relativa y se cifro la misma, aumentando la complejidad a la hora de construir una dirección que nos permitiera acceder al recurso.

La ubicación de la carpeta *moodledata* que hemos explicado anteriormente no varía, y la solución que hemos adoptado es la misma. Ahora bien, la ruta para acceder a un recurso si ha variado de forma significativa, debido a que todos los archivos se cifran con un *hash p*erdiendo su nombre original, algo que antes no ocurría.

El campo *contenthash* de la tabla *mdl_files* nos permite construir la ruta de la siguiente manera. Si un archivo por ejemplo tiene el *contenthash* siguiente:

"c694eae552d3cb8b55196e5ded687e60e4622c7a"

La ruta que nos permitirá llegar a dicho archivo será la siguiente:

"moodledata/filedir/c6/94/ c694eae552d3cb8b55196e5ded687e60e4622c7a"

Siempre se sigue ese esquema, el siguiente campo después de la carpeta filedir, es una carpeta que tiene como nombre los dos primeros caracteres del *contenthash*, luego otra carpeta con los dos siguientes y más tarde el archivo que tiene como nombre el *contenthash* completo.

Nuestro recurso PHP de descarga de archivos, lo que hará será recibir el *timemodified* de un determinado recurso mediante POST, extraerá por un lado el *contenthash* del recurso que tenga dicho *timemodified* y por otro el *source* , o lo que es lo mismo el nombre real del archivo con su extensión.

Utilizamos el parámetro *timemodified* porque entendemos que es realmente complicado que vayan a coincidir dos archivos con este mismo campo, ya que incluye hasta los segundos. Esto nos permite tener un diferenciador individual de cada recurso.

Si atendemos a la tabla *mdl_files* vemos que hay varias filas con un mismo *timemodified* refiriéndose al mismo archivo pero con distintos *contenthash* esto nos puede llevar a pensar a que hay varias rutas para llegar al mismo archivo y no es correcto. El único *contenthash* que nos permite acceder al archivo seleccionado es aquel que tenga un "1" en el campo *sortorder* de la tabla mdl_files. De manera interna nuestro PHP ya se encarga de cifrar todos los distintos *contenthash* y construye la ruta válida con el correcto (Hernández, 2010).

3.3 ESTRUCTURA EMPLEADA DE LA BASE DE DATOS DE MOODLE

En la Firgura 28 se puede observar la relación entre las 10 tablas que intervienen en el desarrollo de este proyecto:

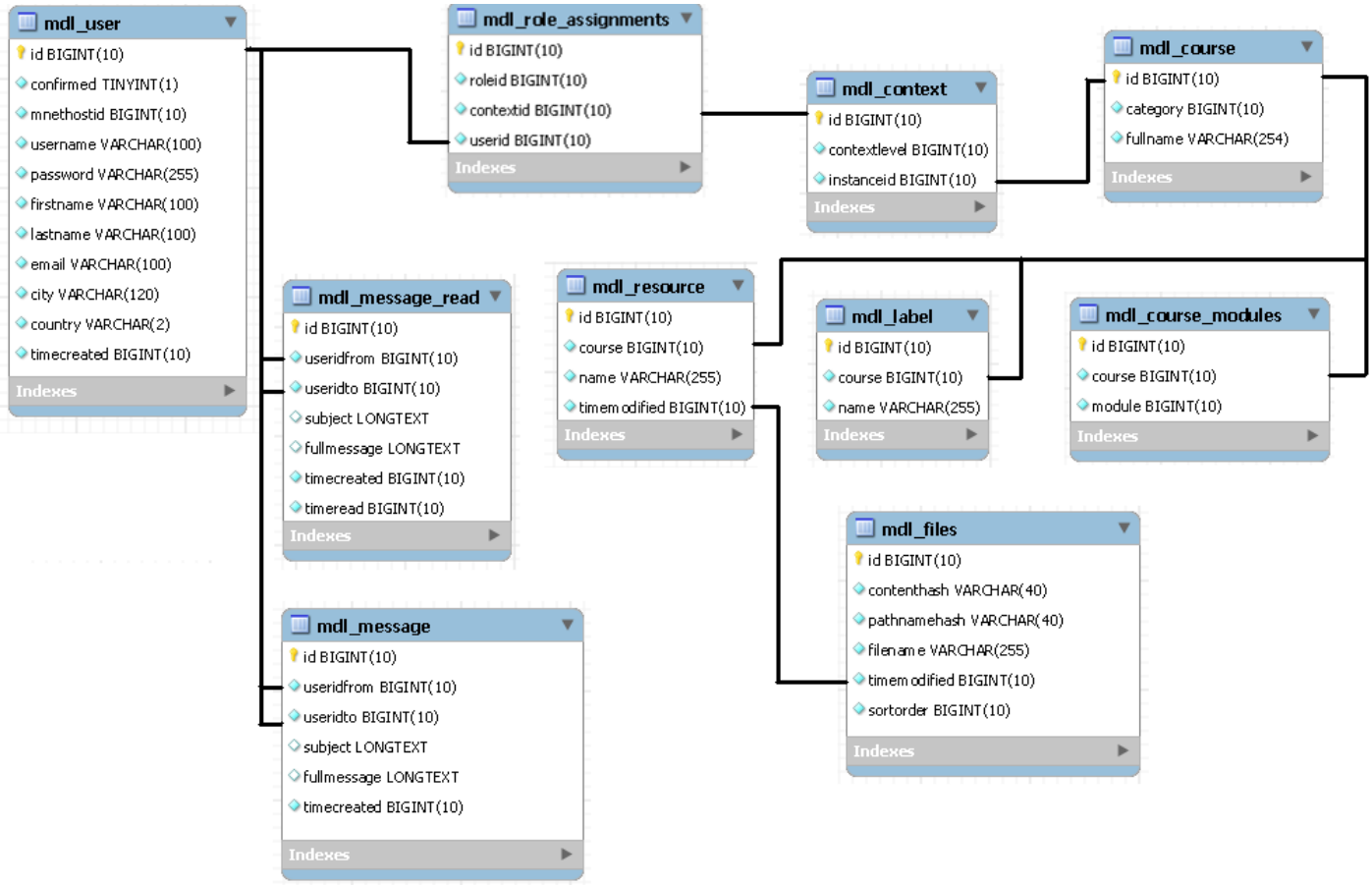

Figura 28 – Relación Tablas Base de Datos Moodle

4. CONEXIÓN DE LA APLICACIÓN MÓVIL CON EL SERVIDOR WEB

4.1 INTRODUCCIÓN

Como ya se indicó a lo largo de este documento, las conexiones necesarias entre la aplicación móvil y el servicio web, se va a llevar a cabo mediante un servicio RESTful. De esta manera a través de conexiones HTTP se podrán realizar comunicaciones en las que

cada mensaje va a contener toda la información necesaria para comprender una petición o una respuesta.

El formato elegido para la representación de los mensajes ha sido JSON, al ser mucho más sencillo en cuanto a estructura que XML, proporcionando más flexibilidad a la hora de representar cualquier estructura de datos o añadir nuevos campos.

4.2 EL PROTOCOLO HTTP

El protocolo HTTP (*HyperText Transfer Protocol*) es un protocolo de la capa de aplicación que se emplea para la transferencia de información entre sistemas, de manera clara y rápida. Es el protocolo que utiliza el *World-Wide Web* desde 1990.

HTTP se basa en un paradigma de peticiones y respuestas. Un cliente envía una petición en forma de método, una URI y una versión del protocolo, seguida de los modificadores de la petición, información sobre el cliente y adicionalmente un posible contenido. Por su parte, el servidor envía una respuesta con una línea de estado que incluye la versión del protocolo y un código de éxito o error, seguido de la información del servidor y un posible contenido.

Generalmente, es el cliente el que inicia la comunicación HTTP, y ésta consiste en la petición de un recurso del servidor. Éste es el caso que ocupa a este proyecto.

La sintaxis de una petición es la siguiente:

"http://" dirección [":" puerto] [path]

Donde dirección es el nombre de un dominio de Internet o una dirección IP, puerto es el número que indica el puerto al que se envía la petición y path hace referencia al recurso a que se quiere acceder. Por defecto, se supone que se accede al puerto 80 y si no se indica el path será "/".

A la hora de realizar las peticiones se ha de incluir el método que se aplica al recurso. Los principales métodos que se utilizan son los que se explican a continuación (Herrero Herrero, 2013).

Método GET

Este método requiere la devolución de información al cliente identificada por la URI. Si la URI se refiere a un proceso que produce información se devuelve la información, y no la fuente del proceso.

Método HEAD

Se trata de un método similar a GET, con la diferencia de que el servidor no tiene que devolver el contenido sino solo las cabeceras. Estas cabeceras deberían ser las mismas que las que se devolverían empleando el método GET.

Método POST

El método POST se utiliza para realizar peticiones en las que el servidor acepta el contenido de la petición como nuevos parámetros del recurso pedido. Este método se creó para cubrir funciones como enviar mensajes a grupos de usuarios, dar un bloque de datos como resultado de un formulario a un proceso o añadir nuevos datos a una base de datos.

La función que lleve a cabo el método POST está determinada por el servidor, y depende de la URI de la petición. El resultado de la acción realizada por el método POST puede ser un recurso que no sea identificable mediante una URI.

Método PUT

Este método permite guardar el contenido de la petición en el servidor bajo la URI de la misma. Si esta URI ya existe, el servidor considera que la petición es una actualización del recurso. Si no existe, el servidor puede crear el recurso con esa URI.

La diferencia entre POST y PUT se halla en el significado de la URI. En el caso de POST, la URI identifica el recurso que va a manejar el contenido de la petición, mientras que en el caso de PUT la URI identifica el contenido de la petición.

Método DELETE

Este método se emplea para que el servidor borre el recurso indicado por la URI de la petición. Aunque la respuesta sea satisfactoria, no se garantiza al cliente que la operación de borrado haya tenido éxito.

De todos estos métodos, para la realización de este proyecto basta con utilizar GET y POST, tal como se verá más adelante.

4.3 EL FORMATO DE DATOS JSON

JSON (*JavaScript Object Notation*) es un formato ligero de intercambio de datos. Se trata de un formato simple, fácil de leerlo y escribirlo para humanos e igualmente fácil de interpretarlo y generarlo para máquinas. JSON es un formato de texto completamente independiente del lenguaje, pero utiliza convenciones ampliamente conocidas por programadores de diferentes lenguajes de programación. Estas propiedades hacen que

JSON sea un lenguaje ideal para el intercambio de datos. JSON está constituido por dos estructuras:

- Una colección de pares nombre-valor. Es lo que se conoce como un objeto.
- Una lista ordenada de valores. Es lo que se conoce como un array.

Estas estructuras son universales, es decir, todos los lenguajes de programación las soportan de una forma u otra, por lo que es bastante interesante que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

Objeto JSON

Un objeto es un conjunto desordenado de pares nombre-valor. Comienza con el carácter "{" y termina con el carácter "}". Cada nombre es seguido por el carácter ":" y cada par nombre-valor está separado por el carácter ",". En la Figura 29 se puede ver una representación de los objetos JSON (JSON, s.f.).

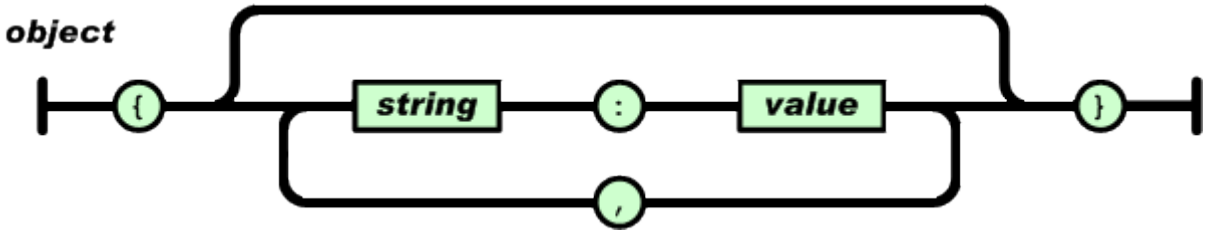

Figura 29 – Objeto JSON

Array JSON

Un array es una colección de valores. Comienza con el carácter "[" y termina con el carácter "]". Los valores de esta lista se separan por el carácter ",". En la Figura 30 se puede ver una representación de los arrays JSON.

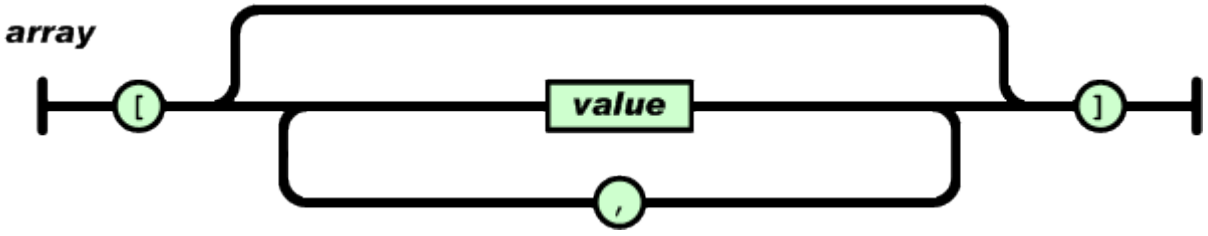

Figura 30 – Array JSON

Valor JSON

Un valor JSON puede ser una cadena de caracteres con comillas dobles, un número, una constante *true*, *false* o *null*, un objeto o un array. En la Figura 31 se puede ver una representación de los valores JSON.

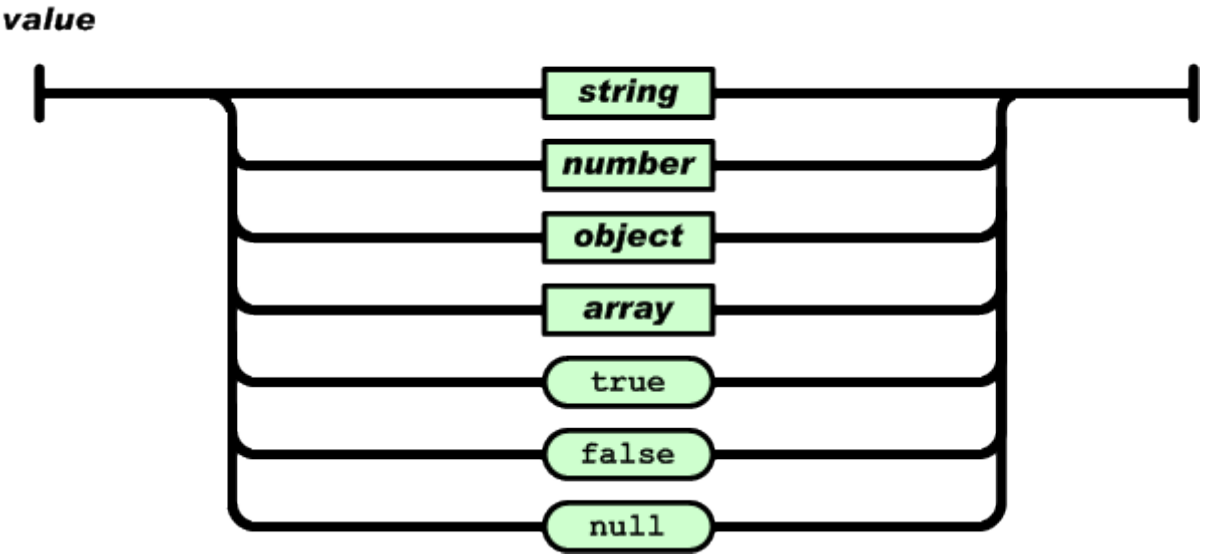

Figura 31 – Valor JSON

Un ejemplo de cómo se estructuran los datos en JSON podría ser el siguiente:

```
{
  - materias: [
      - {
            id: "1",
            nombre: "Dispositivos Moviles"
        },
      - {
            id: "2",
            nombre: "Computo Integrado"
        },
      - {
            id: "3",
            nombre: "Sistemas Distribuidos y paralelos"
        },
      - {
            id: "4",
            nombre: "Modelado y Simulado de sistemas Dinamicos"
        }
    ]
}
```

Nótese que el contenido dentro de los corchetes "[" y "]" es un array JSON, mientras que las llaves "{" y "}" delimitan objetos JSON.

4.4 DIAGRAMAS DE SECUENCIA

Para describir los diagramas de secuencia de este proyecto, es necesario involucrar a tres actores. En primer lugar, se tiene la aplicación *Android*, la cual actuará como cliente y es la encargada por tanto de realizar las peticiones al servidor. En segundo lugar, se tiene el servidor, dentro del cual se va a poder diferenciar claramente, por un lado los recursos que están alojados en el servidor (ficheros PHP que atenderán las peticiones de los clientes) y por otro lado la base de datos de *Moodle*, que almacena toda la información necesaria. A continuación se explicará cada una de las partes en que se ha dividido el proyecto.

4.4.1 AUTENTIFICACIÓN

El cliente *Android* realiza una petición al servidor. Lo que se demanda es el identificador de usuario. Este no es otro más que el parámetro id de la tabla mdl_user de la base de datos. Para poder realizar esta petición el cliente *Android* utiliza el método POST para pasar los datos de usuario y contraseña. El proceso que sigue la aplicación viene representado en el diagrama de la Figura 32:

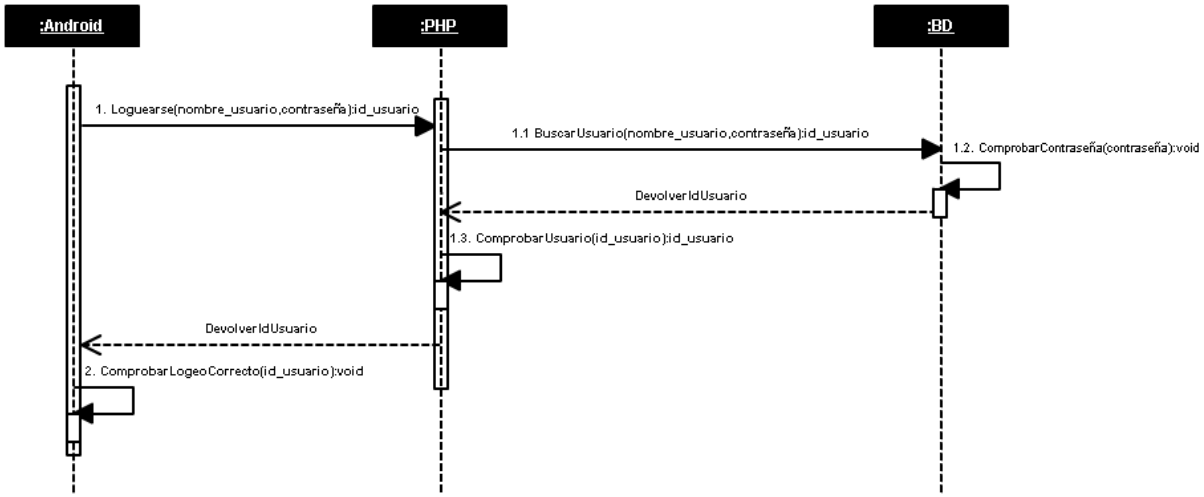

Figura 32– Diagrama Autentificación

Desde el recurso PHP lo que se hace primero es buscar la contraseña que se encuentra en la base de datos vinculada al nombre de usuario que estamos enviando, para después realizar la función *password_verify* con la contraseña que hemos enviado al autentificarnos y la contraseña que acabamos de buscar en la base de datos registrada junto a nuestro nombre de usuario.

Esta función lo que hace es comprobar que la contraseña que el usuario está introduciendo es la que tenemos vinculada al nombre de usuario en nuestra base de datos de *Moodle*.

Si la función *password_verify* nos devuelve un "1" quiere decir que la identificación es positiva y por lo tanto se procederá a buscar el id de la tabla mdl_user que tenga el nombre y contraseña indicados.

Una vez realizada la consulta, se produce una comprobación en el recurso PHP del servidor. Esta consulta se basa en discernir si la consulta ha tenido una respuesta válida o nula. En el caso positivo se devolverá el identificador demandado y por el contrario en el caso negativo se devolverá un "0".

Cuando el cliente *Android* ha recibido la respuesta, se realizará una comprobación de la validez del identificador del usuario. En el caso de que sea "0" se entenderá que los datos introducidos no se corresponden a ningún usuario registrado en la plataforma *Moodle* y saltará un mensaje de aviso. Por el contrario si el identificador es diferente de "0", se entenderá que es el valor correcto del identificador de usuario, y éste parámetro se utilizará en adelante para realizar peticiones al servidor en nombre del usuario que se acaba de autentificar. Acto seguido se realizará una nueva comprobación con otro recurso PHP para ver si este usuario a parte de estar registrado en la plataforma está confirmado por el administrador.

COMPROBACIÓN USUARIO CONFIRMADO

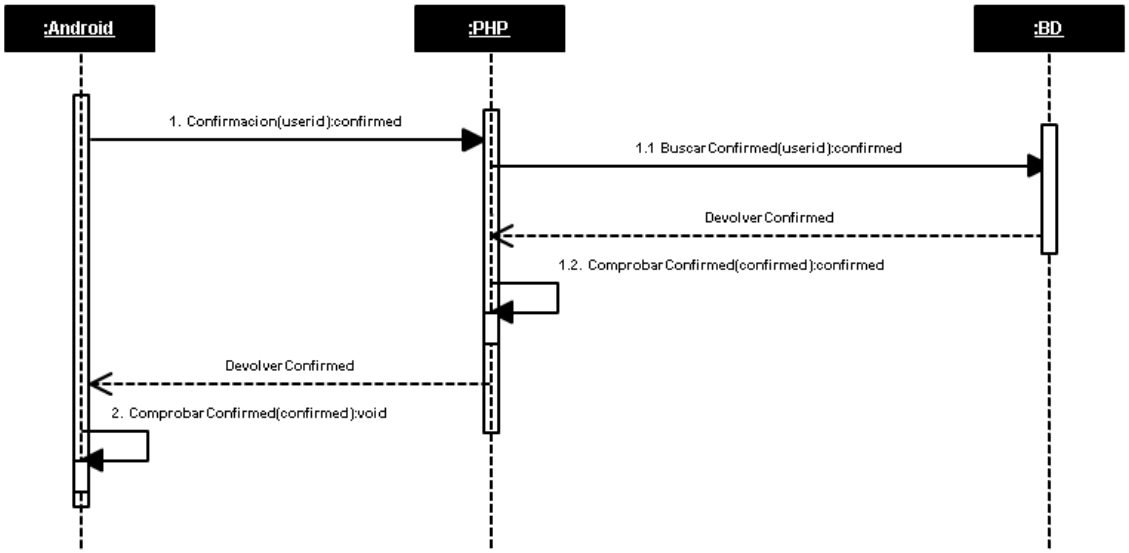

Figura 33 – Diagrama Usuario Verificado

Es un caso más sencillo que el anterior. En este caso lo que se demanda es el parámetro *confirmed* de la tabla mdl_user de la base de datos. Este parámetro indica si el usuario está confirmado o no en el sistema por el administrador. También usando el método POST se manda el *userid* conseguido en el primer paso de autentificación.

Desde el recurso PHP se realiza la búsqueda en la base de datos de *Moodle* del parámetro *confirmed*, comprueba si para el *userid* recibido el parámetro *confirmed* tiene valor "0" (no confirmado) o "1" (confirmado).

Una vez realizada la consulta, se produce una comprobación en el recurso PHP del servidor. Esta consulta se basa en discernir de nuevo si la consulta ha tenido una respuesta válida o nula. En el caso positivo se devolverá un "1" y por el contrario en el caso negativo se devolverá un "0".

Cuando el cliente *Android* ha recibido la respuesta, se realizará una comprobación de la validez del parámetro *confirmed*. En el caso de que sea "0" se entenderá que el usuario registrado aún no ha sido confirmado por el administrador en la plataforma *Moodle* y saltará un mensaje de aviso. Por el contrario si el identificador es "1", se entenderá que el usuario si está confirmado en la plataforma. Acto seguido se accederá al menú principal.

4.4.2 CURSOS, SECCIONES Y DESCARGA DE RECURSOS

VISUALIZACIÓN DE CURSOS

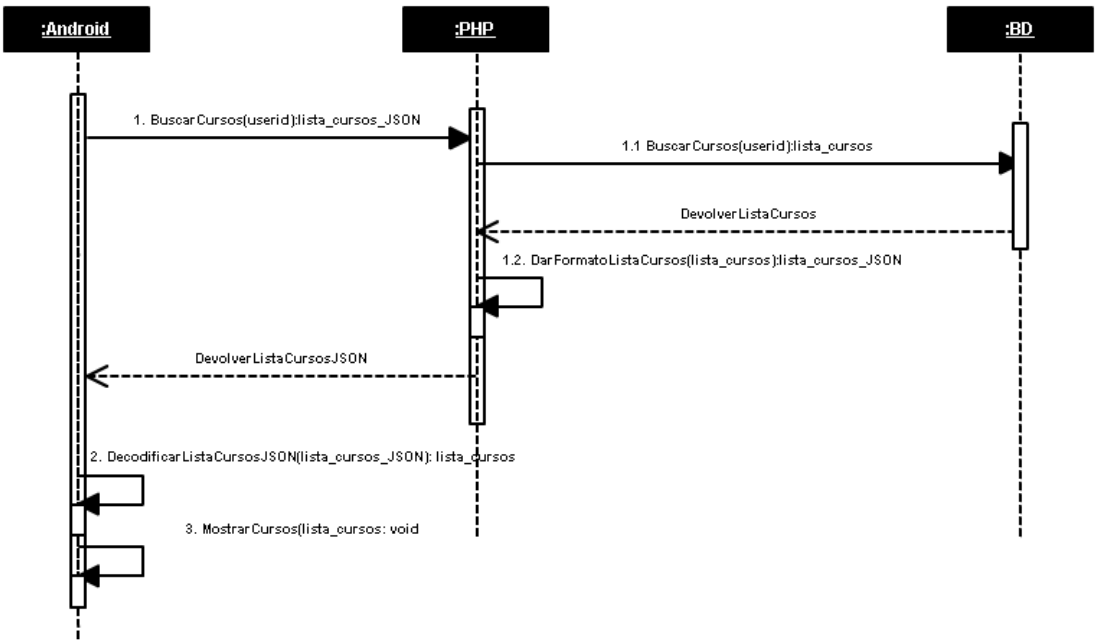

Figura 34 – Diagrama Cursos

Una vez aceptado el proceso de autenticación y tras seleccionar en el menú principal la opción de "asignaturas", el siguiente paso es mostrar los cursos o asignaturas en que está matriculado un usuario para que pueda seleccionar en cuál quiere entrar.

Para ello es necesario que el cliente Android realice una petición al servidor que, al igual que en el caso anterior, empleará el método POST para pasar como parámetro el identificador de usuario obtenido en la secuencia anterior. Gracias a este identificador se podrá buscar en la base de datos en qué cursos está matriculado ese usuario.

De la base de datos entonces se extraerán tanto el identificador del curso como el nombre del mismo. La petición que hará nuestro recurso PHP será la siguiente: Pediremos los ids de los cursos y los nombres completos de los mismos en los que esté inscrito el usuario identificado con el userid que hemos mandado por POST, ambos datos se encuentran en la tabla mdl_course, los nombres con los que se identifican son id y fullname.

Las tres tablas implicadas en nuestra petición serán mdl_course, mdl_context, mdl_role_assignments.

Por un lado intervendrá la tabla mdl_role_assignments la cual guarda información sobre qué rol tiene asignado cada usuario. El campo contextid se relaciona con el campo id de la tabla context.

Por otro lado la tabla context guarda información relativa a donde se encuentra cualquier elemento en Moodle. Los cursos tienen el valor de contextlevel=50, y el campo instandeid se relaciona con el campo id de la tabla course, por tanto si un curso tiene el campo id=8 en la tabla course, los registros de la tabla context que guarden información de ese curso, tendrán el campo instanceid=8.

Cuando se hayan extraído estos parámetros de la base de datos, se procesarán en el recurso PHP del servidor, de manera que queden con formato JSON para poder transmitirlos en la respuesta HTTP. De vuelta al cliente, se realizará un nuevo procesado para decodificar el formato JSON y dejar la información en una lista preparada para ser presentada por pantalla.

En esta lista se visualizará el nombre de cada uno de los cursos en que esté matriculado un usuario. También el identificador de cada curso irá en esa lista, pero estará oculto de cara a la visualización en pantalla. Únicamente se utilizará como parámetro para posteriores peticiones una vez que el usuario haya seleccionado el curso al que quiere acceder.

VISUALIZACIÓN DE SECCIONES

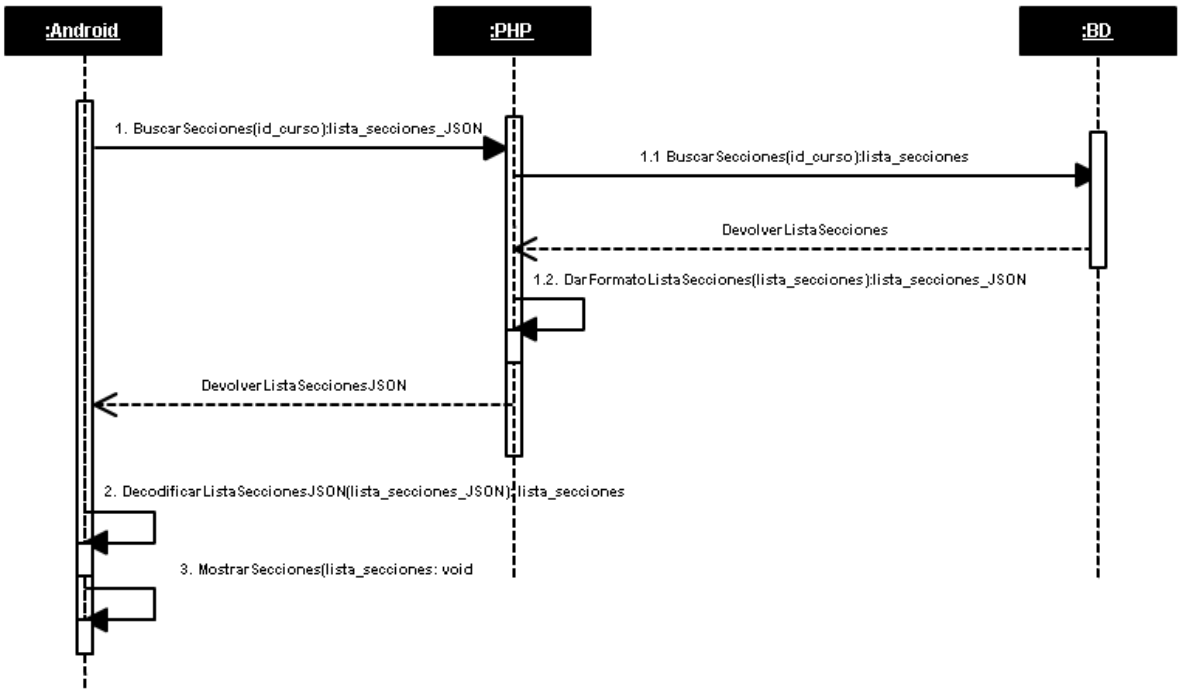

Figura 35 – Diagrama Secciones

Tras seleccionar un curso concreto de nuestra lista llegamos a este punto. Cuando se selecciona un curso, el cliente Android detecta el identificador del mismo y realiza una petición, también utilizando el método POST, pasando este identificador al recurso PHP del servidor que se ha implementado para llevar a cabo esta función.

A mayores del identificador el cliente Android enviará también por POST el parámetro mostrar_etiquetas = "1" indicando así al recurso que lo que queremos visualizar son las etiquetas y no los recursos. Esto lo hacemos debido a que el recurso PHP utilizado en este caso le reutilizamos luego para la parte de visualización de recursos. Este parámetro se manda al seleccionar cualquier curso, de manera interna, con independencia del curso seleccionado.

A continuación, se realiza una serie de consultas a la base de datos para acabar obteniendo una lista de todas las secciones que componen el curso que se ha seleccionado. La complejidad en este caso reside en el entramado de esta parte de la estructura de la base de datos que ya se explicó anteriormente, lo cual da lugar a un recurso PHP bastante complejo.

Para esta consulta intervienen las tablas mdl_course_sections, mdl_course_modules y mdl_label.

Una vez obtenida esta lista de secciones, al igual que ocurría con el caso de los cursos, en el recurso PHP se realizará un formateo de los datos de manera que tomen la apariencia de JSON para enviar la respuesta al cliente Android.

De la misma forma que antes, en el cliente se recibirá esta lista formateada bajo JSON y se decodificará generando una nueva lista dispuesta para ser presentada por pantalla.

En esta lista se visualizará el nombre del rótulo de las secciones que conforman el curso seleccionado. Además, en esa lista también estará vinculado el identificador de cada una de estas secciones, pero oculto de cara a la presentación por pantalla. Y al igual que ocurría en el caso del identificador de los cursos, el identificador de cada sección se utilizará como parámetro para posteriores peticiones al servidor.

VISUALIZACIÓN DE RECURSOS

El procedimiento en este caso es similar al de la visualización de cursos y de secciones (ver Figura 36). Al igual que antes, cuando se seleccione una sección el cliente Android hallará el identificador vinculado a la misma y realizará una petición al recurso PHP del servidor.

Esta petición también utilizará un método POST, pasando los parámetros identificador del curso e identificador de la sección.

Al igual que en el caso anterior el cliente Android enviaba por POST un parámetro mostrar_secciones con valor "1", ahora mandaremos un mostrar_recursos = "1", ya que como habíamos explicado antes, el recurso php de esta parte se reutiliza. Este parámetro se manda con independencia de la sección seleccionada. Se puede observar este proceso en el siguiente diagrama:

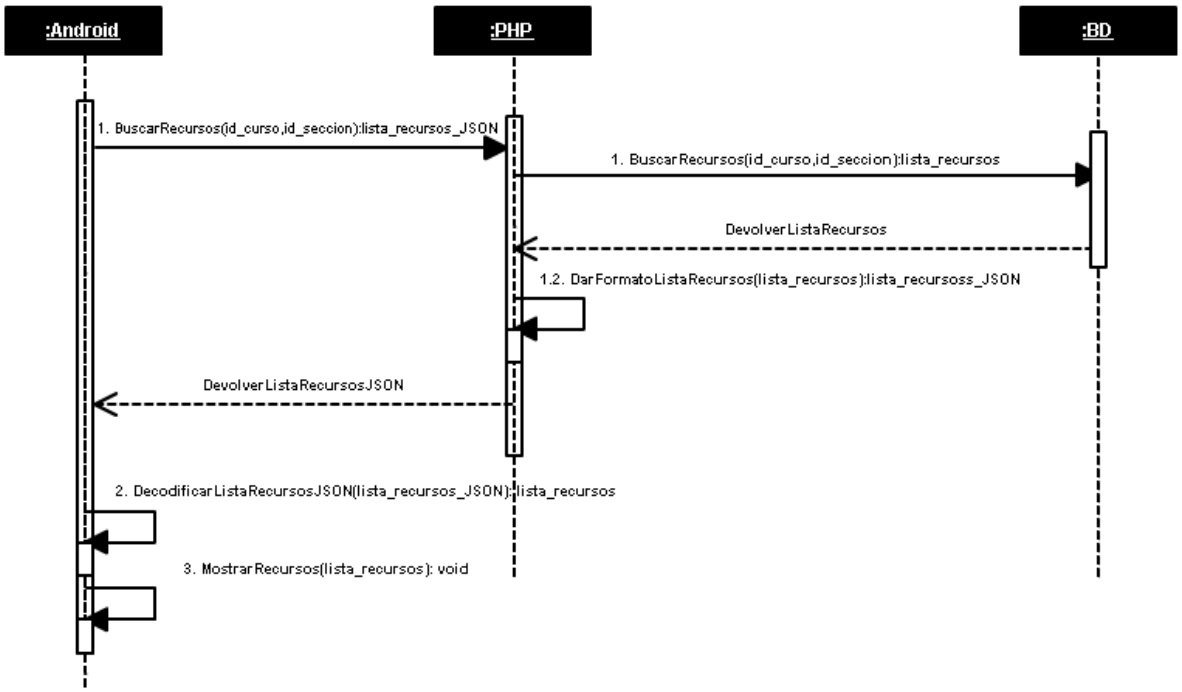

Figura 36 – Diagrama Recursos

El recurso PHP buscará en la base de datos los recursos que se encuentran en el curso seleccionado bajo una determinada sección seleccionada. Para ello, realizará una serie de consultas hasta que se pueda hacer con la lista de estos recursos. Se hace uso del mismo recurso PHP que en el caso anterior pero esta vez enviando nuevos parámetros.

Para esta consulta intervienen las tablas mdl_course_sections, mdl_course_modules y mdl_resource.

Cuando se tenga esa lista de recursos que hay en una sección, el recurso PHP del servidor los parseará a un formato JSON para poder ser enviados como respuesta a la petición HTTP. Una vez en el cliente Android, se decodificarán generando una lista para mostrar estos recursos por pantalla.

Al igual que antes, solo se mostrará el nombre de los recursos, pero en la lista también irá oculto otro dato relevante acerca de estos recursos como es el *timemodified* tiempo en el que fue modificado el recurso, que veremos más adelante como utilizamos para construir la ruta relativa de descarga del recurso.

DESCARGA DE RECURSOS

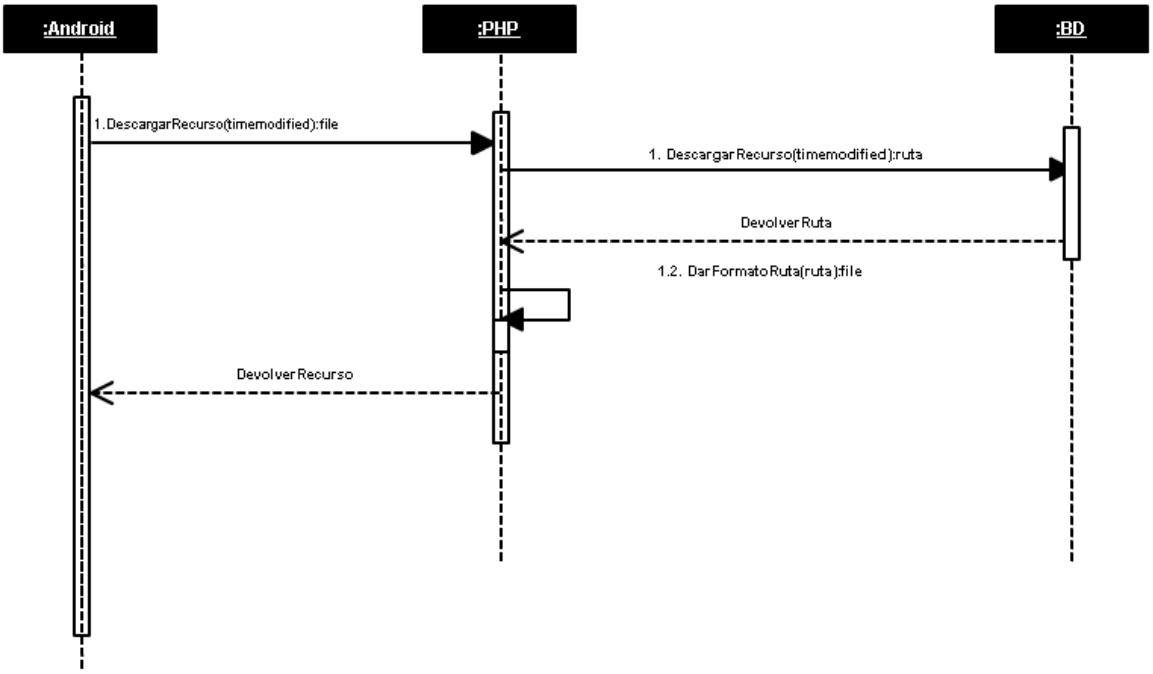

Figura 37 – Diagrama Descarga Recursos

Al seleccionar el recurso que se quiere descargar, se ejecuta un archivo PHP que forzará la descarga del mismo. De manera interna este recurso PHP generará la ruta de acceso en la que está este recurso. En esta consulta únicamente interviene la tabla mdl_files.

Mediante el método GET le se manda al recurso PHP el timemodified del recurso seleccionado, dicho parámetro fue devuelto en la pantalla de visualización de recursos, pero no se encuentra visible.

Se manda dicho parámetro por que al estar el mismo en formato TimeStamp de Unix (con una precisión de segundos) se considera que va a identificar de manera inequívoca el recurso seleccionado, ya que entendemos que va a ser casi imposible que dos profesores vayan a colgar exactamente en el mismo tiempo un recurso haciendo coincidir ese dato para dos recursos diferentes en la base de datos.

La dificultad de este apartado reside en el entramado complejo de cómo se cifran los archivos en la versión de *Moodle* con la que se desarrolla la aplicación, y como con un solo PHP se tiene que construir la ruta y descargar el archivo con su nombre y su formato correcto.

4.4.3 MENSAJERÍA

Tras seleccionar en el menú principal, la opción mensajería nos encontramos con un nuevo menú, esta vez relativo a la función de mensajería interna de *Moodle.* En este caso el menú se compone de 4 opciones:

- Mensajes no leídos: seleccionando esta opción visualizaremos únicamente los mensajes que no se hayan abierto desde la plataforma de *Moodle* para PC o no se hayan marcado como leídos desde la versión móvil.
- Mensajes leídos: si elegimos esta otra opción se visualizaran todos los mensajes marcados como leídos.
- Mensajes enviados: se mostrará una lista de todos los mensajes enviados.
- Enviar nuevo mensaje: nos permitirá acceder a una lista de personas a las que podemos enviar mensajes.

En la Figura 38 se puede observar el diagrama de flujo que se produce al seleccionar la opción de mensajes no leídos. Dicho diagrama es el mismo para los mensajes leídos y para los mensajes enviados, únicamente cambian los recursos PHP, ya que las peticiones a la base de datos son diferentes pero el diagrama no varía.

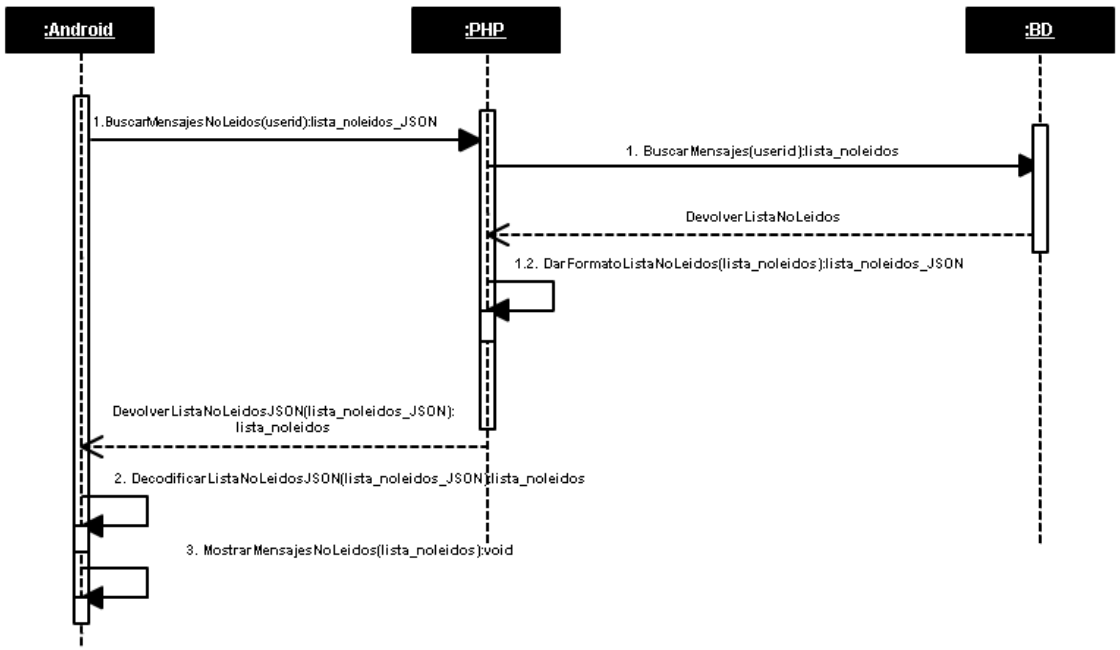

Figura 38 – Diagrama Mensajes No leídos

Enviar nuevo mensaje

Tras seleccionar en el menú de mensajería la opción de nuevo mensaje, se visualizará una lista con aquellos destinatarios a los que el usuario puede enviar un mensaje. El diagrama de la Figura 39 representa el proceso que se produce.

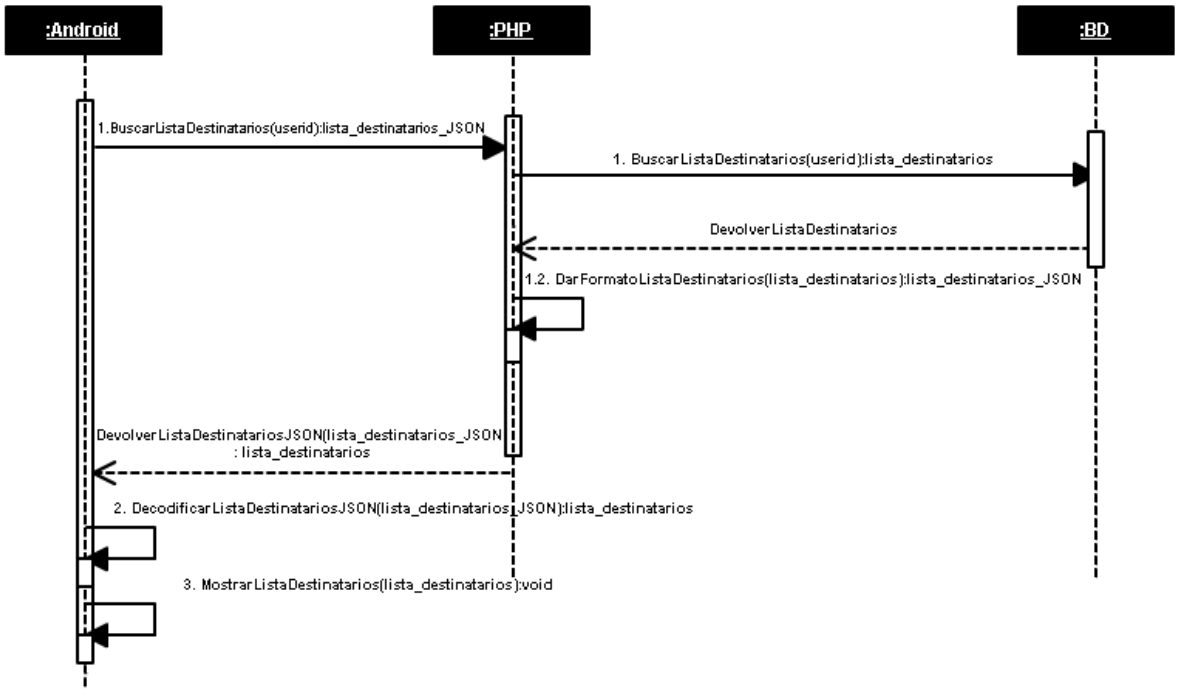

Figura 39 – Diagrama Lista destinatarios

Una vez seleccionado un destinatario, el usuario se encontrará en una pantalla en la que podrá escribir un mensaje nuevo, tras seleccionar la opción de enviar mensaje, se sucederán las acciones que se describen en el siguiente diagrama.

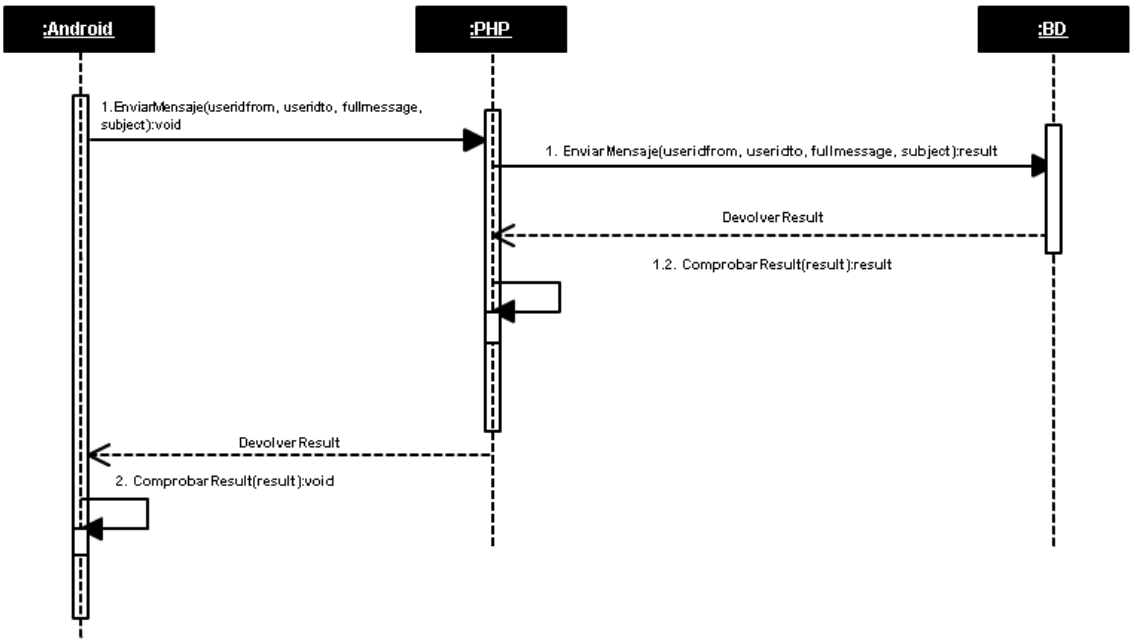

Figura 40 – Diagrama Enviar Nuevo Mensaje

El recurso PHP realizará una inserción de contenidos en la base de datos, de igual manera a la que se introducían los datos al registrar un nuevo usuario, recibiendo una confirmación de que la inserción ha sido correcta tras realizarse. Por lo tanto el diagrama representado en la Figura 40, nos serviría para ambos casos. Únicamente cambia el recurso PHP.

CAPÍTULO VI.

CONCLUSIONES Y LÍNEAS FUTURAS

1. CONCLUSIONES

En este trabajo fin de grado, se ha realizado una segunda versión de una aplicación móvil destinada al uso académico. Dicha aplicación supone la implementación de algunas funcionalidades de la plataforma *Moodle* para dispositivos móviles. La aplicación móvil desarrollada permite sin necesidad de autentificarse registrar nuevos usuarios, y tras realizar una autentificación, se permite al usuario ver los cursos en los que está matriculado, las secciones de cada curso y descargar los contenidos albergados en dichas secciones. También se permite acceder a los usuarios al módulo de mensajería de *Moodle*, permitiéndoles comunicarse desde la aplicación con todos aquellos alumnos y profesores con los que tengan relación académica. Este módulo de mensajería permite visualizar los mensajes ya enviados, los no leídos, los leídos y enviar nuevos mensajes.

El primer objetivo a realizar era el de actualizar y mejorar las funcionalidades que ya se habían realizado en la primera versión, y que en la actualidad se encontraban obsoletas debido a cambios importantes en la configuración de la base de datos del servidor *Moodle*. Estos cambios se deben a que la primera versión de la aplicación se realizó para una versión de *Moodle* bastante antigua, que cambiaba sustancialmente respecto a las más actuales. Las funcionalidades actualizadas son las de autentificación de usuarios, visualización de asignaturas y recursos y la descarga de archivos.

Por otro lado como segundo objetivo se añadieron nuevas funcionalidades a la aplicación, inexistentes en la primera versión. Como son el permitir el registro de nuevos usuarios y el acceso al módulo de mensajería de Moodle de manera completa.

Para esta segunda versión de la aplicación, se han seguido utilizando todas las tecnologías con las que se había implementado la primera versión: Android para el desarrollo de la aplicación móvil, *REST* para la implementación de los servicios Web y *JSON* para la representación de datos. Se ha optado por continuar con dichas tecnologías ya que lo que se buscaba es que el procesado de información fuese lo más ligero posible, y esa es la ventaja principal que presentan *REST* y *JSON* frente a *SOA* y *XML* respectivamente. El utilizar de nuevo estas tecnologías también es favorable desde el punto de vista de reutilizar código.

El duplicar la carpeta *moodledata* es otro aspecto que se ha continuado haciendo respecto a la primera versión. Ese directorio es donde se almacenan estructuradamente los contenidos insertados en *Moodle*. Por motivos de seguridad se encuentra alejado en el servidor pero fuera del directorio web, es decir, no está accesible en ningún momento desde fuera del propio servidor. Respetando las condiciones de seguridad que aconseja *Moodle*, se opta por un duplicado de esta carpeta para desarrollar los objetivos de este proyecto.

También habría que decir que la implementación de esta aplicación ha resultado favorable, y que la posibilidad de ampliar la misma queda abierto a nuevas versiones, ya

que aún quedan muchas funcionalidades de *Moodle* por implementar, que dotarían a esta aplicación de mucho más empaque y utilidad.

Por último y como conclusión personal decir que he cumplido mis expectativas con la realización de este proyecto, no ya con la realización de la aplicación en sí misma, que también, sino en el sentido de que me ha servido para trabajar con tecnologías con las que no me había enfrentado en la carrera como Android o JSON y para indagar más en otras que si había cursado como XML, PHP o las consultas de bases de datos con MySQL. Elegí este proyecto para poder empezar a aprender nuevos lenguajes de programación y eso es lo que he hecho, por lo cual me siento muy satisfecho con la elección de este proyecto y el desarrollo del mismo.

2. LÍNEAS FUTURAS

Teniendo en cuenta el trabajo desarrollado en este proyecto, se proponen las siguientes líneas futuras para la evolución y mejora de las prestaciones de la aplicación:

- Añadir nuevas funcionalidades de la plataforma *Moodle*. Tales como wikis, foros, cuestionarios...
- Implementar una nueva versión de esta aplicación que permita determinadas funcionalidades según los roles que tenga cada usuario, por ejemplo de cara al profesorado estaría bien el poder crear cursos o modificar los mismos desde la aplicación móvil si el usuario autentificado es un profesor y tiene los permisos que se lo permitan.
- Mejorar las funcionalidades ya creadas. Por ejemplo, en el módulo de mensajería permitiendo adjuntar imágenes o archivos en los mensajes, seleccionar varios destinatarios…
- Desarrollar notificaciones que permitan al usuario estar informado de los nuevos recursos que se añadan al instante, o de los mensajes nuevos que ha recibido sin la necesidad de entrar en la aplicación a verlos.
- Otra línea futura a considerar sería la de desarrollar esta aplicación en iOS para los usuarios de iPhone, tiene las complicaciones ya comentadas en este proyecto pero atendería a las necesidades de bastantes usuarios que usan estos dispositivos móviles.
- Por otro lado y al ser una segunda versión, aunque en este caso hemos implementado un par de menús haciendo más agradable la aplicación, la interfaz gráfica es todavía muy mejorable. Esto constituye otro objetivo futuro, la creación de una mejor interfaz gráfica.

BIBLIOGRAFÍA

Amate, C. (2014).*Conoce los Principales Sistemas Operativos Móviles.* Recuperado el 24 noviembre de 2014, de http://blogthinkbig.com/sistemas-operativos-moviles/

Amatellanes (2014). *Integrando un RestFul Web Service en Android*. Recuperado el 25 de Enero de 2015, de https://amatellanes.wordpress.com//01/02/android-integrando-un-restful-web-service-en-android/

Android (2015). *History.* Recuperado el 10 de Marzo de 2015, de https://www.android.com/intl/es_es/history/

Cosmo, J. (2014). *Android 4.4 (KitKat) ya es la versión más usada con el 30,2%.* Recuperado el 10 de Marzo de 2015, de http://www.xatakandroid.com/mercado/android-4-4-kitkat-ya-es-la-version-mas-usada-con-el-30-2

Cosmo, J. (2015). *El 1,6% de los dispositivos Android ya están actualizados a Lollipop.* Recuperado 10 de Marzo de 2015, de http://www.xatakandroid.com/mercado/el-1-6-de-los-dispositivos-android-ya-estan-actualizados-a-lollipop

Dept. Ciencia de la Computación. (Junio de 2014).*Introducción a los Servicios Web. Invocación de servicios web SOAP*. Universidad de Alicante. Recuperado el 15 de Enero de 2015, de http://expertojava.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.pdf

E-abclearning (2011a). *Definición E-Learning.* Recuperado el 12 de Marzo de 2015, de http://www.e-abclearning.com/definicione-learning

E-abclearning (2011b). *Que es una plataforma E-Learning.* Recuperado el 12 de Marzo de 2015, de http://www.e-abclearning.com/queesunaplataformadeelearning

Esquiva Rodríguez, A. (2013). *JSON qué es y para qué sirve*. Recuperado el 20 de Marzo de 2015, de https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/

Fontela Sánchez, A. (2013). *6 alternativas a Moodle para elearning*. Recuperado el 18 de diciembre de 2014, de http://openwebcms.es/2013/6-alternativas-moodle-para-elearning/

Google Play (2015). *Moodle Mobile.* Opiniones. Recuperado el 24 de Marzo de 2015, de https://play.google.com/store/apps/details?id=com.moodle.moodlemobile&hl=es

Hernández, D. (2010). *Donde se Guardan los Contenidos Académicos que Suben a la Plataforma en un Curso Especifico* .Recuperado el 12 de Octubre de 2014, de https://moodle.org/mod/forum/discuss.php?d=164482&parent=721142

Herrero Herrero, F. (2013).*Desarrollo de una Aplicación m-Learning Android con Acceso a Moodle.* Proyecto Fin de Carrera. Valladolid: Universidad de Valladolid.

JSON (s.f.). *Introducing JSON.* Recuperado el 15 de Octubre de 2014, de http://json.org/

Marín, M. (2014). *Android domina el mercado mundial de 'smartphones' con un 85% de cuota.* Recuperado el 24 de noviembre de 2014, de http://www.ticbeat.com/tecnologias/android-domina-mercado-mundial-smartphones/

Moodle (2015a). *Antecedentes*. Recuperado el 10 de Marzo de 2015 de https://docs.moodle.org/all/es/Antecedentes

Moodle (2015b). *Arquitectura*. Recuperado el 10 de Marzo de 2015 de https://docs.moodle.org/all/es/Arquitectura

Moodle (2015c). *Características*. Recuperado el 10 de Marzo de 2015 de https://docs.moodle.org/all/es/Características

Moodle (2015d). *Filosofía*. Recuperado el 10 de Marzo de 2015 de https://docs.moodle.org/all/es/Filosofía

Moodle (2015e). *Manual de Estilo de Código*. Recuperado el 12 de Marzo de 2015, de https://docs.moodle.org/all/es/Manual_de_Estilo_de_Codigo

Moodle (2015f). *Paquetes de Idioma*. Recuperado el 10 de Marzo de 2015 de https://docs.moodle.org/all/es/Paquetes_de_idioma

Paszniuk, R. (2014). *Acceso a Web Service SOAP en Android*. Recuperado el 20 de Enero de 2015, de http://www.programacion.com.py/moviles/android/acceso-a-web-service-soap-en-android

Pes, C. (2014). *Tutorial de XML*. Recuperado el 20 de diciembre de 2014, de http://www.abrirllave.com/xml/

Php (2013). *Funciones de hashing de contraseñas*. Recuperado el 20 de Noviembre de 2014, de http://php.net/manual/es/function.password-hash.php

Picurelli, L. (2014). C*ómo darse de alta como desarrollador iOS*. Recuperado el 15 de diciembre de 2014, de http://www.yeeply.com/blog/como-darse-de-alta-como-desarrollador-ios/

Pueyo, J. (2009). *Ejemplo con roles, usuarios y contextos.* Recuperado el12 Octubre de 2014, de http://moodle-chile.cl/blog/noticias/moodle-foro/103-para-administradores/3050-ejemplo-con-usuarios-roles-y-contextos.html

World Wide Web Consortium(2013). *Guía Breve de Servicios Web*. Recuperado el 20 de Noviembre de 2014, de http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

Zamora, J.A. (2014), *Aprende Android en 20 conceptos.* Recuperado el 20 de Octubre de 2014, de http://www.elandroidelibre.com/2014/02/aprende-android-en-20-conceptos-empezando-a-programar-para-android.html

ANEXO I.

VERSIONES DE ANDROID

Android es el sistema operativo móvil que utilizan más de mil millones de smartphones y tablets. Como se puede observar en la página oficial la compañía hace referencia a mejorar o endulzar la vida de sus usuarios a través de su sistema operativo, por lo que las diferentes versiones del mismo llevan nombre de un dulce en inglés, ordenados alfabéticamente, ordenación que coincide con su fecha de aparición: *Apple Pie, Banana Bread, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat y Lollipop*(Android, 2015). En la imagen y tabla siguientes (Figura 41 y Tabla 2) se pueden ver la distribución de las diferentes versiones en el panorama actual. En la imagen no se recogen los dispositivos actualizados a la versión Lollipop, los cuáles únicamente conforman en la actualidad el 1,6% (Cosmo, 2015).

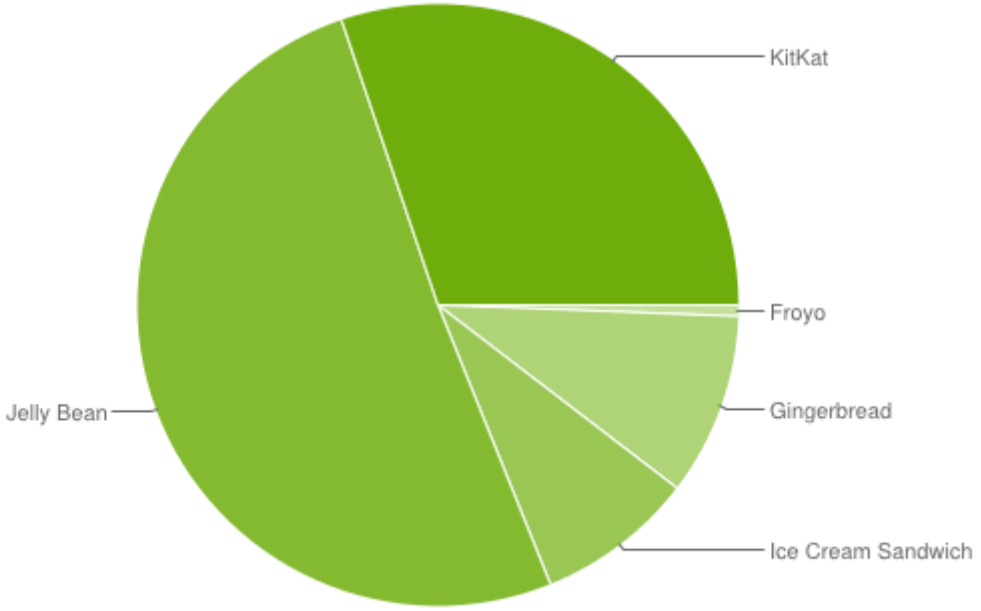

Figura 41 – Cuota de Mercado Versiones Android 2014

Version	Codename	API	Distribution
2.2	Froyo	8	0.6%
2.3.3 - 2.3.7	Gingerbread	10	9.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	8.5%
4.1.x	Jelly Bean	16	22.8%
4.2.x		17	20.8%
4.3		18	7.3%
4.4	KitKat	19	30.2%

Tabla 2 – Cuota de Mercado Versiones Android

ANDROID 0.X, PRIMEROS PASOS

Esta primera versión de Android no estaba pensada para pantallas táctiles y su aspecto difiere mucho del Android que la mayoría de nosotros pudo apreciar por primera vez. De hecho estas versiones únicamente estaban disponibles en formato beta y no aptas para su venta comercial.

Lo que tenemos ante nosotros (ver Figura es un clon de una Blackberry con una versión de Android primigenia, ni estaba Google ni el iPhone había sido presentado todavía. Tenemos una resolución de 320×240, sin widgets, sin el botón de atrás, sin notificaciones y con una configuración mínima.

A medida que pasó el tiempo con las sucesivas betas se fueron añadiendo y mejorando todos los menús, contactos, mensajes, llamadas, navegador, maps... se añadieron las notificaciones, algunos widgets. De las primeras betas a la versión 0.9 pasó mucho tiempo. Los cambios fueron sucesivos y muy diversos (Android, 2015).

Figura 42 – Interfaz Android Básica

ANDROID 1.0/1.1 – APPLE PIE – NIVEL DE API 1

Android 1.0, la versión que introdujo las Google Apps. Además de ellas, también representó la llegada de la app más importante de nuestros días; el Android Market, que luego se convertiría en lo que hoy día es Google Play.

Esta nueva versión añadía nuevas opciones, como el patrón de desbloqueo, las alarmas, las notificaciones de batería y algunas otras aplicaciones como Gmail, Google Calendar, el reproductor de videos YouTube…

Figura 43 – Interfaz Android ApplePie

ANDROID 1.1 BANANA BREAD, NIVEL DE API 2

Aparece en febrero de 2009. Es una versión en la que apenas se añadieron funcionalidades. Básicamente, se trató de corregir algunos errores de la versión anterior.

ANDROID 1.5 CUPCAKE – NIVEL DE API 3

Fue la primera versión Android con apodo propio, al menos que se hiciera público. Fue lanzada en abril de 2009. Se basó en el núcleo Linux 2.6.27.

La mayor novedad fue el teclado táctil con las opciones de mayúsculas, minúsculas y adaptado a los distintos fabricantes. El panel de notificaciones fue rediseñado y se permitió por primera vez widgets de terceros.

En Cupcake también hubo grandes actualizaciones referentes a las aplicaciones móviles, entre las que se encuentra la posibilidad de subir vídeos a Youtube por primera vez.

Otras mejoras importantes fueron:

- *Widgets* de escritorio y *live folders*.
- Grabación avanzada de audio y vídeo.
- Copiar y pegar.
- Subir vídeos a *YouTube* de forma directa.
- Transiciones animadas entre pantallas.
- Función de auto-rotación de la pantalla.

En la Figura 44 se puede observar una comparación gráfica de esta versión respecto a la versión 1.1

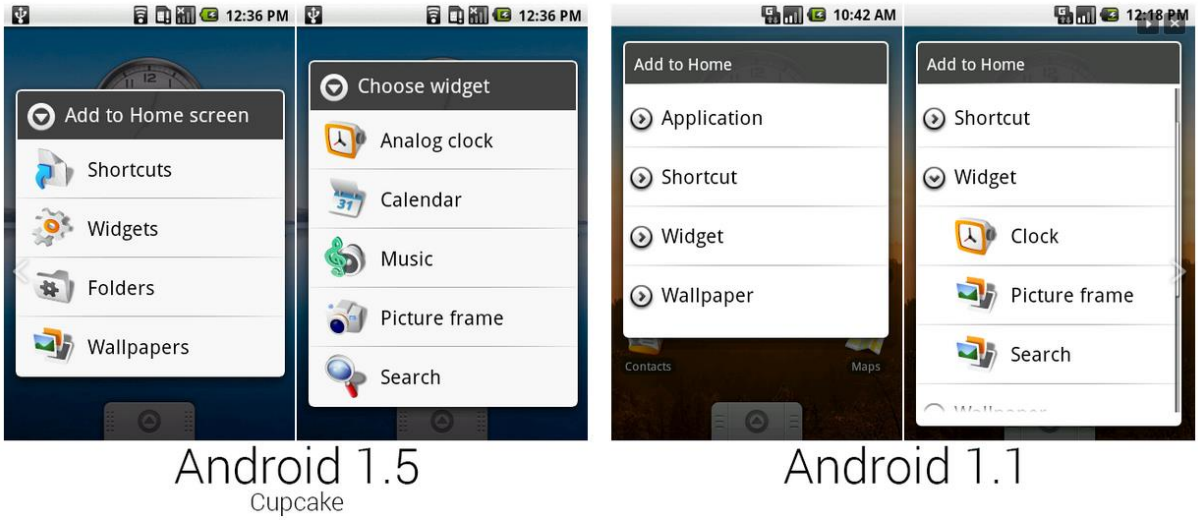

Figura 44 – Comparación Cupcake con Android 1.1

ANDROID 1.6 DONUT – NIVEL DE API 4

Esta versión se lanzó en septiembre de 2009. Con Donut llegó el soporte para múltiples tamaños de pantalla distintos, soporte CDMA, y el TTS. Más allá de esto, fue una actualización menor que a pesar de corregir bugs y mejorar algún diseño no aportó grandes cambios. Los más reseñables son los siguientes:

- Capacidades de búsqueda avanzada en todo el dispositivo, incluso por voz.
- Puede trabajar con diferentes densidades de pantalla.
- Soporte para pantallas WVGA.
- Soporte para CDMA/EVDO, 802.1x y VPN.
- Mejoras en la aplicación de la cámara.
- Mejoras en el *Android Market* permitiendo realizar al usuario búsquedas de aplicaciones de forma más sencilla.
- Aparece un nuevo atributo, *onClick*, que puede especificarse en una vista.

En la Figura 45, se puede observar los cambios producidos a nivel de interfaz en la tienda de aplicaciones Google Market respecto a la versión 1.5. Podemos observar como el entorno estás más trabajado y la carga gráfica es mayor.

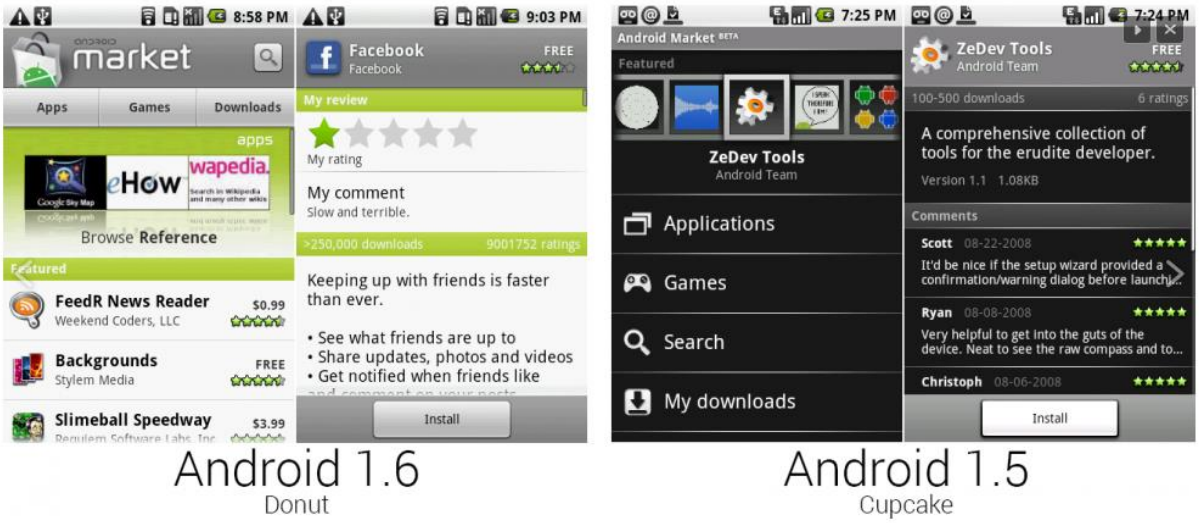

Figura 45 – Comparación Interfaz Donut con Cupcake

ANDROID 2.0/2.1 ECLAIR – NIVEL DE API 5 y 7

Android 2.0 Eclair fue lanzada en diciembre de 2009, vino acompañada en un principio del smartphone Nexus One aunque en realidad este vino ya con la 2.1.

Esta versión apenas cuenta con usuarios, debido a que la mayoría e fabricantes pasaron de la versión 1.6 a la versión 2.1 (nivel de API 7). Las novedades que introdujo la versión 2.0 son las siguientes:

- Incorpora un API para manejar el *bluetooth 2.1*.
- Permite sincronizar adaptadores para conectarlo a cualquier dispositivo.
- Ofrece un servicio centralizado de manejo de cuentas.
- Mejora la gestión de contactos y ofrece más ajustes en la cámara.
- Incrementa el número de tamaños de ventana y resoluciones soportadas.
- Incorpora una nueva interfaz del navegador y soporte para HTML5.

La versión 2.1 se considera una actualización menor, por lo que conservó el nombre *Eclair*. Su lanzamiento se produjo en enero de 2010. En la Figura 45 podemos ver una

comparativa del Nexus One con Eclair con el Iphone 3, el gran competidor del mismo en ese momento.

Figura 46 – Comparación Nexus One con Iphone 3

Como se ha explicado, la versión 2.1 fue más bien una actualización de la versión 2.0, más que una nueva versión, en la Figura 47 se pueden observar unos pequeños cambios de interfaz entre una y otra.

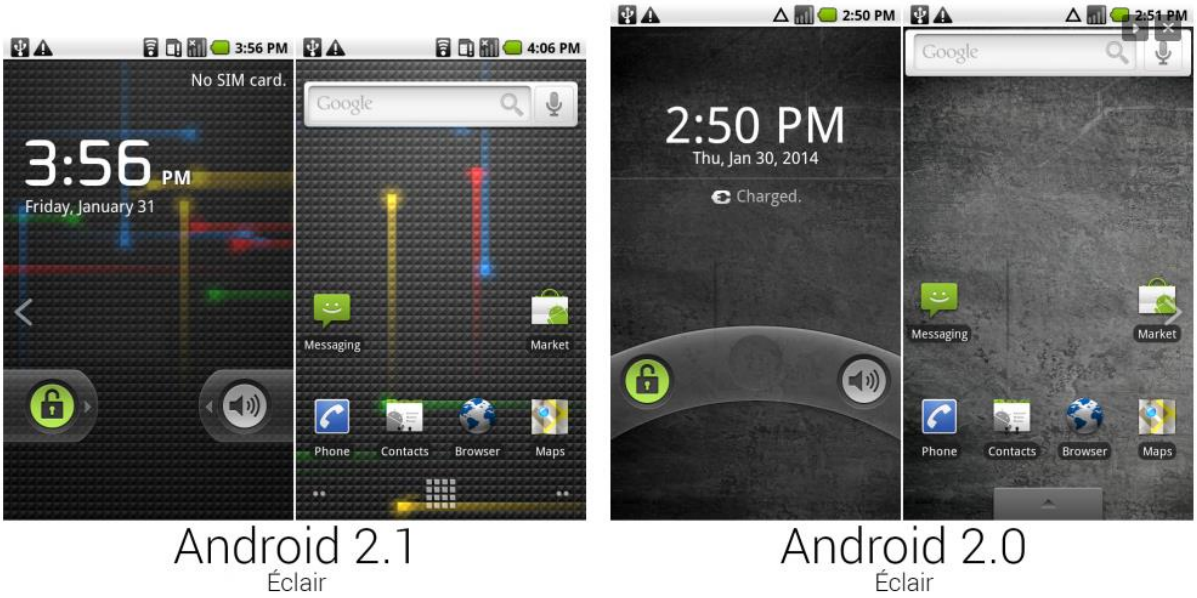

Figura 47 – Comparación Eclair 2.1 vs 2.0

ANDROID 2.2 FROYO – NIVEL DE API 8

Esta versión tuvo su lanzamiento en mayo de 2010. Se mejoró muy significativamente el rendimiento. Fue gracias sobre todo a la inclusión del motor V8 javascript en Chrome y la inclusión de JIT. También en Android 2.2 se insertó la barra de búsqueda en el escritorio como algo permanente y característico. Mientras, en el Android Market se añadía la posibilidad de "actualizar todo".

Android 2.2 Froyo se hizo también famosa por soportar Flash, a diferencia de su principal rival.

La característica más destacable son las optimizaciones en velocidad, memoria y rendimiento. También cabe destacar las siguientes:

- Nuevas mejoras en el navegador web.
- Permite instalar aplicaciones en medios de almacenamiento externo.
- Ofrece actualizaciones automáticas de las aplicaciones cuando aparecen nuevas versiones.
- Permite interacciones con el reconocimiento de voz.
- Mejoras en la conectividad.

En la siguiente imagen (Figura 48) se pueden observar algunos cambios de interfaz significativos entre la versión Froyo y la versión anterior.

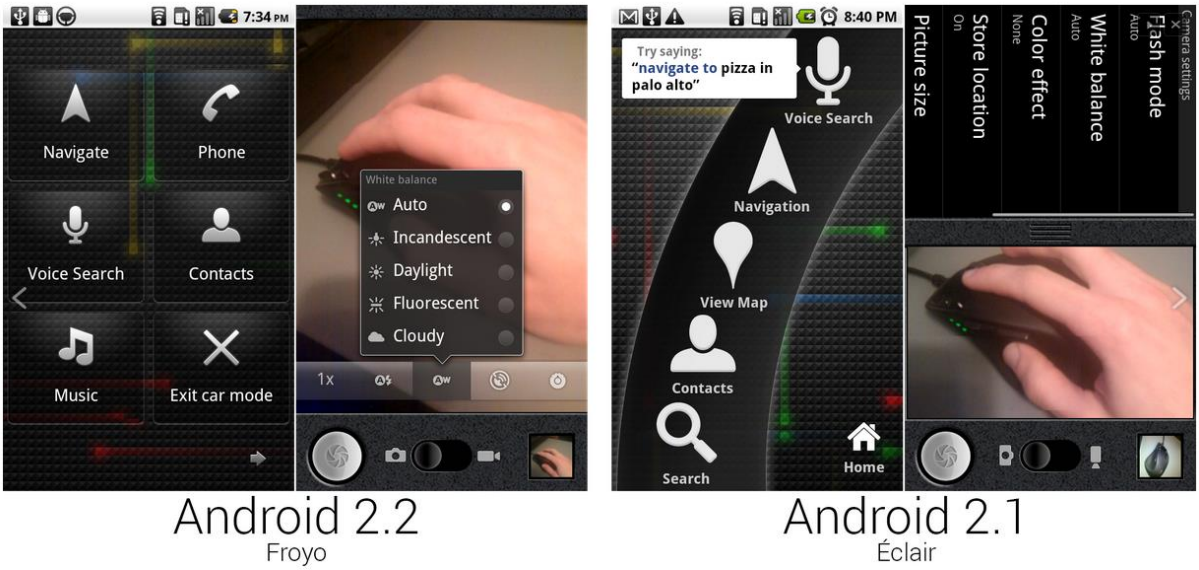

Figura 48 – Comparación Froyo con Eclair

ANDROID 2.3 GINGERBREAD, NIVEL DE API 9

Esta versión sale a la luz en diciembre de 2010. Las principales mejoras que introdujo esta versión son las siguientes:

- Mejoras en la funcionalidad de "cortar, copiar y pegar".
- Mejoras en el rendimiento, optimizando el cierre de aplicaciones en desuso y permitiendo así reducir el consumo de batería.
- Soporte nativo para telefonía sobre Internet VoIP/SIP.

En la siguiente imagen se puede ver la interfaz de bienvenida que presentaba esta versión de fábrica

Figura 49 – Diseño Nexus S (Versión Android 2.3 Gingerbread)

Android 2.3 Gingerbread fue la primera versión de Android donde Google prestó especial importancia del diseño. El Nexus S fue el primer móvil fruto de la unión entre el buscador y Samsung y el encargado de presentar Gingerbread.

Los menús grisáceos de las versiones anteriores se aligeraron, el panel de notificaciones se modificó y el "copiar y pegar" se resaltó para facilitar su uso. Esto se puede observar en la imagen X. En la cual se puede ver una mayor estilización de las formas respecto a la versión 2.2.

Figura 50 – Comparación Gingerbread con Froyo

ANDROID 3.0 HONEYCOMB, NIVEL DE API 11

Android 3.0 Honeycomb sorprendió con su propuesta radical para adaptarse a los tablets, un dispositivo de moda cuando fue lanzada esta versión (febrero de 2011). Los cambios sentaron el precedente de lo que luego tendríamos con ICS. El diseño adquirió los tonos azulados que fueron mantenidos hasta las versiones más actuales.

También fue a partir de esta versión cuando el botón de multitarea empezó a cobrar importancia respecto al de menú. La pantalla de desbloqueo cambió, se empezaron a ver las primeras apps Android adaptadas a pantallas grandes.

Esta versión, lanzada en febrero de 2011, se desarrolló para mejorar la experiencia de *Android* en las *tablets*, ya que estaba optimizada para dispositivos con pantallas grandes. Las principales novedades son las siguientes:

- Resolución por defecto WXGA, escritorio 3D con *widgets* rediseñados, nuevos componentes y vistas, mejoras en las notificaciones y otras características para optimizar el uso en las pantallas más grandes.

- Mejoras en la reproducción de animaciones 2D/3D gracias al *renderizador OpenGL* acelerado por hardware.

- Ejecución más rápida de las aplicaciones.

- Compatibilidad con las aplicaciones creadas para versiones anteriores, a pesar de que la interfaz gráfica está optimizada para *tablets*.

En mayo de 2011 se lanzó la versión 3.1 (nivel de API 12), cuya principal mejora es que permite manejar dispositivos conectados por USB. En julio de ese mismo año apareció la versión 3.2 (nivel de API 13), que ofrecía optimizaciones para distintos tipos de *tablets*. En la imagen contigua (Figura 51) se puede observar la pantalla inicial de una Tablet con esta versión.

Figura 51 – Interfaz Tablet Nivel API 13

ANDROID 4.0 ICE CREAM SANDWICH, NIVEL DE API 14

Con la cuarta versión se alcanzó el status de sistema operativo moderno, capaz de empezar a dominar el mercado móvil. El Samsung Galaxy Nexus fue el elegido para representarlo.

Los cambios de diseño probados en Honeycomb llegaron también a las pantallas pequeñas, la pantalla de notificaciones volvió a ocupar todo el espacio, se insertó la pestaña de widgets, NFC recibió soporte total y también el dialer sufrió cambios. Youtube se actualizó a una versión mejor y el menú de ajustes permitió la posibilidad de controlar el uso de datos.

Ice Cream Sandwich fue la versión en la que las pautas de diseño de Android se crearon y aún hoy siguen más o menos vigentes. También fue en ICS cuando vimos el cambio del Android Market a Google Play, una evolución que sería el inicio de la estrategia comercial de Google en el sector móvil.

La versión 4.0 de Android, lanzada en octubre de 2011, tuvo como principal objetivo unificar las dos versiones anteriores (2.x para teléfonos y 3.x para tablets), con el fin de buscar la compatibilidad con cualquier tipo de dispositivo. Las principales características son las siguientes:

- La interfaz de usuario aparece totalmente renovada, reemplazando los botones físicos por botones en pantalla como ocurría en las versiones 3.x.
- Mejoras en las aplicaciones de reconocimiento, tanto facial como de voz.
- Acceso a notificaciones y navegación entre aplicaciones abiertas mucho más cómodo, gracias a mejoras en las barras de sistema y de acción.

En diciembre de 2011 se lanzó la versión 4.0.3 (nivel de API 15), en la que aparecen ligeras mejoras en algunas APIs, como el calendario, el revisor ortográfico y las bases de datos, entre otros.

En la Figura 52 podemos ver la pantalla de inicio predeterminada de esta versión:

Figura 52 – Interfaz Nexus IV

Por otra parte en la Figura 53 se pueden observar cambios significativos en la interfaz respecto a la versión inmediatamente inferior.

Figura 53 – Comparación Ice Cream Sandwich con Honeycomb

ANDROID 4.1 – 4.3 JELLY BEAN – NIVEL DE API 16-17

Jelly Bean sigue siendo a Junio de 2014 la versión más extendida y utilizada de Android. Uno de los mayores cambios que se introdujo fue Project Butter con el que se mejoró nuevamente el rendimiento, el sistema de notificaciones se expandió para permitir respuestas rápidas, se mejoró la cámara, se introdujo Hangouts y se añadió Google Now, el sistema inteligente que se anticipa a nuestras búsquedas.

Paralelamente los Google Play Services empezaron a integrarse de manera significativa en el sistema, formando parte de la columna vertebral de Android. Para relanzar Jelly Bean se lanzaron diversos Nexus, desde la primera tablet de siete pulgadas hasta el Nexus 4, todos ellos con una calidad/precio muy ajustada.

ANDROID 4.4 – KITKAT NIVEL DE API 20

Vino acompañada con la presentación del Nexus 5, el cual se puede observar en la Figura 54 mostrando a su vez la pantalla de inicio predeterminada de esta versión. Se permitió que dispositivos con 512MB de RAM pudieran ejecutar Android sin problemas lo que ha permitido una mejora muy significativa de la gama baja. Actualmente es la versión más utilizada (Cosmo, 2014).

Figura 54 – Interfaz Nexus V

La interfaz pasó a tener un tono más grisáceo, las carpetas se modificaron y se lanzó el Google Experience Launcher para los Nexus. También ha sido desde KitKat cuando el comando por voz de "OK, Google" se empieza a utilizar. Otro cambio es la manera de añadir widgets y wallpapers o el dialer con interfaces distintas a las anteriores.

ANDROID 5.0 – LOLLIPOP

Fue presentada el 3 de noviembre del 2014, es la última versión conocida. Las características principales y novedosas de la siguiente versión son las siguientes (Cosmo, 2015):

- Material Design:

  Un diseño intrépido, colorido, y sensible interfaz de usuario para las experiencias coherentes e intuitivas en todos los dispositivos. Movimiento de respuesta natural, iluminación y sombras realistas y familiares, elementos visuales hacen que sea más fácil de navegar su dispositivo. Nuevos colores vivos, tipografía e imágenes de ayuda de borde a borde de enfocar su atención.

- Notificaciones:

  Nuevas formas de controlar cuándo y cómo se reciben mensajes - sólo ser interrumpido cuando se quiere ser. Ver y responder a mensajes directamente desde la pantalla de bloqueo. Incluye la capacidad de ocultar contenido sensible para estas notificaciones. Se puede programar el tiempo durante el cual sólo las notificaciones de prioridad aparecen. También, las llamadas entrantes no interrumpen lo que estés haciendo. Se puede optar por responder a la llamada o simplemente seguir haciendo lo que se esté haciendo. Clasificación más inteligente de notificaciones. Ver todas las notificaciones en un solo lugar tocando la parte superior de la pantalla.

- Batería:

  Una característica de ahorro de batería que se extiende el uso de dispositivos de hasta 90 minutos. El tiempo estimado de batería restante aparece cuando el dispositivo está enchufado. El tiempo restante de batería antes de tener que cargar el dispositivo de nuevo ahora se puede encontrar en la configuración de la batería.

ANEXO II.

ESTILO DE CÓDIGO Y NORMAS MOODLE

ESTILO DEL CÓDIGO

Cualquier proyecto colaborativo necesita que la consistencia y la estabilidad sean fuertes. Siguiendo el manual de estilo de código desarrollado en la documentación de *Moodle* se ha tratado de cumplir todas las reglas explicadas en él. Todo el código nuevo definitivamente deberá adherirse a estos estándares de la forma más exacta posible (Moodle, 2015e).

Reglas generales

Las reglas generales que en este manual se especifican son las siguientes:

- Todos los archivos de código deberían utilizar la extensión .php.
- Todas las plantillas deberían utilizar la extensión .html.
- Todos los archivos de texto deberían utilizar el formato de texto Unix (la mayoría de los editores de texto tienen esto como una opción).
- Todas las etiquetas PHP deben ser 'completas' como <?php ?> ... no 'reducidas' como <? ?>.
- Todos los avisos de copyright deben ser mantenidos. Puede incluir los suyos propios si resulta necesario.
- Todos los archivos deben incluir el archivo principal config.php.
- Cualquier otro include/require debería utilizar una ruta absoluta que comience por $CFG->dirroot o $CFG->libdir, nunca relativos, ya que estos en algunas ocasiones funcionan de forma extraña en PHP.
- Cada archivo debería comprobar que el usuario está autenticado correctamente, utilizando las funciones require_login() y isadmin(), isteacher(), iscreator() o isstudent().
- Todos los accesos a la base de datos deberían utilizar las funciones definidas en lib/datalib.php cuando sea posible, esto permite la compatibilidad con un gran número de bases de datos. Si se quiere escribir código SQL entonces se deberá comprobar que: funciona en cualquier plataforma; restringido a funciones específicas de su código (normalmente un archivo lib.php); y claramente comentado.

- No se deben crear o utilizar variables globales distintas de las estándar $CFG, $SESSION, $THEME, $SITE, $COURSE y $USER.

- Todas las variables deberían ser inicializadas o, al menos, comprobada su existencia utilizando isset() o empty() antes de ser utilizadas.

- Todas las cadenas deberían ser traducibles. Para ello, se deben crear nuevos textos en los archivos lang/es_utf8 con palabras reducidas en inglés y su traducción completa al español. Para recuperarlas en el código se utilizan las funciones get_string() o print_string().

- Todos los errores deberían ser visualizados utilizando la función print_error() para maximizar la traducción y ayudar a los usuarios (automáticamente se enlaza con Moodle Docs).

- Todos los ficheros de ayuda deben ser traducibles. Para ello, se deben crear nuevos textos en el directorio lang/es_utf8/help y llamarlos utilizando la función helpbutton(). Si se necesita actualizar un fichero de ayuda:

- Para un pequeño cambio, donde la traducción antigua del fichero podría tener todavía sentido, está permitido que se haga el cambio, pero se debería notificarlo a translation@moodle.org.

- Para un cambio importante se tendría que crear un nuevo fichero añadiéndole en el nombre un número incrementado (p.ej. filename2.html) para que los traductores puedan ver fácilmente que se trata de una nueva versión del archivo. Obviamente el nuevo código y los índices de las páginas de ayuda deben ser modificados para apuntar a las versiones más recientes.

- La información que llega desde el navegador (enviada con los métodos GET o POST) automáticamente tiene las magic_quotes aplicadas (sin importar la configuración de PHP) por lo que se puedes insertar con total seguridad en la base de datos. El resto de la información (obtenida desde los archivos, o desde la base de datos) debe ser escapada con la función addslashes() antes de insertarla en la base de datos.

- Muy importante: Todos los textos dentro de Moodle, especialmente aquellos que han sido introducidos por los usuarios, deben ser mostrados utilizando la función format_text(). Esto asegura que el texto es filtrado y limpiado correctamente.

- Al generar enlaces HTML, se deben hacer siempre relativos a la raíz del sitio Moodle. Esto permite que su código funcione aunque sea llamado por un script que se encuentre en otra carpeta diferente.

Estilo del código

Aunque pueda resultar un poco complicado modificar el estilo de programación personal de cada uno, es comprensible que, debido a que Moodle evoluciona gracias a una comunidad de desarrolladores, se trate de llegar a un estilo común puesto que puede resultar ciertamente complicado encontrarle sentido al código de Moodle si está compuesto por una mezcla de estilos.
Hay muchos puntos a favor y en contra de cada estilo que la gente utiliza, pero el que se detalla a continuación, es el que se debe emplear.

- El sangrado del texto debe ser siempre de 4 espacios. No se deben utilizar los tabuladores nunca.

- Los nombres de las variables tienen que ser siempre fáciles de leer, procurando que sean palabras en minúsculas con significado en inglés. Si realmente se necesita más de una palabra, se deben poner juntas, pero procurando que sean tan breves como sea posible. Se deben utilizar nombres en plural para las matrices de objetos.

- Las constantes tienen que definirse siempre en mayúsculas, y empezar siempre por el nombre del módulo al que pertenecen. Deberían tener las palabras separadas por guiones bajos.

- Los nombres de las funciones tienen que ser palabras sencillas en minúsculas y en inglés, y empezar con el nombre del módulo al que pertenecen para evitar conflictos entre módulos. Las palabras deberían separarse por guiones bajos. Los parámetros, si es posible, tendrán valores por defecto. Se debe comprobar que no haya espacio entre el nombre de la función y lo siguiente (paréntesis).

- Los bloques de código siempre deben estar encerrados por llaves (incluso si solo constan de una línea).

- Las cadenas tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.

- Los comentarios deben ser añadidos de forma que resulten prácticos, para explicar el flujo del código y el propósito de las funciones y variables. Los

comentarios en línea deberían utilizar los caracteres //, alineados con cuidado por encima de las líneas de código que comenta.

- El espacio en blanco se puede utilizar con bastante libertad para ganar claridad. Generalmente, debería haber un espacio entre llaves y líneas normales y ninguno entre llaves y variables o funciones.

- Cuando se esté realizando una copia de un objeto, se debe utilizar siempre la función clone() originalmente sólo disponible en php5 (en caso contrario simplemente tendrá una referencia al primer objeto). Moodle garantiza que este método funcionará también bajo php4. Si lo que se quiere copiar no es un objeto, pero puede contener objetos (p.ej. un array de objetos) utilice la función fullclone() en su lugar.

Reglas en la gestión de la base de datos

A continuación se repasan las reglas básicas para la gestión de la base de datos.

- Cada tabla debe tener un campo autonumérico id (INT10) como clave primaria.

- La tabla principal que contiene instancias de cada módulo debe tener el mismo nombre que el módulo y contener, por lo menos, los siguientes campos:

    - id: descrito arriba.

    - course: el identificador del curso al que la instancia pertenece.

    - name: el nombre completo de la instancia.

- El resto de las tablas asociadas con un módulo que contiene información sobre 'cosas', deberían ser llamadas módulo_cosas.

- Los nombres de las tablas y de los campos tienen que evitar el uso de palabras reservadas por las bases de datos.

- Los nombres de los campos (columnas) deberían ser sencillos y cortos, siguiendo las mismas reglas que los nombres de las variables.

- Cuando sea posible, las columnas que contengan una referencia al campo id de otra tabla (por ejemplo, módulo) debería ser llamado móduloid.

- Los campos booleanos serán implementados como enteros cortos (por ejemplo, INT4) con los valores 0 o 1, para permitir la futura expansión de los valores si fuera necesario.

- La mayoría de las tablas tienen que tener un campo timemodified (INT10) que será actualizado con la fecha actual (timestamp de UNIX) obtenida con la función time() de PHP.

- Se debe definir siempre un valor por defecto con sentido para cada campo.

- Cada tabla debe comenzar con el prefijo de la base de datos ($CFG->prefix). En muchos casos esto es gestionado automáticamente. Además, bajo PostgreSQL, el nombre de cada índice debe empezar también con el prefijo.

- Para garantizar la compatibilidad entre bases de datos, se deben seguir las reglas siguientes sobre el uso del comando AS (sólo si se necesita alias en tablas y campos):

- No utilizar el comando AS para alias de tablas.

- Utilizar el comando AS para alias de campos (columnas).

- Nunca se deben emplear UNIQUE KEYs (restricciones) para nada. En su lugar se deben utilizar UNIQUE INDEXes. En el futuro, si se decide añadir integridad referencial a Moodle y si se necesitan UNIQUE KEYs, serán utilizadas, pero no por ahora.

- Esas UNIQUE KEYs creadas en el Editor XMLDB sólo deben ser definidas si el campo/campos van a ser el objetivo para alguna FOREIGN KEY (a nivel de Editor). En caso contrario, se deben crear como UNIQUE INDEXes.

- Nunca se deben realizar cambios a la base de datos en ramas estables. Si se hace eso, entonces los sitios actualizando de una versión estable a la siguiente pueden encontrarse con cambios por duplicado, lo cual puede producir errores serios.

- Cuando se haga referencia a una variable entera en consultas SQL, no se debe entrecomillar el valor. Por ejemplo, get_records_select('question' "category=$catid") es correcto, mientras que get_records_select ('question', "category='$catid'") es incorrecto. Ese uso oculta posibles errores cuando $catid está sin definir.

Normas de seguridad

- No basarse en register_globals. Cada variable debe ser correctamente inicializada.
- Inicialice todos los arrays y objetos aunque estén vacíos.
- No utilizar la función optional_variable(). En su lugar, utilizar la función optional_param().
- No utilizar la función require_variable(). En su lugar, utilizar la función required_param().
- Utilizar data_submitted() con cuidado. La información debe ser limpiada antes de utilizarla.
- No utilizar $_GET, $_POST o $_REQUEST. En su lugar, utilizar las funciones required_param() u optional_param().
- No comprobar las acciones con código como: if (isset($_GET['algo'])).
- Cuando sea posible agrupar todas las llamadas a required_param(), optional_param() y el resto de inicialización de variables en el principio de cada fichero.
- Utilizar el mecanismo sesskey para proteger el envío de formularios de ataques.
- Todos los nombres de ficheros deben ser 'limpiados' utilizando la función clean_filename().
- Cualquier información leída desde la base de datos debe tener la función addslashes() aplicada antes de volver a enviar la información a la base de datos.
- La información que se almacenará en la base de datos debe venir de peticiones POST.
- No utilizar información obtenida de $_SERVER.
- La información enviada a la base de datos debe ser filtrada mediante la función clean_param().
- Asegurarse de que el código SQL es correcto.

- Comprobar toda la información (especialmente la que es enviada a la base de datos) en cada archivo que es utilizada.

- Los bloques de código que se incluyan deben presentar una estructura PHP correcta.

- Para utilizar funciones que invoquen un Shell hay que asegurarse de que se han limpiado los parámetros anteriormente con escapeshellcmd() o escapeshellarg().