Universidad de Valladolid

E.T.S.I TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN TELEMÁTICA

TRABAJO FIN DE GRADO

Diseño y Desarrollo de una Aplicación Móvil Android para el apoyo a la docencia en el ámbito de la Traducción de Textos Económicos EN-ES

Autor:

Don. Jonatan Rafael Serna Pérez

Tutores:

Dña. Míriam Antón Rodríguez

Dña. Mª Ángeles Pérez Juárez

TÍTULO:	Diseño y Desarrollo de una Aplicación Móvil Android para el apoyo a la docencia en el ámbito de la Traducción de Textos Económicos EN-ES
AUTOR:	Jonatan Rafael Serna Pérez.
TUTORES:	Míriam Antón Rodríguez Mª Ángeles Pérez Juárez
DEPARTAMENTO:	Teoría de la señal y comunicaciones e ingeniería telemática

Miembros del tribunal

PRESIDENTE: Mª Ángeles Pérez Juárez

SECRETARIO: Míriam Antón Rodríguez

VOCAL: David González Ortega

SUPLENTE 1: Javier Manuel Aguiar Pérez

SUPLENTE 2: Mario Martínez Zarzuela

FECHA DE LECTURA: 03/06/2016

CALIFICACIÓN:

RESUMEN DEL PROYECTO

En la era de la tecnología y las comunicaciones, es de gran importancia adaptar los medios educativos utilizando las nuevas tecnologías de las que se dispone, haciendo la educación más accesible, atractiva y efectiva fuera del aula física tradicional. A este aprendizaje electrónico se le conoce con el anglicismo *e-learning*, que se define como: "las aplicaciones y servicios que, tomando como base las Tecnologías de la información y la comunicación (TIC), se orientan a facilitar el proceso de enseñanza-aprendizaje".

La gran penetración de los dispositivos móviles en la sociedad actual debe ser aprovechada como una oportunidad que se brinda al ámbito educativo para conectar a profesores y alumnos. El *e-learning* llevado al campo de la tecnología móvil es conocido como *m-learning*.

El ámbito del aprendizaje de idiomas y traducción de textos, es un tipo de aprendizaje que necesita un trabajo constante y continuo, por lo tanto, el *e-learning* y el *m-learning* cobran gran importancia, ya que permiten que el alumno tenga ese aprendizaje continuo y constante fuera del aula.

Una de las plataformas educativas más utilizadas en el ámbito del aprendizaje electrónico es la plataforma *Moodle*, que permite conectar a profesores y alumnos fuera de las aulas a través de internet, mediante foros de discusión, compartición de materiales educativos y posibilitando la realización de cuestionarios evaluables y/o formativos.

En este Trabajo Fin de Grado se propone el desarrollo de una aplicación móvil *Android* que se conecte con la plataforma *Moodle*, de forma que los alumnos puedan realizar desde un *Smartphone* los cuestionarios propuestos por el profesor en la plataforma. Esta aplicación proporcionará al alumno mayor accesibilidad, simplicidad y amigabilidad a la hora de realizar los cuestionarios ya que no será necesario que el alumno acceda a la plataforma web para realizarlos.

PALABRAS CLAVE

Aprendizaje electrónico, Android, Moodle, Servicios Web, Cuestionarios, PHP.

ABSTRACT

In the technology and communications age, it is very important to adapt the education resources using available new technologies, making education more accessible, attractive and effective outside the traditional physical classroom. This electronic learning is known by the anglicized *e-learning*, it´s defined as: “applications and services, based on information technology and communications (ITC), it´s directed to facilitate the teaching-learning process”.

The huge penetration of mobile devices in current society must be utilized as an opportunity for education to connect teachers and students. E-learning in the mobile technology is known as m-learning.

The area of language learning and text translation is a type of learning that needs a constant and continuous work, therefore, e-learning and m-learning receive a great importance, and they allow the students to have a continuous and constant learning outside the classroom.

One of the most used educational platforms in the area of electronic learning is Moodle, it allows to connect teachers and students outside the classroom by Internet, through discussion forums, sharing of educational materials and the possibility of making evaluable or training quizzes.

This Final Project aims to develop an Android mobile application to connect with Moodle Platform, so students can make quizzes proposed by the teacher on the platform with a Smartphone. This application provides students with greater accessibility, simplicity and friendliness when they make the quizzes.

KEYWORDS

Electronic learning, Android, Moodle, Web Services, quizzes, PHP.

Agradecer a mi familia que siempre confía en mí, a mis amigos y a mi pareja por su apoyo constante durante 4 años importantes.

Agradecer a todos mis profesores y compañeros que me han ayudado a disfrutar de esta carrera y transformar un *hobby* en una profesión.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPITULO 1.
INTRODUCCIÓN

1. Introducción general

La era de las comunicaciones, esta es la forma en la que numerosos autores de libros y tesis describen el cambio que la sociedad actual está sufriendo a partir de los avances tecnológicos que día a día aparecen.

La sociedad y los aspectos que rodean a esta se han visto en la obligación de acoger estos avances tecnológicos. La economía, los mercados, las industrias, los medios de comunicación… un gran conjunto de sectores se han adaptado rápidamente a los cambios tecnológicos que esta nueva era ha traído consigo. La educación, sin embargo, se muestra algo más reticente a acoger estos cambios y no aprovechar al completo la gran variedad de herramientas que la tecnología pone a su disposición.

Las tecnologías de la información obligan a modificar la organización de la educación, porque crean entornos educativos que amplían considerablemente las posibilidades del sistema, no sólo de tipo organizativo, sino también de transmisión de conocimientos y desarrollo de destrezas, habilidades y actitudes. La clave está en transformar la información en conocimiento y éste, en educación y aprendizaje significativo (Ruiz, 1996).

El pensamiento de utilizar la tecnología como una ayuda imprescindible en la enseñanza coge fuerza con la llegada del *e-learning*, son cada vez más los centros educativos que cuentan con *tablets*, profesores que utilizan redes sociales para mantener contacto con sus alumnos, centros que cuentan con plataformas online, etc.

Una de las plataformas más utilizadas por centros educativos, en especial por universidades, es la plataforma *Moodle*. Esta plataforma posibilita a profesores y alumnos compartir recursos, crear foros de discusión e incluso permite a los profesores evaluar los conocimientos de sus alumnos. *Moodle* es una plataforma *online* a la que se accede a través de un navegador web (Moodle, 2016).

La tendencia que se está siguiendo en la actualidad es dirigir el *e-learning* al *m-learning (e-learning* sobre dispositivos móviles*)* (América, 2016). Según un informe de la fundación telefónica de 2014, "el tiempo de acceso a medios digitales utilizado en dispositivos en movilidad (*smartphone* + *tablet*) supera al empleado en el PC, 53% frente a 47%" (Telefónica, 2014).

Los *smartphone* son dispositivos con capacidades limitadas: memoria, velocidad de procesamiento, resolución de pantalla, velocidad de acceso a red… lo que provoca la necesidad de crear aplicaciones de escritorio optimizadas para esos dispositivos en lugar de acceder desde ellos a aplicaciones web. Las aplicaciones de escritorio están desarrolladas para adaptarse a las limitaciones de los dispositivos móviles, sencillez, velocidad, bajo consumo de datos de red, pantallas de tamaño reducido, etc.

2. Motivación

Como se ha explicado en la sección anterior, "Introducción general", el *m-learning* es el camino a seguir en la enseñanza debido a las posibilidades que este ofrece. Estas posibilidades se hacen aún más patentes en ámbitos de la educación que necesitan de un aprendizaje continuo y constante como es el caso aprendizaje de idiomas. Aprender un idioma necesita de un trabajo continuo fuera del aula y el *m-learning* proporciona esta posibilidad. Este factor es la motivación principal para realizar este proyecto en colaboración con la facultad de traducción e interpretación de Soria perteneciente a la Universidad de Valladolid (UVA). El proyecto se dirige de forma global a la facultad anteriormente mencionada y de forma algo más específica se encamina a satisfacer las necesidades que presenta la asignatura "Traducción económica Lengua B (inglés), optativa de cuarto curso del Grado en Traducción e Interpretación". Sin embargo, al compartir todas las facultades de la Universidad de Valladolid el mismo campus virtual (http://campusvirtual2015.uva.es/), este proyecto es aprovechable por cualquiera de ellas.

Las posibilidades que ofrece el *m-learning*, unido a las peculiaridades que tiene el aprendizaje de idiomas y las necesidades que presenta la facultad de traducción e interpretación de la universidad de Valladolid en cuanto a facilitar el aprendizaje de sus alumnos fuera del aula, motivan el desarrollo de este proyecto.

La comunidad *Moodle*[1]ha observado la tendencia al *m-learning* y la necesidad de crear una aplicación móvil compatible con sus plataformas web, por lo que ha desarrollado la aplicación móvil oficial de *Moodle*, "Moodle Mobile", esta aplicación es compatible con todas las plataformas *Moodle* actualizadas a la versión 2.4 y únicamente necesita una sencilla configuración en la plataforma (Moodle Mobile, 2012).

La aplicación se encuentra en continuo desarrollo, como se muestra en la figura 1, la última actualización se realizó el 29 de febrero de 2016. Sin embargo, la aplicación aun no es compatibilidad con algunos módulos, como se muestra en las figuras 2 y 3 con el módulo cuestionario. En la figura 2 se muestra como el curso consultado contiene un cuestionario, sin embargo, en la figura 3 se observa que la aplicación no soporta este tipo de actividad e insta al usuario a acceder al sitio web para realizar el cuestionario, algo que como ya se ha explicado anteriormente no es óptimo para dispositivos móviles ya que se degrada la experiencia de usuario.

La aplicación "Moodle Mobile" no tiene soporte para el módulo cuestionario debido a que el módulo "cuestionario" de la plataforma *Moodle*, en la versión actual *Moodle 3.0.3*, no tiene desarrolladas las funciones externas que posibilitan el acceso a este módulo desde un cliente externo, a diferencia de otros módulos como encuestas, cursos... que si tienen desarrolladas estas funciones externas.

[1] *Moodle es una plataforma de código abierto, desarrollada y actualizada por la comunidad de desarrolladores de Moodle.*

Que la versión actual de la plataforma *Moodle* no tengan desarrolladas las funciones externas del módulo "cuestionario". Que la aplicación oficial de *Moodle*, "Moodle Mobile" no ofrezca la posibilidad de realizar cuestionarios desde ella. Y la necesidad que presenta la facultad de traducción e interpretación de la universidad de Valladolid de ofrecer una plataforma móvil desde la cual poder resolver los cuestionarios que los profesores publican en la plataforma *Moodle*, motivan el desarrollo de este proyecto.

Figura 1- Fecha de actualización de Moodle Mobile

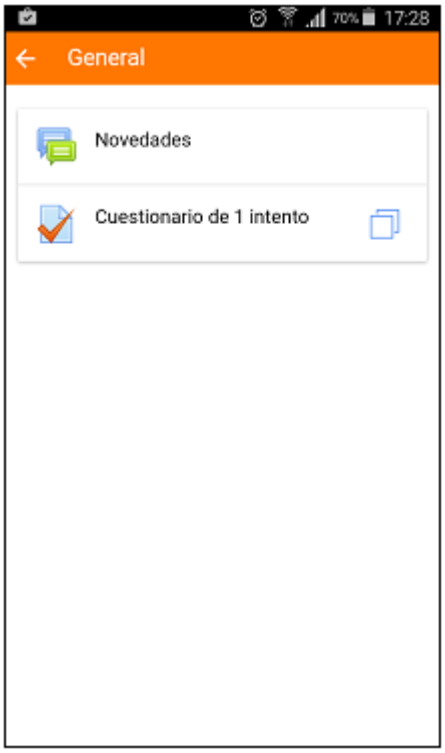

Figura 2 - Moodle Mobile, curso con cuestionario.

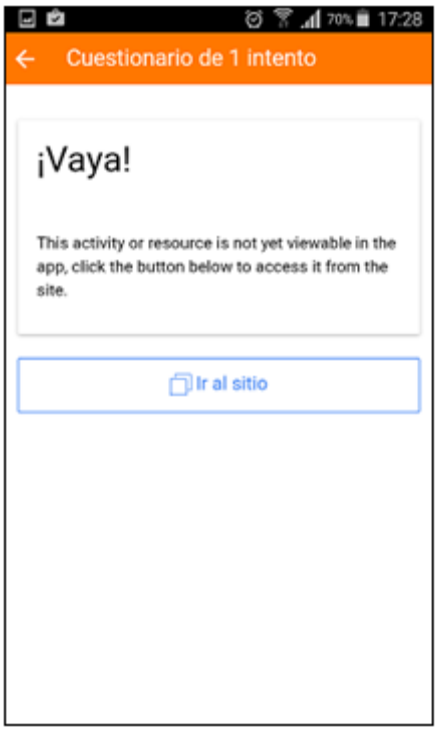

Figura 3 - Moodle Mobile, redirección de cuestionario

3. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación móvil que permita a los alumnos de la facultad de traducción e interpretación de la universidad de Valladolid realizar desde su dispositivo móvil, los cuestionarios que sus profesores "cuelguen" en la plataforma *Moodle* de la universidad.

Como objetivo secundario se perseguirá que la aplicación móvil sea fácilmente adaptable a las plataformas *Moodle* de otras facultades y universidades, necesitando cambios mínimos tanto en la aplicación móvil como en la plataforma *Moodle*.

Para conseguir estos objetivos se desarrollarán los siguientes procesos que sumados darán la solución buscada:

- Instalación de una plataforma *Moodle* de desarrollo en la que realizar las pruebas de integración con la aplicación móvil. En la plataforma se dará de alta a varios usuarios, algunos con roles de profesores y otros con roles de alumno, se crearán diferentes cursos y se matriculará a cada alumno en varios cursos de forma aleatoria. Accediendo a la plataforma un profesor debe poder crear un cuestionario en la asignatura de la que sea profesor, y un alumno matriculado en esa asignatura debe poder resolver el cuestionario accediendo a la aplicación móvil.

- Programación de las funciones internas del módulo "Cuestionario" de *Moodle* necesarias, tales como: obtener el número de intentos realizados por un usuario para un cuestionario, obtener las preguntas de un cuestionario y responder un cuestionario. Para el desarrollo de estas funciones se utilizarán y aprovecharán las funciones del módulo "Cuestionario" que vienen por defecto en *Moodle 3.0* definidas en los ficheros lib.php y locallib.php dentro del directorio "quiz".

- Desarrollo de los servicios web o funciones externas del modulo "Cuestionarios" que permitan a la aplicación móvil interactuar con las funciones internas del módulo. Para el desarrollo de estas funciones se utilizarán y aprovecharán las funciones del módulo "Cuestionario" que vienen por defecto en *Moodle 3.0* definidas en los ficheros lib.php y locallib.php dentro del directorio "quiz".

- Integrar los servicios web en la plataforma *Moodle* de modo que estos sean accesibles desde la aplicación móvil.

- Desarrollo de la aplicación móvil que interactúe con la plataforma *Moodle*, teniendo este proceso varios pasos:
    - Desarrollo de los modelos de objetos necesarios.
    - Desarrollo de las peticiones web y las tareas asíncronas que utilizará la aplicación para interactuar con la plataforma *Moodle*.
    - Desarrollo de *layouts* y *widgets* necesarios.

- Desarrollo de las actividades que formarán la aplicación.

- Realizar los cambios pertinentes en la plataforma *Moodle* que utiliza la facultad de de traducción e interpretación de la universidad de Valladolid (http://campusvirtual2015.uva.es/), para desplegar la aplicación móvil.

4. Introducción al documento

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

- El primer capítulo es una introducción al proyecto y al presente documento, explicando los objetivos y las motivaciones que han llevado al desarrollo del mismo.

- En el segundo capítulo se realizará un análisis de las tecnologías que se han utilizado en el proyecto, de posibles alternativas y se expondrán los motivos de la elección de cada una.

- En el tercer capítulo se aborda el desarrollo del proyecto, tanto los cambios en la plataforma *Moodle*, como el desarrollo de la aplicación móvil.

- El cuarto capítulo estará formado por tres guías de usuario: guía del administrador de la plataforma *M*oodle, guía del profesor y guía del usuario de la aplicación móvil.

- En los últimos 3 capítulos se presentarán, el presupuesto, las conclusiones del proyecto y las líneas futuras.

CAPITULO 2.
TECNOLOGÍAS

1. Introducción

En este capítulo se expondrán las tecnologías empleadas en el proyecto, se analizarán las posibles alternativas y se dará una justificación a las tecnologías utilizadas.

El proyecto a desarrollar es un sistema basado en cliente-servidor. La parte del servidor contendrá la información del sistema, información de usuarios, información de cursos, cuestionarios, foros, etc. La parte cliente estará compuesta por una aplicación móvil que accede a la información del servidor para mostrársela al usuario.

La parte servidora se desarrollará con un sistema de gestión de contenido denominado *Moodle* ya que, como se explicará a continuación, los sistemas de gestión de contenidos simplifican notoriamente el trabajo en comparación a si se codificasen a mano todas las funcionalidades con un editor de texto. La justificación de que sea *Moodle* el sistema de gestión de contenido elegido se explicará en el apartado 2 de este capítulo.

Para la parte cliente se realizará una aplicación móvil en lugar de utilizar la interfaz web que trae *Moodle* por defecto debido a las razones expuestas en el capítulo introducción. Se realizará un desarrollo nativo para el SO Android, esta elección se justificará en el apartado 3 de este capítulo.

2. Sistemas de gestión de contenido (CMS)

Realizar un sitio web que gestione gran cantidad de información puede ser un trabajo complicado y muy laborioso si no se dispone de las herramientas adecuadas. En el pasado las herramientas eran básicamente editores que permitían generar una página, que evolucionaron para incorporar el control de la estructura de la web y otras funcionalidades, pero en general estaban enfocadas más a la creación que al mantenimiento (García, 2004). En los últimos años se ha desarrollado el concepto de sistema de gestión de contenidos (Content Management System o CMS). Se trata de herramientas que permiten crear y mantener un sitio web con facilidad, encargándose de los trabajos más tediosos que hasta ahora ocupaban el tiempo de los administradores de las webs.

Teniendo en cuenta el ahorro que supone la utilización de estas herramientas, y el coste de desarrollarlas, sería lógico esperar que su precio fuera muy elevado. Eso es cierto para algunos productos comerciales, pero existen potentes herramientas de gestión de contenidos de acceso libre, disponibles con licencias de código abierto.

Los gestores de contenidos proporcionan un entorno que posibilita la actualización, mantenimiento y ampliación de la web de forma sencilla. En cualquier entorno virtual ésta es una característica importante y aún en mayor medida en entornos de trabajo donde la parte de administración y de desarrollo debe llevarse a cabo por un grupo reducido de personas, como es el caso de este proyecto.

A la hora de seleccionar un gestor de contenidos se deben tener en cuenta los objetivos que se quieran alcanzar, ya que existen multitud de gestores de contenido, cada cual dirigidos a satisfacer en mayor medida unos objetivos frente a otros.

2.1 Historia de los CMS

A principios de los años noventa, el concepto de sistemas de gestión de contenidos era desconocido. Algunas de sus funciones se realizaban con aplicaciones independientes: editores de texto y de imágenes, bases de datos y programación a medida.

Ya el año 1994 Illustra Information Technology utilizaba una base de datos de objetos como repositorio de los contenidos de una web, con el objetivo de poder reutilizar los objetos y ofrecía a los autores un entorno para la creación basado en patrones. La idea no cuajó entre el público y la parte de la empresa enfocada a la Web fue comprada por AOL, mientras que Informix adquirió la parte de bases de datos.

RedDot es una de las empresas pioneras que empezó el desarrollo de un gestor de contenidos el año 1994. No fue hasta a finales del año siguiente que presentaron su CMS basado en una base de datos.

Entre los CMS de código abierto uno de los primeros fue Typo 3, que empezó su desarrollo el año 1997, en palabras de su autor, Kasper Skårhøj, "antes de que el término gestión de contenidos fuera conocido sobradamente".

PHPNuke, la herramienta que popularizó el uso de estos sistemas para las comunidades de usuarios en Internet, se empezó a desarrollar el año 2000. La primera versión supuso tres semanas de trabajo al creador, rescribiendo el código de otra herramienta, Thatware (García, 2004).

Tendencia que han seguido los CMS

Aparte de la ampliación de las funcionalidades, uno de los campos más interesantes en la evolución de los CMS ha sido la utilización de estándares que mejoran la compatibilidad de componentes dentro de una plataforma, facilitan el aprendizaje al cambiar de sistema y aportan calidad y estabilidad (García, 2004).

Algunos de estos estándares son CSS, que permite la creación de hojas de estilo; XML, un lenguaje de marcas que permite estructurar un documento; XHTML, que es un subconjunto del anterior orientado a la presentación de documentos vía web; WAI, que asegura la accesibilidad del sistema; y RSS, para sindicar contenidos de tipo noticia.

También el resto de componentes que rodean los CMS acostumbran a ser estándar (de facto), como los servidores web Apache e ISS; los lenguajes PHP, Perl y Python; y las bases de datos MySQL y PostgreSQL. La disponibilidad para los principales sistemas operativos de estas aplicaciones y módulos, permite que los CMS puedan funcionar en diversas plataformas sin muchas modificaciones.

2.2 Funcionalidades de un CMS

James Robertson (2003) propone una división de la funcionalidad de los sistemas de gestión de contenidos en cuatro categorías: creación de contenido, gestión de contenido, publicación y presentación.

Creación de contenido

Un CMS aporta herramientas para que los creadores sin conocimientos técnicos en páginas web puedan concentrarse en el contenido. Lo más habitual es proporcionar un editor de texto WYSIWYG (What You See Is What You Get), en el que el usuario ve el resultado final mientras escribe, al estilo de los editores comerciales, pero con un rango de formatos de texto limitado. Esta limitación tiene sentido, ya que el objetivo es que el creador pueda poner énfasis en algunos puntos, pero sin modificar mucho el estilo general del sitio web.

Hay otras herramientas como la edición de los documentos en XML, utilización de aplicaciones ofimáticas con las que se integra el CMS, importación de documentos existentes y editores que permiten añadir marcas, habitualmente HTML, para indicar el formato y estructura de un documento.

Un CMS puede incorporar una o varias de estas herramientas, pero siempre tendría que proporcionar un editor WYSIWYG por su facilidad de uso y la comodidad de acceso desde cualquier ordenador con un navegador y acceso a Internet.

Para la creación del sitio propiamente dicho, los CMS aportan herramientas para definir la estructura, el formato de las páginas, el aspecto visual, uso de patrones, y un sistema modular que permite incluir funciones no previstas originalmente.

Gestión de contenido

Los documentos creados se depositan en una base de datos central donde también se guardan el resto de datos de la web, cómo son los datos relativos a los documentos (versiones hechas, autor, fecha de publicación y caducidad, etc.), datos y preferencias de los usuarios, la estructura de la web, etc.

La estructura de la web se puede configurar con una herramienta que, habitualmente, presenta una visión jerárquica del sitio y permite modificaciones. Mediante esta estructura se puede asignar un grupo a cada área, con responsables, editores, autores y usuarios con diferentes permisos. Eso es imprescindible para facilitar el ciclo de trabajo (workflow) con un circuito de edición que va desde el autor hasta el responsable final de la publicación. El CMS permite la comunicación entre los miembros del grupo y hace un seguimiento del estado de cada paso del ciclo de trabajo.

El almacenamiento de todos los datos en una base de datos hace que el diseño del sitio web sea fácilmente adaptativo, ya que por un lado están los datos y por otro lado la

estructura, modificar la estructura de la web no afecta a los datos y viceversa. Esta separación de datos y estructura de la web es uno de los puntos más importantes que tiene un CMS y se explicará con algo más profundidad en el siguiente punto "Publicación".

Publicación

Una página aprobada se publica automáticamente cuando llega la fecha de publicación, y cuando caduca se archiva para futuras referencias. En su publicación se aplica el patrón definido para toda la web o para la sección concreta donde está situada, de forma que el resultado final es un sitio web con un aspecto consistente en todas sus páginas. Esta separación entre contenido y forma permite modificar el aspecto visual de un sitio web sin afectar a los documentos ya creados y libera a los autores de preocuparse por el diseño final de sus páginas.

Presentación

Un CMS puede gestionar automáticamente la accesibilidad del web, con soporte de normas internacionales de accesibilidad como WAI (World Wide Web Consortium, 2005), y adaptarse a las preferencias o necesidades de cada usuario. También puede proporcionar compatibilidad con los diferentes navegadores disponibles en todas las plataformas (Windows, Linux, Mac, Palm, etc.) y su capacidad de internacionalización lo permite adaptarse al idioma, sistema de medidas y cultura del visitante.

El sistema se encarga de gestionar muchos otros aspectos como son los menús de navegación o la jerarquía de la página actual dentro del web, añadiendo enlaces de forma automática. También gestiona todos los módulos, internos o externos, que incorpore al sistema. Así por ejemplo, con un módulo de noticias se presentarían las novedades aparecidas en otra web, con un módulo de publicidad se mostraría un anuncio o mensaje animado, y con un módulo de foro se podría mostrar, en la página principal, el título de los últimos mensajes recibidos. Todo eso con los enlaces correspondientes y, evidentemente, siguiendo el patrón que los diseñadores hayan creado.

2.3 Criterios de selección

Antes de empezar el proceso de selección de un CMS concreto, hay que tener claros los objetivos que se persiguen, teniendo en cuenta al público destinatario, y estableciendo una serie de requerimientos que tendría que poder satisfacer el CMS.

A continuación se enumeran los requerimientos generales mínimos que debe tener un *CMS* siguiendo el estudio realizado por el "Centro de Apoyo Tecnológico a Emprendedores" (CATE, 2012) y las funcionalidades mínimas que se busca para este proyecto.

Requerimientos generales mínimos:

- **Código abierto**. El CMS debe ser de código abierto (o libre) de modo que se puedan realizar cambios en pro de las necesidades de cada momento del proyecto.
- **Arquitectura técnica**. Tiene que ser fiable y permitir la escalabilidad del sistema para adecuarse a futuras necesidades con módulos. También tiene que haber una separación de los conceptos de contenido, presentación y estructura que permita la modificación de uno de ellos sin afectar a los otros.
- **Grado de desarrollo**. Madurez de la aplicación y disponibilidad de módulos que le añaden funcionalidades.
- **Soporte**. La herramienta tiene que tener soporte tanto por parte de los creadores como por otros desarrolladores. De esta manera se puede asegurar de que en el futuro habrá mejoras de la herramienta y que se podrá encontrar respuesta a los posibles problemas.
- **Posición en el mercado y opiniones**. Una herramienta poco conocida puede ser muy buena, pero hay que asegurar de que tiene un cierto futuro.
- **Usabilidad**. La herramienta tiene que ser fácil de utilizar y aprender. Los usuarios no siempre serán técnicos, por lo tanto hace falta asegurar que podrán utilizar la herramienta sin muchos esfuerzos y sacarle el máximo rendimiento.
- **Accesibilidad**. Para asegurar la accesibilidad de una web, el CMS debe cumplir un estándar de accesibilidad. El más extendido es WAI (Web Accessibility Initiative) del World Wide Web Consortium.
- **Velocidad de descarga**. Teniendo en cuenta que no todos los usuarios disponen de líneas de alta velocidad, las páginas se deben cargar rápidamente o dar la opción.
- **Servicios web.** Debe tener soporte para servicios web que permitan la interacción de una aplicación cliente con los datos del CMS.

Funcionalidades mínimas:

Cada herramienta *CMS* presenta unas funcionalidades diferentes ya que están encaminadas a un uso diferente, en la siguiente lista se presentan las funcionalidades mínimas necesarias para este proyecto:

- Editor de texto WYSIWYG a través del navegador.
- Herramienta de búsqueda.
- Comunicación entre los usuarios (foros, correo electrónico, chat).
- Ciclo de trabajo (workflow) con diferentes perfiles de usuarios y grupos de trabajo.
- Fechas de publicación y caducidad.
- Carga y descarga de documentos y material multimedia.
- Avisos de actualización de páginas o mensajes en los foros, y envío automático de avisos por correo electrónico.
- Disponibilidad o posibilidad de traducción a diferentes idiomas.
- Soporte de múltiples formados (HTML, Word, Excel, Acrobat, etc.).
- Soporte de múltiples navegadores (Internet Explorer, Netscape, etc.).
- Estadísticas de uso e informes.
- Administración de profesores y alumnos.
- Aulas virtuales que contengan toda la información de un curso y permitan la comunicación con foros o con chats.
- Creación, mantenimiento y publicación del material de un curso, con soporte de diferentes formatos, incluidos audio y vídeo.
- Talleres virtuales.
- Exámenes y *testes* con valoraciones.
- Trabajos con fecha de límite de entrega y aviso al profesor en caso de incumplimiento.
- Seguimiento estadístico de las acciones del estudiante.

2.4 CMS generales de código abierto

En el mercado de los CMS existen gran variedad de productos, tanto gratuitos de código abierto como de pago propietarios, los más utilizados son los CMS de código abierto ya que presentan multitud de ventajas, entre las que destacan:

- Son gratuitos.
- Poseen comunidades de desarrolladores multitudinarias por lo que están constantemente actualizándose y la ayuda de soporte es elevada.
- Cumplen los requerimientos necesarios para la mayoría de los proyectos.
- Sencillez de utilización.

Los CMS de código abierto más populares son 3, Joomla!, Drupal y WordPress, dependiendo la documentación que se consulte el orden de popularidad varía entre los tres, pero todas las fuentes coinciden en que estos son los CMS que ocupan el podio (CATE, 2012).

Cada uno de los 3 CMS nombrados tiene unas características que deben tenerse en cuenta a la hora de seleccionar uno u otro para realizar un proyecto. Ambos tres podrían utilizarse para realizar el proyecto que en este informe se propone, sin embargo, necesitarían un alto nivel de adaptación y desarrollo para cumplir con las necesidades de este proyecto ya que se tratan de CMSs de carácter general y no orientados al *e-learning.*

2.5 Los CMS en E-Learning (LMS)

Ante la alta demanda y utilización de los CMS en el ámbito del *e-learning*, el mercado de los CMS ha generado un conjunto de herramientas orientadas a este sector específico denominadas sistemas de gestión de aprendizaje *o Learning Management System (LMS)*

El *e-learning* tiene unas necesidades específicas que un CMS general no siempre cubre, o si lo hace, no da las mismas facilidades como una herramienta creada específicamente para esta función.

En general, los sistemas de gestión de aprendizaje facilitan la interacción entre los profesores y los estudiantes, aportan herramientas para la gestión de contenidos académicos y permiten el seguimiento y la valoración de los estudiantes. Es decir, facilitan una translación del modelo real en el mundo virtual.

A continuación se presentan los 3 sistemas de gestión de aprendizaje más populares según el estudio realizado por la firma estadounidense Capterra (2015):

Edmodo:

Edmodo es una plataforma social educativa gratuita que permite la comunicación entre los alumnos y los profesores en un entorno cerrado y privado a modo de *micro-blogs.*

Edmodo proporciona al docente un espacio virtual privado en el que se pueden compartir mensajes, archivos y enlaces, un calendario de aula, así como proponer tareas y actividades y gestionarlas. Actualmente cuenta con más de 3.000.000 de usuarios activos (Edmodo, 2016).

En la actualidad, Edmodo permite:

Crear grupos privados con acceso limitado a docentes, alumnos y padres. Disponer de un espacio de comunicación entre los diferentes roles mediante mensajes y alertas. Gestionar las calificaciones de los alumnos. Compartir diversos recursos multimedia: archivos, enlaces, vídeos, etc. Enviar encuestas a los alumnos. Asignar tareas a los alumnos y gestionar las calificaciones de las mismas. Gestionar un calendario de clase. Crear comunidades donde agrupar a todos los docentes y alumnos de un centro educativo Dar acceso a los padres a los grupos en los que estén asignados sus hijos, permitiendo estar informados de la actividad de sus hijos y tener la posibilidad de comunicación con los profesores. Conceder insignias a los alumnos como premios a su participación en el grupo; posibilidad de crear cuestionarios de evaluación (en fase de desarrollo). Gestionar los archivos y recursos compartidos a través de la biblioteca. Crear subgrupos para facilitar la gestión de grupos de trabajo. Disponer de un espacio público donde mostrar aquella actividad del grupo que el profesor estime oportuna. (Edmodo, 2016).

Según el estudio realizado por la firma estadounidense Capterra (2015) que se basa en número de usuarios, clientes e impacto en las redes sociales, Edmodo es la plataforma LMS más popular. Edmodo cuenta con 300.000 clientes y casi 50.000.000 de usuarios. Sin embargo, Edmodo no es apta para realizar este proyecto debido a varios factores, entre los que destacan que la funcionalidad de crear cuestionarios de evaluación, funcionalidad en la que se basa este proyecto, está en fase de desarrollo y que la plataforma web que utiliza la universidad de Valladolid es *Moodle* y no Edmodo.

Blackboard:

Según el estudio realizado por Capterra (Capterra, 2015), Blackboard (http://es.blackboard.com) es la tercera plataforma LMS más popular.

Blackboard es un LMS que puede funcionar con Java o .NET y está escrito bajo Java EE, es decir, la ingeniería de software es de más alto nivel que los LMS desarrollados con PHP. Puede funcionar tanto en UNIX como en Windows y utiliza ORACLE o SQL Server como contenedor de datos.

Es una plataforma licenciada, por lo que su mayor ventaja respecto a los productos de código abierto radica en el soporte que ofrecen.

Aunque Blackboard cumple con todas las características necesarias para realizar este proyecto, no se ha elegido como la solución más optima, debido a que se trata de una

plataforma licenciada que puede llegar a tener un alto coste no asumible para este proyecto.

Moodle:

Según el estudio realizado por Capterra (Capterra, 2015), *Moodle* (https://moodle.org) es la segunda plataforma más popular del mercado, sin embargo, en número de usuarios es la más popular con cerca de 80.000.000 de usuarios.

Moodle es una plataforma LMS de código abierto desarrollada en PHP, con un elevado número de desarrolladores que forman la "comunidad Moodle". El elevado número de desarrolladores ayudan a tener un soporte comparable al de las herramientas licenciadas gracias a la interactuación de los desarrolladores en foros, esto también permite que la plataforma esté en continuo desarrollo con nuevas versiones y actualizaciones.

Moodle cubre todos los requerimientos y funcionalidades mencionadas en el punto 2.3 (Criterios de selección) de este informe, por esta razón, por ser una plataforma de código abierto, por ser la plataforma más popular que cumple los requerimientos necesarios y por ser la plataforma que actualmente utiliza la universidad de Valladolid, *Moodle* ha sido elegida como la solución a adoptar para desarrollar este proyecto.

2.6 Moodle 3.0

La versión de *Moodle* con la que se ha desarrollado este proyecto es Moodle 3.0 ya que era la versión más actual en el momento de iniciar el desarrollo. En esta sección se hablará de la estructura y características generales de *Moodle* y se expondrán las características añadidas a Moodle 3.0.

2.6.1 Introducción

Moodle es un Sistema de Gestión de Cursos de Código Abierto (*Open Source Course Management System*). *Moodle* es un software que permite crear comunidades educativas, es decir, espacios donde un centro educativo, institución o empresa, gestiona recursos educativos proporcionados por unos docentes y organiza el acceso a estos recursos por los estudiantes, además de permitir la comunicación entre todos los implicados (alumnos y profesores).

Moodle es una plataforma cuya estructura principal es una portada generada en PHP a la que accede un navegador, esta portada usualmente incluye información acerca del sitio mismo y puede ser altamente personalizada. A esta portada se le añaden funcionalidades como autenticación de usuarios, cursos, calendario, foros, etc. Estas funcionalidades se generan a partir de módulos cargados en la plataforma, contra más módulos se añadan mayores serán las funcionalidades que ofrezca la plataforma.

En conclusión, *Moodle* es un LMS para el Manejo del Aprendizaje en línea gratuito, que permite a los educadores la creación de sus propios sitios web privados, llenos de

cursos dinámicos que extienden el aprendizaje, en cualquier momento y en cualquier sitio. (Moodle, 2015)

2.6.2 Estructura

Como se ha explicado en la sección anterior, *Moodle* tiene una serie de características modulares, incluyendo temas, actividades, interface 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 (Moodle, 2015).

El directorio principal de *Moodle* del servidor tiene almacenados todos los contenidos de la plataforma. Las carpetas y ficheros correspondientes a cada módulo se almacenan en una carpeta del directorio principal en función del objetivo que persigan. A continuación se detallan los elementos más importantes de este directorio para la versión 3.0 (Campos, 2015):

- **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 *Moodle.*
- **version.php:** Define la versión actual del código de *Moodle.*
- **index.php:** La página principal del sitio, donde hay que autenticarse.
- **help.php:** Muestra la página de ayuda cuando se pulsa sobre el icono de ayuda.
- **admin/:** Código para administrar el servidor.
- **auth/:** Módulos para la autenticación de usuarios.
- **blocks/:** Módulos para los 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 cursos.
- **enrol/:** Código para los módulos de inscripción.
- **error/:** Código sobre errores de la plataforma.
- **files/:** Código para representar y gestionar los archivos cargados.
- **filter/:** Código para gestionar los filtros de 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 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 (iconos).
- **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 usuario y sus intereses.
- **theme/:** Paquetes de temas para cambiar la apariencia del sitio.
- **user/:** Código para mostrar y gestionar los usuarios.
- **webservice/:** Código para la gestión de los servicios web.

En los siguientes apartados se desarrollan los tres módulos más importantes con los que se trabajará en este proyecto, curso, cuestionario (Quiz) y servicios web, se explicará su estructura y sus características.

2.6.3 Módulo curso

Uno de los módulos más importantes de *Moodle* son los "cursos", este módulo permite a un profesor crear un curso al que tendrán acceso los alumnos matriculados si el curso es privado, o cualquier usuario que acceda a la plataforma si el curso es público. Dentro de un curso el profesor deberá añadir actividades o recursos que serán accesibles para sus alumnos, las posibles actividades a añadir dependerán de la versión de *Moodle* con la que se trabaje o de los módulos añadidos a la plataforma.

El código para gestionar este módulo se encuentra en el directorio course/ que se encuentra en el directorio principal. El fichero más importante de este módulo para este proyecto es el fichero externallib.php que define el código de las funciones externas que se pueden invocar sobre este módulo.

En la versión 3.0 y desde la versión 2.2 *Moodle* trae los *webservices* (funciones externas) creados para el módulo curso.

2.6.4 Módulo cuestionario

El módulo de actividad con el que se trabajará en este proyecto será el módulo Cuestionario o *Quiz*, este módulo viene por defecto integrado en la plataforma *Moodle* desde la versión 1.5.4 (Moodle, 2016g), en la versión 3.0 se han añadido nuevas características que se comentarán en el apartado 2.6.6 de este informe. Este módulo permite al profesor crear exámenes con diferentes configuraciones y tipos de preguntas para examinar a sus alumnos.

Este es un módulo de actividad, por lo que al igual que el resto de módulos de actividad, su código se encuentra dentro del directorio mod/ del directorio raíz de *Moodle*. El código de este módulo está agrupado dentro del directorio /mod/quiz y para este proyecto, los ficheros más importantes son los siguientes:

- **db/install.xml:** define la estructura de las tablas de la base de datos para este módulo y alguna meta información.
- **lib.php y locallib.php:** todas las funciones definidas para el módulo deben estar aquí, estas funciones se utilizan para interactuar con las tablas de la base de datos y obtener, almacenar o modificar la información.

En la versión 3.0 de *Moodle*, el módulo Cuestionario no trae creados los servicios web que hacen a este módulo accesible desde un cliente externo, por lo que si estos se necesitan deben crearse de la manera que se explica en el siguiente apartado.

2.6.5 Servicios web

Los servicios web vienen por defecto en *Moodle* desde la versión 2.0, esto permite a un cliente externo obtener, modificar o generar información en las bases de datos de *Moodle* mediante la llamada a procedimientos remotos (funciones externas en *Moodle*). Sin embargo, no todos los módulos que vienen por defecto en la versión 3.0 de *Moodle* traen estas funciones externas generadas.

El código de los servicios web de *Moodle* se puede agrupar en dos grupos, servidor del protocolo de comunicación que nos permite invocar funciones externas y el código de las funciones externas correspondientes a cada módulo que son invocadas a través de ese servidor de protocolo de comunicación.

Servidor del protocolo de comunicación:

El código que nos permite la invocación de funciones externas se encuentra dentro de la carpeta *webservice* que se encuentra en el directorio raíz de *Moodle*, el código dentro de esta carpeta está agrupado en función del protocolo de comunicación que se utiliza. *Moodle 3.0* trae 4 posibles protocolos de comunicación para intercambiar información, AMF, SOAP, REST y XMLRPC. Tanto SOAP como REST son protocolos muy utilizados a la hora de crear servicios web, AMF y XMLRPC son menos populares. Para este proyecto se ha elegido el protocolo REST como protocolo de comunicación por tres motivos principales:

- Android no incluye en su SDK ningún tipo de soporte para el acceso a servicios web tipo SOAP, por lo que sería necesario utilizar una librería externa para facilitar esta tarea. Por el contrario, REST comunica cliente y servidor a través de una interfaz estándar (HTTP), por lo que se puede utilizar la librería *http-request* de android sin necesidad de utilizar librerías externas.

- El desarrollador de este proyecto tiene conocimientos previos sobre el protocolo REST por lo que utilizar este protocolo reducirá el tiempo de desarrollo.
- El protocolo REST es el protocolo más utilizado en la actualidad para crear servicios Web, por lo que utilizar este protocolo aumenta la probabilidad de que un segundo desarrollador que quiera ampliar el proyecto esté familiarizado con este protocolo.

Otra elección que se debe tener en cuenta es la representación de los datos, el protocolo REST permite representar los datos en formato XML (World Wide Web Consortium, s.f.) o JSON (JSON, s.f.), para este proyecto se ha decido utilizar el formato de representación XML por la experiencia previa del desarrollador con este formato. El desarrollador cuenta con conocimientos previos en la conversión de datos de lenguaje XML a objetos Java, por lo que el tiempo de desarrollo se reducirá al utilizar XML en lugar de JSON. Ambas formas de representar los datos son legibles para el ser humano, la mayor ventaja que presenta XML está en ser un lenguaje universal no ligado a la tecnología que se utiliza, la mayor ventaja de JSON está en la forma que estructura los datos, más orientada a un lenguaje orientado a objetos como es java, lo que a priori facilitaría la conversión de los datos a objetos java. Sin embargo, como ya se ha explicado antes, esto no supone una ventaja para este proyecto debido al conocimiento previo de la conversión de XML a objetos Java que tiene el desarrollador, de no ser así se hubiese utilizado JSON, ya que como se ha indicado JSON utiliza una estructura orientada a objetos.

Funciones externas:

La segunda parte en la que se divide el código de los servicios web son las funciones externas que son llamadas a través de los protocolos de comunicación arriba descritos.

Cada módulo contiene sus propias funciones externas, estas funciones están definidas en dos ficheros dentro del directorio del módulo correspondiente:

- **services.php:** Se suele encontrar dentro de la carpeta “db”. Este fichero define el nombre de las funciones externas de un módulo, aparte de otros parámetros como: método que invocará esa función, nombre de la clase que contiene ese método, ubicación del fichero que tiene definida la clase anteriormente nombrada, descripción de las tareas que realiza la función, permisos necesario para llamar a esa función y el tipo de operaciones que realiza la función (lectura o escritura). En la figura 4 se muestra un ejemplo de la función externa “mod_assign_get_grades” definida dentro del fichero services.php del módulo *assign*, esta definición indica que esta función invocará el método “get_grades” de la clase “mod_assign_external” que se encuentra en el fichero “mod/assign/externallib.php”, la descripción indica que la función devolverá las notas de una tarea y que se realizará una operación de tipo lectura.

```
'mod_assign_get_grades' => array(
        'classname'   => 'mod_assign_external',
        'methodname'  => 'get_grades',
        'classpath'   => 'mod/assign/externallib.php',
        'description' => 'Returns grades from the assignment',
        'type'        => 'read'
),
```

Figura 4 - Ejemplo de definición de un método remoto.

- **externallib.php:** Este fichero contiene el código de la clase que contiene los métodos invocados por una función externa. Cada método que se invoca a través de un función externa está compuesto por 3 métodos cuyos nombres deben ser los siguientes en función del *methodname* definido en el fichero services.php (figura 4):
    - *methodname_parameter():* define los parámetros que deben pasarse al llamar a la función externa, en el caso de este proyecto, estos parámetros se enviarán en el cuerpo de la petición *http* o en la propia URL de la petición.
    - *methodname_return():* define los valores que serán devueltos en la llamada a la función.
    - *methodname(parameters):* contiene el código de las tareas que se realizarán al invocar a la función, este método tomará como entrada los parámetros enviados en la llamada a la función, utilizará funciones locales del módulo que se encuentran en los ficheros lib.php y locallib.php para realizar las tareas oportunas y devolverá los valores que se indican en el método *methodname_return().*

En la versión 3.0 de *Moodle* el módulo *quiz* no trae generadas las funciones externas, por lo que serán desarrolladas por el desarrollador. Se desarrollarán las siguientes funciones:

- *mod_quiz_get_quiz_status_for_user*: Devuelve “completado” o “no completado” en función de si un alumno completó todos los intentos posibles para un cuestionario.
- *mod_quiz_start_one_attempt:* Genera un nuevo intento de un usuario para un cuestionario.
- *mod_quiz_process_attempt:* Procesa las respuestas correspondientes a un intento.

Se debe generar el fichero service.php con la definición de cada función y el fichero externallib.php con el código de la clase “mod_quiz_external” que contiene los tres métodos correspondientes a cada función.

Pasos para utilizar servicios web en Moodle:

Una vez elegido el protocolo de comunicación que se desea utilizar para las llamadas a las funciones y están generados todos las funciones externas que se desean utilizar, se deben seguir ciertos pasos para activar y utilizar los servicios web en *Moodle* (Moodle, 2012).

Lo primero que se debe hacer es activar la extensión de servicios web en la plataforma *Moodle* y seleccionar el protocolo de comunicación a utilizar, en este caso REST. En la versión 3.0 de *Moodle* el rol estudiante viene por defecto con el permiso deshabilitado para utilizar el protocolo REST, por lo que será necesario habilitar este permiso al rol estudiante.

Una vez habilitada la extensión y especificado el protocolo, se debe crear un servicio. Un servicio es un conjunto de funciones, se agrupan en un servicio para dar permiso a un usuario a utilizar o no ese conjunto de funciones. En el caso de este proyecto se generará un único servicio denominado *QUIZ_WS* que contendrá todas las funciones necesarias y que estarán disponibles únicamente para usuarios autorizados.

Una vez creado el servicio se debe añadir a este servicio las funciones que se quiere que sean accesibles.

Como último paso solo queda agregar al servicio los usuarios autorizados y generar un *token* para cada usuario autorizado, este *token* lo recuperará un usuario al inicio de una sesión y será utilizado por *Moodle* para comprobar la identidad y los permisos del usuario que solicita un servicio.

Llamada a una función:

Una vez habilitada la configuración anterior, un usuario autorizado para utilizar un servicio ya puede realizar llamadas a todas las funciones que contiene ese servicio, para ello lo que se debe hacer desde el cliente (aplicación móvil) es (Moodle, 2016):

- Obtener el *token* del usuario para un servicio, para ello se envía el nombre de usuario y la contraseña al *script* del servicio web de *login*, indicando de que servicio se quiere obtener el *token*.
  Ejemplo: petición http://dominio/login/token.php con parámetros username=nombre, password=contraseña y service=servicio.

- La petición anterior devolverá el *token* del usuario para ese servicio por lo que ya se puede llamar a una función de ese servicio web a través de un servidor de protocolo de comunicación, incluyendo el *token*, en este caso el servidor a utilizar será el del protocolo REST.
  Ejemplo: petición http://dominio/webservice/rest/server.php con parámetros wstoken=token, wsfunction=nombre_función y los parámetros requeridos por la función.

El servidor del protocolo utilizará el *token* para comprobar si el usuario puede llamar a la función. Por defecto el método de autenticación es mediante *token*, sin embargo, modificando el código del fichero server.php se puede utilizar el nombre de usuario y la contraseña como método de autenticación y los parámetros a pasar serían wsusername y wspassword en lugar de wstoken.

- El servidor del protocolo llamará a la función externa que está localizada en el fichero externallib.php dentro del módulo implicado.
- La función externa comprueba si el usuario que ha llamado a esa función tiene los permisos necesarios para realizar las operaciones que esa función realiza.
- La función externa llama a las funciones del núcleo de *Moodle*, localizadas normalmente en los ficheros lib.php o locallib.php del módulo implicado.
- La función del núcleo devuelve el resultado a la función externa.
- La función externa devuelve el resultado al servidor del protocolo
- El servidor del protocolo devuelve el resultado al cliente.

2.6.6 Nuevas características de Moodle 3.0

Las características más relevantes que añade Moodle 3.0 son las siguientes (Moodle, 2015f):

Para profesores:

- Añade cuatro tipos de preguntas nuevas para el módulo cuestionario: Seleccionar palabra, Arrastrar y soltar al texto, Arrastrar y soltar sobre imagen, Arrastrar y soltar marcadores.
- Reporte de envíos, el profesor puede filtrar quien ha enviado un trabajo y quién no.
- El profesor puede mostrar la fecha en la que fue subido o cambiado un recurso.
- Mejoras en las opciones de filtrado de los participantes de un curso.
- Edición de sección más sencilla y simplificada.

Para administradores:

- Mejoras en la interfaz para instalar y actualizar extensiones.
- Mejoras en el manejo de las etiquetas (tags).
- Mejoras en la seguridad.

Para todos los usuarios:

- Mejoras en el editor de texto Atto.
- Se añade la opción de eliminar mensajes de la lista de mensajes propia de un usuario.

3. Aplicaciones Móviles

El desarrollo de aplicaciones móviles y las tecnologías que se utilizan han ido evolucionando desde que las aplicaciones móviles llegaron al mercado, el avance de las tecnologías ha ido dirigido a agilizar el proceso de desarrollo.

Cada tecnología tiene sus ventajas e inconvenientes ya que las alternativas al desarrollo nativo, al igual que ocurría con los CMS, son creadas para facilitar y agilizar el desarrollo, aunque esto en la mayoría de los casos implica pérdida de rendimiento y de flexibilidad.

En esta sección se analizarán las 3 posibles formas de desarrollar una aplicación móvil que son: crear una *webApp*, desarrollar de forma nativa o utilizar tecnologías hibridas.

webApps:

Una *webApp* es una versión de una página web optimizada al tamaño de pantalla de los dispositivos móviles. Dicho de otra manera, es una página que se puede abrir desde el navegador de cualquier terminal independientemente del sistema operativo que utilice (Qode, 2014).

La principal ventaja que tienen es su capacidad de adaptación a cualquier dispositivo móvil. Es suficiente con que el dispositivo cuente con un navegador móvil actualizado. Eso sí, para poder visualizar la página correctamente en los diferentes navegadores es necesario incluir algún fragmento de códigos especiales para cada uno.

Otra de las ventajas de este tipo de aplicaciones es que no ocupa memoria en los dispositivos y no hacen falta actualizaciones ya que siempre se accederá a la última versión de la página como cualquier página web.

La mayor desventaja que presenta este tipo de desarrollo está en que al tratarse de una web optimizada para móviles que se abre desde el navegador no accede a las capacidades del teléfono, como geolocalización, notificaciones *push*…

Otra de las desventajas está en la perdida de usabilidad ya que el acceso a una aplicación de escritorio es más rápido y la experiencia de usuario es mejor que acceder a una página web desde el navegador.

Uno de los puntos intermedios entre ventajas y desventajas del desarrollo de una *webApp* está en el consumo de recursos del dispositivo. Un navegador web suele consumir bastantes recursos del dispositivo. Sin embargo, una aplicación de escritorio mal desarrollada puede llegar a consumir mayor cantidad de recursos que un navegador, por lo que el consumo de recursos se entenderá como una desventaja de una *webApp* pero teniendo en consideración que el desarrollo de la aplicación de escritorio debe ser optimo en cuanto a consumo de recursos se refiere.

Para este proyecto se ha desechado la opción de crear una *webApp* ya que por las razones que se expusieron en el capítulo 1 de este informe y las desventajas arriba expuestas se considera más conveniente desarrollar una aplicación de escritorio que pueda aprovechar las capacidades de los dispositivos móviles y donde la experiencia de usuario sea mayor.

Aplicaciones híbridas:

Generalmente consisten en Apps que contiene en su interior el navegador web del dispositivo. Para su desarrollo se utilizan *frameworks* de desarrollo basados en lenguajes de programación web (HTML, CSS y JS). Actualmente *Phonegap* es el más conocido (aunque no el único) y el que concentra mayor número de desarrolladores a su alrededor.

En este tipo de Apps el nivel de integración con el SO dependerá del *framework* de desarrollo utilizado y como de abierto sea el SO. La mayor ventaja frente al desarrollo de una *webApp* está en la posibilidad de acceder al hardware del teléfono e incluso en algunos casos a las librerías del SO, aunque de momento no se ha conseguido igualar la respuesta y la experiencia de usuario de una App nativa (Appio, 2013).

El desarrollo híbrido consiste en ejecutar ciertas funcionalidades como una web y otras en nativo, tal y como hace la aplicación oficial de *Moodle*, *Moodle Mobile* (2012) mencionada en el capítulo 1 de este informe.

Las limitaciones de esta tecnología y la opción que ya presenta *Moodle Mobile* de tener una aplicación móvil híbrida, hace que se desestime esta opción de desarrollo para este proyecto.

Aplicaciones Nativas:

Una App nativa es una aplicación que se desarrolla directamente en el lenguaje nativo de cada terminal. Por eso, si se va a desarrollar una App nativa se debe utilizar un lenguaje diferente para cada Sistema Operativo. Los lenguajes de programación en función del SO son los siguientes:

iOS: Objective C o Swif**t**

Android: Java

Windows: C# y Visual Basic .NET.

BlackBerry 10: C++

Una App nativa es la opción cuyo resultado es el más robusto y fluido ya que se desarrolla directamente para integrarse en el Sistema Operativo, siempre teniendo en cuenta que el desarrollo sea correcto.

Las mayores ventajas del desarrollo nativo son:

- Acceso a todo el hardware del móvil como puede ser el GPS, la cámara y demás accesorios.
- Acceso a todas las librerías gráficas del SO.
- Envío de notificaciones push, una de las herramientas de comunicación más potentes de las Apps, en este proyecto no se desarrolla pero estará presente en líneas futuras.
- Sincronizar o cachear datos para funcionar sin conexión a internet.
- Mejora de la experiencia de usuario.

La principal desventaja radica en la necesidad de desarrollar la aplicación en diferentes lenguajes para cada sistema operativo, esto tendrá principalmente un impacto económico en el desarrollo de la aplicación. No se podrá reutilizar código de un SO a otro y las actualizaciones y mantenimiento deberán hacerse para cada SO y directamente en el código de la plataforma.

El desarrollo nativo es el que más se adapta a las necesidades de este proyecto que busca una experiencia de usuario optima, sin embargo, realizar el proyecto para todos los sistemas operativos del mercado supondría un coste demasiado elevado.

Según el estudio realizado en febrero de 2016 por la consultoría de investigación Kantar world panel (www.kantarworldpanel.com) y según muestra la figura 5, obtenida de este estudio, el SO android domina claramente en ventas en España al resto de sistemas operativos. Esta es razón suficiente para decantar el desarrollo de la aplicación a este sistema operativo, ya que se pretende llegar al máximo número de usuarios/alumnos posibles y desarrollando para android se llegará casi al 90% de los alumnos. Sin

embargo, se han tenido también en consideración las razones económicas para decantarse por Android en contra del segundo sistema operativo más popular, Apple. Para desarrollar para Android únicamente es necesario un PC con cualquier sistema operativo y que cuente con una herramienta de desarrollo como Eclipse, NetBeans o Android Studio que son gratuitas. Sin embargo, el desarrollo para el SO Apple debe hacerse desde un ordenador Mac de la Marca Apple y pagar una licencia anual con un precio de 100$ (https://developer.apple.com).

Spain	3 m/e Feb 15	3 m/e Feb 16	% pt. Change
Android	88	90	2.0
iOS	8.7	9.1	0.4
Windows	2.9	0.9	-2.0
Other	0.4	0.0	-0.4

Figura 5 - Cuota de mercado de SOs móviles en España, Feb 2016.

3.1. Sistema operativo Android

Como se ha indicado en el apartado anterior, Android es el sistema operativo número uno en cuanto popularidad, con una cuota de mercado cercana al 90% en España. El sistema operativo de *Google* se caracteriza por ser abierto y disponible para cualquier fabricante interesado en utilizarlo para sus dispositivos móviles. Esta característica hace que su cuota de mercado sea muy amplia, ya que no necesita una marca de dispositivo concreta para poder ser implementado (Campos, 2015).

Arquitectura Android:

El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas *XML*, 2,8 millones de líneas de lenguaje *C*, 2,1 millones de líneas *Java* y 1,75 millones de líneas de *C++*. Se puede observar la arquitectura de este sistema operativo de manera gráfica en la Figura 6.

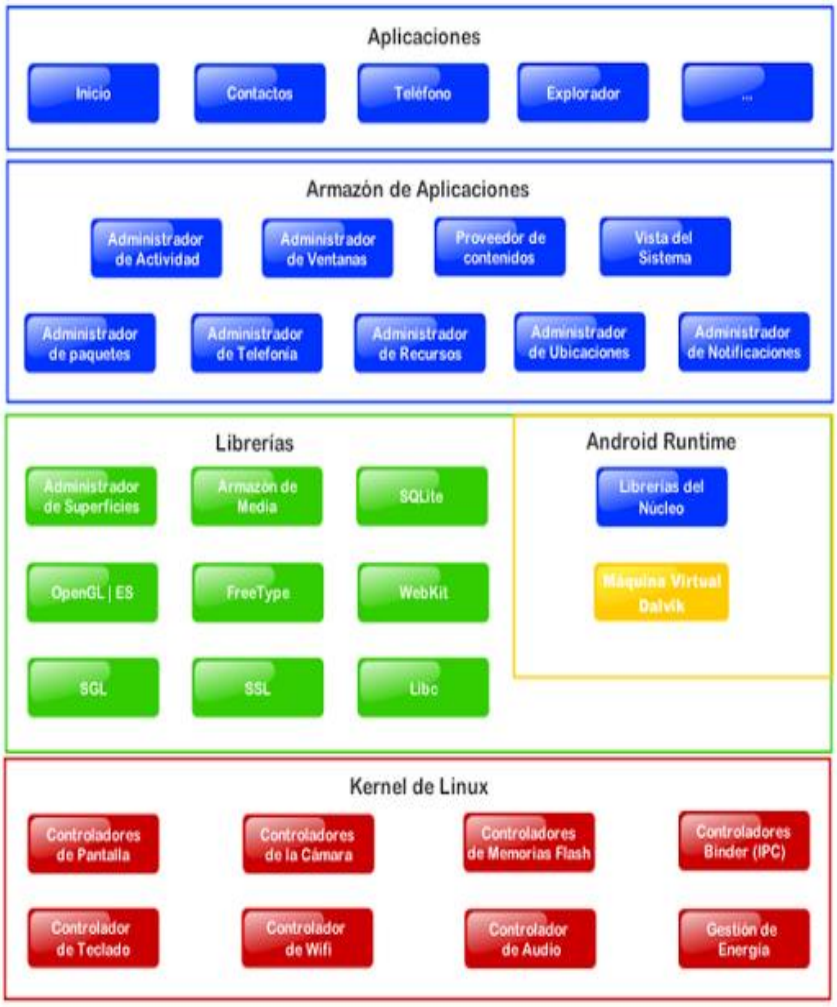

Figura 6 - Arquitectura Android.

- 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.*

Desarrollo de aplicaciones Android:

Una aplicación Android consiste fundamentalmente y de forma simplificada en un conjunto de actividades que se crean y se destruyen en función de la interacción del usuario con la aplicación.

Actividades

Una aplicación que tiene una UI visible se implementa con una actividad. Cuando un usuario selecciona una aplicación desde la pantalla de inicio o el iniciador de aplicación, se inicia una actividad (Ableson, 2013).

La UI de la actividad, que estará formada por elementos de las librerías gráficas del SO operativo, botones, imágenes, etc, se puede generar de forma estática a través de un fichero XML o de forma dinámica durante la construcción de la actividad, la UI de una actividad puede cambiar debido a la interacción del usuario.

El ciclo de vida de una actividad está representado en la figura 7, representa los estados que puede atravesar una actividad desde que es creada hasta que es destruida.

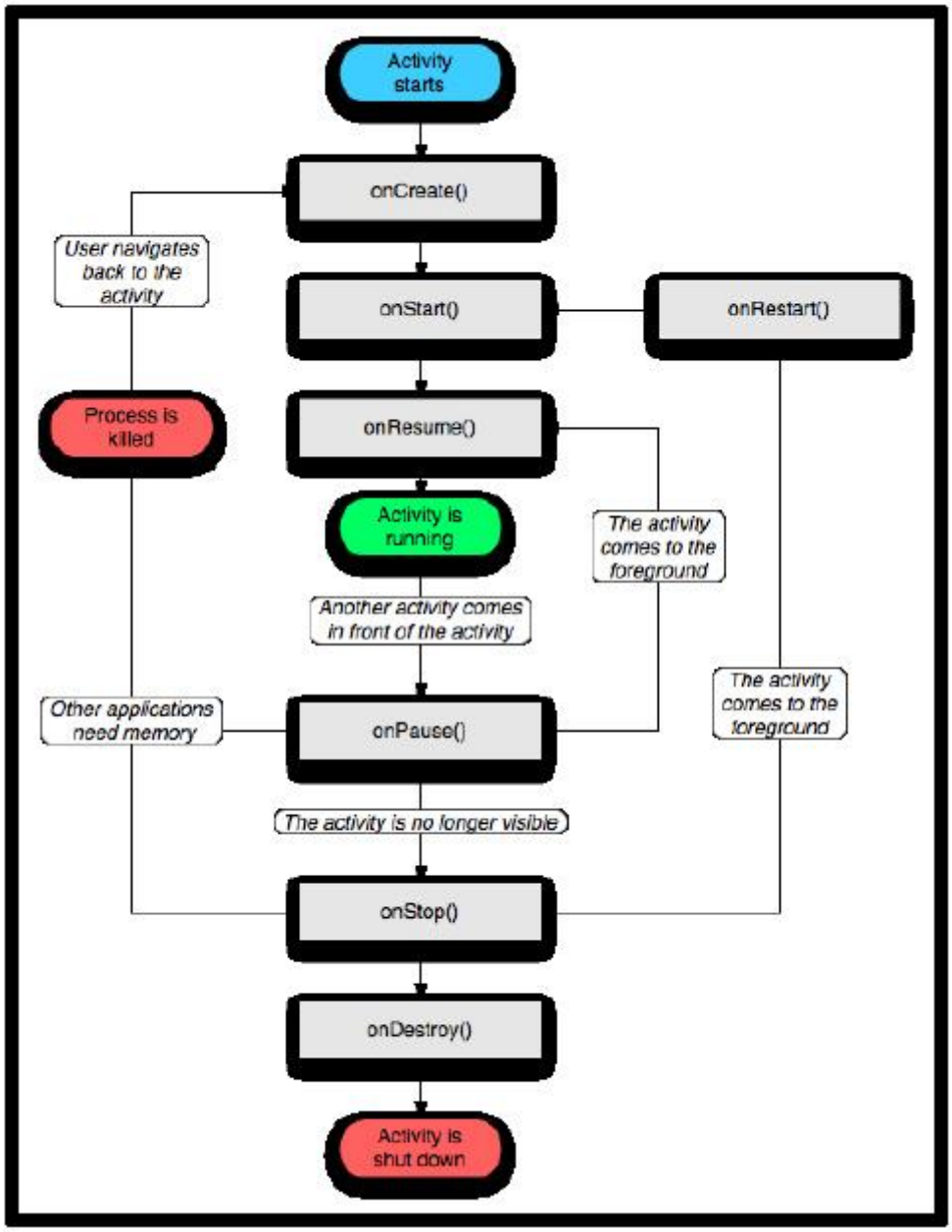

Figura 7 - Ciclo de vida de una actividad.

La interacción del usuario con la aplicación se realiza mediante *Listeners* asociados a los elementos de la UI, la aplicación responderá a las interacciones del usuario en función del código desarrollado en la actividad asociada a esa UI.

Otros contenidos importantes, aunque en este proyecto no se utilizarán, por los que puede estar formada una aplicación son los siguientes:

Servicios

Se debe usar un servicio para cualquier aplicación que necesite persistir por mucho tiempo, como por ejemplo un supervisor de red o una aplicación de comprobación de actualización.

Proveedores de contenido

Es posible considerar los proveedores de contenido cuando una aplicación móvil requiere ofrecer sus datos a otras aplicaciones.

Receptores de difusión

Se puede lanzar una aplicación Android para procesar un elemento de datos o para responder a un evento, como por ejemplo la recepción de un mensaje de texto.

Una aplicación Android, junto con un archivo llamado AndroidManifest.xml, se despliega para un dispositivo. AndroidManifest.xml contiene la información de configuración necesaria para instalarlo adecuadamente en el dispositivo. Incluye los nombres de clases requeridos y los tipos de eventos que la aplicación puede procesar y los permisos requeridos que la aplicación necesita para ejecutarse. A modo de ilustración, si una aplicación requiere de acceso a la red — para descargar un archivo, por ejemplo — este permiso debe estar mencionado explícitamente en el archivo manifiesto.

CAPITULO 3. DESARROLLO DEL PROYECTO

1. Introducción

En el capítulo anterior se han analizado las diferentes posibles tecnologías a utilizar. A modo de resumir las elecciones realizadas se concreta que, la parte del servidor estará generada con *Moodle*, un CMS dirigido al *e-learning* e implementado con *PHP-MySQL*. La parte cliente (aplicación móvil), estará desarrollada de forma nativa para el sistema operativo *Android*. Por último, para la parte servicios web, la elección ha sido la arquitectura REST, con representación XML.

A continuación, se exponen y explican las tareas que se han realizado durante el desarrollo de la aplicación móvil.

2. Plataforma Moodle de desarrollo

El funcionamiento de la aplicación móvil se desarrollará y depurará contra una plataforma *Moodle* de desarrollo similar a la que utiliza en producción la universidad de Valladolid (http://campusvirtual2015.uva.es/). Esta plataforma se utilizará para desarrollar las funciones externas necesarias y depurar el funcionamiento de la aplicación antes de desplegarla en producción.

La plataforma de desarrollo tendrá una configuración inicial como se describe a continuación:

- Tres cursos denominados Curso1, Curso2, Curso3.
- Tres usuarios de los cuales, uno tiene el rol profesor (profesor1) y los otros dos tienen el rol alumno (alumno1 y alumno2).
- Alumno1 está matriculado en los Cursos, Curso1 y Curso3.
- Alumno2 está matriculado en los Cursos, Curso2 y Curso3
- Profesor1 es profesor de los tres cursos.
- La plataforma debe tener habilitada la utilización de servicios web con arquitectura REST del modo que se explicará en el siguiente capítulo en la “guía del administrador”.
- La plataforma debe tener creado un servicio web denominado QUIZ_WS que contendrá todas las funciones externas necesarias para el funcionamiento de la aplicación móvil y al que estarán autorizados a utilizar alumno1 y alumno2. El profesor creará los cuestionarios desde la plataforma web, no accederá a la *app* por lo que no necesitará estar autorizado para acceder utilizar este servicio.

3. Diagramas de flujo.

El objetivo que se persigue a partir de la configuración inicial descrita en el punto anterior es desarrollar una aplicación móvil en la cual el alumno1 pueda ver y responder los cuestionarios que profesor1 cree en los cursos Curso1 y Curso3 y el alumno2 pueda hacer lo mismo con los cuestionarios creados en los cursos Curso2 y Curso3. Para ello la aplicación móvil debe seguir los diagramas de flujo que se describen a continuación.

Autenticación del usuario:

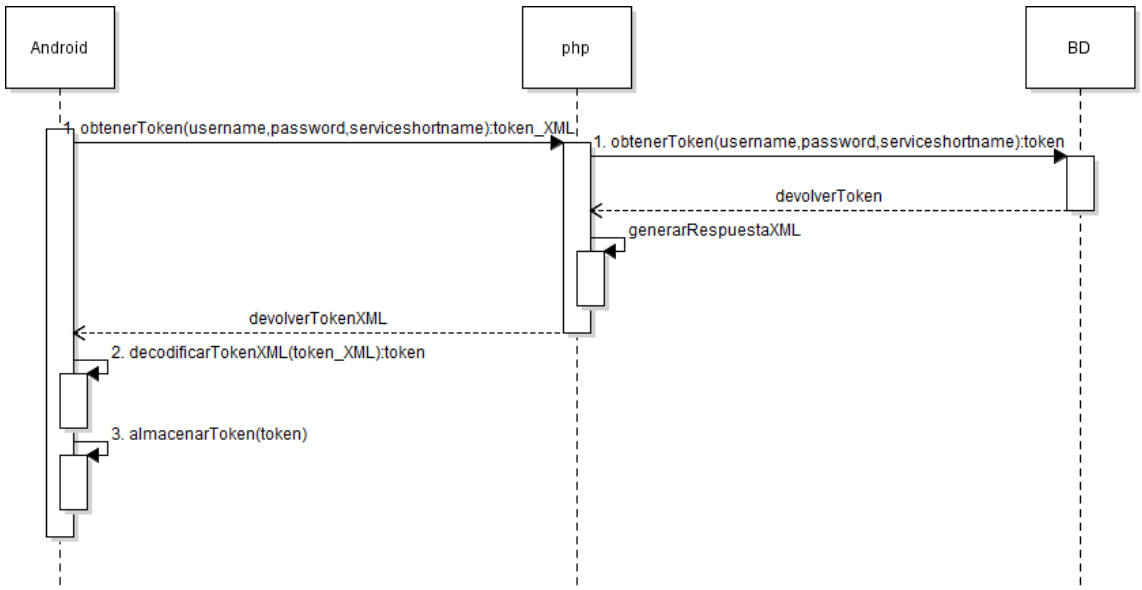

Figura 8 - Diagrama "autenticación".

Lo primero que hará la aplicación móvil al ser abierta por un usuario será pedir las credenciales de acceso a la plataforma, usuario y contraseña. La aplicación utilizará estos credenciales para autenticar al usuario contra el servidor y obtener el token asociado a ese usuario y al servicio web QUIZ_WS. El servicio web QUIZ_WS, creado previamente en la plataforma como se ha explicado anteriormente, contendrá todas las funciones externas necesarias para el funcionamiento de la aplicación (figura 8).

Una vez obtenido el token, este se almacenará el tiempo que dure la sesión, ya que se necesitará constantemente durante el funcionamiento de la aplicación siempre que se requiera acceder a un servicio web.

Para obtener el token del usuario se utiliza el Script */login/token.php*, este Script consulta varias tablas de la BD como se muestra en la figura 9. Se consulta la tabla *mdl_user* de la BD para autenticar al usuario y obtener el identificador de ese usuario,

esta tabla contiene los campos *username*, *password* (cifrado[2] por seguridad) e *id (userid)* que son los consultados para autenticar al usuario y obtener su identificador. Una vez autenticado el usuario y obtenido el *userid*, se consulta la taba *mdl_external_services*, para obtener el identificador del servicio a partir del nombre del servicio. Para finalizar, una vez que está autenticado el usuario y se tienen los identificadores *userid* y *externalserviceid* se consulta la tabla *mdl_external_token* para obtener el token, si ningún token tiene asociados el *userid* y el *externalserviceid* obtenidos en los pasos anteriores, significa que ese usuario no está autorizado a utilizar ese servicio y se devolvería una excepción.

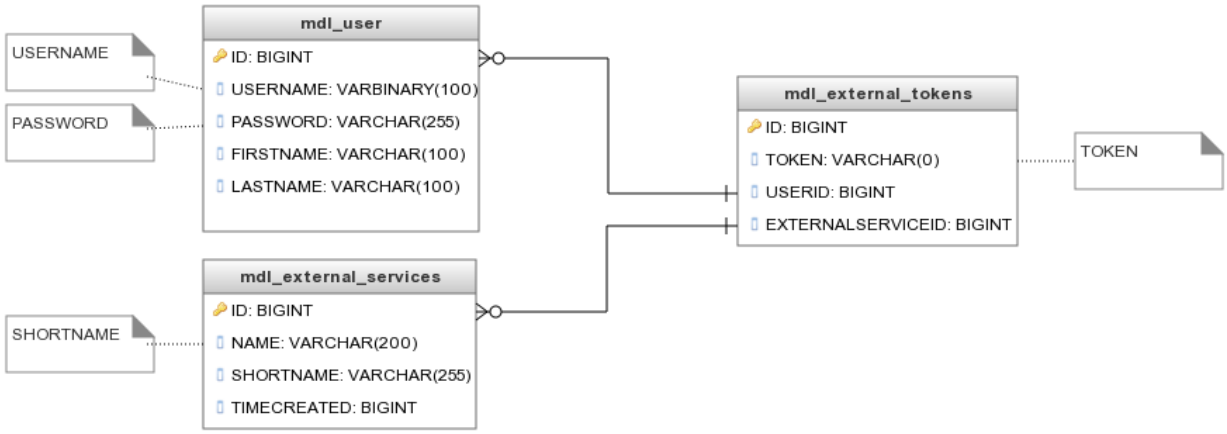

Figura 9 -Tablas relacionadas en la obtención del Token.

Obtención del identificador de usuario:

Después de obtener el token que autentica al usuario, se solicitará al servidor el identificador de ese usuario (*userid*), identificador que se almacenará y se utilizará para llamadas futuras a funciones (figura 10).

Para obtener el identificador de usuario se llama a la función *core_user_get_users_by_field*, esta función viene definida por defecto en la versión *Moodle 3.0* y se encuentra en el fichero */mod/user/externallib.php*. A esta función se le pasan dos parámetros, el primer parámetro es *field* que será el campo por el que se filtrará a los usuarios, en este caso el *field* será *username*. El segundo parámetro será *value*, el valor por el que se filtrará, en este caso el valor será el nombre del usuario del que se quiere obtener el identificador.

[2] La contraseña no se almacena en texto plano en la Base de Datos por seguridad sino que esta se almacena cifrada con una clave, cuando el usuario introduce su contraseña, esta se cifra con la misma clave y se compara con el valor almacenado.

La tabla de la BD consultada por la función *core_user_get_users_by_field* es *mdl_user* y sus campos más importantes están representados en la figura 11.

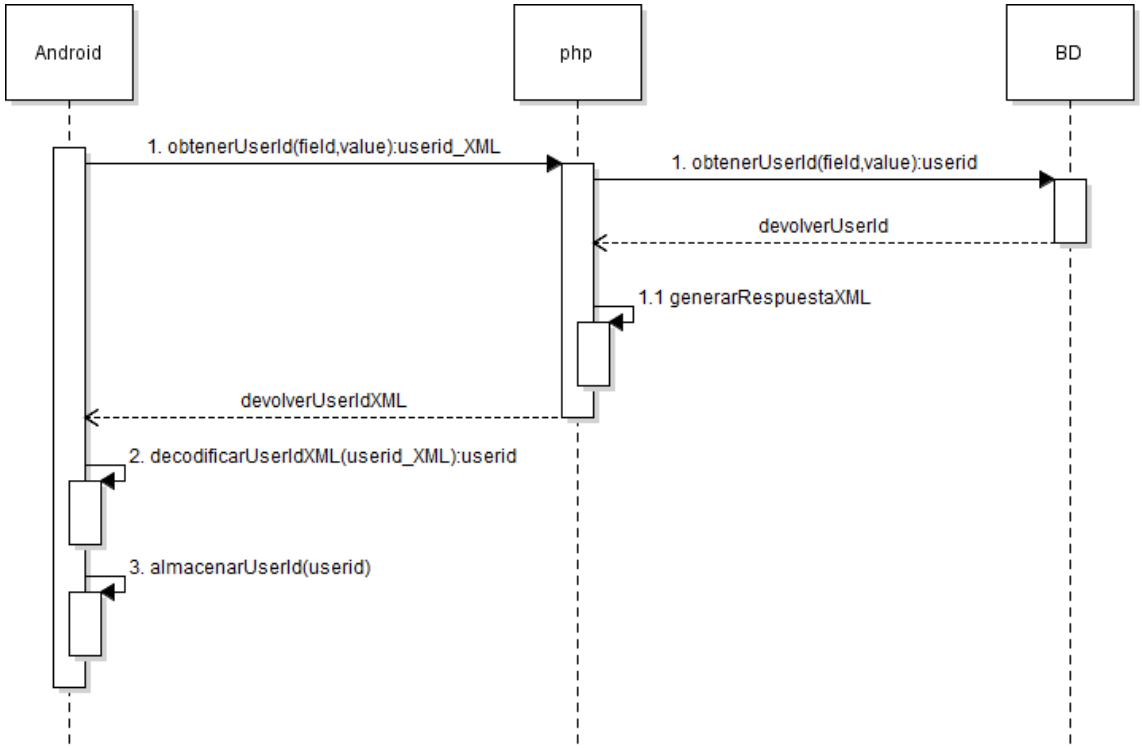

Figura 10 - Diagrama "obtención del identificador de usuario".

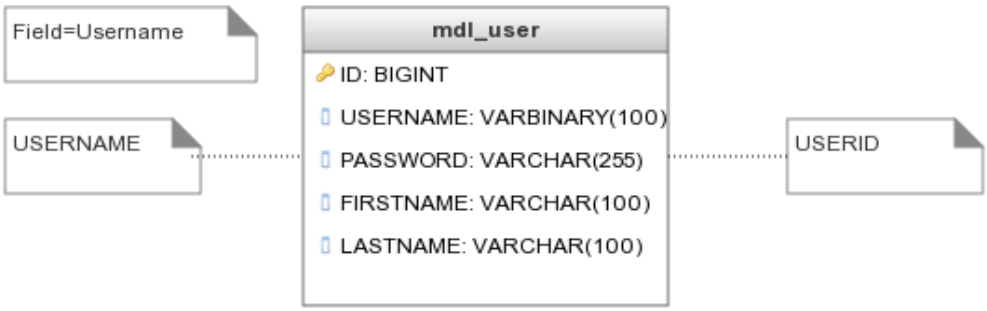

Figura 11 - Tabla mdl_user.

Siempre que se llame a una función se hará mediante el *Script /webservice/rest/server.php* al que se le pasará como parámetros el nombre de la función que se desea llamar, el token de autenticación del usuario y los parámetros que requiera la función llamada. El *Script* antes de llamar a la función externa realiza una tarea para comprobar si el usuario tiene autorización a llamar a esa función. Lo primero que hace es consultar la tabla *mdl_external_tokens* para obtener el identificador del servicio que se solicita (a partir del token), con ese identificador se consulta la tabla

mdl_external_services_functions para comprobar que la función solicitada está permitida para ese servicio (figura 12). Si todo está correcto el *Script* llama a la función externa, si la función solicitada no está disponible para el token enviado se devolverá una excepción y no se llamará a la función externa.

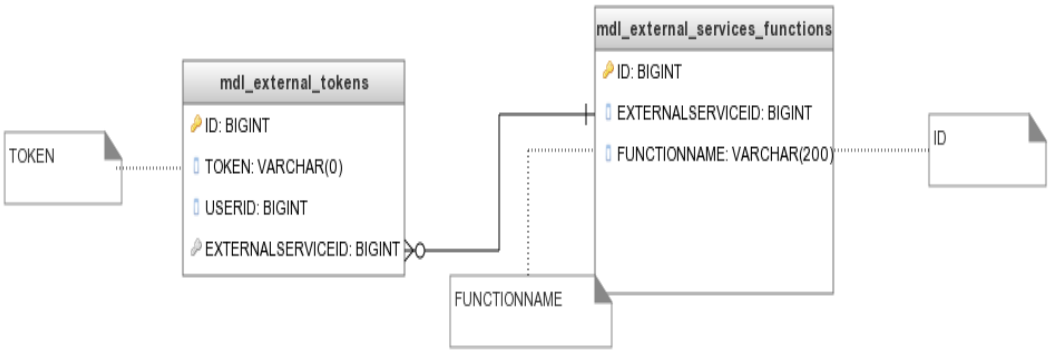

Figura 12 - Tablas relacionadas para comprobar la autorización a llamar una función.

Obtención de cursos en los que está matriculado un usuario:

Después de obtener el identificador del usuario (*userid*), se solicitará al servidor una lista con los cursos en los que está matriculado el usuario (figura 13).

Para obtener la lista de cursos en los que está matriculado el usuario se llama a la función *core_enrol_get_users_courses*, esta función viene definida por defecto en la versión *moodle 3.0* y se encuentra en el fichero */enrol/externallib.php*. A esta función se le pasa como parámetro el *userid* del usuario del que se quiere obtener la lista de cursos en los que está matriculado.

Las tablas de la BD que indican en que cursos está matriculado un usuario son *mdl_user_enrolments* y *mdl_enrol* (figura 14), de la primera se obtiene los *enrolid* asignados a un usuario y consultando esos *enrolid* en la segunda tabla se obtienen los *courseid* de los cursos en los que está matriculado un usuario. Consultando la tabla *mdl_course* (figura 14) a partir del *courseid* se obtienen detalles de un curso cursos.

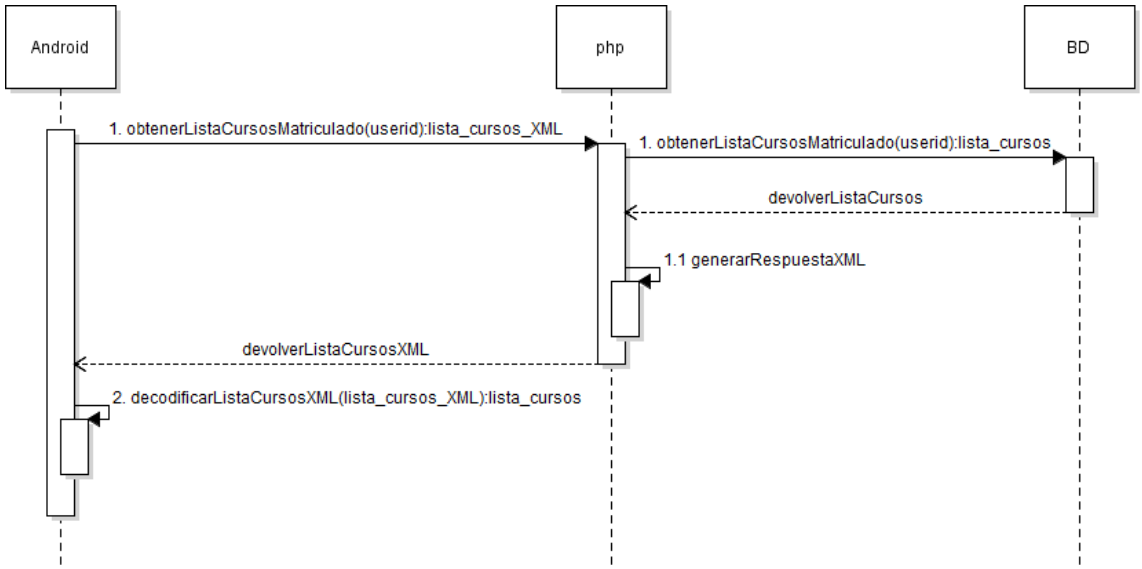

Figura 13 - Diagrama "obtención de la lista de cursos matriculados".

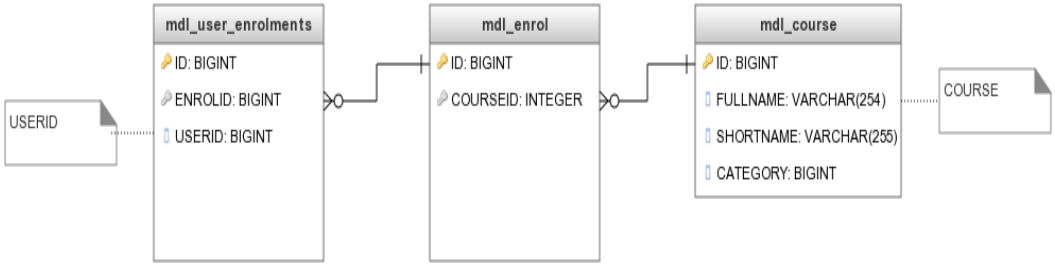

Figura 14 - Tablas relacionadas para obtener la lista de cursos en los que está matriculado un estudiante.

Obtención de la lista de los cuestionarios de un curso:

A este paso se llegará cuando el usuario seleccione el curso del que desea obtener la lista de cuestionarios, se solicitará al servidor una lista de todas las actividades del curso y la aplicación móvil seleccionará y mostrará una lista de las actividades que sean tipo cuestionario. Esto se ha hecho así ya que al desarrollador le resultaba más sencillo filtrar el resultado en la aplicación que modificar la función del servidor que devuelve las actividades, aunque lo óptimo hubiese sido la segunda opción (figura 15).

Para obtener la lista de las actividades de un curso se llama a la función *core_course_get_contents*, esta función viene definida por defecto en la versión *moodle*

3.0 y se encuentra en el fichero */course/externallib.php.* A esta función se le pasa como parámetro el *courseid* del curso del que se quiere obtener la lista de actividades.

Las tablas de la BD que se consultan para saber qué actividades tiene un curso son todas las que tienen la siguiente sintaxis *mdl_nombreModuloActividad.* Un ejemplo es el que se muestra en la figura 16 con la tabla *mdl_quiz* donde a partir del *courseid* se obtienen todos los IDs de las actividades de tipo cuestionario que pertenecen a un curso.

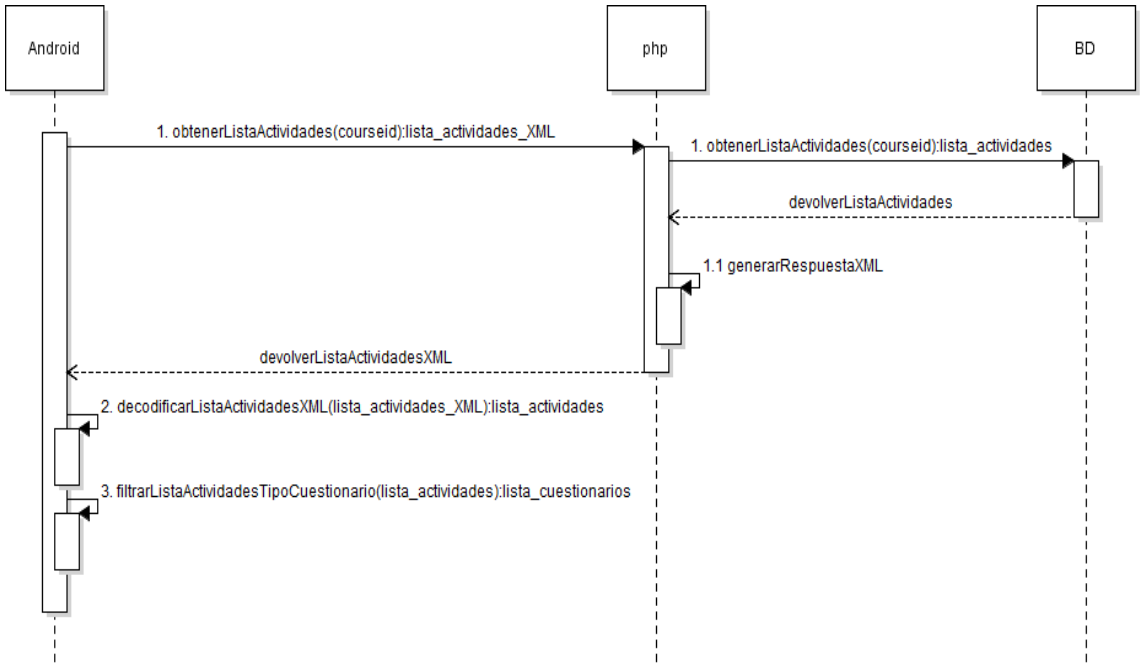

Figura 15 - Diagrama "obtención de la lista de cuestionarios".

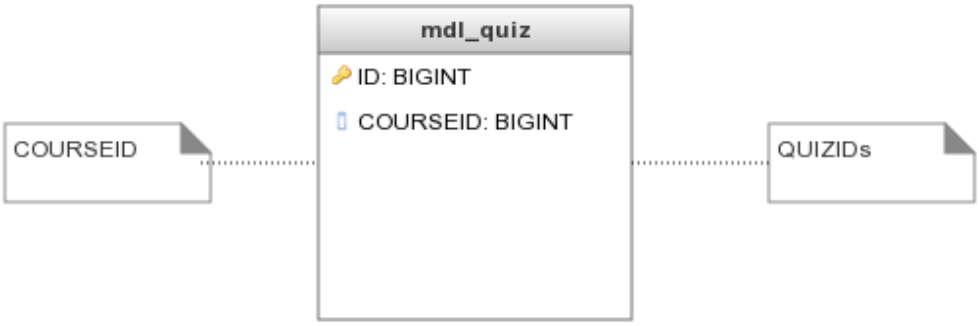

Figura 16 - Tabla mdl_quiz.

Obtención del estado de un cuestionario:

Una vez se selecciona el cuestionario que se desea responder, la aplicación solicitará al servidor el estado del cuestionario para ese usuario. El servidor responderá "available" si al usuario le quedan intentos por realizar, en cuyo caso se debe iniciar un intento, o

responderá “finished” si el usuario ha realizado todos los intentos posibles, en cuyo caso se informará al usuario de este hecho (figura 17).

Para obtener el estado de un cuestionario se llama a la función *mod_quiz_get_quiz_status_for_user* desarrollada por el desarrollador del proyecto, como se ha indicado anteriormente en este informe, el módulo Cuestionario (*quiz*) de *Moodle* para la versión *Moodle 3.0* no trae creadas las funciones externas para el módulo cuestionario. Los parámetros de entrada para esta función son *userid* y *quizid* y devuelve el valor “available” o“finished” en función de si al usuario le quedan o no intentos para un cuestionario.

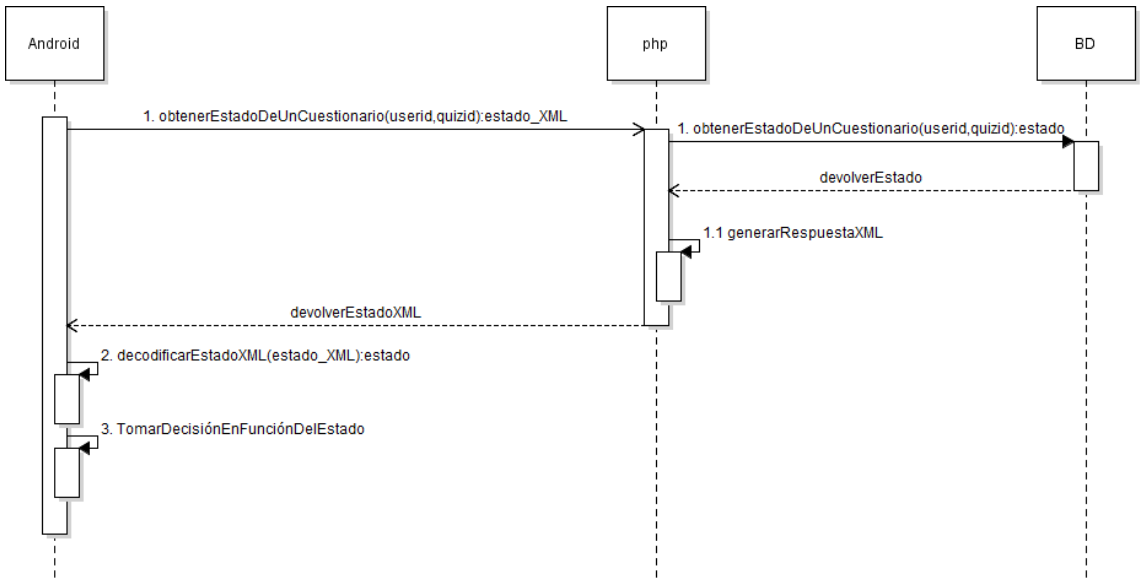

Figura 17 - Diagrama "obtención del estado de un cuestionario".

Para realizar las funciones externas del módulo cuestionario no es estrictamente necesario conocer la estructura de la base de datos ya que se han utilizado funciones locales del módulo cuestionarios que se encuentran en los ficheros lib.php y locallib.php en lugar de consultas directas a la base de datos. Estas funciones abstraen al desarrollador de las consultas que se realizan sobre la base de datos, sin embargo, a modo formativo se presenta una estructura de la relación de las tablas que utiliza el módulo cuestionario (figura 18) (Campos, 2015).

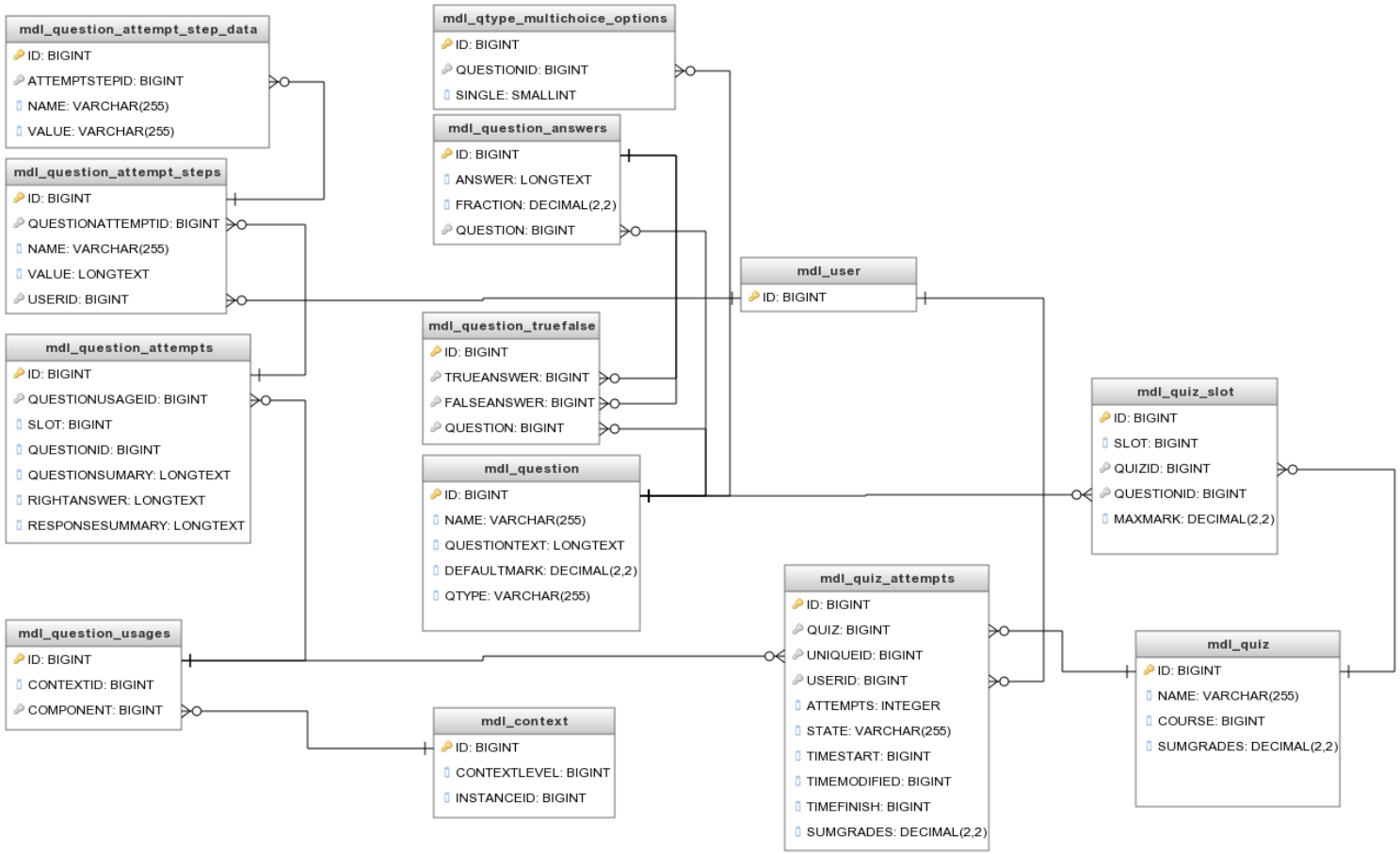

Figura 18 - Relación de la base de datos de la sección cuestionario.

Iniciar un intento de un cuestionario:

Una vez comprobado que al usuario le quedan intentos para realizar el cuestionario la aplicación solicitará al servidor que se inicie un intento. El servidor responderá con la información relativa a ese intento, enunciado de las preguntas, respuestas posibles, orden de las respuestas, identificador del intento, etc. Si no es posible iniciar el intento debido a que este ya está en curso o la fecha límite para realizar el cuestionario expiró el servidor devolverá una excepción y se informará al usuario de la aplicación (figura 19).

Para iniciar un intento se llama a la función *mod_quiz_start_one_attempt* el código de esta función se ha obtenido del proyecto desarrollado por la comunidad (Github Moodle, 2016), este código ha sido modificado por el desarrollador para que obtenga y devuelva las preguntas correspondientes al intento iniciado y así poder presentarlas en la aplicación con una única llamada a un servicio web. La función original desarrollada por la comunidad inicia el intento y devuelve valores referentes a ese intento como identificador, curso, etc. Pero no devuelve las preguntas de ese intento que deben obtenerse con una llama a otro servicio web con el consumo de tiempo y recursos que ello conlleva.

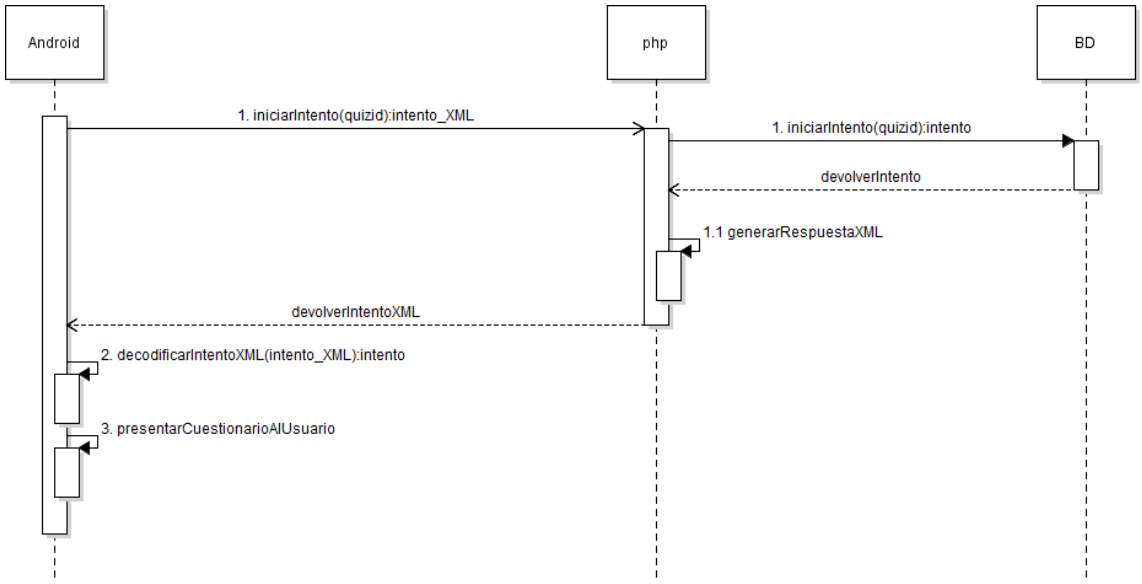

Figura 19 - Diagrama "iniciar un intento".

Al igual que ocurría con la función *mod_quiz_get_quiz_status_for_user*, para el desarrollo de la función *mod_quiz_start_one_attempt* no se realiza ninguna consulta directa a la base de datos con una petición SQL, sino que se realizan a través de funciones locales del módulo cuestionario contenidas en los ficheros lib.php y locallib.php que vienen por defecto en la versión *Moodle 3.0*.

Responder y terminar cuestionario:

Cuando el usuario haya respondido las preguntas del cuestionario presentado en el paso anterior y confirme el envío de las respuestas, la aplicación debe enviar estas al servidor y finalizar el intento. El servidor responderá a cada envío de respuesta confirmando que se ha recibido la respuesta y que el intento sigue activo, cuando la aplicación envíe el mensaje de terminar el intento, el servidor responderá con la nota obtenida en ese intento y una confirmación de que el intento se ha terminado correctamente (figura 20).

Para enviar las respuestas de un intento se utilizará la función *mod_quiz_process_attempt* el código de esta función se ha obtenido del proyecto desarrollado por la comunidad (Github Moodle, 2016), este código ha sido modificado por el desarrollador para que tras finalizar el intento obtenga y devuelva la nota obtenida por el usuario en ese intento.

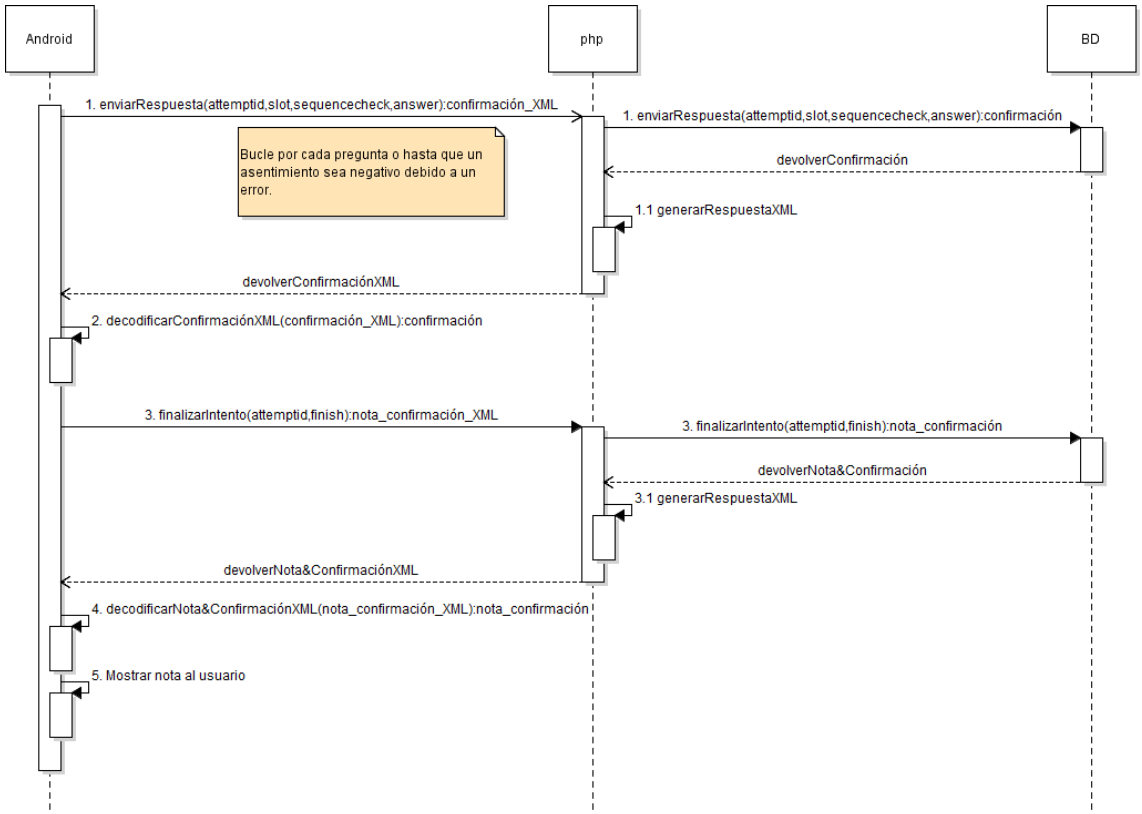

Figura 20 - Diagrama "responder y terminar cuestionario".

Al igual que ocurría con las funciones *mod_quiz_get_quiz_status_for_user* y *mod_quiz_start_one_attempt*, para el desarrollo de la función *mod_quiz_process_attempt* no se realiza ninguna consulta directa a la base de datos con una petición SQL, sino que se realizan a través de funciones locales del módulo cuestionario contenidas en los ficheros lib.php y locallib.php que vienen por defecto en la versión *Moodle 3.0.*

4. Estructura del código de la aplicación

A continuación se detalla la estructura que se ha seguido a la hora de desarrollar el proyecto, cómo se han estructurado las clases Java y cómo se han estructurado los ficheros XML. Con esta explicación se pretende reducir el trabajo de comprensión del código para un posible segundo desarrollador que continúe con el proyecto

4.1 Ficheros XML de layout

Los ficheros XML que definen los *layouts* se diferencian en tres tipos:

Ficheros que definen la interfaz de una actividad, los que comienzan por *activity* (figura 21).

Ficheros que definen la interfaz personaliza de los cuadros de diálogo que se muestran en la aplicación, ya que no se ha utilizado la interfaz que viene por defecto para los diálogos en Android. Estos ficheros son los que comienzan por *dialog* (figura 21).

Ficheros que definen la interfaz de los *widgets* creados, se ha creado un *widget* para cada tipo de pregunta de cuestionario que soporta la aplicación. Si se quiere que la aplicación soporte más tipos de preguntas el primer paso será crear el fichero XML que defina la interfaz de ese tipo de pregunta. Estos ficheros son los que comienzan por *ítem* (figura 21).

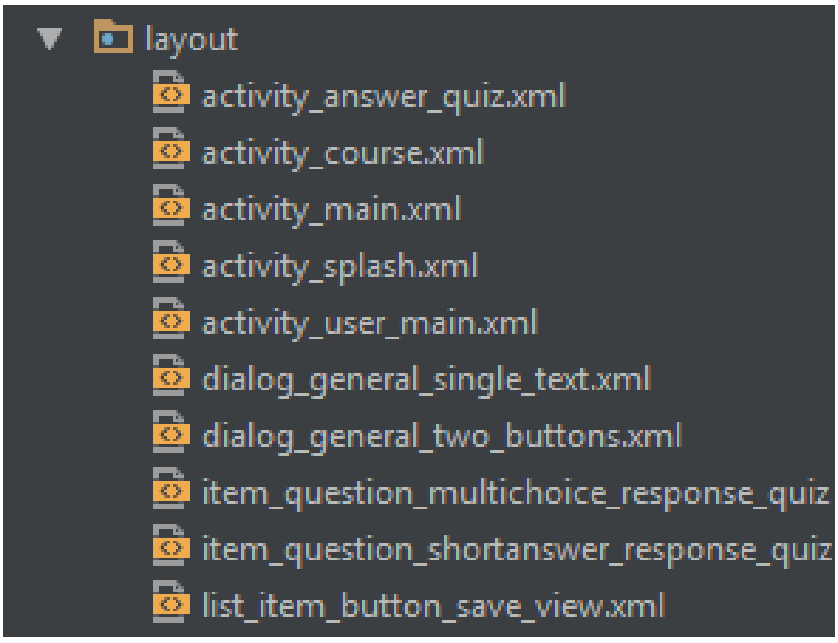

Figura 21 - Ficheros XML de layout.

4.2 Ficheros XML de cadenas

La internacionalización de una aplicación en Android se resuelve implementando varios archivos *strings.xml*, uno por cada idioma que se necesite implementar (Developer Android, 2016a). Este proyecto únicamente se pretende que esté soportado para el idioma español, sin embargo, es una buena práctica utilizar ficheros *string.xml* para definir las cadenas ya que simplificará la tarea de internacionalización en un futuro.

Los ficheros de *strings* por defecto, se almacenan en la carpeta *values*. Para conseguir que la aplicación soporte varios idiomas hay que crear otra carpeta *values* con el identificador de idioma correspondiente. Ejemplo: para que la aplicación soporte el idioma inglés se debe crear la carpeta *values-en* y dentro de esa carpeta crear los ficheros *string.xml* correspondientes.

Para decidir qué valor (idioma) de cada cadena presenta la aplicación, esta detectará el idioma en el que está configurado el terminal y se irá a buscar el valor de los *strings* necesarios a la carpeta correspondiente de ese idioma *(values-identificador)*. Si esa carpeta no existiese cogerá el valor de la carpeta por defecto (*values)*.

La aplicación de este proyecto únicamente está soportada en el idioma español, el valor de las cadenas de la carpeta por defecto (*values*) está en español. Los ficheros *string.xml* se han organizado por actividades de modo que sea más sencillo encontrar una cadena si se necesita reemplazar su valor, también existe un fichero *strings.xml* que contiene las cadenas generales que involucran a varias actividades (figura 22).

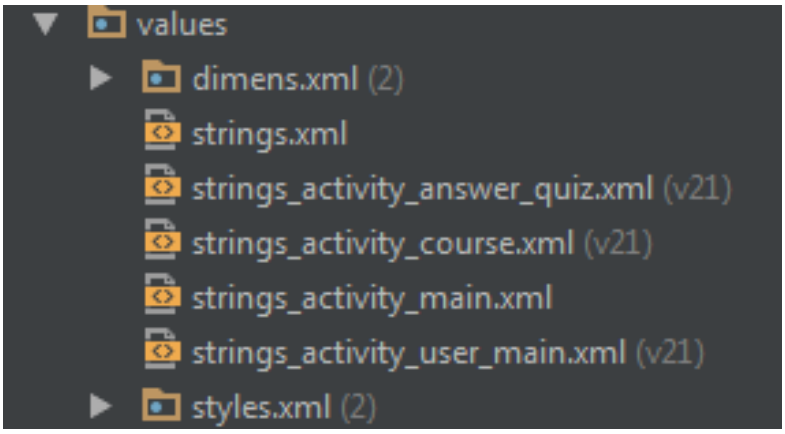

Figura 22 - Ficheros XML de cadenas.

Un ejemplo de un fichero *string.xml* se representa en la figura 23, los elementos de las cadenas deben definirse dentro de un elemento “resources” y cada elemento debe contener un atributo “name” que será el mismo en los ficheros *string.xml* de los diferentes idiomas.

```
<resources>
    <string name="app_name">ETSO</string>
    <string name="generic_loading_text">Loading...</string>
    <string name="dialog_general_title">ETSO Message</string>
    <string name="continu">Continuar</string>
    <string name="cancel">Cancelar</string>
    <string name="loading">Cargando...</string>

    //Network error
    <string name="no_internet_title">Error de conexión</string>
    <string name="no_internet_text">Por favor, asegurese de que su dispositivo está conectado a internet antes de iniciar la aplicación,
    <string name="close_session_button">Cerrar la aplicación</string>
    <string name="title_activity_splash">SplashActivity</string>
</resources>
```

Figura 23 - Ejemplo fichero string.xml.

4.3 Ficheros Java

Los ficheros java desarrollados en la aplicación se organizan como muestra la figura 24. A continuación se expone una breve explicación del contenido de cada carpeta.

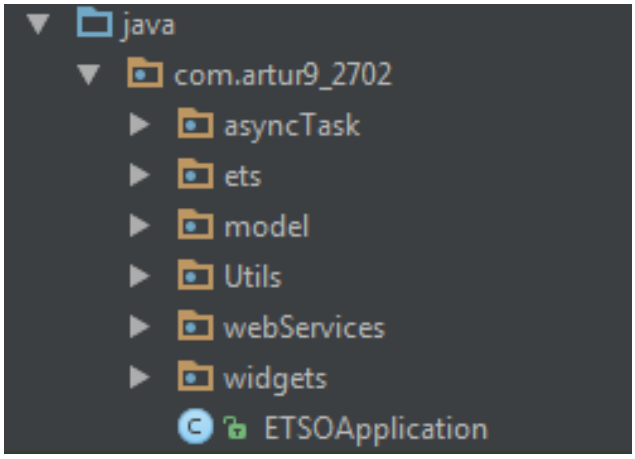

Figura 24 - Ficheros Java.

Carpeta Utils:

La carpeta “Utils” contiene el código de dos clases que desarrollan utilidades empleadas en diferentes partes del código de la aplicación (figura 25).

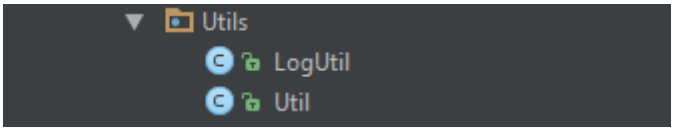

Figura 25 - Carpeta Utils.

La clase *LogUtil* implementa métodos de log de modo que los mensajes únicamente se escriban si la aplicación se encuentra en modo depuración. Se muestra un ejemplo en la figura 26 con el método “d”.

```
/*
      Logcat silent on release
     */
public static void d(String msg) { d(TAG_GENERAL, msg); }

public static void d(String tag, String msg) {
    if (!DEBUG_ENABLE) {
        return;
    } else {
        android.util.Log.d(tag, msg);
    }
}
```

Figura 26 - Método "d" de la clase LogUtil.

La clase *Util* contiene métodos de utilidades genéricas que se emplean en diferentes clases java de la aplicación.

Carpeta model:

La carpeta “model” contiene el código de los modelos de objetos que se utilizan a lo largo de la aplicación (figura 27). Cada modelo implementa los métodos *get* y *set* correspondientes para cada propiedad de la clase, además de métodos propios de cada clase, como por ejemplo: La clase *Question* desarrolla el método *OrderAnswers* para ordenar las respuestas de una pregunta.

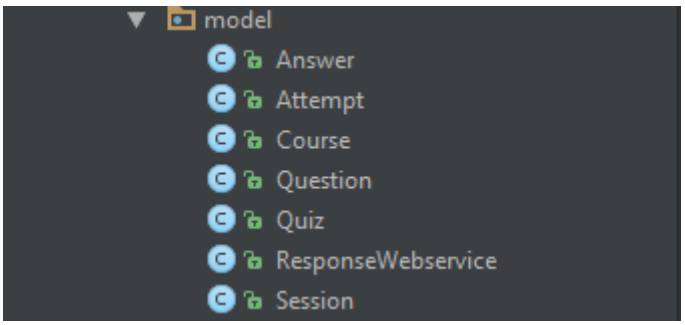

Figura 27 - Carpeta model.

Carpeta widgets:

La carpeta “widgets” contiene el código de la clase *QuizQuestionWidget* (Figura 28).

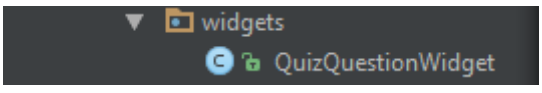

Figura 28 - Carpeta widgets.

La clase *QuizQuestionWidget* contiene diferentes métodos que, a partir de un objeto de la clase *Question*, devuelve un objeto de la clase *View* cuya interfaz y funcionalidad está definida por el tipo de pregunta que contiene el objeto *Question* y el número y orden de las respuestas de ese objeto.

La clase *QuizQuestionWidget* es utilizada en la actividad *AnsweQuizActivity*, que se mencionará más adelante, para generar dinámicamente la interfaz de esa actividad en función de las preguntas que tenga un cuestionario.

Clase ETSOApplication:

La clase "ETSOApplication" (figura 24) hereda de la clase *Application* y contiene los métodos para almacenar y obtener valores de las propiedades de la aplicación.

Carpeta ets:

La carpeta "ets" contiene el código de todas las actividades de la aplicación (figura 29).

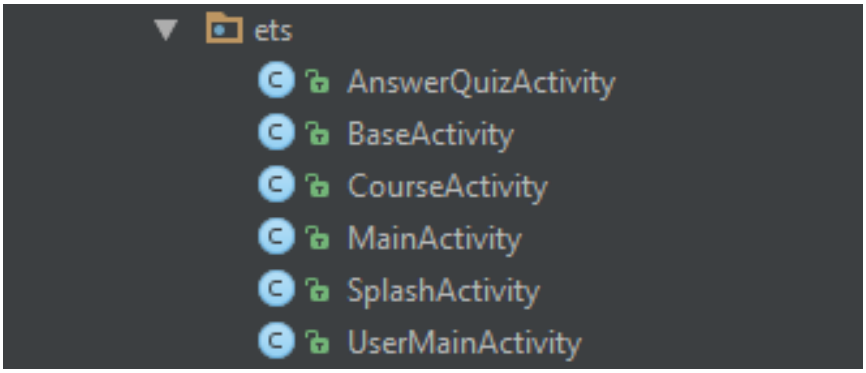

Figura 29 - Carpeta ets.

Se ha creado una clase *BaseActivity* que desciende de la clase *Activity*, esta clase base contiene los métodos comunes a todas las actividades, todas las actividades descienden de esta clase.

La actividad *SplashActivity* es únicamente una pantalla de *splash* que se retira a los 2.5 segundos para dar a la aplicación un aspecto más profesional.

La actividad *MainActivity* muestra dos campos para introducir usuario y contraseña, cuando el usuario los introduce la aplicación autentica al usuario contra el servidor.

Las tareas que realizan las actividades *UserMainActivity*, *CourseActivity* y *AnswerQuizActivity* son muy similares. Las tres actividades llaman a una tarea asíncrona (*AsyncTask*, explicará más adelante) que llama a un servicio web, con la respuesta de la tarea asíncrona se construye la interfaz de la actividad y diferentes *listerns* esperan la interacción del usuario con la aplicación.

Carpeta asyncTask:

La carpeta “asyncTask” contiene el código de las tareas asíncronas que se llaman desde las actividades para solicitar servicios web (figura 30).

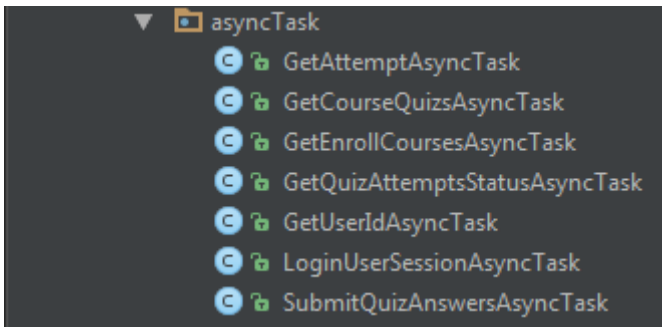

Figura 30 - Carpeta asyncTask.

Las clases de esta carpeta descienden de la clase *AsyncTask* (Developer Android, 2016b), esta clase permite realizar operaciones en segundo plano y publicar los resultados sobre el hilo de interfaz de usuario sin tener que manipular los hilos. Al ejecutar operaciones en un hilo diferente del de la interfaz de usuario esta no se bloquea y se consigue una experiencia de usuario de mayor calidad.

Las clases definidas en esta carpeta llaman a un servicio web en *background*, cuando reciben la respuesta del servicio web decodifican la respuesta y llaman a un método implementado en la actividad que creó la tarea asíncrona, pasándole como parámetro la respuesta obtenida. Se llamará a un método u otro de la actividad origen en función de la respuesta obtenida del servicio web.

Carpeta webServices:

La carpeta “webServices” contiene el código de las clases que realizan las peticiones web para solicitar los servicios web de *Moodle* (figura 31).

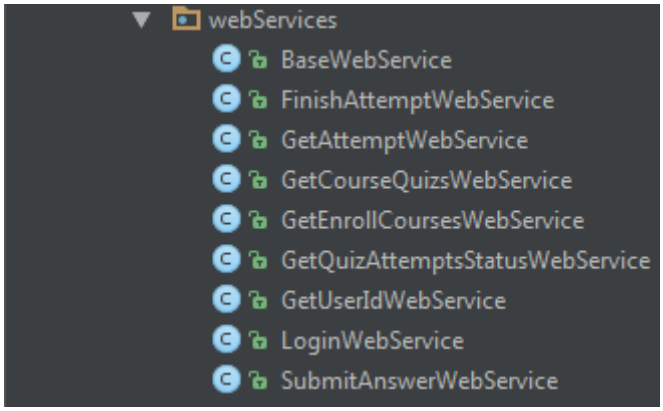

Figura 31 - Carpeta WebServices.

Se ha desarrollado una clase por cada servicio web que se solicita. Todas las clases heredan de la clase *BaseWebService*. Esta clase base principalmente implementa dos

tipos de métodos: un método que a partir del nombre de un servicio web genera la *url* del servicio que se quiere solicitar y los métodos para realizar las peticiones web. Este segundo tipo de métodos tienen como entrada la *url* que se quiere consultar y los parámetros que se desean enviar en el cuerpo o en la cabecera de la petición.

5. Pasos para desplegar en producción

Como se explica en el apartado 2 de este capítulo, el proyecto se ha realizado sobre una plataforma de desarrollo creada para construir la aplicación móvil y depurar su funcionamiento. Sin embargo, el objetivo final que persigue este proyecto es poder desplegar esta aplicación sobre la plataforma *Moodle* que la universidad de Valladolid tiene en producción (http://campusvirtual2015.uva.es/), plataforma que utiliza la facultad de traducción e interpretación y que consultan a diario sus alumnos.

Para desplegar la aplicación móvil sobre la plataforma en producción de la universidad de Valladolid o sobre cualquier otra plataforma *Moodle* se deben realizar diferentes tareas tanto en la plataforma *Moodle* como en el código de la aplicación móvil.

Tareas a realizar sobre la plataforma *Moodle:*

- Habilitar la utilización de servicios web con arquitectura REST del modo que se explicó en el apartado 2.6.5 de este informe, en el punto "Pasos para utilizar servicios web en *Moodle"*.

- Se deben copiar los ficheros referentes a las funciones externas creadas para el módulo cuestionarios que se encuentran en la plataforma de desarrollo, y depositarlos en la plataforma de producción dentro del directorio del módulo cuestionario (/mod/quiz). Estos ficheros son *externallib.php* y *services.php,* el primero contiene el código de las funciones externas referente a las funciones *mod_quiz_get_quiz_status_for_user, mod_quiz_start_one_attempt* y *mod_quiz_process_attempt,* el segundo contiene la interfaz de estas funciones.

- Se deben escribir en el fichero */mod/quiz/lib.php* de la plataforma de producción las funciones, *quiz_validate_new_attempt, quiz_prepare_and_start_new_attempt* y *quiz_process_attempt* cuyo código se encuentra en el fichero */mod/quiz/lib.php* de la plataforma de desarrollo o en el repositorio de la referencia (Github Moodle, 2016). Estas funciones son utilizadas por las funciones externas del paso anterior.

- Se debe actualizar la versión del módulo cuestionario para que la plataforma *Moodle* detecte los cambios y actualizar la plataforma.

- Como último paso en la plataforma, se debe crear un servicio web que contenga las funciones *core_user_get_users_by_field, user_get_user_details_courses, core_enrol_get_users_courses, core_course_get_contents, mod_quiz_get_quiz_status_for_user, mod_quiz_start_one_attempt* y

mod_quiz_process_attempt. Además de autorizar a todos los estudiantes de la escuela a utilizar este servicio.

Tareas a realizar sobre la aplicación móvil:

- Añadir al código de la figura 32 la dirección URL de producción y cambiar la constante URL_DESA por URL_PRO en los dos métodos que se muestran en la misma figura.

```
public class BaseWebService {
    private static final String URL_DESA = "http://plataformadesarrollo.es";
    private static final String URL_PRO = "";
    private static final String URL_REST_SERVICE = "/webservice/rest/server.php?";
    public static final int MAX_REQUEST_ATTEMPTS = 3;

    public static URL getUrlForWebservice(final String ws) {
        try {
            String urlService = URL_DESA + ws;
            return new URL(urlService);
        }catch (Exception exception){
            LogUtil.d("Url bad formed");
            return null;
        }
    }

    public static URL getUrlForRestservice(final String ws) {
        try {
            String urlService = URL_DESA + URL_REST_SERVICE + ws;
            return new URL(urlService);
        }catch (Exception exception){
            LogUtil.d("Url bad formed");
            return null;
        }
    }
```

Figura 32 - Código de la aplicación donde se indica la URL a la que apuntan los servicios web.

- Modificar el valor de la constante QUIZ_SERVICE por el nombre del servicio que se creó en la plataforma de producción, si este no fuese el mismo que el de la plataforma de desarrollo (ETSO_WS[3]) (figura 33).

```
public class LoginWebService extends BaseWebService{

    private static final String QUIZ_SERVICE = "ETSO_WS";
    private static final String WS_LOGIN = "/login/token.php?";
```

Figura 33 - Código de la aplicación donde se indica el nombre del servicio utilizado.

[3] ESTO_WS es únicamente un nombre que se le da al servicio web, es un nombre ambiguo, no relacionado con la aplicación de forma intencionada para dificultar los posibles intentos de *hackeo.*

CAPITULO 4. Manuales de usuario

En este punto cabe recordar cuáles son los objetivos de este proyecto: Permitir a un profesor crear un cuestionario en una plataforma *Moodle* gestionada por un administrador, dar al alumno la posibilidad de realizar el cuestionario desde su dispositivo móvil y obtener una retroalimentación en forma de nota de ese intento.

A continuación se detallan diversos manuales de usuario, tanto de la plataforma *Moodle* como de las funcionalidades de la aplicación móvil.

1. Plataforma Moodle

La plataforma *Moodle* será gestionada por un administrador, sin embargo, será un profesor quien cree los cuestionarios que realizarán los alumnos. Debido a que serán dos usuarios con roles diferentes quienes accedan a la plataforma *Moodle*, es necesario desarrollar manuales de usuario diferenciados.

1.1 Manual del administrador

El administrador es el encargado de mantener la plataforma *Moodle* de modo que los servicios web que se llamen desde la aplicación móvil obtengan respuesta. En este manual se explica cómo habilitar los servicios que debe tener la plataforma *Moodle* para que los servicios web funcionen correctamente. Este manual debe ser consultado por el administrador en la configuración inicial o en caso de fallo.

Para realizar las tareas que se presentan a continuación el usuario debe entrar en la plataforma *Moodle* con el rol de administrador.

Activación de los servicios web y el protocolo REST en Moodle:

Para activar los servicios web y el protocolo *REST* se debe acceder a la página "Vista general" de los servicios web, que se encuentra en:

Administración del sitio→ Extensiones → Servicios Web → Vista general

Cuando el administrador acceda a la ventana "Vista general" le aparecerán las entradas 1 (Habilitar Servicios Web) y 2 (Habilitar los protocolos) que se muestran en la figura 34, el administrador debe acceder a estas entradas y habilitar los servicios web (figura 35) y el protocolo REST (figura 36).

Figura 34 - Vista general de los servicios web.

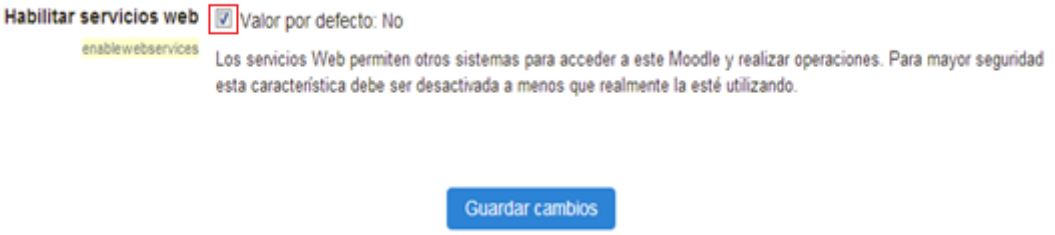

Figura 35 - Habilitar los servicios web.

Figura 36 - Habilitar el protocolo REST.

Habilitar el permiso "protocolo REST" al rol estudiante:

Por defecto, al igual que la plataforma *Moodle* trae deshabilitados los servicios web, también trae deshabilitado el permiso a acceder a los servicios web para el rol de estudiante. Para que los estudiantes puedan utilizar los servicios web, se debe habilitar este permiso para este rol, esto se hace accediendo a la ventana "Definir roles" de la sección permisos de usuarios que se encuentra en:

Administración del sitio → Usuarios →Permisos →Definir roles

Una vez dentro de la ventana "Definir roles" (figura 37), se debe editar el rol estudiante y habilitar el protocolo REST (figura 38).

Figura 37 - Ventana Definir roles.

Figura 38 - Habilitar el protocolo REST para el rol estudiante.

Crear un servicio:

Un servicio es un conjunto de funciones a las que tendrán acceso un conjunto de usuarios autorizados. Primeramente se debe crear un servicio accediendo a:

Administración del sitio → Extensiones → Servicios Web → Servicios Externos

En la ventana "Servicios externos" (figura 39), el administrador debe agregar un nuevo servicio marcando las casillas "Habilitado" y "Únicamente usuarios autorizados" (figura 40).

Figura 39 - Ventana servicios externos, añadir servicio.

Figura 40 - Agregar un nuevo servicio.

Agregar funciones al servicio:

Una vez se ha creado un servicio, se deben añadir a ese servicio las funciones que serán accesibles a través de él, accediendo a:

Administración del sitio → Extensiones → Servicios Web → Servicios Externos

En la ventana "Servicios externos", el administrador debe entrar en "Funciones" (figura 41) del servicio al que desee agregar las funciones (figuras 42 y 43).

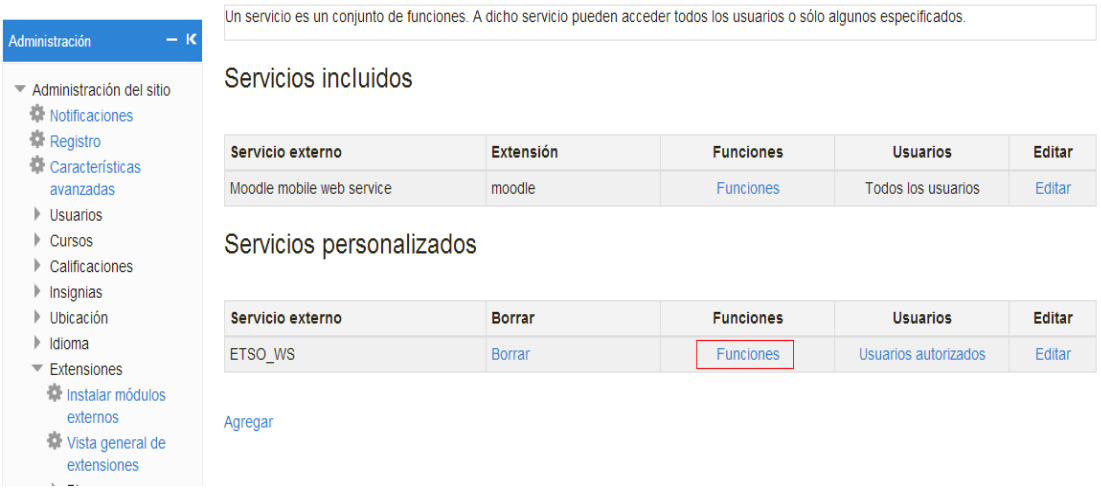

Figura 41 - Ventana servicios externos, añadir función.

Agregar funciones al servicio "ETSO_WS"

Función	Descripción	Permisos requeridos	Editar
core_user_get_users_by_field	Retrieve users information for a specified unique field - If you want to do a user search, use core_user_get_users()	moodle/user:viewdetails, moodle/user:viewhiddendetails, moodle/course:useremail, moodle/user:update	Eliminar
core_enrol_get_users_courses	Get the list of courses where a user is enrolled in	moodle/course:viewparticipants	Eliminar
core_course_get_contents	Get course contents	moodle/course:update,moodle/course:viewhiddencourses	Eliminar
mod_quiz_get_quiz_status_for_user	return finished if user has used all atemps for a quiz or available if the user has not completed all atemps		Eliminar
mod_quiz_start_one_attempt	it starts one attempt and return it		Eliminar
mod_quiz_process_attempt	Process responses during an attempt at a quiz and also deals with attempts finishing.	mod/quiz:attempt	Eliminar

Agregar funciones

Figura 42 - Agregar una función (1).

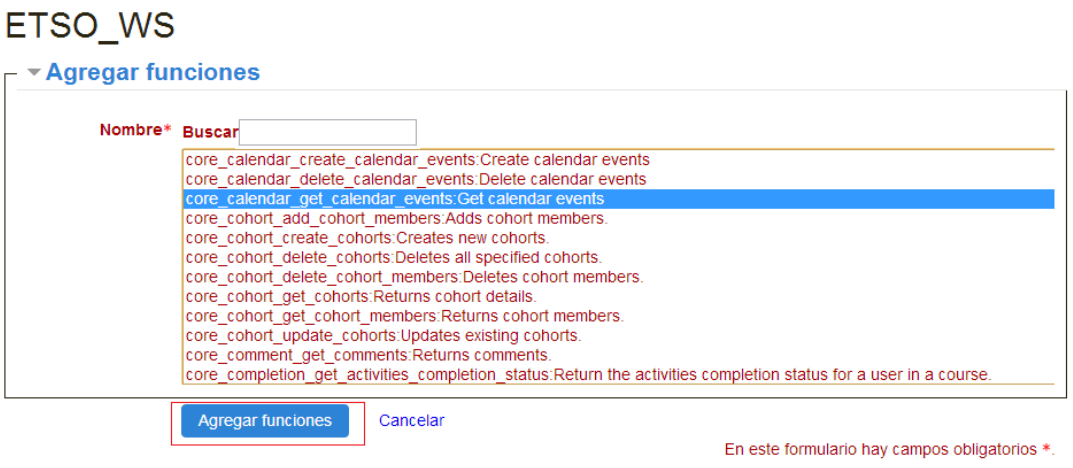

Figura 43 - Agregar una función (2).

Agregar usuarios autorizados a un servicio:

El último paso que debe realizar el administrador es añadir los usuarios autorizados a utilizar un servicio, este paso generará un token para ese servicio, este token es el se ha mencionado previamente en la memoria. Para autorizar a un usuario se debe acceder a:

Administración del sitio → Extensiones → Servicios Web → Administrar tokens

Dentro de la ventana "Administrar tokens" (figura 44) el administrador debe pulsar en "Agregar" y seleccionar el/los usuarios que desea agregar y a qué servicio desea hacerlo (figura 45).

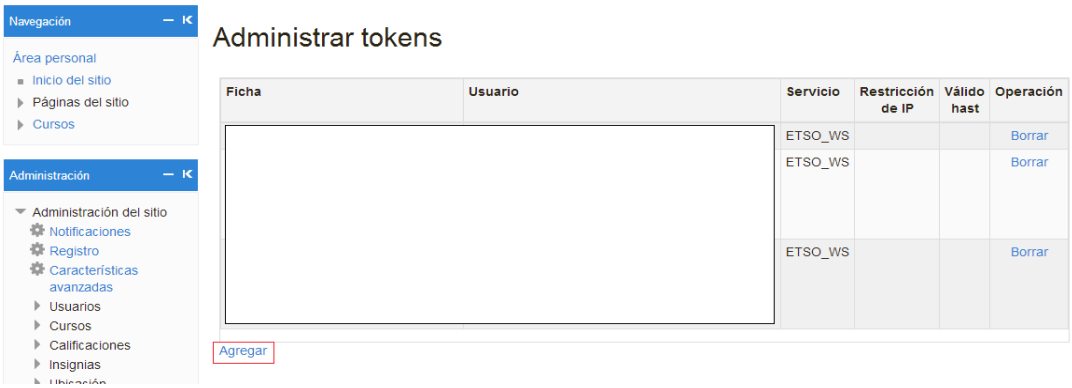

Figura 44 - Ventana administrar tokens[4].

Figura 45 - Agregar usuario autorizado.

[4] Por seguridad se han ocultado datos de esta tabla.

1.2 Manual del profesor

El profesor es el encargado de generar los cuestionarios en la plataforma *Moodle* en las asignaturas de las que sea tutor. En este manual se explica cómo crear cuestionarios y las restricciones que se deben tener en cuenta a la hora de crear el cuestionario para que este se pueda visualizar y funcione correctamente en la aplicación móvil.

Para poder crear un cuestionario el usuario debe entrar en la plataforma *Moodle* con el rol de profesor.

Crear un cuestionario:

Para crear un cuestionario el profesor debe seleccionar el curso del que es tutor y en el que desea crear el cuestionario.

Área personal →Mis cursos → "Curso donde crear el cuestionario"

Una vez dentro del curso el profesor debe activar la opción editar:

Administración del curso → Activar edición

Una vez activada la edición del curso, el profesor debe elegir la sección donde se añadirá el cuestionario, seleccionar la opción "añadir una actividad o un recurso" (figura 46) y seleccionar que la actividad que desea añadir es de tipo cuestionario (figura 47).

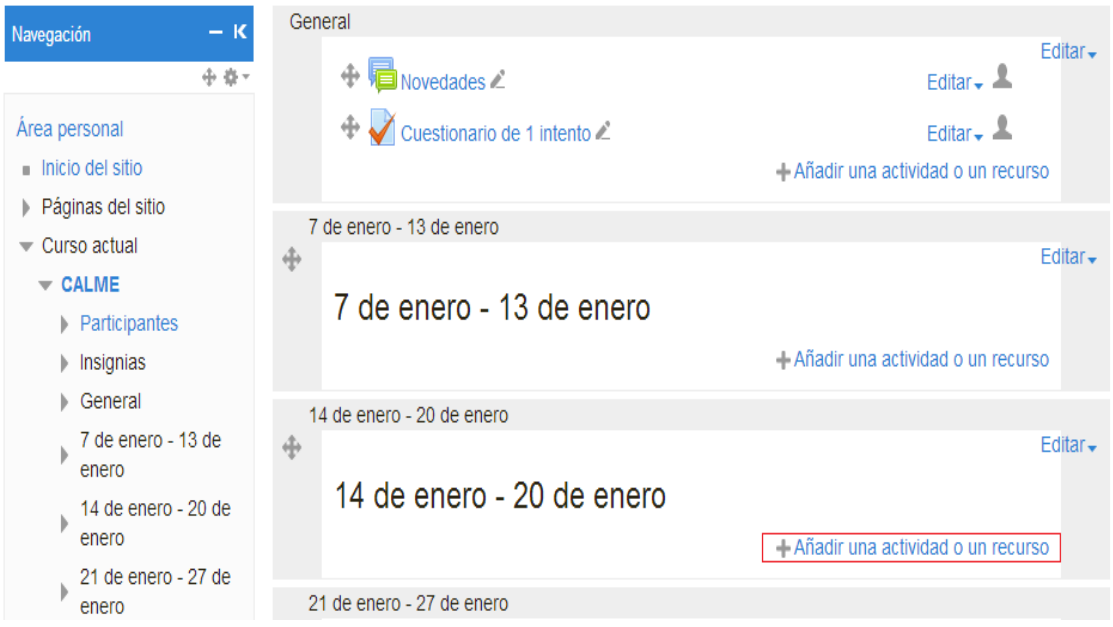

Figura 46 - Añadir una actividad o un recurso.

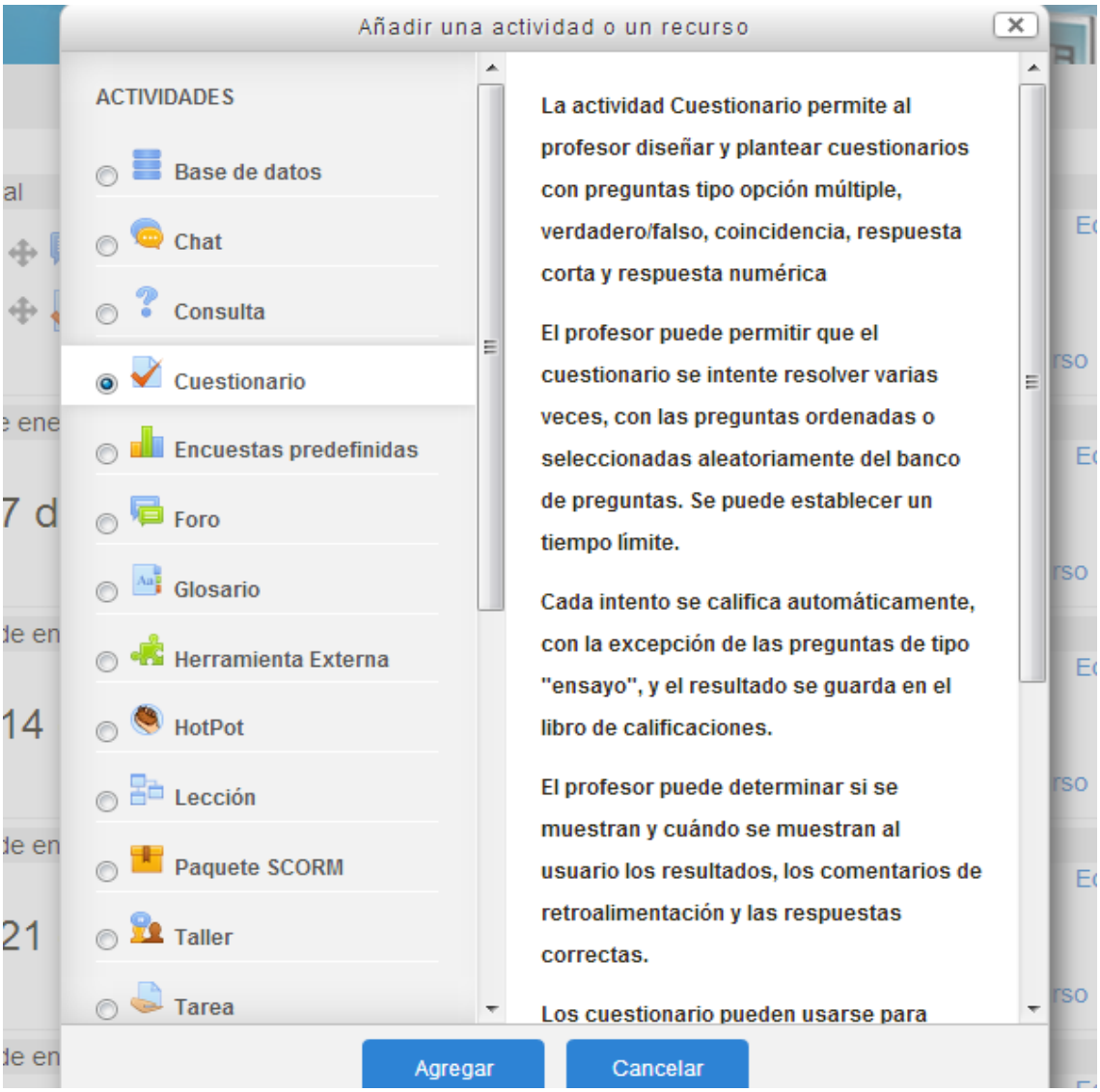

Figura 47 - Actividad tipo cuestionario.

Una vez añadida la actividad del tipo cuestionario se deben configurar diferentes propiedades para el cuestionario, tales como: calificación, temporalización, comportamiento de las preguntas, etc. Estas propiedades no pueden configurarse al azar ya que la aplicación móvil no está desarrollada para soportar todas las posibles configuraciones, se deben tener en cuenta ciertas restricciones que se comentan a continuación, para un correcto funcionamiento de la aplicación móvil:

- General (figura 48):
  Se puede:
  - Se puede y es obligatorio asignar un nombre a un cuestionario, este aparecerá en la aplicación para identificar al cuestionario dentro de un curso.

  No se puede:
  - La descripción del cuestionario no aparecerá en la aplicación móvil.

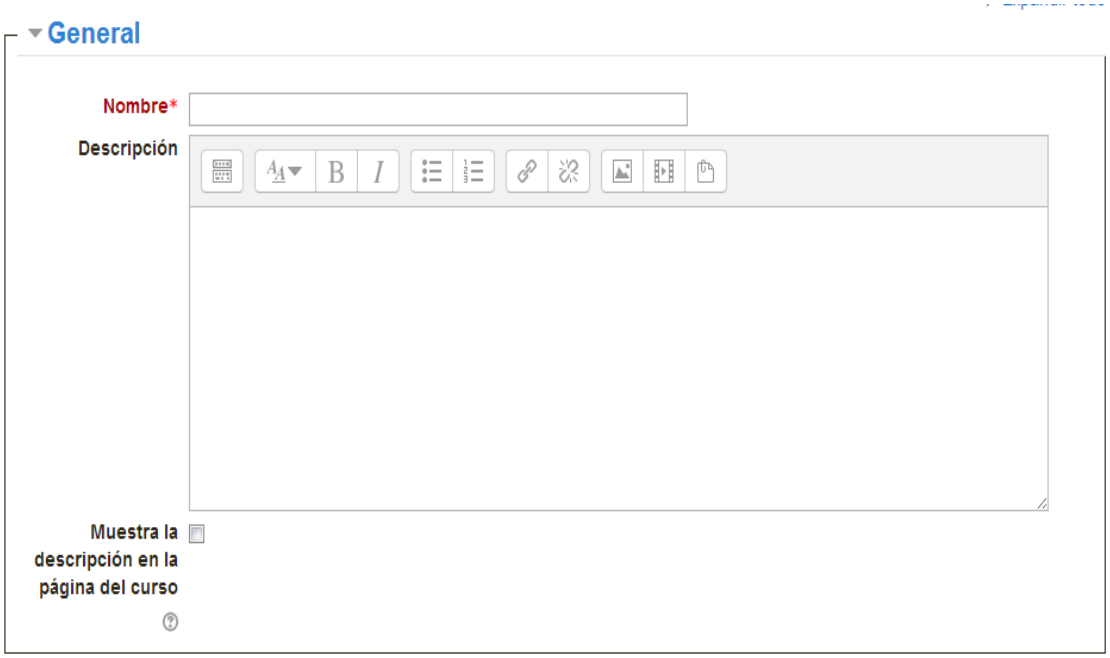

Figura 48 - Propiedades generales de un cuestionario.

- Temporalización (figura 49):
  Se puede:
  - Habilitar una fecha para abrir y cerrar el cuestionario y un periodo de gracia. Al iniciar un intento desde la aplicación móvil si el cuestionario está cerrado (fecha pasada) o aún no abierto se informará al usuario.

  No se puede:
  - Habilitar un tiempo límite para realizar el intento de un cuestionario, ya que la aplicación no está preparada para informar al estudiante del tiempo que le resta para terminar el intento.

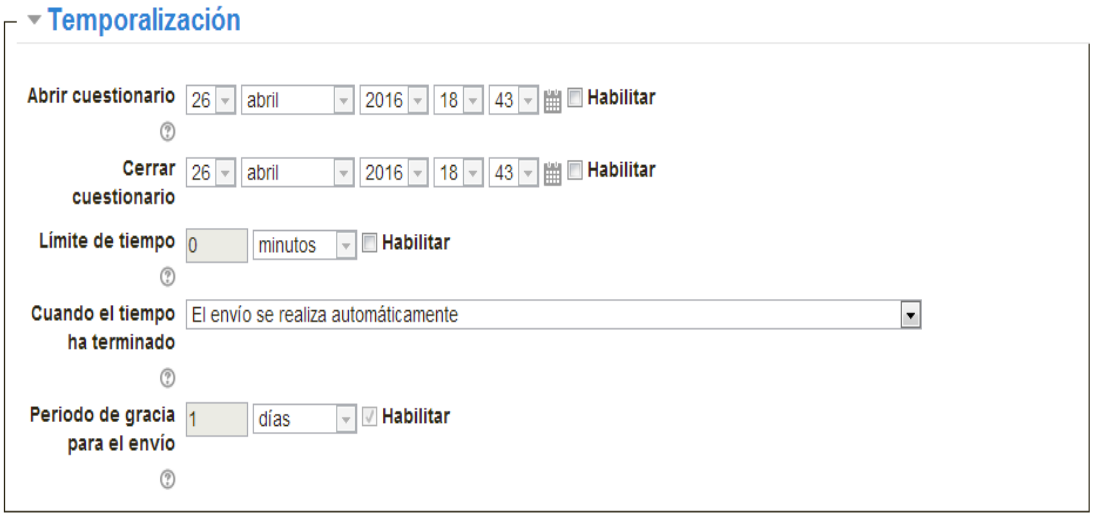

Figura 49 - Propiedades de temporalización de un cuestionario.

- Calificación (figura 50):

Se puede:

- Todas las configuraciones son posibles. Si desde la aplicación móvil se desea iniciar un intento de un cuestionario y ese usuario ha consumido todos los intentos se le informará. La calificación de un intento y si este está aprobado o no, se calcula en el servidor y la aplicación móvil únicamente obtiene esta información.

Figura 50 - Propiedades de calificación de un cuestionario.

- Esquema (figura 51):

  Se puede:

  - Es indiferente la elección del número de preguntas que se mostrarán por página, en la aplicación móvil todas las preguntas de un cuestionario se mostrarán en la misma página.

Figura 51 - Propiedades de esquema de un cuestionario.

- Comportamiento de las preguntas (figura 52):

  Se puede:

  - Ordenar las respuestas al azar, la aplicación está preparada para obtener el orden de las respuestas y mostrarlas ordenadas.

  No se puede:

  - Mostrar una retroalimentación desde la aplicación a las respuestas del estudiante, este deberá acceder a la plataforma web para obtener la retroalimentación.
  - Desde la aplicación móvil no es posible recuperen las respuestas de un intento anterior al iniciar un nuevo intento por lo que las opciones “basar un intento en el anterior” no deben configurarse.

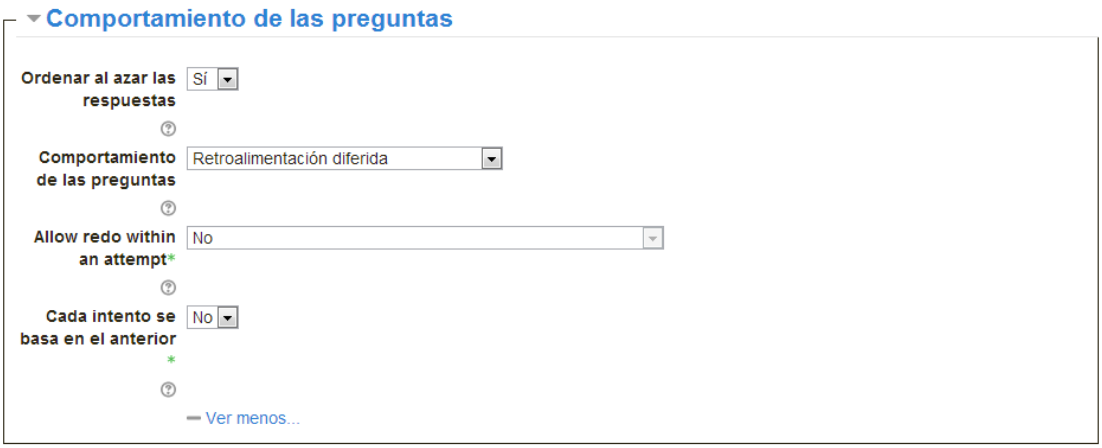

Figura 52 - Propiedades del comportamiento de las preguntas de un cuestionario.

- Revisar intento (figura 53):
  Se puede:
  - Mientras el intento está en curso el estudiante podrá revisar la respuesta dada a una pregunta y cambiar esta las veces que sea necesario.

  No se puede:
  - Una vez que el intento ha finalizado, el estudiante no podrá comprobar las respuestas dadas desde la aplicación móvil.
  - La única retroalimentación que obtendrá el estudiante una vez que un intento ha finalizado es la nota obtenida en el cuestionario en el instante posterior a finalizar el intento.

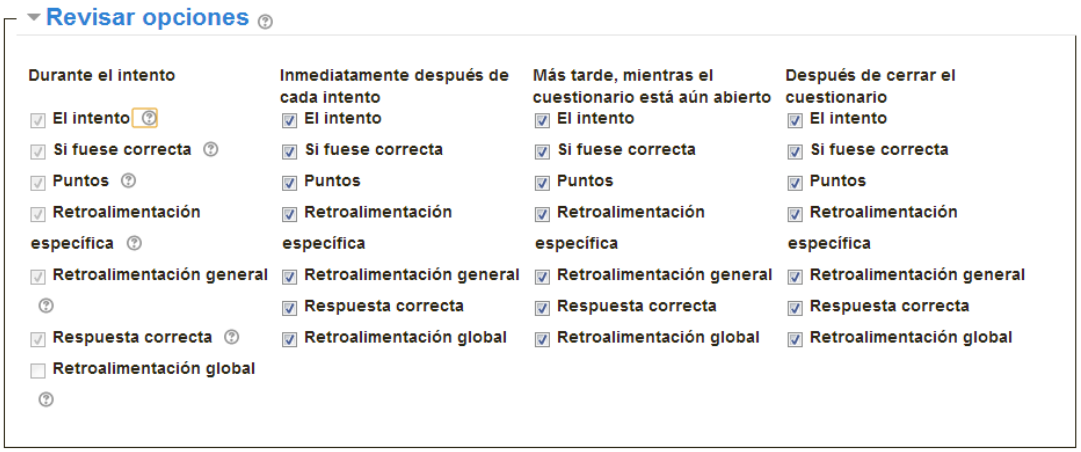

Figura 53 - Propiedades sobre la revisión del intento de un cuestionario.

- Apariencia (figura 54):
  Se puede:

- El número de decimales en las calificaciones de las preguntas y del cuestionario se tendrá en cuenta a la hora de mostrar la calificación final del cuestionario al usuario.

No se puede:

- En la aplicación móvil no se mostrará la imagen del usuario ni los bloques durante el intento aunque estas opciones se seleccionen.

Figura 54 - Propiedades de apariencia de un cuestionario.

- Restricciones extra sobre intentos (figura 55):

Se puede:

- Configurar una demora entre intentos, el servidor no permitirá iniciar un nuevo intento si el tiempo entre intentos no es el configurado, por lo que aunque la aplicación móvil desee iniciar un nuevo intento este no se iniciará y la aplicación informará de este suceso al estudiante.

No se puede:

- Configurar opciones de seguridad como contraseña, dirección de red o seguridad del navegador ya que estas opciones no están soportadas en la aplicación móvil y provocaría que el cuestionario no se pudiese realizar.

Figura 55 - Restricciones extra sobre intentos de un cuestionario.

- Retroalimentación global (figura 56):
  No se puede:
  - Una vez terminado el intento ninguna retroalimentación (que no sea la nota) se mostrará en la aplicación móvil. Los comentarios en función de la nota obtenida únicamente serán accesibles desde la plataforma web.

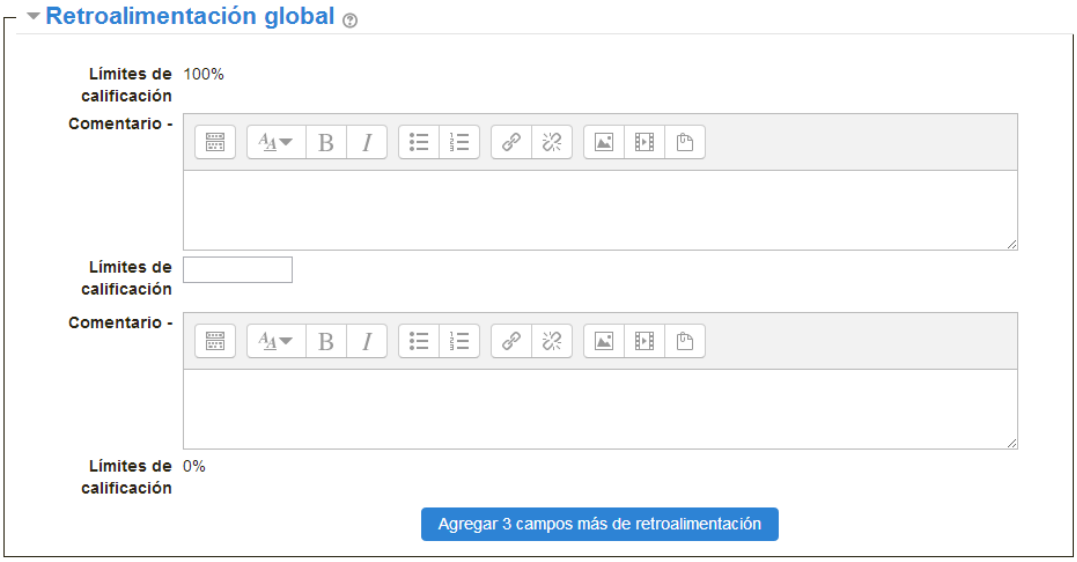

Figura 56 - Retroalimentación global de un intento.

Añadir preguntas a un cuestionario:

Una vez creado el cuestionario, el profesor debe añadir las preguntas. Para añadir las preguntas el profesor accederá al cuestionario creado y pulsará en la opción editar

cuestionario, una vez realizado esto se mostrará la pantalla para editar el cuestionario (figura 57).

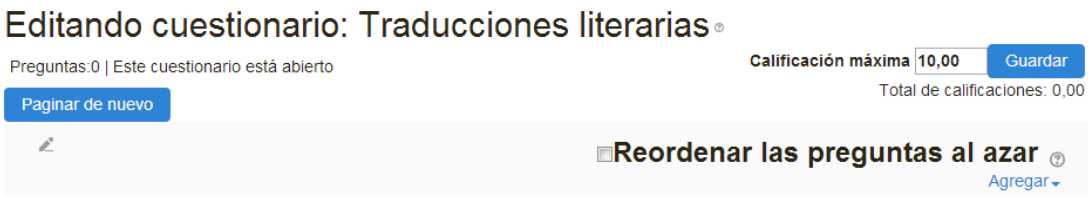

Figura 57 - Ventana editar cuestionario.

En la ventana "editar cuestionario" el profesor podrá seleccionar una calificación máxima (la suma de la calificación de todas las preguntas debe ser igual a la calificación máxima), elegir si las preguntas se ordenarán al azar o no (ambas opciones soportadas en la aplicación móvil) y añadir preguntas pulsando en la opción "agregar".

Al agregar una nueva pregunta aparecerá una lista con todos los tipos de preguntas que soporta la versión *Moodle* con la que se esté trabajando (figura 58), en este caso *Moodle 3.0*.

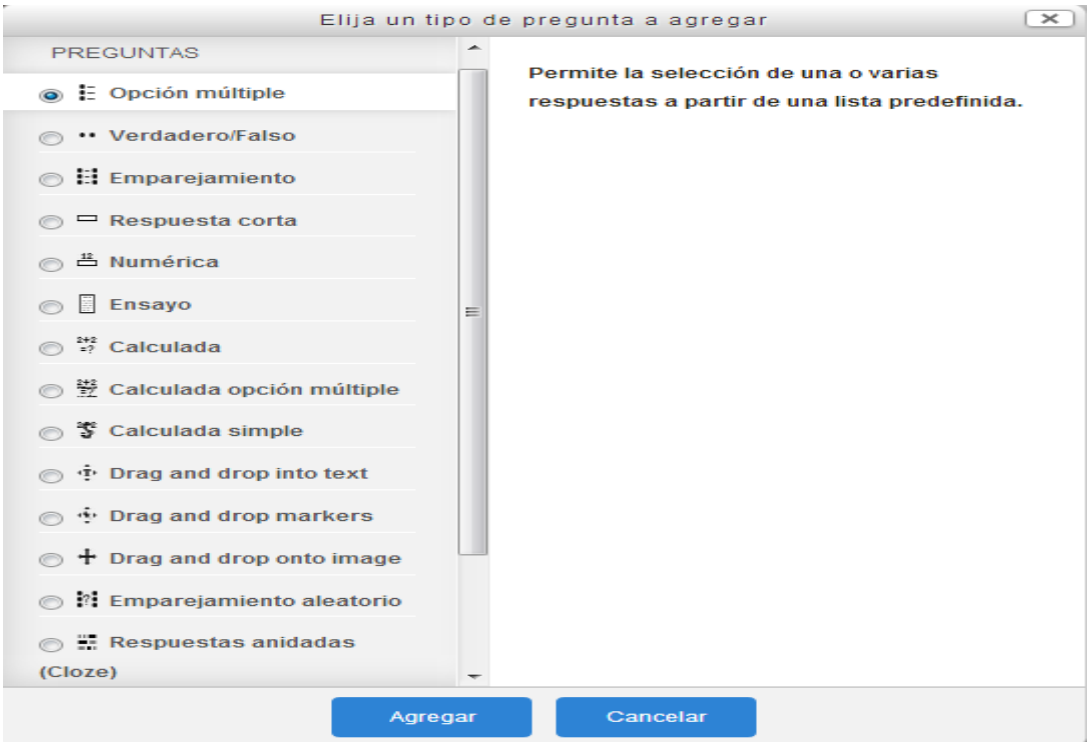

Figura 58 - Tipos de preguntas soportadas por Moodle 3.0.

La aplicación móvil únicamente soporta tres tipos de preguntas de las que *Moodle* proporciona: Opción múltiple, Verdadero/Falso y Respuesta corta. Se han elegido estas tres opciones como las soportadas por la aplicación móvil ya que son las más utilizadas por la mayoría de profesores en los cuestionarios.

Una vez seleccionado el tipo de pregunta, el profesor deberá rellenar diferentes campos en función del tipo de pregunta, teniendo en cuenta ciertas restricciones al igual que ocurría al crear un cuestionario. Para las preguntas, estas restricciones son principalmente dos:

- La aplicación móvil no soporta retroalimentación por lo que la retroalimentación que se añada a las preguntas en función de la respuesta del usuario únicamente será visible desde la plataforma web.

- La aplicación móvil no soporta la opción de múltiples intentos para una pregunta, una vez que la respuesta a un pregunta ha sido enviada no es posible obtener desde la aplicación la respuesta que el usuario envió.

2. Aplicación móvil

La aplicación móvil será accedida por los estudiantes para realizar los cuestionarios que los profesores hayan creado previamente en la plataforma web.

Para el desarrollo de la interfaz se ha optado por seguir uno de los principios básicos de diseño del estilo global de *Moodle:* la simplicidad. Según este principio, se debe utilizar el mínimo de interfaz necesario para obtener la funcionalidad que se desee implementar.

Para mostrar las listas de cursos en los que está matriculado un usuario y los cuestionarios disponibles en un curso se ha optado por utilizar *ListViews*, los cuales permiten visualizar de una forma muy fácil e intuitiva estas listas. Por otro lado, la interfaz de los cuestionarios se crea de forma dinámica en función del número y tipo de preguntas que constituyen el cuestionario.

A continuación, se va a proceder a detallar todas las funcionalidades de la aplicación móvil de este proyecto.

2.1 Manual de usuario

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 60. 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.

Los datos a introducir en los campos antes mencionados se tratan de las credenciales que cada usuario utilice para acceder a su cuenta *Moodle* vía web. Para acceder a la aplicación es necesario introducir el valor correcto de estos 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 (figura 59).

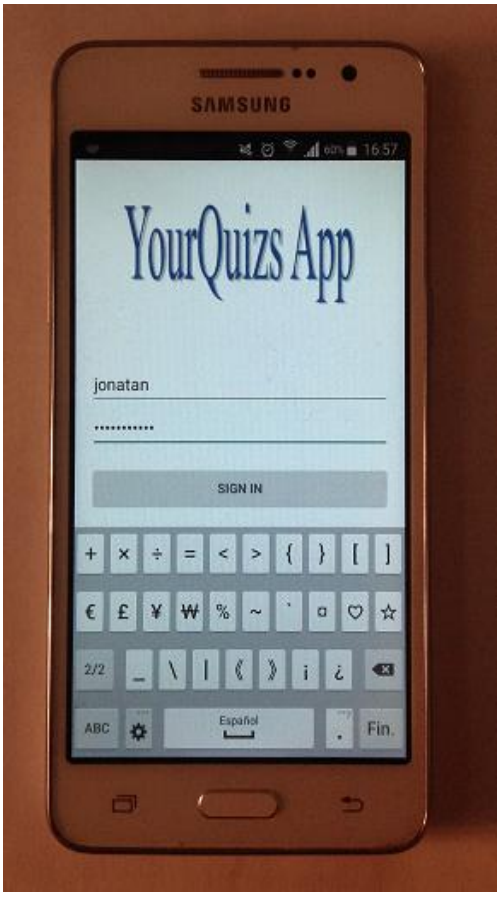

Figura 62 - Pantalla de inicio.

Figura 61 - Error de autenticación.

Pantalla de cursos o asignaturas:

Una vez autenticado en la aplicación, al usuario le aparecerá un listado de los cursos o asignaturas en las que está matriculado, además del nombre del usuario autenticado (figura 61). Simplemente seleccionando un curso se podrá acceder a la lista de los cuestionarios que contiene el curso.

Figura 63 - Pantalla asignaturas.

Pantalla de cuestionarios:

Una vez seleccionada la asignatura de la que se quieren consultar los cuestionarios, aparecerá un listado de los cuestionarios que contiene ese curso, además del nombre del curso en el que el usuario se encuentra (figura 63). Al pulsar sobre un cuestionario se comprobará si al usuario le quedan intentos para ese cuestionario, si al usuario no le quedan intentos se le informará (figura 64), si por el contrario al usuario le quedan intentos se le pedirá que confirme si realmente desea iniciar un intento (figura 62).

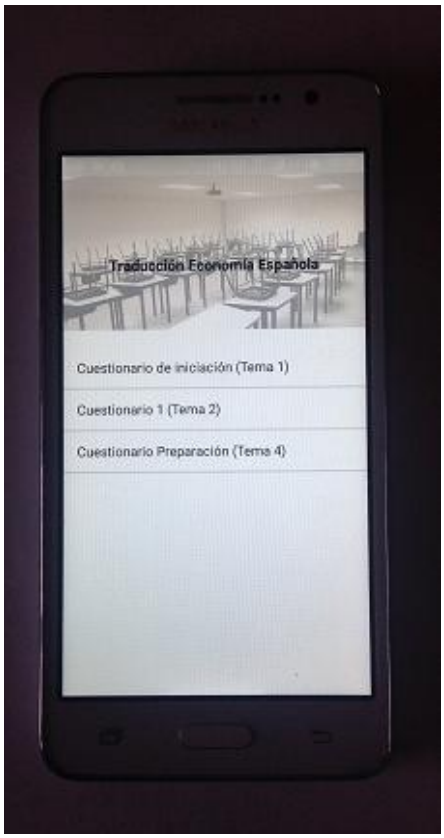

Figura 65 - Pantalla cuestionarios.

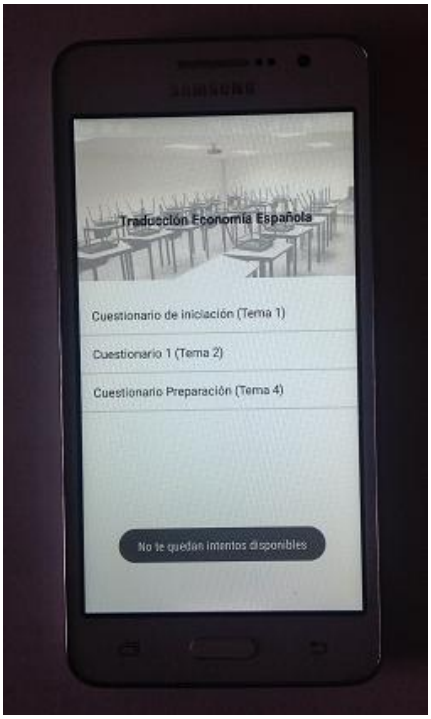

Figura 66 - Cuestionario sin intentos.

Figura 64 - Iniciar intento de cuestionario.

Al iniciar un cuestionario se comprobará si este se encuentra dentro de fecha para poder realizarlo, de no ser así se informará a usuario (figura 65).

Figura 67 - Cuestionario fuera de fecha.

Pantalla detalle del cuestionario:

Una vez se ha comprobado que al usuario le quedan intentos para realizar el cuestionario, que el cuestionario se encuentra dentro de fecha y que el usuario ha aceptado iniciar un nuevo intento, se mostrará una pantalla con las preguntas del cuestionario (figura 66). La interfaz de esta pantalla se crea de forma dinámica en función del número y tipo de preguntas que contiene el cuestionario, el usuario deberá contestarlas y pulsar el botón de enviar (que se encuentra en la parte baja de la pantalla) para enviar y terminar el cuestionario.

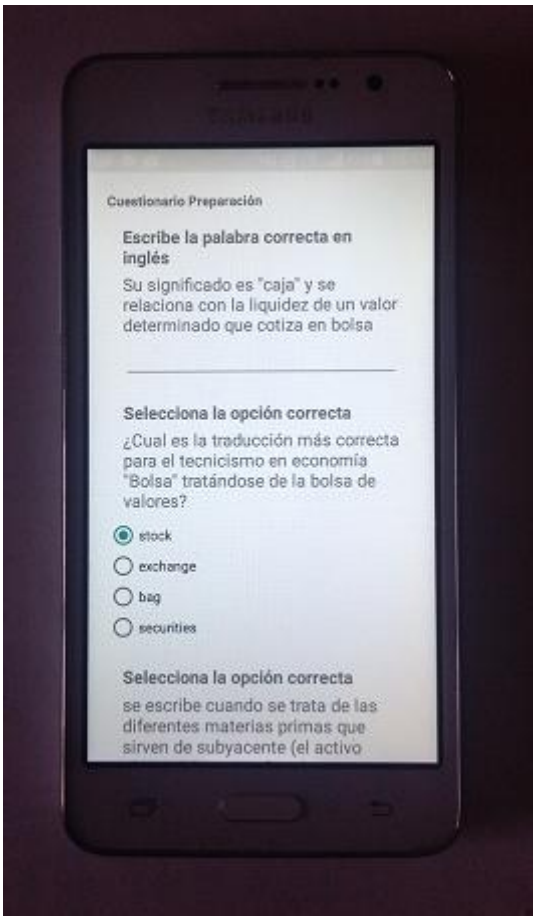

Figura 68 - Pantalla detalla del cuestionario.

Cuando el usuario pulse el botón enviar se le pedirá que confirme que desea enviar esas respuestas (figura 67), al igual que si el usuario pulsa el botón “volver”, se le informará que va a abandonar el intento y que confirme que desea abandonar el cuestionario (figura 68).

Finalmente si el usuario confirma que desea enviar las respuestas, el intento se finalizará y al usuario se le mostrará la nota obtenida en ese intento (figura 69). Si el usuario desea obtener mayor retroalimentación sobre las respuestas correctas o comentarios del profesor deberá acceder a la plataforma *Moodle* de la universidad ya que esas funcionalidades no están disponibles en la versión actual de la aplicación.

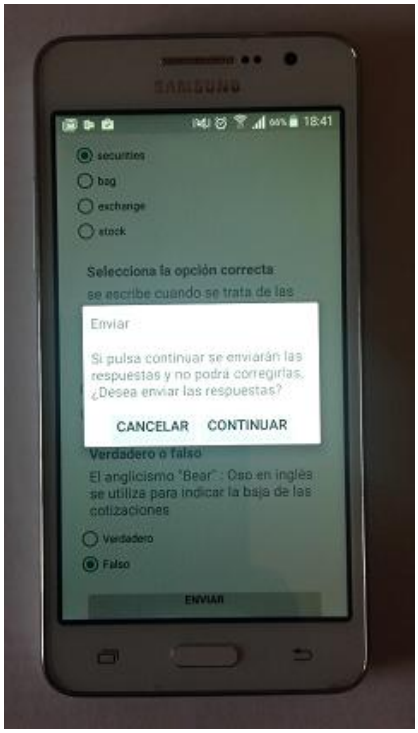

Figura 69 - Pantalla envío de respuestas.

Figura 70 - Pantalla salir del cuestionario.

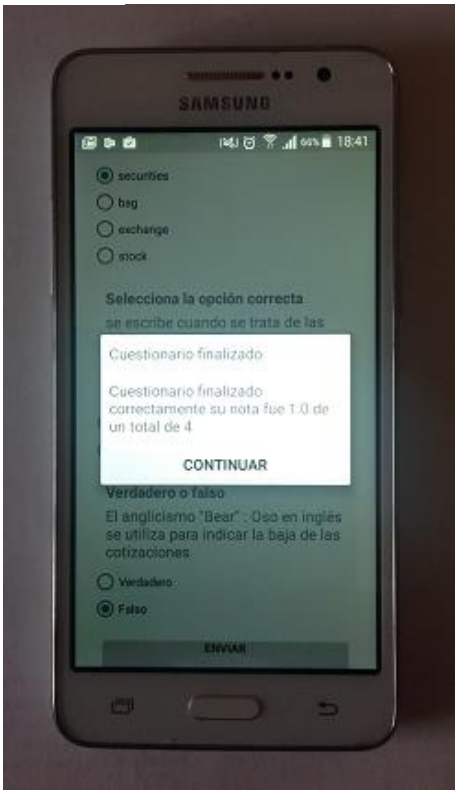

Figura 71 - Nota del cuestionario.

CAPITULO 5. Presupuesto

En este capítulo se estimará un presupuesto económico del proyecto realizado. Siguiendo las diferentes fases seguidas en la realización de ese proyecto, se irán incluyendo los diferentes gastos tanto software como de hardware que han surgido durante este desarrollo.

Antes de comenzar con dicha estimación, se comentará que el mayor coste para el desarrollador al realizar este proyecto ha sido el tiempo invertido en formarse en la tecnología *Moodle* ya que este desconocía la tecnología por completo. Este coste no se repercutirá en el coste del proyecto sino que se tratará de aprovecharse este conocimiento adquirido para realizar nuevas ofertas al cliente de modo que se aumenten las funcionalidades de la aplicación. Al cliente únicamente se le presupuestarán las horas utilizadas para desarrollo, tratando de ajustar el precio de la aplicación al precio de mercado que ofrecería un desarrollador con experiencia en *Moodle,* servicios web y desarrollo android nativo. Con esto se pretende fidelizar al cliente de modo que se aprovechen y rentabilicen los conocimientos obtenidos por el desarrollador ya que se trata de una aplicación con muchas líneas futuras que poder desarrollar.

El presupuesto estimado para esta aplicación, suponiendo que esta aplicación la hubiese realizado un trabajador autónomo ha sido:

Estimación presupuesto económico		
Desarrollo de los servicios web de *Moodle*	20€/h * 20h	400€
Desarrollo de la aplicación Android	15€/h * 140h	2.100€
Cuota autónomo	264.44€/mes =8.59€/día → En 20 días: 170€	170€
Software Libre	Android Estudio	0€
	Xampp	0€
	FireBug (php debug)	0€
Coste de ordenador portátil con Windows 7	Coste total 550€. Amortización fiscal 25% al año: 150€. 20 días: 8.50€	8.50€
Coste de teléfono Samsung S3 mini con Android 4.4 y pantalla de 4.0 pulgadas	Coste total 350€. Amortización fiscal 25% al año: 87.25. 20 días: 4.80€	4.80€
Coste de teléfono Samsung	Coste total 674€. Amortización fiscal 25% al	9.3€

S7 con Android 5 y pantalla de 5.1 pulgadas	año: 168.5. 20 días: 9.3€	
Coste oficina y gastos corrientes	Alquiler + Luz + Conexión a internet + Gas + Teléfono (20 días)	225€
Asesoría Moodle, permisos, solución de fallos…	2horas*35€/hora = 50€	70€
	TOTAL	2.987.6 €

El ordenador portátil ha sido necesario durante todo el proceso de desarrollo de la aplicación. Primero se usó para la búsqueda de documentación inicial y para el aprendizaje de *Moodle*, el aprendizaje de *Moodle* se ha realizado fundamentalmente gracias a la documentación que ofrece la web oficial (www.moodle.org) y los foros de esta misma web. Una vez finalizada la fase previa de documentación y aprendizaje, se comenzó a plantear lo que sería la aplicación en sí. Para ello se tuvo en cuenta siempre un enfoque de sencillez y usabilidad, buscando que su uso fuese agradable y atractivo para el usuario final, el estudiante.

La primera tarea de desarrollo realizada fue generar los servicios web de la plataforma *Moodle*, se han tenido en cuenta 20 horas, en las que no se ha contabilizado la fase inicial de aprendizaje: estructuración del código de *Moodle*, funcionamiento general, estructura de la base de datos de *Moodle*, el estudio sobre la creación de servicios web en *Moodle*, etc. Sí se han tenido en cuenta, tanto las horas de desarrollo del código de los servicios web creados, como el tiempo de estudio de los servicios web que trae *Moodle 3.0* que se han utilizado en la aplicación móvil. El precio por hora es de 20€, superior al precio por hora de desarrollo Android (15€), ya que la oferta de desarrolladores que puedan generar código para la plataforma *Moodle* es inferior a la oferta de desarrolladores Android.

La segunda tarea realizada fue el desarrollo de la aplicación móvil. Se han tenido en cuenta 140 dedicadas tanto al desarrollo del código, como a pruebas de funcionamiento. Las tareas de estudio en el desarrollo de la aplicación móvil han sido mínimas, ya que el desarrollador tenía un sólido conocimiento de desarrollo de aplicaciones móviles Android. Teniendo en cuenta el precio por hora de mercado de un desarrollador autónomo, se ha estimado un precio por hora de 15€.

En total se han necesitado 160 horas de mano de obra, son las horas equivalentes a 20 días de trabajo con una jornada laboral de 8 horas al día. Es por este motivo por el que todos los cálculos de los gastos se estiman para 20 días.

Aclaraciones a las diferentes partidas de este presupuesto:

- Salario del programador, representa el mayor coste del presupuesto y la mayor partida de ingresos.
- Cuota mensual de la seguridad social en régimen de autónomos (266.44€/mes). En este presupuesto este gasto se ha prorrateado para 20 días.
- Coste de terminales móviles utilizados para depurar el funcionamiento de la aplicación en diferentes versiones Android y la interfaz de usuario en diferentes tamaños de pantalla. Estos terminales tienen una depreciación y según el Ministerio de Hacienda se permite que anualmente se incluya como gasto de depreciación en la declaración de impuestos hasta un 25% del valor total.
- Coste del ordenador portátil utilizado para el desarrollo y búsqueda de documentación. La depreciación del ordenador portátil es similar a la de los terminales móviles.
- Gastos generales correspondientes al lugar de trabajo. Estos son, el coste del alquiler, la luz, el gas, la conexión a internet y el teléfono. La estimación de estos gastos fijos se ha prorrateado para 20 días.
- Asesoría: Se ofrecerán 2 horas de asesoría al administrador de la plataforma *Moodle* en producción. Las tareas de asesoramiento se dirigirán hacia: las implicaciones que tiene habilitar los servicios web en la plataforma *Moodle*, como autorizar a usuarios, como solucionar diversos problemas que puedan surgir, etc.

Finalmente, se ha estimado que el presupuesto para el desarrollo completo de la aplicación elaborada en este TFG, es de 2.987,6€.

De este importe, se estima un beneficio aproximado para el desarrollador de 2.570€

CAPITULO 6.
Conclusiones.

En este trabajo fin de grado, se ha realizado una primera versión de una aplicación móvil para dispositivos *Android* destinada al uso académico, en la cual se permite realizar los cuestionarios que un profesor crea en una plataforma *Moodle.*

Se habla de una primera versión, ya que como se mencionará en el capítulo siguiente, la posible proyección y las líneas futuras de este proyecto son muy amplias. Con la posibilidad tanto de mejorar la funcionalidad desarrollada (realizar cuestionarios), como de añadir nuevas funcionalidades (foros, mensajería, etc.) o desarrollar la aplicación para otros sistemas operativos.

Debido a que se pretende que esta aplicación sea una primera versión a continuar por otros desarrolladores se ha prestado gran atención en la estructuración y limpieza del código. A demás de utilizar tecnologías ampliamente conocidas como es *REST* para la implementación de los servicio Web o el sistema operativo Android. Android es el SO económicamente más asequible y la comunidad de desarrolladores en España es muy superior a la de otros SOs.

En el apartado técnico, mencionar la experiencia que he adquirido en el desarrollo de aplicaciones Android, he mejorado en cuanto a metodología de trabajo, me he dado cuenta de que en proyectos con gran cantidad de líneas de código hay que trabajar de una forma estructurada, creando modelos de objetos, servicios web, testear esos servicios web, desarrollar las interfaces de las actividades, etc. El orden y la organización son de gran importancia en proyectos grandes.

También en el desarrollo Android he mejorado en cuanto al conocimiento de la tecnología y diferentes posibles soluciones para un mismo problema, como puede ser utilizar *Asyncktask* o no, utilizar un *listview* o un layout lineal, etc. A la hora de desarrollar una aplicación móvil son muchas las posibles soluciones que se pueden utilizar, y el conocer mejor la tecnología y sus posibilidades hace que la solución obtenida sea la más óptima en cada caso.

Personalmente, el conocimiento adquirido en este proyecto me servirá de gran ayuda en mi vida laboral, ya que al inicio del proyecto tenía bastantes conocimientos sobre el desarrollo de aplicaciones móviles, sin embargo eran conocimientos desordenados que este proyecto me ha ayudado a ordenar. Aunque tenía conocimientos previos sobre el desarrollo de aplicaciones móviles, nunca había trabajado con la parte del servidor. Este proyecto me ha servido para conocer cómo trabaja un CMS y que requisitos debe tener un CMS en función de las necesidades del proyecto. He conocido y analizado muchas de las posibles soluciones tanto de LMS como de CMS que ofrece el mercado. Estos conocimientos adquiridos me servirán para facilitar y agilizar mi aprendizaje en otras tecnologías CMS como *Drupal, Joomla, PrestaShop, WordPress…* que son el día a día de las empresas dedicadas al desarrollo de soluciones tecnológicas.

Para finalizar y como reflexión personal, con el desarrollo de este proyecto se pretende algo más que realizar una aplicación móvil. Se pretende hacer ver a los profesores que

no están habituados a utilizar la tecnología en las aulas, que los tiempos están cambiando y deben adaptarse a los recursos y posibilidades que esta nueva era trae si quieren realmente enseñar y conectar con sus alumnos. En una carrera como Telecomunicaciones resulta difícil pensar una clase sin tecnología, sin embargo, en un gran número de carreras se siguen pretendiendo enseñar con cientos de alumnos observando a distancia una tiza y una pizarra.

CAPITULO 7. Líneas futuras.

La aplicación desarrollada en este proyecto cubre las necesidades básicas y más importantes del cliente, sin embargo, teniendo en cuenta el trabajo desarrollado en este proyecto, se proponen las siguientes líneas futuras que ofrecer al cliente:

- Desarrollo del código necesario tanto en los servicios web del módulo cuestionario, como en la aplicación móvil, para que todos los tipos de preguntas disponibles en *Moodle 3.0* se puedan realizar desde la aplicación móvil.

- Desarrollo del código necesario tanto en los servicios web del módulo cuestionario, como en la aplicación móvil, para que el usuario pueda acceder desde la aplicación móvil a los intentos de cuestionarios ya realizados y obtener retroalimentación de estos intentos.

- Desarrollo de notificaciones que permitan al usuario estar informado de cuando un nuevo cuestionario es añadido a un curso.

- Desarrollo del código necesario tanto en los servicios web del módulo cuestionario, como en la aplicación móvil, para que los intentos tengan un límite de tiempo para desarrollarse.

- Modificaciones en la interfaz de usuario para que este tenga constancia del estado en el que se encuentran los diferentes cuestionarios sin la necesidad de terne que acceder a ellos.

- Desarrollo de todas las opciones que permiten configurarse en los cuestionarios desde la plataforma *Moodle* pero que no están soportadas en la aplicación: número de preguntas por página, intentos referidos a intentos anteriores, etc. (sección 1.2 “manual del profesor” del capítulo 4).

- Interfaz adaptable a tamaños de pantalla superiores a las de un *smartphone* como son las *tablets.*

- Internalización de la aplicación en diferentes idiomas.

- Desarrollo para que otros módulos como foros, mensajes… estén disponibles desde la aplicación móvil.

- Otra línea futura a considerar sería la de desarrollar esta aplicación en *iOS* para los usuarios de iPhone, esto atendería a las necesidades de bastantes usuarios que usan estos teléfonos.

- Mejorar la seguridad de la aplicación para que utilice el protocolo web de comunicaciones https en lugar de http, donde el intercambio de información va cifrado. Esto implicaría un pequeño cambio en la aplicación móvil y un cambio algo más importante en la parte del servidor.

BIBLIOGRAFÍA

Ableson F. (2013). Introducción al desarrollo en Android. Consultado el 7 de Abril de 2016 de https://www.ibm.com/developerworks/ssa/library/os-android-devel/

América Learning Media (2016). *Mobile Learning*, tomografía de una tendencia en expansión. Consultado el 15 de Marzo de 2016 de http://www.americalearningmedia.com/component/content/article/30-tendencias/144-mobile-learning-tomografia-de-una-tendencia-en-expansion

Appio (2013). Tipos de Apps. Consultado el 7 de Abril de 2016 de http://appio.es/tipos-de-apps/

Baelo, R. (2009). El e-Learning, una respuesta educativa a las demandas de las sociedades del siglo XXI. *Pixel-Bit. Revista de Medios y Educación*. Consultado el 3 de Marzo de 2016 de http://www.sav.us.es/pixelbit/pixelbit/articulos/n35/7.pdf

Campos Gutiérrez, E.M. (2015). Aplicación Android de comunicación a través de Moodle. Trabajo Fin de Grado. Valladolid: Universidad de Valladolid.

Capterra (2015). Top LMS Software. Consultado el 18 de Marzo de 2016 de http://www.capterra.com/learning-management-system-software/#infographic

Centro de Apoyo Tecnológico a Emprendedores (2012). Estudio de los sistemas de gestión de contenidos web. Albacete, España: Fundación Parque Científico y Tecnológico de Albacete

Developer Android (2016a). Supporting different languages. Consultado el 21 de Abril de 2016 de http://developer.android.com/training/basics/supporting-devices/languages.html

Developer Android (2016b). Async Task. Consultado el 25 de Abril de 2016 de http://developer.android.com/intl/es/reference/android/os/AsyncTask.html

Edmodo (2016). Sobre Edmodo. Consultado el 18 de Marzo de 2016 de https://www.edmodo.com/about

García, X (2004). Introducción a los sistemas de gestión de contenido de código abierto. Consultado el 9 de Marzo de 2016 de http://mosaic.uoc.edu/2004/11/29/introduccion-a-los-sistemas-de-gestion-de-contenidos-cms-de-codigo-abierto/

Github Moodle (2016). Repositorio Github de la comunidad Moodle. Consultado el 21 de Abril de 2016 de https://github.com/moodle/moodle/blob/master/mod/quiz/

JSON (s.f.). Introducción a JSON. Consultado el 5 de Abril de 2016 de http://www.json.org/json-es.html

Moodle (2012). Usando servicios web en Moodle. Consultado el 5 de abril de 2016 de https://docs.moodle.org/22/en/Using_web_services

Moodle (2015a). Sobre Moodle. Consultado el 15 de Marzo de 2016 de https://moodle.com/moodle-lms/

Moodle (2015b). Arquitectura. Consultado el 31 de Marzo de 2016 de https://docs.moodle.org/all/es/Arquitectura

Moodle (2015c). Características. Consultado el 31 de Marzo de 2016 de https://docs.moodle.org/all/es/Características

Moodle (2015d). Filosofía. Consultado el 31 de Marzo de 2016 de https://docs.moodle.org/all/es/Filosofía

Moodle (2015e). Paquetes de Idioma. Consultado el 31 de Marzo de 2016 de https://docs.moodle.org/all/es/Paquetesd_de_idioma

Moodle (2015f). Nuevas características de Moodle 3.0. Consultado el 31 de Marzo de 2016 de https://docs.moodle.org/all/es/Nuevas_características_de_Moodle_3.0

Moodle (2015g). Versiones de Moodle. Consultado el 4 de Abril de 2016 de https://docs.moodle.org/dev/Releases

Moodle (2015h). Manual de Estilo de Código. Consultado el 24 de Mayo de 2016, de https://docs.moodle.org/all/es/Manual_de_Estilo_de_Codigo

Moodle (2016). Servicios web en Moodle. Consultado el 5 de Abril de 2016 de https://docs.moodle.org/dev/Web_services

Moodle Mobile (2012). Configuración de Moodle Mobile en la plataforma Moodle. Consultado el 15 de Marzo de 2016 de https://docs.moodle.org/22/en/Mobile_web_services

Qode (2014). ¿Qué es una webApp?. Consultado el 5 de Abril de 2016 de http://qode.pro/blog/que-es-una-web-app/

Robertson, J (2003). So, what is a content management system?. Consultado el 10 de Marzo de 2016 de http://www.steptwo.com.au/papers/kmc_what/index.html

Ruiz, F. (1996). Herramientas tecnológicas para la realización de cursos por computador. Revista de Enseñanza y Tecnología, 5, pp. 21-31.

Telefónica (2014). Informe de la fundación telefónica. Consultado el 8 de Marzo de 2016 de http://www.fundaciontelefonica.com/2015/01/21/sie14-informe-sociedad-informacion-espana-pais-conectado-europa/.

World Wide Web Consortium (2005). Introducción a la accesibilidad Web. Consultado el 31 de Marzo de 2016 de http://www.w3c.es/Traducciones/es/WAI/intro/accessibility

World Wide Web Consortium (s.f.). Tecnología XML. Consultado el 5 de Abril de 2016 de http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML

ANEXO I. INSTALACIÓN DE UNA PLATAFORMA MOODLE 3.x.

En el siguiente anexo se desarrolla una guía detallada de como instalar una plataforma *Moodle* en un servidor Windows. Esta guía está pensada para instalar *Moodle* en una plataforma en producción ya que se explica de forma detallada algunos aspectos que para una plataforma en desarrollo no serían necesarios de controlar.

Si lo que desea es únicamente desplegar una plataforma de desarrollo vea la guía rápida de instalación de *Moodle*, en caso de querer desplegar una plataforma en producción o conocer en detalle la instalación de *Moodle* continúe con este anexo.

1. Requisitos

Moodle está desarrollado para utilizarse principalmente en GNU/Linux usando Apache, PostgreSQL / MySQL / MariaDB y PHP (también conocida como plataforma LAMP). Sin embargo en este caso instalaremos *Moodle* en un servidor Windows, con la consideración de que a partir de php5.5 en adelante se necesitará tener tener*Visual C++ Redistributable for Visual Studio 2012* instalado.

Los requisitos básicos de Moodle son los siguientes:

1.1 Hardware

- Espacio de disco: 200 MB para el código de Moodle, sin embargo, para almacenar el material que se subirá a la plataforma de un sitio en producción, se necesitarán entorno a 5GB.
- Procesador: 1GHz (mínimo), se recomienda 2GHZ doble núcleo o más.
- Memoria: 512 (mínimo), 1GB o más es altamente recomendado. La regla usual es que Moodle puede soportar de 10 a 20 usuarios concurrentes por cada 1GB de RAM, pero esto variará dependiendo de su combinación específica de *hardware* y *software*. Concurrente realmente significa procesos de servidor web en memoria al mismo tiempo (por ejemplo: usuarios interactuando con el sistema dentro de una ventana de unos pocos segundos). NO significa personas 'ingresadas al sitio'.

Todos los requisitos anteriores variarán dependiendo de las combinaciones del *hardware* y *software* específicos, además del tipo de uso y la carga; los sitios muy concurridos muy probablemente requerirán recursos adicionales.

1.2 Software

- Se necesitará un servidor web en funcionamiento, una base de datos y tener PHP configurado.
- Moodle requiere un cierto número de extensiones de PHP. Sin embargo, Moodle lo comprueba durante el proceso de instalación y se puede solucionar el problema y reiniciar el guión de instalación si falta alguna extensión.

Los requisitos de *software* dependen de la versión específica de Moodle que desea instalar, en este caso se expondrán los requisitos descritos en las notas de la versión de Moodle 3.0:

Requisitos del servidor web:

Versión mínima de PHP: PHP 5.4.4 (siempre utilizar el más reciente, PHP 5.4.x o 5.5.x en Windows - http://windows.php.net/download/) PHP 7 NO ESTÁ SOPORTADA

Requisitos de la base de datos:

Moodle soporta los servidores de Base de Datos descritos en la figura 72. Los números de versiones son la mínima versión soportada, pero siempre se recomienda correr la versión más reciente estable de cualquier *software.*

BasedeDatos	Versión mínima	Recomendada
PostgreSQL	9.1	La más reciente
MySQL	5.5.31	La más reciente
MariaDB	5.5.31	La más reciente
Microsoft SQL Server	2008	La más reciente
Oracle Database	10.2	La más reciente

Figura 72 - Bases de Datos soportadas por Moodle 3.0

Requisitos del cliente:

Moodle está soportado en los navegadores descritos en la figura 73.

Navegador	Versión mínima	Versión recomendada	Notas
Google Chrome	30.0	La más reciente	
Mozilla Firefox	25.0	La más reciente	
Apple Safari	6	La más reciente	
Microsoft Internet Explorer	9	La más reciente	Se necesita la versión 10 para arrastrar y soltar al subir contenidos desde afuera del navegador hacia el interior de Moodle

Figura 73 - Navegadores soportados por Moodle 3.0.

2. Configuración del servidor

Hay muchas posibilidades para instalar el *software* básico del servidor dependiendo de las elecciones particulares, en este caso se explicará cómo ha de ser la instalación del servidor web que distribuye Microsoft en sus sistemas operativos destinados a ofrecer servicios (Windows server) y otras distribuciones de Windows (Windows 7, Windows XP Pro, etc), este servidor web es IIS (Internet Information Services).

Instalación del servidor:

El servidor IIS en los sistemas operativos de Microsoft se instala como una característica de Windows y en función de la distribución de Windows que utilicemos esta instalación se realizará de diferentes formas. En Windows 7 se instala desde el panel de control, programas y características (figura 74) mientras que en las versiones Windows Server se instala desde el administrador del servidor, añadir roles y características (figura 75 y 76).

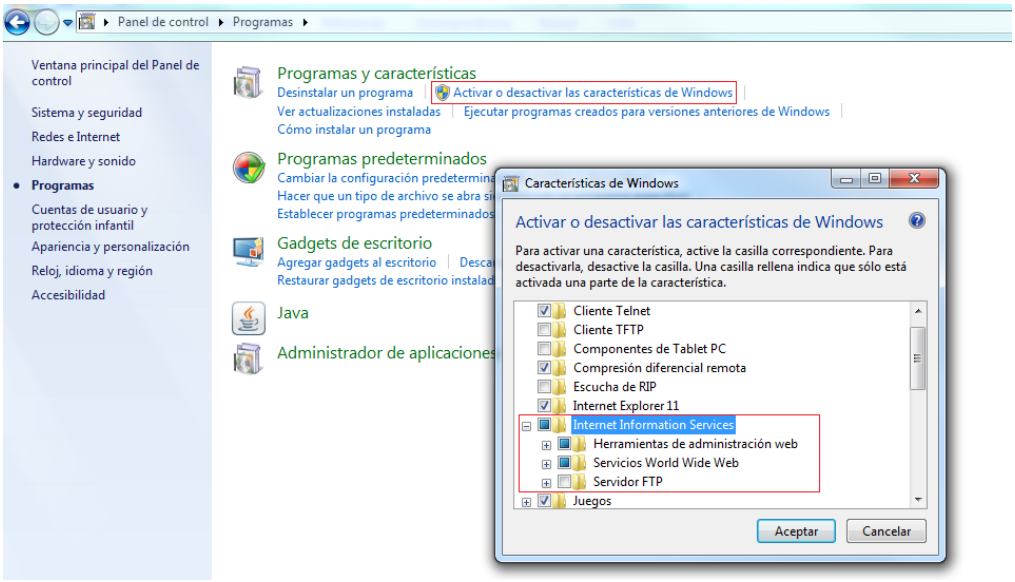

Figura 74 - Instalar IIS en Windows 7

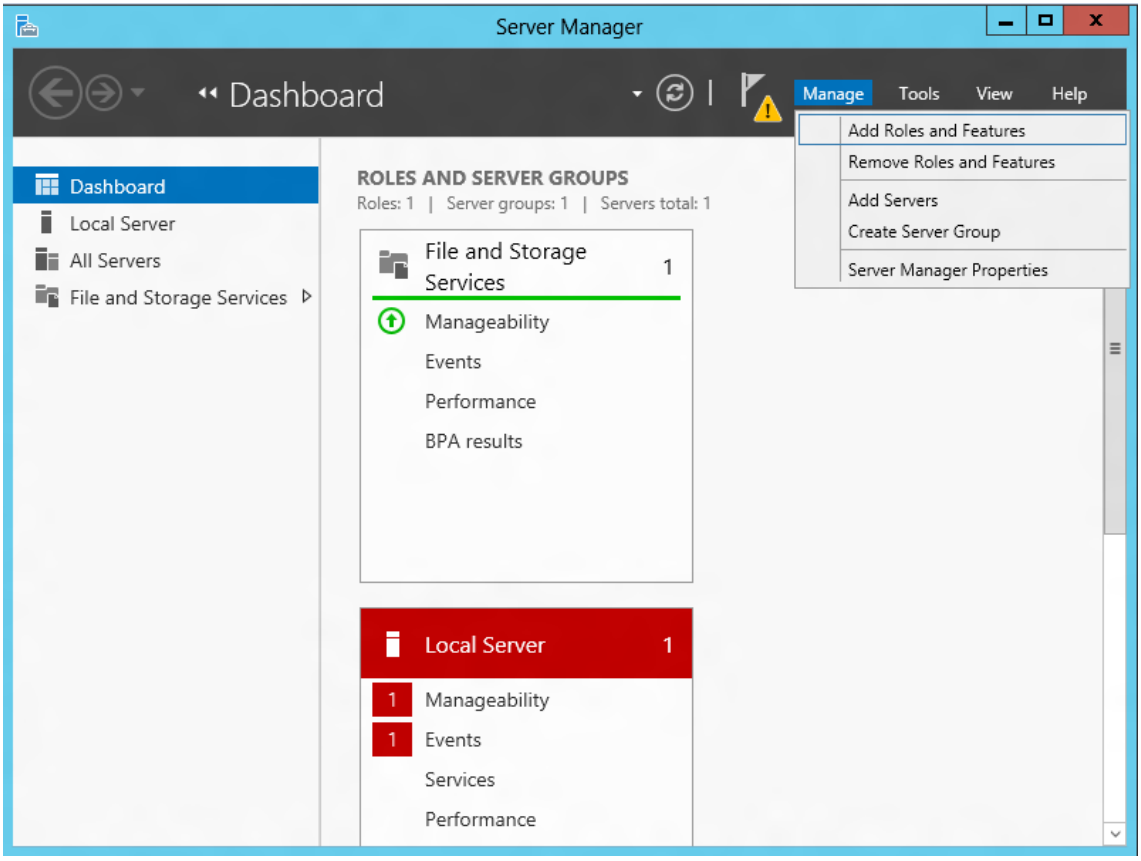

Figura 75 - Instalar IIS en Windows Server (I)

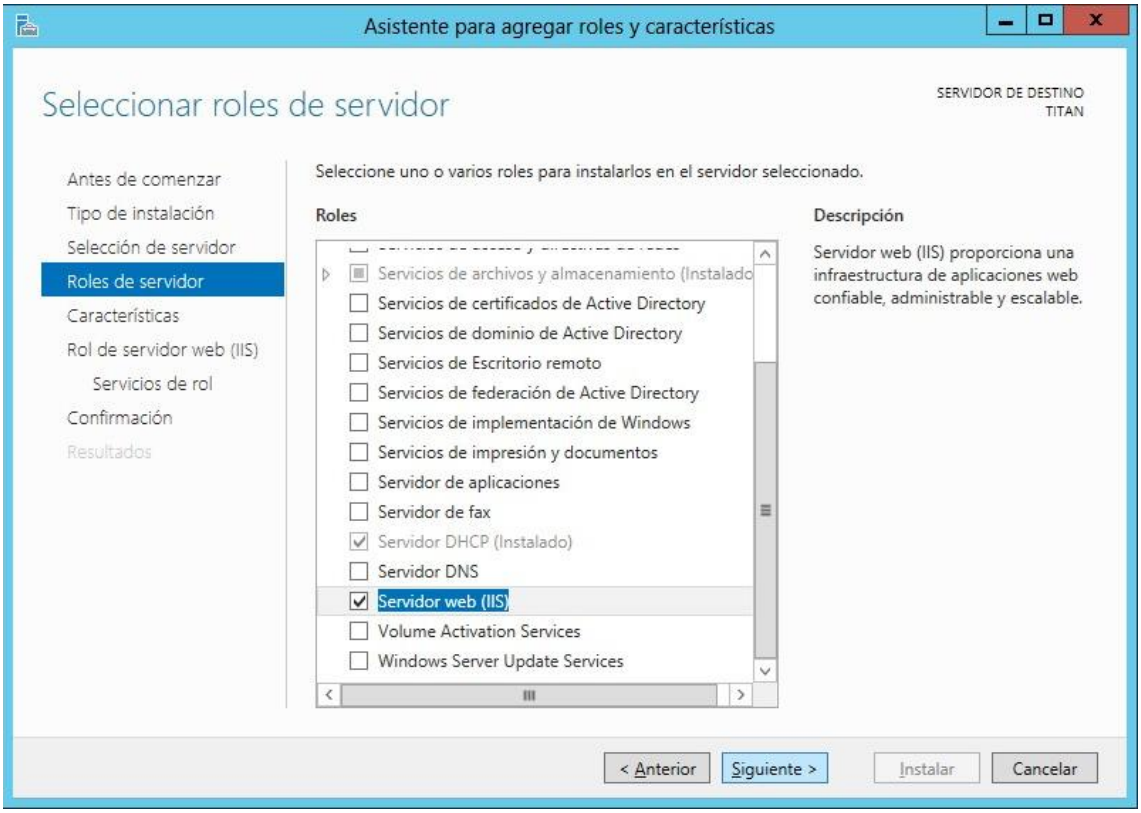

Figura 76 - Instalar IIS en Windows Server (II)

Instalación de PHP

Los pasos para instalar PHP de forma manual y tener un mayor control sobre lo que se instala y donde se instala son los siguientes:

- Descargar e instalar *PHP manager for IIS* que permitirá controlar en el IIS las versiones registradas de PHP, donde almacenar los logs de errores de PHP, etc.
- Descargar la versión que se desee de PHP y extraerla en el directorio deseado, en este caso la versión será PHP 5.5.1 y se extraerá en el directorio C:\PHP\.
- Instalar *Visual C++ Redistributable for Visual Studio 2012*.
- Abrir el *IIS Manager*.
- Acceder al *PHP Manager*.
- Registrar la nueva versión de PHP, seleccionando C:\PHP\php-cgi.exe

Otra posible solución es utilizar *Microsoft Web Platform Installer* que reduce el trabajo de tener que descargar y descomprimir manualmente las versiones de PHP, Visual C++ y tener que registrar la versión de PHP.

Configuración de PHP

Una vez instalado PHP se debe configurar este:

- Configurar los valores recomendados por el *PHP Manager*.

- Habilitar en el *PHP Manager* las extensiones requeridas: php_intl.dll, php_pgsql.dll.
- Habilitar la extensión OPcache.
- Configurar la *timezone* en el PHP.ini.
- Configurar los límites de memoria apropiados en el PHP.ini.

3. Descarga de *Moodle*

Una vez instalado el servidor IIS y Configurado PHP se deben descargar la versión deseada de *Moodle* desde el sitio oficial (https://moodle.org/downloads) y descomprimir esta en el directorio de documentos del servidor web, en este caso el sitio estará localizado en http://servidorweb/moodle.

Si se desea que la localización sea simplemente http://servidorweb debe copiar el contenido del directorio *moodle*, que se ha generado al descomprimir, directamente en el directorio principal de documentos del servidor web.

4. Instalación de una base de datos

Al igual que todos los CMS, *Moodle* utiliza una base de datos para almacenar la información, por lo que deberemos instalar en el mismo equipo que el servidor o en otro equipo diferente un gestor de base de datos de los indicados en la figura 72 de la sección requerimientos, vista con anterioridad.

Creación de una base de datos vacía

Una vez instalado el sistema gestor de base de datos deseado debemos crear una base de datos nueva, vacía, y tomar en nota la siguiente información que se utilizará durante la etapa final de la instalación:

- **dbhost:** El nombre del host del servidor de base de datos. Probablemente sea *localhost* si la base de datos y el servidor web son la misma máquina, en caso contrario debe ser el servidor de base de datos.
- **dbname:** El nombre de la base de datos. El nombre que deseemos poner (sin acentos ni letra ñ), por ejemplo: *Moodle.*

- **dbuser:** El nombre de usuario para la base de datos. Es recomendable no utilizar la cuenta de root/superusuario. Crearemos una cuenta con el mínimo de permisos necesarios.

- **dbpass:** la contraseña para el usuario que acabamos de crear.

5. Creación del directorio de datos (moodledata)

Moodle requiere un directorio donde almacenar todos sus archivos (archivos que los profesores suban, datos temporales, datos de la sesión, etc). El servidor web necesita poder escribir en este directorio. En sistemas grandes se debe considerar cuanto espacio libre se va a necesitar emplear al crear este directorio.

Debido a la manera en la que *Moodle* cachea los datos, nos podríamos encontrar con problemas si se utiliza un almacenamiento relativamente lento (por ejemplo, NFS) para este directorio.

Este directorio no debe ser directamente accesible vía web. Esto sería un agujero serio de seguridad. No se debe poner dentro de la raíz de la web ni dentro del directorio de archivos de *Moodle* o *Moodle* no se instalará.

6. Iniciar la instalación de Moodle

Durante la instalación de *Moodle,* se crearán las tablas de la Base de Datos y se configurará el sitio. El método recomendado es utilizar el instalador por línea de comandos, sin embargo, Windows no soporta este instalador por lo que se utilizará el instalador basado en web.

Para ejecutar el *script* del instalador web, únicamente hay que acceder a la URL principal de nuestro *Moodle* empleando un navegador web.

Durante el proceso de instalación se mostrarán un número de páginas en las cuales se le pedirá que confirme el *copyright*, que proporcione información para acceder a la Base de Datos, que vea que se crean las tablas de la Base de Datos, que proporcione detalles para la cuenta del administrador principal y los valores por defecto del sitio, etc.

Si durante la instalación se produce algún error, siga las instrucciones que se le indicarán para solucionar estos errores.

7. Finalizar la instalación

Una vez terminada la instalación, se nos situará en la portada de *Moodle*, donde con la cuenta de administrador ya podremos registrar a nuevos usuarios con diferentes roles, crear cursos y matricular a cada usuario en los cursos correspondientes.

ANEXO II. ESTILO DE CÓDIGO Y NORMAS DE MOODLE.

1. 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 deberá adherirse a estos estándares de la forma más exacta posible (Moodle, 2015h).

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 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 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 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 notificar al 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 los magic_quotes aplicadas (sin importar la configuración de PHP) por lo que se pueden insertar con total seguridad en la base de datos. El resto de la información (obtenida desde los archivos, o desde la base de datos) deber 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 de 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 tiene 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 deben 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 está 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 la 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 moduloid.

- Los campos booleanos serán implementados como enteros cortos (por ejemplo, INT4) con los valores 0 y 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 este 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.

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

- 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

A continuación se enuncian diferentes normas de seguridad que deben llevarse a cabo a la hora de desarrollar código en *Moodle*:

- No basarse en register_globals. Cada variable de ser correctamente inicializada.

- Inicializar 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 require_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().