Universidad de Valladolid

Escuela Técnica Superior
Ingenieros de Telecomunicación

Trabajo de fin de grado

Grado en Ingeniería de Tecnologías
Específicas de Telecomunicación
Mención en Ingeniería Telemática

Diseño y desarrollo de aplicación de aprendizaje basado en juegos eLiza para iOS

Autor: Jorge Saúl Alonso Buitrago
Tutora: Míriam Antón Rodríguez

TÍTULO: "DISEÑO Y DESARROLLO DE LA APLICACIÓN DE APRENDIZAJE BASADO EN JUEGOS ELIZA PARA IOS"

AUTOR: JORGE SAÚL ALONSO BUITRAGO

TUTORA: MÍRIAM ANTÓN RODRÍGUEZ

DEPARTAMENTO: TEORÍA DE LA SEÑAL, COMUNICACIONES E INGENIERÍA TELEMÁTICA

Miembros del Tribunal

PRESIDENTE: MÍRIAM ANTÓN RODRÍGUEZ

SECRETARIO: DAVID GONZÁLEZ ORTEGA

VOCAL: MARIO MARTÍNEZ ZARZUELA

SUPLENTE: FRANCISCO JAVIER DÍAZ PERNAS

SUPLENTE: Mª ÁNGELES PÉREZ JUÁREZ

FECHA DE LECTURA: 18 DE JULIO DE 2016

CALIFICACIÓN:

Resumen del proyecto

Este Trabajo Fin de Grado nace de la necesidad del alumnado de un entorno más accesible y flexible en la realización de tareas de aprendizaje y evaluación. Con los grandes avances tecnológicos disponibles hoy en día que derivan de Internet y el gran número de dispositivos que se manejan a diario, la enseñanza ha sufrido un proceso de cambio. En la actualidad, gracias a los ordenadores, dispositivos móviles y el uso que éstos hacen de Internet, la enseñanza ha avanzado hacia un modelo más cómodo para los docentes, más instructivo para el alumnado y más accesible para todos.

eLiza es una herramienta de aprendizaje y evaluación basada en juegos, desarrollada para el gestor de contenidos educativos Moodle por la Universidad de Valladolid, y que se mantiene en uso.

El desarrollo de este Trabajo Fin de Grado ha tenido como actividad principal la adaptación de este herramienta para su uso en dispositivos móviles, permitiendo hacer uso de todas sus funcionalidades desde estos dispositivos y añadiendo alguna herramienta adicional.

Desde el punto de vista del desarrollador, esto se traduce en la inserción, obtención y modificación de datos alojados en las bases de datos de eLiza, ya existentes, mediante la definición de un servicio web haciendo uso de la tecnología REST.

La versión móvil de eLiza no se plantea como un sustituto de la versión web de Moodle, si no que convive con ella y la complementa al otorgar mayor facilidad de acceso a sus usuarios.

El objetivo principal de esta aplicación es el de servir de apoyo a la metodología de aprendizaje de las asignaturas impartidas por el Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática de la Universidad de Valladolid y también el de diversificar los métodos de aprendizaje disponibles para los alumnos.

Palabras clave

iOS, eLiza, Swift, eLearning, Moodle, servicio web

Abstract

This project was born from the need of students of more accesible and flexible platforms in the way of learning and evaluation. Due to the technological advances available today derived from Internet and the many devices that are handled every time, learning process has changed. Nowadays, thanks to computers, mobile devices and the use that they do of Internet, the teaching has evolve to be more comfortable for teachers, more instructive for students and more accesible for everyone.

eLiza is a learning and evaluation tool developed for the educational content manager called Moodle by University of Valladolid. This tool is still operative.

The development of this end of grade work aims at the adaptation of the eLiza tool to improve its use in mobile devices, allowing users to get same functionalities in this devices.

From the developer point of view, it means the data insertion, obtaining and updating that are hosted in the already existing eLiza databases, defining a REST web service.

eLiza mobile application is not propose as Moodle web version replacement, it coexist with it and complements it due to easier users access.

The application goal is to provide support to the learning methodology of the subjects teaching by the Signal Theory, Communications and Telematics Engineering Department of University of Valladolid and it is also aimed to diversify the learning methods that are available for students.

Key words

iOS, eLiza, Swift, eLearning, Moodle, web service

AGRADECIMIENTOS

A mis padres y mi hermana por su apoyo incondicional.

A mis compañeros y amigos por hacer de estos años una experiencia extraordinaria.

Y a mi tutora, Míriam, por haberme guiado y ayudado en la realización de este Trabajo.

Gracias.

"La mejor manera de predecir el futuro es inventarlo."
Alan Kay

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1: Introducción general

La creación de Internet ha supuesto un cambio evolutivo en la forma, lugar y duración del desempeño de nuestras actividades, tanto laborales como de ocio y educativas. Hoy en día nos encontramos con que muchas de las aplicaciones Web que usábamos años atrás, antes de la irrupción de los smartphones, tienen su versión móvil.

El principal motivo del proceso de movimiento hacia plataformas móviles es la accesibilidad que confiere a estos servicios. El usuario puede acceder desde su terminal móvil independientemente de su localización a servicios web adaptados específicamente para terminales móviles en aplicaciones.

La educación no es un campo ajeno a esta apremiante actualización de los canales de comunicación con el público. En la actualidad la Universidad de Valladolid ya dispone de ciertas aplicaciones móviles para la publicación de información, consulta de notas o biblioteca. Con la realización de este Trabajo Fin de Grado, la Universidad de Valladolid sigue extendiendo la presencia de los contenidos web en terminales móviles.

Este Trabajo Fin de Grado parte del módulo eLiza desarrollado para la plataforma Moodle como un método de aprendizaje y evaluación basado en juegos. Habitualmente, se conoce como eLearning a la filosofía de aprendizaje a través de Internet. En ella participan multitud de plataformas, entre las cuales destaca como la más utilizada Moodle.

Definimos eLearning como el conjunto de procesos de enseñanza llevados a cabo a través de Internet, en los cuales el docente y el alumnado no tienen contacto físico pero sí están comunicados (Barberá, 2008).

1.1 Objetivos

Con la realización de este Trabajo Fin de Grado se pretende trasladar el módulo de eLiza existente en Moodle a dispositivos móviles con sistema operativo iOS, de modo que sea más accesible y flexible para los alumnos y confiera una motivación extra al alumno en su proceso de aprendizaje en línea. Se plantea este desarrollo como un añadido, no como un sustituto de la plataforma web actual, de modo que la aplicación móvil y la plataforma web puedan convivir y ser utilizadas indistintamente por el usuario. Para ello, se han trasladado las funciones existentes en la plataforma web a la aplicación, y se ha añadido una nueva funcionalidad de clasificación ajustada al curso y asignatura del alumno, con el fin de fomentar el uso de la aplicación.

1.2 Fases y métodos

Para lograr los objetivos presentados en el punto anterior, el proyecto consta de las siguientes fases:

- Fase de documentación, en la que se estudian las principales características de la plataforma de eLearning Moodle, del funcionamiento del módulo para Moodle eLiza y de las posibilidades de implantación de APIs basadas en servicios web.
- Fase de iniciación y familiarización con el lenguaje de programación Swift y con el entorno de desarrollo Xcode, y documentación de las posibilidades y funcionalidades que ofrece este lenguaje.
- Desarrollo en Swift mediante la herramienta de desarrollo Xcode, de la interfaz gráfica y las funcionalidades de la aplicación en conexión con la plataforma Moodle y el módulo eLiza.
- Realización de pruebas de la aplicación completa en terminal real y depuración de fallos.
- Extracción de conclusiones y establecimiento de futuras líneas de desarrollo.

1.3 Medios

Para la realización del proyecto se han tomado como referencia otros Trabajos Fin de Grado y Proyectos Fin de Carrera, relacionados con el desarrollo de plataformas web y aplicaciones móviles.

También se han consultado artículos y libros que analizan el desarrollo de aplicaciones en las distintas plataformas, así como la utilización de tecnología en el campo de la educación.

Para la implementación y las pruebas a realizar, se dispone de los siguientes equipos y programas:

- Ordenador MacBook Pro Intel Core 2 Duo @ 2,53GHz y 4 GB de RAM. Sistema Operativo Mac OS, versión 10.11.5 “El Capitan”. Empleado para el diseño y desarrollo de la aplicación y para la implantación del servidor web.
- Teléfono móvil iPhone 6 procesador A8 de 64 bits y 1 GB RAM. Sistema operativo iOS, versión 9.3. Utilizado para realizar pruebas de ejecución de la aplicación.

- Xcode 7.3: Entorno de Desarrollo creado por Apple Inc. para el desarrollo de aplicaciones para sistemas operativos Mac OS X, iOS y tvOS.
- Sublime Text 3: editor de texto utilizado para la edición de ficheros PHP en la parte del servidor.
- Servidor MySQL versión 14.14 distribución 5.5.48 para Mac OS X.
- Servidor Apache versión 2.4.18.

1.4 Estructura del documento

El documento está organizado de la siguiente manera:

- En el primer capítulo, se realiza una introducción sobre el contenido del trabajo y los objetivos, fases, metodologías y medios que se han empleado para la realización del mismo.
- En el segundo capítulo, se estudia la aplicación de aplicaciones móviles en el proceso de aprendizaje.
- En el tercer capítulo, se estudian las diferentes tecnologías disponibles que pueden utilizarse para la realización del Trabajo.
- En el cuarto capítulo se explica el método empleado en la resolución del problema y las características técnicas de la solución.
- En el quinto capítulo se documenta un manual de usuario que describe las distintas acciones que se pueden realizar con la aplicación.
- En el sexto capítulo se realiza un estudio económico de lo que supone la realización del trabajo.
- En el capítulo séptimo, se exponen las conclusiones del proyecto y las líneas futuras.
- Por último, en el capítulo octavo se recoge la bibliografía empleada para la realización del proyecto.

Capítulo 2: Aplicaciones móviles en aprendizaje

2.1 eLearning

El e-learning consiste en la educación y capacitación a través de Internet. Este tipo de enseñanza en línea permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas (Barberá, 2008).

Este nuevo concepto educativo es una revolucionaria modalidad de capacitación que posibilitó Internet, y que hoy se posiciona como la forma de capacitación predominante en el futuro. Este sistema ha transformado la educación, abriendo puertas al aprendizaje individual y organizacional. Es por ello que hoy en día está ocupando un lugar cada vez más destacado y reconocido dentro de las organizaciones empresariales y educativas.

Las grandes ventajas del eLearning son:

- Reducción de costos: permite reducir y hasta eliminar gastos de traslado, alojamiento, material didáctico, etc.
- Rapidez y agilidad: Las comunicaciones a través de sistemas en la red confiere rapidez y agilidad a las comunicaciones.
- Acceso just-in-time: los usuarios pueden acceder al contenido desde cualquier conexión a Internet, cuando les surge la necesidad.
- Flexibilidad de la agenda: no se requiere que un grupo de personas coincidan en tiempo y espacio.

La diferencia fundamental entre el eLearning y la enseñanza tradicional a distancia está en la combinación de tres factores esenciales: el seguimiento, el contenido y la comunicación (en proporción variable y en función de la materia a tratar en cada caso). A la hora de aplicar y gestionar de forma virtual estos tres elementos o factores, puede resultar interesante conocer a priori las diferencias entre los principales Sistemas de Gestión de Contenidos con los que podemos contar: CMS (Content Management Systems), LMS (Learning Management Systems) y LCMS (Learning Content Management Systems).

2.2 CMS

Un Sistema de Gestión de Contenido (Content Management System, en inglés) es un software que permite la creación y administración de los contenidos de una página Web, principalmente, de forma automática. Así, con él se puede publicar, editar, borrar, otorgar permisos de acceso o establecer los módulos visibles para el visitante final de la página (Cañellas Mayor, 2010). El CMS está formado por 2 elementos:

- La aplicación gestora de contenidos (CMA): El elemento CMA permite al gestor de contenidos o autor realizar la creación, modificación y eliminación de contenido en un sitio Web sin necesidad de tener conocimientos de lenguaje HTML.
- La aplicación dispensadora de contenidos (CDA): El CDA usa y compila la información para actualizar el sitio Web.

2.3 LMS

Un Sistema de Gestión de Aprendizaje (Learning Management System) es un software que automatiza la administración de acciones de formación. Son variadas las funcionalidades de un LMS: registra a todos los actores que intervienen en el acto de aprendizaje (alumnos, profesores, administradores, etc.), organiza los diferentes cursos en un catálogo, almacena datos sobre los usuarios, realiza un seguimiento del aprendizaje y la temporización de los trámites y genera informes automáticamente para tareas de gestión específicas. También desarrolla procesos de comunicación, e incluso algunos LMS permiten posibilidades de autoría de contenidos (Cañellas Mayor, 2010).

La mayoría de LMS están basadas en la Web para facilitar en cualquier momento y en cualquier lugar y a cualquier ritmo el acceso a los contenidos de aprendizaje y administración.

Un LMS proporciona por tanto:

- Tecnologías transmisivas (por ejemplo mediante una presentación que se agregue al curso).
- Tecnologías interactivas (mediante cuestionarios, actividades, etc.).
- Tecnologías colaborativas (mediante foros, wikis, etc.).

Algunos ejemplos de LMS son:

- ATutor: proyecto distribuido de forma libre por el departamento Adaptative Technology Resource Centre (ATRC) de la Universidad de Toronto. Utiliza las tecnologías PHP, Apache y MySQL. Entre sus características principales están el soporte de estándares, facilidades de importación/exportación de contenidos y usuarios, herramientas de

seguimiento, posibilidad de gestionar los contenidos de los cursos, es fácil crear contenidos y tests dentro de la plataforma e importar paquetes SCORM 1.2 e IMS.

- Bazaar: proyecto de libre distribución desarrollado por la Universidad de Athabasca, Canadá. Comenzó como un sistema de "web board conferencing" para evolucionar después hacia un sistema integrado de información. Se trata de un sistema flexible y configurable. Utiliza las tecnologías Apache, Perl y MySQL. Su gran ventaja es el alto grado de personalización, gracias a la utilización de plantillas configurables para gestionar todo tipo de contenidos.
- Claroline: el proyecto fue iniciado por la Universidad Católica de Lovaina, Bélgica, basándose en la experiencia pedagógica de los docentes y en base a sus necesidades. Más adelante surgió el Consorcio Claroline, fundado por varias universidades de Bélgica, España, Canadá y Chile, para coordinar el desarrollo de la plataforma y promover su uso. Se trata de uno de los LMS más utilizados en el mundo. Utiliza las tecnologías Apache, PHP y MySQL. Su gran ventaja es la facilidad de uso, su navegación es intuitiva y la administración completa.

2.4 LCMS

Un Sistema de Gestión de Contenidos de Aprendizaje-LCMS (Learning Content Management System, en inglés) es una aplicación de software que combina las capacidades de gestión de cursos de un LMS con las capacidades de almacenamiento y creación de contenidos de un CMS. Los LCMS se acercan a la denominación en castellano de "campus virtuales". Permite la creación y el desarrollo eficiente de contenidos para el aprendizaje proporcionando las herramientas necesaria a autores, diseñadores instruccionales y expertos del tema (Primeau, 2012).

Normalmente se crean partes de contenido en forma de módulos que se pueden personalizar, manejar, y que se pueden usar en diferentes ocasiones (son los denominado objetos de aprendizaje u OAs). Así, en lugar de apoyar el desarrollo totalizador de cursos, lo que hace un LCMS es ayudar a diseñadores instruccionales a crear módulos o bloques de contenido reutilizable (OAs), que luego se distribuirán según convenga en cada caso, por los diseñadores de los cursos.

Un LMS y un LCMS, aunque complementarios, son dos sistemas muy diferentes que sirven para distintos propósitos. Podemos decir que un LMS es un software que planifica y gestiona los eventos de aprendizaje dentro de una organización, incluyendo el aula en línea o virtual y los cursos dirigidos por un instructor. Por el contrario, un LCMS es un software para la gestión de contenidos

de aprendizaje de los diversos programas de capacitación que se configuran en el desarrollo en toda la organización. En este segundo caso, se proporciona a los desarrolladores, autores, diseñadores instruccionales y expertos en la materia los medios para crear y reutilizar el contenido de aprendizaje y reducir la duplicación de los esfuerzos de desarrollo, ya que un LCMS crea, almacena, ensambla y entrega de forma personalizada el contenido en forma de objetos de aprendizaje específicos.

Un LCMS añade al concepto de LMS una particularidad propia de los CMS: el hecho de poder administrar todos los contenidos del sistema.

Algunos ejemplos de LCMS son:

- AContent: se trata de la variante LCMS de ATutor. También desarrollado por la Universidad de Toronto, se trata de un gestor y repositorio de contenidos para la enseñanza basado en la web. Utiliza las tecnologías Apache, PHP y MySQL. Su gran ventaja es la interoperabilidad con cualquier sistema que soporte contenido IMS (estándar técnico que define el intercambio de datos entre sistemas de eLearning).
- Moodle: se trata de uno de los LCMS más populares y utilizados. Su comunidad de usuarios y desarrolladores es muy numerosa, lo que gracias a su sistema de libre distribución, permite una constante evolución de la plataforma. Utiliza las tecnologías Apache, PHP y MySQL (Moodle, 2016).

De todas las alternativas, sin duda el de mayor éxito actualmente es Moodle. Además, se trata del sistema utilizado por la Universidad de Valladolid para su Campus Virtual, por lo que será el sistema en el que se base la aplicación a desarrollar.

2.5 Moodle

Moodle es una plataforma de enseñanza diseñada para porporcionar a educadores, administradores y estudiantes un sistema robusto, seguro e integrado para crear entornos de enseñanza personalizados (Moodle, 2016).

Fue creado por Martin Dougiamas, administrador WebCT en la Universidad Tecnológica de Curtin, Australia, investigando un nuevo modo de enseñanza en línea. En 1999 comenzó a desarrollar simples prototipos de un nuevo sistema LCM, en los que basó el artículo “Aumentando la eficacia de la enseñanza en línea”. También registró el término Moodle como marca registrada del grupo Moodle Trust. Dougiamas basó su diseño en las ideas del constructivismo en pedagogía, teoría según la cual el conocimiento se ha de desarrollar en la mente del estudiante, no siendo igual de eficaz el conocimiento adquirido a través de libro o enseñanzas magistrales. Según estas ideas, la misión del profesor sería crear un entorno

centrado en el estudiante que le ayudara a construir ese conocimiento con base en sus habilidades y conocimientos previos.

Moodle 1.0 fue lanzado en 2002. Los usuarios estaban utilizando Moodle en un nuevo foro, traduciendo Moodle a diferentes idiomas y creando temas. Un año después, se desarrolló el primer módulo creado por usuarios y Moodle.org se convirió en el gran punto de unión entre la comunidad de Moodle, con Moodle.com representando el aspecto comercial.

En 2004, Oxford realizó el primer debate acerca de Moodle y las compañías comenzaron a solicitar convertirse en socios de Moodle.

El éxito de Moodle se basaba en la facilidad de uso, para lo cual bastaba con tener habilidades básicas para navegar por Internet. Además, para instalarlo tampoco se necesitaban muchos conocimientos, ya que una guía oficial facilitaba mucho la labor.

Moodle es un sistema para el Manejo del Aprendizaje en línea gratuito, que permite a educadores crear su propio sitio web privado. Algunas de sus características son:

- Interfaz de fácil utilización: la interfaz de Moodle fue diseñada para ser accesible y fácil de navegar.
- Tablero Personalizado: permite adaptar y organizar los cursos, mensajes y tareas del modo que se desee.
- Actividades y herramientas colaborativas: facilita el trabajo colectivo mediante la participación de los usuarios.
- Calendario: permite mantener al día el calendario académico, mostrando fechas importantes para el curso, como fechas de entrega, reuniones u otros eventos.
- Gestión de ficheros: mediante gestos de arrastre, permite subir ficheros locales o de plataformas de almacenamiento en la nube.
- Editor de texto: simple e intuitivo, permite dar formato y añadir elementos multimedia mediante un editor que funciona en cualquier navegador o dispositivo.
- Monitorización del progreso: tanto los profesores como los alumnos pueden seguir el progreso de actividades o recursos.

En 2010 se lanzó la versión 2.0 de Moodle, la cual introdujo mejoras en las características básicas de la primera versión, y corrigió un gran número de errores de la versión anterior. Algunas de las nuevas características fueron:

- Community Hubs: directorio de cursos para uso público o por una comunidad privada. El código se distribuye como un módulo externo de Moodle. Cualquier plataforma basada en Moodle puede registrarse en uno de estos directorios, y compartir en él sus cursos.
- Soporte de repositorios: se mejora notablemente la administración de ficheros, tanto su funcionalidad como su interfaz. Moodle se integra con

repositorios externos de contenido. Añade también atributos, como licencia y autor, para los ficheros.

- Los datos de módulos pueden exportarse a sistemas externos, creando un portafolio en diversos formatos (HTML, LEAP2A, Imágenes y Texto).
- Los profesores pueden especificar una condición estándar para completar un curso que se aplica a todos los alumnos. De igual modo, pueden fijarse estas condiciones como prerrequisitos para otros cursos.
- Actividades condicionales: se puede restringir el acceso a ciertas actividades basándose en fechas, calificaciones, o la realización de otra actividad.
- Soporte para servicios web: se añade soporte para los servicios web estándares en todo el código de Moodle, permitiendo al administrador publicar funcionalidades de Moodle para su uso por aplicaciones externas. La seguridad se gestiona mediante un sistema de token y un control total de los permisos sobre las funciones expuestas. Todas las funciones son accesibles vía XML-RPC, AMF, REST y SOAP.
- Mensajes: todos los emails enviados a través de Moodle se tratan como un mensaje. El usuario puede acceder a un panel de control con los mensajes que ha recibido.
- Permisos: se simplifica la lógica de evaluación de permisos y se definen interfaces AJAX simplificadas para la creación y asignación de roles.

En 2015 se lanza la versión 3.0 de Moodle, la cual incluye mejoras como:

- Encuestas: se mejora el sistema de encuestas.
- Simplificación del sistema de etiquetado: agiliza el cambio de nombre, y estado de etiquetas y permite el autocompletado.
- Herramientas externas para cursos: permite añadir herramientas externas como actividades de un curso simplemente indicando la URL del recurso.
- Nuevo editor de texto más completo y con más funcionalidades.

2.6 mLearning

El mLearning se define como una metodología de enseñanza y aprendizaje valiéndose del uso de pequeños dispositivos móviles, tales como por ejemplo: teléfonos móviles, tabletas y todo dispositivo de mano que tenga alguna forma de conectividad inalámbrica.

El mLearning puede ser visto como un subconjunto del eLearning. El eLearning es el concepto macro que incluye los entornos de aprendizaje móvil y en línea. De este modo, puede simplificarse la definición de mLearning como un proceso de eLearning a través de dispositivos móviles.

El mobile learning o m-learning tiene ventajas pedagógicas sobre otros modelos educativos, incluso sobre su predecesor e-learning. Entre las ventajas principales radican:

- Mayor libertad y flexibilidad de aprendizaje: no depende del acceso a un ordenador, si no a un dispositivo más portable.
- Utilización de juegos de apoyo en el proceso de formación: es muy común generar juegos para móviles que impulsen la colaboración en la actividad formativa.
- Navegación sencilla y adaptación de contenidos a los requisitos de este tipo de dispositivos.
- Acceso inmediato a datos y avisos: los usuarios pueden acceder de forma rápida a mensajes, correos, recordatorios y noticias generados en tiempo real.

En los años 80, el Grupo de Investigación de Aprendizaje de Xerox Palo Alto Research Center (PARC) propuso el Dynabook, una computadora del tamaño de un libro, portátil, con red inalámbrica y pantalla plana.

En la década de los 90, las primeras Universidades europeas y asiáticas comienzan a desarrollar y probar el aprendizaje móvil para los estudiantes. La Empresa Palm ofreció ayudas a las universidades y a las empresas que crearan y probaran el uso de Mobile Learning en la plataforma PalmOS. También se crea el primer móvil de módulos de aprendizaje con certificación Microsoft.

A partir del año 2000 La Comisión Europea financia las grandes empresas nacionales MOBIlearn M-Learning para creación de proyectos de desarrollo de contenidos.

2.7 Serious Games

Constituyen un sistema de aprendizaje online, muy de moda en la actualidad. Conocidos también con el nombre de "juegos formativos", son juegos diseñados para un propósito principal distinto del de la pura diversión, orientándose fundamentalmente hacia fines sociales como la educación, la exploración científica o sanitaria, la planificación cívica, la ingeniería, etc.

El desarrollo e integración de las TIC en los entornos educativos contribuye al soporte de las metodologías activas de aprendizaje basadas en la interacción entre docentes y discentes. Aunque el uso de las TIC para la enseñanza y el aprendizaje solo explotan una parte de su potencial, su aplicación ofrece un significativo soporte para las metodologías activas, mediante el aprendizaje basado en juegos (Game Based Learning). Los Serious Games constituyen potentes entornos para la mejora de la motivación e implicación de los participantes, básicamente, en el contexto específico de los juegos multi-jugadores. Destacan las dinámicas de cooperación y competición que pueden tener lugar en estos contextos y, el potencial de combinación de las dinámicas de cooperación intragrupal que favorecen la interdependencia y transferencia de conocimientos, junto a las dinámicas de competición intergrupal que promueven la implicación y reto en el uso de los juegos (Peter Vorderer, 2009).

Como posibles beneficios de los serious game, se pueden identificar:

- Que los cada vez más numerosos usuarios de videojuegos están sobradamente familiarizados con los procedimientos de trabajo de la plataforma.
- Que la utilización de las infraestructuras existentes para el desarrollo de los videojuegos reducen ostensiblemente el desarrollo de los propios serious game.
- Que los juegos serios no requieren de grandes soportes informáticos, exactamente igual que los videojuegos tradicionales. Su distribución se limita a enviarlos por correo o permitir su acceso mediante un sitio web dedicado.
- Y, por último, que al tiempo que los juegos serios están pensados para formar o educar a los usuarios, lo están también para entretener.

2.8 eLiza

Ilustración 1: Logo de eLiza

eLiza es un módulo para la plataforma Moodle desarrollado por el Grupo de Telemática e Imagen de la Universidad de Valladolid. Se trata de un juego de cuestionarios que se presentan como actividades de la plataforma virtual Moodle para la autoevaluación y adquisición de conocimientos por parte de los alumnos, y la evaluación y seguimiento por parte de los profesores (Romero Oraá, 2014).

Para ello, hace uso de un amplio abanico de tecnologías, librerías y plataformas para el desarrollo web (HTML5, CSS3, JavaScript, PHP, MySQL, XML, Moodle y Ajax). Con ello muestra una interfaz gráfica basada en HTML y CSS, ejecuta las acciones con código PHP y JavaScript y almacena toda la información en una base de datos MySQL.

Ilustración 2: Página principal de juego eLiza

El menú principal del módulo en la vista del alumno permite jugar a contestar cuestionarios, bien individualmente o en grupo, proponer preguntas para que se incluyan en el cuestionario o visualizar una serie de estadísticas respecto a las veces que se ha jugado.

En la vista como profesor, el menú permite añadir preguntas a los cuestionarios, así como visualizar unas estadísticas más extensas con datos de todos los alumnos para realizar un seguimiento de la realización de encuestas por los alumnos.

Capítulo 3: Estudio de las diferentes tecnologías

El mercado de aplicaciones móviles está en continuo aumento desde la aparición de los denominados teléfonos móviles inteligentes o smartphones. Cada día salen al mercado nuevas y novedosas aplicaciones que responden a nuevas necesidades de los millones de usuarios a nivel global que utilizan sus dispositivos móviles.

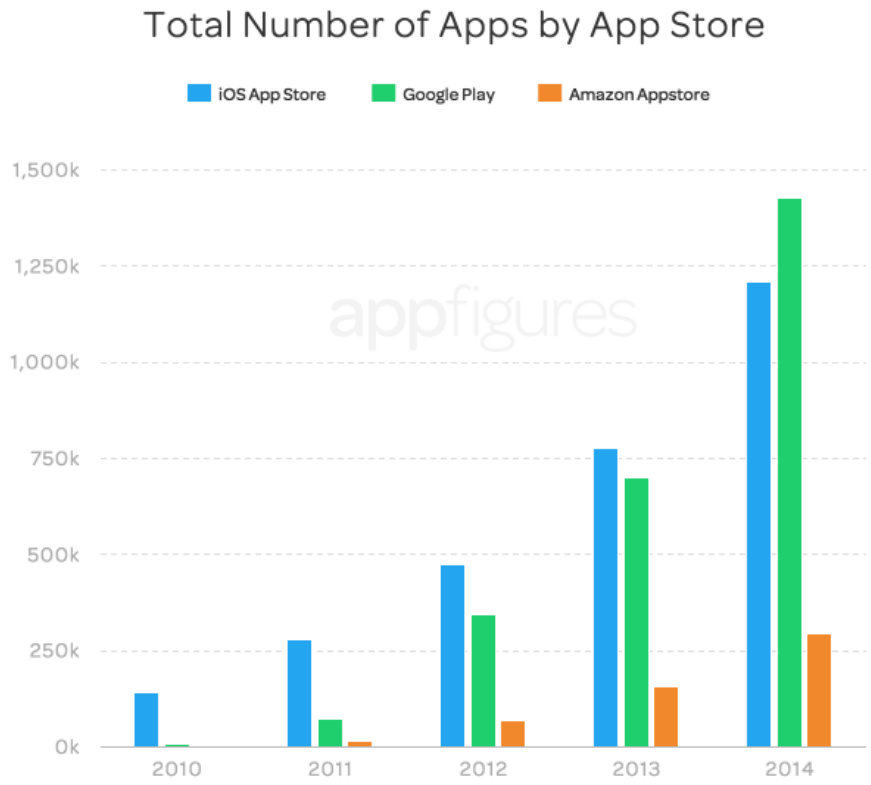


Ilustración 3: Comparativa del número de aplicaciones publicadas en las tiendas de los distintos sistemas operativos móviles (appfigures, 2015)

Los dispositivos móviles están controlados por diversos sistemas operativos, los cuales se detallarán más adelante, explicando las diferencias entre ellos.

A la hora de desarrollar una aplicación móvil, se puede optar por hacerlo utilizando tecnologías web, híbridas o nativas.

En este proyecto, en el describiré una aplicación móvil enfocada al mundo de la educación, más concretamente a la plataforma de aprendizaje en línea Moodle,

explicaré las ventajas e inconvenientes del desarrollo de aplicaciones según las distintas tecnologías y en función del sistema operativo y finalmente, expondré los motivos que me han llevado a programar mi aplicación con una tecnología nativa y para el sistema operativo iOS.

3.1 Tipos de aplicaciones

Existen diversos sistemas operativos móviles en el mercado mundial. El mercado está dominado por Android, el sistema operativo móvil creado por Google, y por iOS, sistema operativo móvil creado por Apple. Por detrás de ellos quedan varios sistemas (Windows Phone, Java ME, Symbian, Blackberry, etc.) con mucha menos penetración en el mercado, bien por una sustancial reducción de su popularidad ante la salida de sistemas más avanzados, como es el caso de Java ME o Blackberry, o bien porque no ha podido competir con los dos sistemas predominantes, como es el caso de Windows Phone (Ortiz, 2013).

Entre Android y iOS se reparten casi el 90% de las ventas de dispositivos móviles.

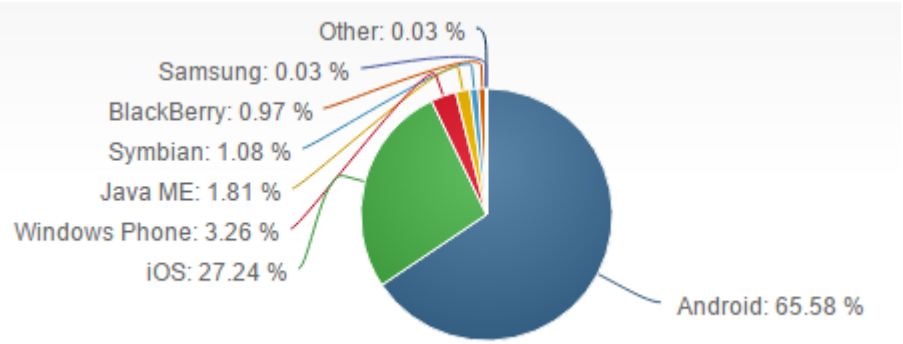


Ilustración 4: Comparativa de ventas de dispositivos móviles según el sistema operativo que corren

La empresa VisionMobile realiza desde hace tiempo su informe Developer Economics, en el que se analizan las tendencias del mercado en cuanto al interés de los desarrolladores por el segmento de la movilidad. En su informe de 2016, muestra la proporción de desarrolladores que utilizan cada plataforma para crear aplicaciones.

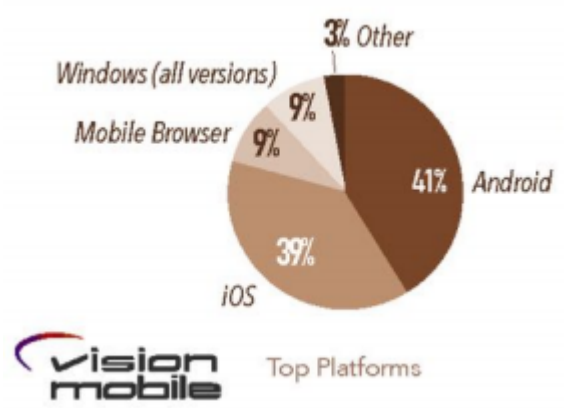


Ilustración 5: Tecnologías utilizadas por los desarrolladores para crear aplicaciones (Vision Mobile, 2016)

Según estos datos, un 41% de los desarrolladores priorizan Android, respecto a un 37% que lo hacían según los datos de 2015, y un 39% de los desarrolladores crean sus aplicaciones en iOS, respecto al 32% que lo hacía en 2015. El porcentaje de desarrolladores que utilizan Windows Phone también ha aumentado hasta un 9% respecto al 6% de 2015. El resto de plataformas tienen una comunidad de desarrolladores testimonial.

Cabe destacar, comparando los datos de ventas y de desarrollo de aplicaciones, que a pesar de que la cuota de mercado de dispositivos con sistema operativo Android es más de dos veces mayor que la de dispositivos iOS, la cantidad de desarrolladores que priorizan las plataformas es prácticamente igual.

Si comparamos los lenguajes de programación utilizados para el desarrollo de aplicaciones móviles en 2016 y la evolución respecto a 2015, se nota un descenso en la utilización de Java (lenguaje utilizado para el desarrollo de aplicaciones Android), mientras que en los lenguajes utilizados para el desarrollo en iOS (Objective C y Swift) se aprecia la tendencia a abandonar Objective C, con un notable descenso en su utilización, en favor de Swift, el cual aumenta ligeramente.

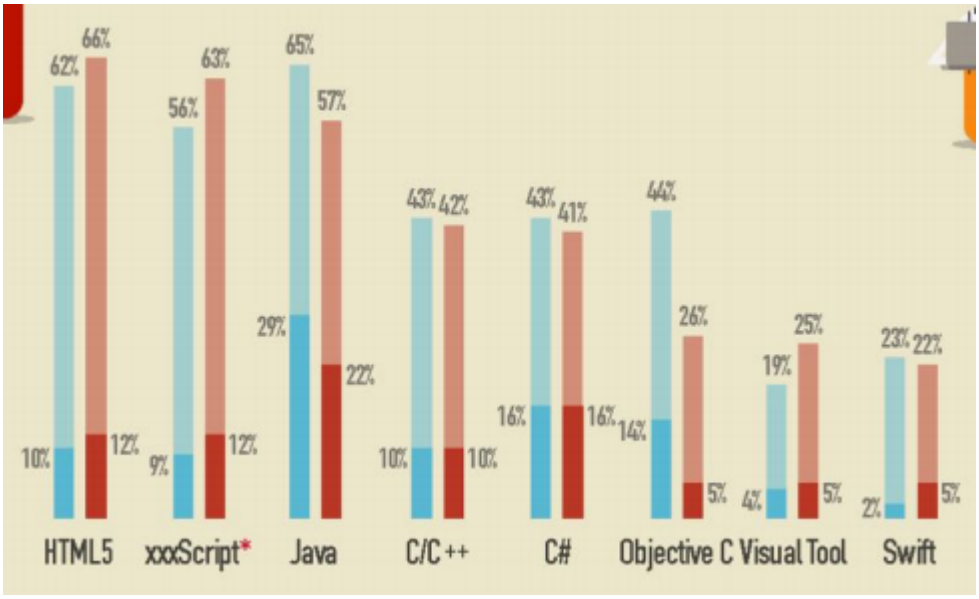


Ilustración 6: Evaluación de los lenguajes utilizados para el desarrollo de aplicaciones móviles (Vision Mobile, 2016)

3.1.1 Aplicaciones móviles web

Una aplicación web o webapp es la desarrollada con lenguajes muy conocidos por los programadores, como es el HTML, Javascript y CSS. La principal ventaja de este tipo de aplicaciones es la posibilidad de programar independiente del sistema operativo en el que se usará la aplicación. De esta forma se pueden ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones.

Las aplicaciones web se ejecutan dentro del propio navegador web del dispositivo a través de una URL. Por ejemplo en Safari, si se trata de la plataforma iOS. El contenido se adapta a la pantalla adquiriendo un aspecto de navegación como si se tratase de una aplicación completa.

Ventajas	Inconvenientes
· El mismo código puede reutilizarse en múltiples plataformas · Proceso de desarrollo más sencillo y económico · No es necesario pasar por el proceso de aprobación para publicar la aplicación · El usuario siempre dispone de la última versión sin necesidad de realizar un proceso de actualización	· Requieren de conexión a Internet · Acceso muy limitado a los elementos y características del hardware del dispositivo · La experiencia del usuario (navegación, interacción) es peor que en una aplicación nativa · Requiere de mayor esfuerzo para asegurar la visibilidad al no publicarse en las tiendas de aplicaciones comunes

Tabla 1: Ventajas e inconvenientes del desarrollo de aplicaciones móviles web

3.1.2 Aplicaciones móviles híbridas

Las aplicaciones híbridas son aplicaciones en parte nativas, en parte web. Al igual que las aplicaciones nativas, se publican en las tiendas de aplicaciones de cada plataforma y tienen acceso a muchas de las funcionalidades del dispositivo. Al igual que las aplicaciones web, delegan en un navegador que renderiza el código HTML, con la diferencia de que en este caso el navegador está contenido en la propia aplicación. En muchas ocasiones, las compañías construyen sus aplicaciones híbridas adaptando una página web existente, de este modo tienen presencia en las tiendas de aplicaciones sin tener que desarrollar una aplicación diferente desde cero. Las aplicaciones híbridas también son muy populares por permitir reutilizar el código HTML para desarrollar aplicaciones para los distintos sistemas operativos sin demasiadas modificaciones, lo que permite reducir significativamente los costes.

Para crear una aplicación híbrida, se desarrollan ficheros HTML, CSS y JavaScript a ejecutar en un navegador como si de una aplicación web se tratase. Pero, como sucedería con una aplicación nativa, el código generado se compila y se forma un paquete ejecutable que puede ser compartido en las tiendas de aplicaciones.

El acceso a las herramientas del terminal se lleva a cabo mediante frameworks. Apache Cordova es uno de los más utilizados por los programadores para el desarrollo de aplicaciones híbridas.

Hoy día existen muchos frameworks y tecnologías basadas en HTML5 y Javascript que permiten programar interfaces de uso en los móviles que casi igualan la experiencia de usuario de aplicaciones nativas. A continuación se van a mencionar algunos de los múltiples frameworks que existen hoy en día:

- Apache Cordova (también denominado PhoneGap): es un framework que permite el desarrollo de una aplicación móvil simultáneamente para varios sistemas operativos. Fue creado por la empresa Nitobi, la cual fue adquirida por Adobe, quien nombró el software como PhoneGap, y lanzó posteriormente la versión open source llamada Apache Cordova. El desarrollo utilizando este framework se basa en CSS3, HTML5 y JavaScript en lugar de obligar a utilizar las APIs específicas de los lenguajes de desarrollo de los diferentes sistemas operativos. Permite convertir casi todos los códigos basados en tecnologías web (HTML, CSS y JavaScript) en código con apariencia de nativo, preparado para compilar en el SDK. Evitando al programador el largo proceso de aprendizaje de los lenguajes y los métodos de las APIs específicos de cada plataforma. El producto final de cada aplicación en un archivo binario (IPA, APK, XAP, etc.) listo para ser distribuido en las diferentes tiendas o markets propios de cada sistema. Apache Cordova consiste en

un conjunto de APIs, basadas en JavaScript, que permiten al desarrollador acceder a las distintas funciones y elementos nativos del dispositivo (como cámara, sensores, GPS, contactos). El uso de estas librerías se combina a su vez con otros frameworks de desarrollo web móvil para permitir acceder a funcionalidades del dispositivo utilizando únicamente lenguajes de programación web.

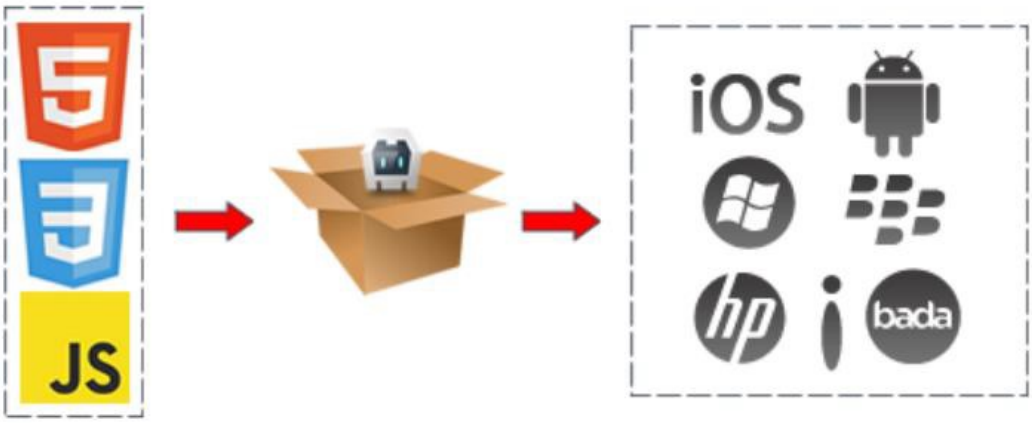


Ilustración 7: Proceso de creación de aplicación móvil híbrida

- IONIC: es uno de los frameworks para aplicaciones móviles en HTML5 más utilizados. Está creado utilizando el lenguaje de hojas de estilo SASS. Proporciona muchos componentes de interfaz de usuario para el desarrollo de aplicaciones complejas e interactivas. Utiliza el framework MVVM de JavaScript y el lenguaje AngularJS.
- jQuery Mobile: se trata de un framework optimizado para dispositivos táctiles. Está siendo desarrollado actualmente por el equipo de proyectos de jQuery. jQuery Mobile es compatible con otros frameworks móviles y plataformas como Apache Cordova y Worklight. El framework se focaliza en crear sitios web-responsive para ser convertidos en aplicaciones híbridas (compilado como una aplicación nativa).

Ventajas	Inconvenientes
· Es posible distribuir las aplicaciones en las tiendas de iOS y Android. · Instalación nativa pero construida con JavaScript, HTML y CSS. ·El mismo código base para múltiples plataformas. · Acceso a parte del hardware del dispositivo.	· Experiencia del usuario similar a la de una aplicación web · Diseño visual no siempre relacionado con el sistema operativo en que se muestre

Tabla 2: Ventajas e inconvenientes del desarrollo de aplicaciones móviles híbridas

3.1.3 Aplicaciones móviles nativas

Una aplicación nativa es la que se desarrolla de forma específica para un determinado sistema operativo, llamado Software Development Kit o SDK. Cada una de las plataformas, Android, iOS o Windows Phone, tienen un sistema diferente, por lo que si se pretende que una aplicación (App) esté disponible en todas las plataformas se deberán de crear varias apps con el lenguaje del sistema operativo seleccionado.

Sistema operativo	Compañía	Lenguaje de programación
iOS	Apple	Objective C / Swift
Android	Google	Java
Windows Phone	Microsoft	C#

Tabla 3: Distintos sistemas operativos móviles

Cuando se habla de desarrollo móvil casi siempre se refiere a aplicaciones nativas. La principal ventaja con respecto a los otros dos tipos (aplicaciones híbridas o aplicaciones Web – WebApp), es la posibilidad de acceder a todas las características del hardware del móvil: cámara, GPS, agenda, dispositivos de almacenamiento y otras muchas. Esto hace que la experiencia del usuario sea mucho más positiva que con otro tipo de aplicaciones.

La descarga e instalación de estas aplicaciones se realiza siempre a través de las tiendas de aplicaciones (app store de los fabricantes). Esto facilita el proceso de marketing y promoción lo que es vital para dar visibilidad a una aplicación.

Está claro que si el coste no es un obstáculo a la hora de programar, si se busca un gran rendimiento o si se tiene la certeza de que la aplicación será rentable, la mejor opción será siempre el desarrollo de una aplicación nativa para cada plataforma (iOS, Android y Windows Phone).

Ventajas	Inconvenientes
· Acceso completo al dispositivo · Rendimiento superior · Buena experiencia de usuario · Visibilidad en las tiendas de aplicaciones de los sistemas operativos · Envío de notificaciones o avisos a los usuarios	· Diferentes procesos/lenguajes de programación según la plataforma de destino · Coste de desarrollo elevado · Código no reutilizable entre las diferentes plataformas

Tabla 4: Ventajas e inconvenientes del desarrollo de aplicaciones móviles nativas

3.1.3.1 Android

Ilustración 8: Aspecto de la pantalla de inicio en un dispositivo Android

Google compró al desarrollador original de Android, Android inc., en 2005. El sistema es desarrollado a partir de ese momento por la Open Handset Alliance. Liderada por Google, es una alianza comercial de 84 compañías que buscan desarrollar estándares abiertos para dispositivos móviles. Entre ellas, se encuentra

las principales compañías del sector tecnológico, a excepción de Apple, Microsoft, Nokia y algunos competidores más. Varios operadores móviles de renombre internacional, como Telefónica, Vodafone y T-Mobile, también forman parte de la alianza.

La última versión de Android lanzada por Google es Android 6.0 Marshmallow, aunque el porcentaje de dispositivos en que se encuentra instalada esta última versión es de aproximadamente un 5%, según datos de la compañía Google. La gran fragmentación entre los dispositivos basados en sistema operativo Android, fabricados por numerosas compañías, limita en gran medida la distribución ágil de las actualizaciones del sistema.

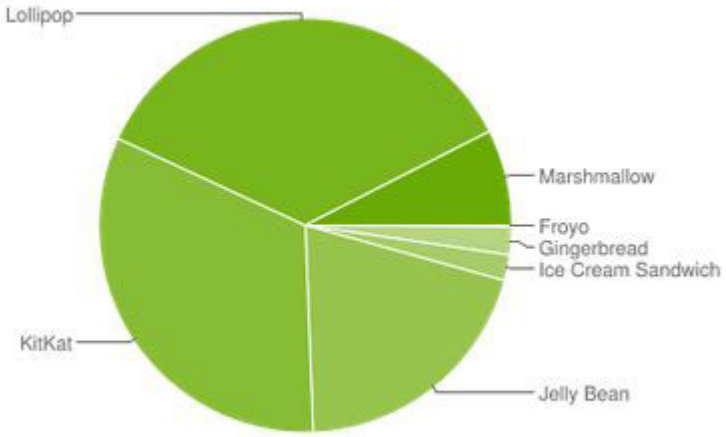


Ilustración 9: Porcentaje de utilización de las diferentes versiones de sistema operativo Android (Google, 2016)

Android es una pila de software para dispositivos móviles que incluye un sistema operativo, middleware y diversas aplicaciones. El Android SDK aporta las herramientas y APIs necesarias para empezar a desarrollar aplicaciones para la plataforma Android usando el lenguaje de programación Java. En la siguiente ilustración se puede apreciar un diagrama en el que se muestra la arquitectura de Android (Rodríguez Cayetano, 2014).

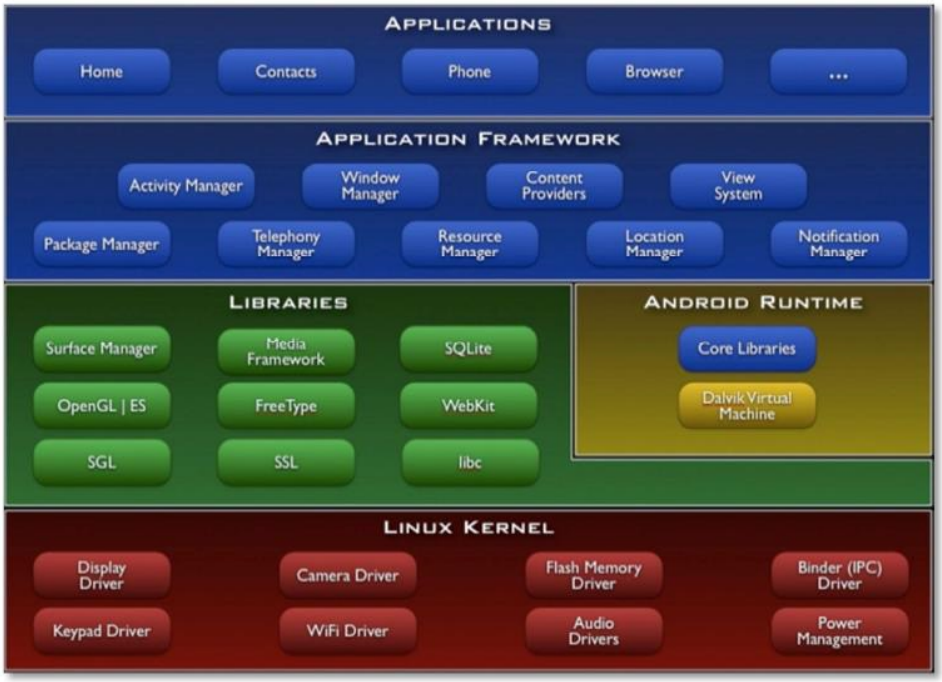


Ilustración 10: Arquitectura del sistema operativo Android

Antes de estudiar la arquitectura de Android se harán algunas aclaraciones sobre Java. Java es un lenguaje orientado a objetos que fue diseñado para funcionar de forma independiente a la arquitectura sobre la que trabaja. Para ello, tal y como se puede observar en la ilustración siguiente, la compilación del código fuente (archivos .java) genera archivos de clases (archivos .class), es decir bytecode, los cuales serán ejecutados por la máquina virtual JVM (Java Virtual Machine), utilizada como abstracción entre el hardware de la máquina y los programas Java. Los distintos archivos class suelen ser compilados en un único archivo JAR (Java Archive). En Android sucede algo parecido (Google, 2016).

La Open Handset Alliance optó por diseñar un sistema similar teniendo en cuenta las limitaciones de los dispositivos móviles, en principio una baja capacidad de almacenamiento y procesamiento y poca potencia de cálculo.

La máquina virtual empleada por Android recibe el nombre de DVM (Dalvik Virtual Machine). Los programas generalmente son escritos en Java y compilados a bytecode. Posteriormente son convertidos a archivos compatibles con la DVM (archivos.dex) antes de su instalación en el dispositivo. Estos archivos son comprimidos en archivos APK (Application Package), los cuales pueden ser instalados en dispositivos Android compatibles (a la hora de crear una aplicación es necesario especificar la versión para la que está diseñada por lo que el terminal donde se instalará ha de tener una versión de Android igual o superior). En la ilustración siguiente se puede observar cómo se crean los archivos con extensión dex a partir de los archivos de clase class. La principal diferencia radica en la forma

de empaquetar la información. Los archivos de clase son transformados en un único archivo .dex en el que se intenta despreciar la información repetida. Es más que evidente el enfoque de Android hacia dispositivos con memoria pequeña y almacenamiento similar.

Otro factor importante y a tener en cuenta es que Android implementa algunas de las APIs de Java pero no todas ellas.

Consecuentemente, Android y Java no son el mismo sistema, aunque tengan muchos puntos en común.

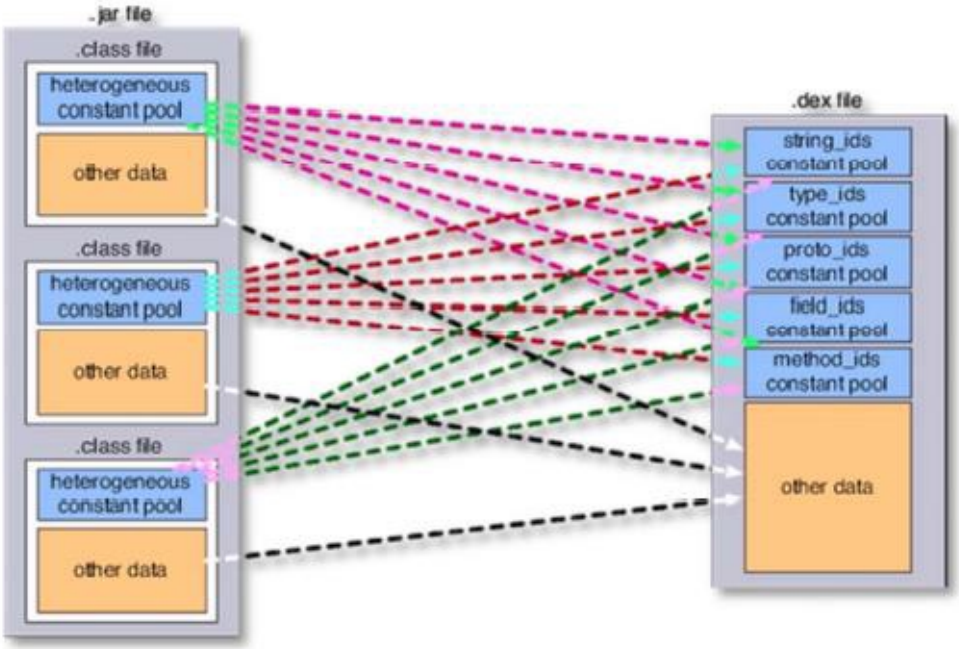


Ilustración 11: Comparativa entre Android y Java

A continuación se exponen las principales características de la arquitectura Android (Rodríguez Cayetano, 2014):

- Applications: Android se entrega conjuntamente con una serie de aplicaciones que incluyen un cliente de email, un programa gestor de SMS, calendario, mapas, explorador y gestor de contactos entre otros. Todas ellas están escritas en Java.
- Application Framework: los desarrolladores de Android tienen acceso a las mismas APIs del framework usadas por las aplicaciones del núcleo. La arquitectura está diseñada con el objetivo de simplificar la reutilización de componentes. Por debajo de todas las aplicaciones hay un conjunto de servicios y sistemas entre los que se incluyen:
    - Conjunto de Views que pueden ser utilizadas para construir la aplicación en cuestión: listas, tablas, cajas de texto…

    - Proveedores de contenido que permiten compartir información entre aplicaciones, como por ejemplo la destinada a gestionar los contactos.
    - Un Resource Manager capaz de proporcionar acceso a recursos como cadenas de texto, imágenes, layouts…
    - Un Notification Manager que permite gestionar las notificaciones en pantalla.
    - Un Activity Manager que gestiona el ciclo de vida de las aplicaciones.
- Libraries: Android incluye un conjunto de librerías escritas en C/C++ que son usadas por el sistema Android. Algunas de estas librerías son: System C library, Media libraries, Surface Manager…
- Android Runtime: cada aplicación de Android es un proceso diferenciado que tiene su propia instancia de la máquina DVM (Dalvik Virtual Machine). La máquina virtual Dalvik se sustenta en el Linux Kernel para proporcionar funcionalidades como la gestión de threads y de la memoria a bajo nivel.
- Linux Kernel: Android se basa en la versión 2.6 de Linux para servicios del núcleo de sistema como seguridad, gestión de memoria, gestión de drivers…El núcleo Linux también es usado como capa de abstracción entre el hardware y el resto del software.

Por su parte, las principales ventajas de Android son las siguientes:

- Uso del lenguaje de programación Java. Este lenguaje se encuentra ampliamente extendido y su comprensión si se desconoce es fácil para desarrolladores con conocimientos de programación orientados al objeto.
- Software libre: gracias a la licencia Apache en la que se basa Android, cualquier programador puede realizar modificaciones de las partes más internas de cualquier programa. Además, cualquier persona puede realizar un programa para dispositivos Android sin necesidad de cuotas anuales como sucedía con iOS.
- Accesibilidad: la participación de terceros en el desarrollo del sistema operativo conlleva la aparición y creación de multitud de APIs.
- Al igual que sucede con iOS, existe una gran comunidad de desarrolladores gracias a la cual es sencillo encontrar información para programadores noveles en la red.

En cuanto a las desventajas se pueden citar las siguientes:

- Fragmentación: Android ha sido criticado múltiples veces a causa de la inmensa variedad de terminales que soportan el sistema operativo. Este hecho no es en sí mismo negativo, pero la diferencia de

hardware existente entre sus terminales y la influencia de las operadoras móviles y de los fabricantes en el ritmo de llegada de las actualizaciones de Android a los terminales, hacen que exista una diversidad enorme de versiones del sistema operativo utilizados por los clientes. Existen numerosas y distintas versiones del sistema operativo de Google. Esto supone un gran inconveniente para los desarrolladores de software, obligados a revisar sus aplicaciones cada vez que Android saca al mercado una nueva versión.

- Aplicaciones: Google no supervisa las aplicaciones que se suben a Google Play, por lo que suele haber un mayor número de aplicaciones no demasiado útiles en el mercado.
- La gestión multitarea. El sistema de Google gestiona la multitarea de modo que si una aplicación pasa a segundo plano, ésta se sigue ejecutando en el procesador del terminal. El problema en que deriva este hecho, es que si un usuario tiene muchas tareas abiertas, el dispositivo se ralentizará.

En cuanto al entorno de desarrollo, programar para Android tiene bastantes menos requisitos que hacerlo para iOS. Prueba de ello es la posibilidad de elección del sistema sobre el que se quiere programar (Unix, Windows, Mac OS,…) y del entorno de desarrollo a utilizar. El Entorno de Desarrollo oficial es Android Studio, que fue lanzado por Google en 2013, aunque también pueden utilizarse otras opciones, como Eclipse.

3.1.3.2 iOS

Ilustración 12: Aspecto de la pantalla de inicio en un dispositivo iOS

iOS (anteriormente denominado iPhone OS) es un sistema operativo móvil de Apple desarrollado originalmente para el iPhone, siendo después usado en el iPod touch e iPad. Es un derivado de Mac OS X, que a su vez está basado en Darwin (Sistema Operativo). El iOS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios de comunicación" y la capa de "Cocoa Touch" (Antón Rodríguez, 2014).

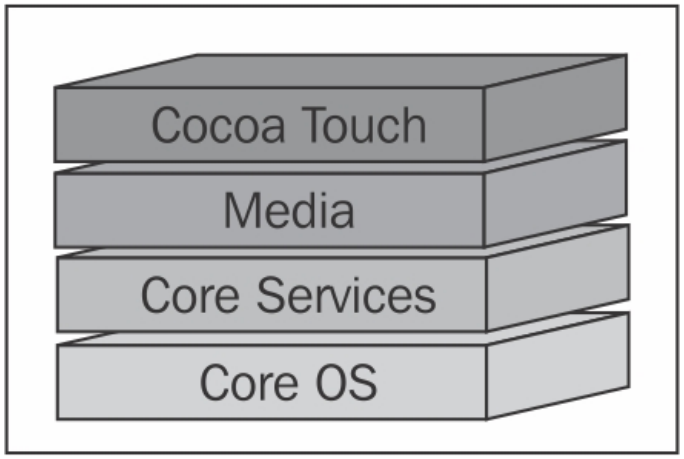


Ilustración 13: Arquitectura del sistema operativo iOS

La versión del sistema iOS más actual es iOS9, la cual se estima que está instalada en el 84% de los dispositivos iPhone, iPod touch o iPad, según información de Apple (Apple, 2016).

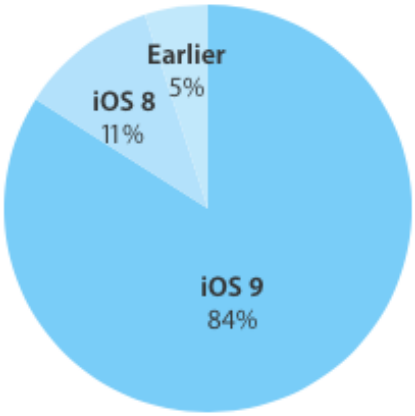


Ilustración 14: Porcentaje de utilización de las distintas versiones de sistema operativo iOS (Apple, 2016)

Originalmente Apple desarrolló el sistema operativo iOS para su teléfono inteligente iPhone. Apple reveló por primera vez la existencia de iOS, por aquel entonces llamado iPhone OS, en la MacWorld Conference el 9 enero del 2007, aunque el sistema no tuvo nombre oficial hasta marzo del 2008, fecha en la que salió la primera versión beta del iPhone SDK (Start Development Kit). Como se ha mencionado el sistema original estaba diseñado para funcionar en los iPhones de

Apple, pero lo emplean otros de sus productos como son el iPod Touch, el iPad y Apple TV. Apple no permite la instalación de iOS en hardware de terceros.

En junio del 2010 Steve Jobs, CEO de Apple, anunció que iPhone OS pasaría a llamarse iOS. iOS está basado en el sistema operativo Mac OS X, que a su vez está basado en Darwin BSD y por lo tanto es un sistema operativo Unix. En concreto ambos sistemas operativos comparten el mismo núcleo Mach/FreeBSD, y utilizan como lenguajes de programación principales C, Objective-C y Swift (Macprogramadores, 2013).

El sistema Unix es el utilizado en publicaciones de Linux, así que iOS, OS X y Linux, guardan más similitudes de las que nos podemos imaginar, tan solo que los dos primeros son sistemas operativos propiedad de Apple y cerrados al uso en dispositivos de la propia compañía, mientras que Linux es un código abierto y valido para multitud de dispositivos, abierto a implementaciones y al uso e inclusión en los dispositivos y marcas que lo consideren.

El lenguaje de programación C es el predecesor de Objetive-C y ha sido el principal lenguaje estructurado y utilizado en gran cantidad de entornos diferentes. C nació a principios de los años 70 en AT&T Bell Laboratories a manos de Dennis Ritchie. A su vez, Objective-C es el predecesor del lenguaje Swift, lanzado por Apple en 2014.

Las aplicaciones nativas para OS X e iOS pueden programarse en el lenguaje Swift o en Objective-C, aunque este último cada vez está más en desuso en favor de Swift. Se trata de lenguajes orientados a objetos, dinámicos. El lenguaje Swift está en constante mejora por parte de Apple.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles (eventos multi-touch). Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo o rotarlo en tres dimensiones.

Para entender mejor la arquitectura de iOS se debe comenzar por saber que la capa Cocoa Touch es la de más alto nivel y la de Core OS la de más bajo nivel. En general, las dos capas superiores ofrecen servicios y tecnologías más sofisticadas que el resto de capas. Los detalles de cada una de ellas son:

- Cocoa Touch: es un conjunto de frameworks orientados a objetos que permiten el desarrollo de aplicaciones nativas para iOS. El lenguaje utilizado es Swift. Cabe destacar la presencia de los siguientes servicios:
    - Eventos multi-touch.
    - Multi-tasking.
    - Acelerómetro y giroscopio.

    - Jerarquía de vistas.
    - Localización e internalización: framework que permite adoptar diferentes idiomas y regiones sin necesidad
- Media: contiene servicios orientados a multimedia como:
    - OpenAL (Open Audio Library).
    - Mezcla de audio y grabación.
    - Reproducción de video.
    - Formatos de archivo de imágenes
    - Quartz: framework para manipular gráficos 2D.
    - Core Animation: framework para la visualización de datos.
    - OpenGL: framework para manipular gráficos 3D.
- Core Services: recopila servicios básicos como pueden ser:
    - Networking.
    - Base de datos SQLite.
    - Core Location: framework que permite un fácil acceso al GPS.
    - Threads.
    - Core Motion.
- Core OS: actúa como núcleo del sistema operativo. Controla el sistema de memoria virtual, los hilos, los ficheros del sistema, la red e interprocesa la comunicación con los marcos de la capa Core Services. En general rodea el entorno del kernel, los controladores y las interfaces básicas del sistema operativo
    - TCP/IP
    - Sockets
    - Gestión de la batería
    - Sistema de ficheros
    - Seguridad

Para poder desarrollar aplicaciones en este sistema operativo, el único entorno de desarrollo disponible es Xcode, el cual es propiedad de Apple. Para poder instalarlo es necesario tener un ordenador con el sistema operativo Mac OS X y una cuenta de Apple para acceder a la App Store, método necesario para descargarlo.

Además, para la publicación de aplicaciones desarrolladas para iOS es necesario tener una cuenta de desarrollador de Apple. Hay varios tipos de cuentas de desarrollador (Apple, 2015):

- Cuenta gratuita: permite descargar el entorno de desarrollo y el SDK, así como probar las aplicaciones en un dispositivo propio.

- Cuenta individual: cuenta para un único desarrollador. Crea un identificador de desarrollador para poder publicar aplicaciones en la tienda App Store. Tiene un coste anual de $99.
- Cuenta de organización: cuenta para empresas con un gran número de desarrolladores. Permite incluir cuentas personales de cada desarrollador con las mismas características que la cuenta individual. Un miembro asume la gestión de usuarios dentro del equipo. Tiene un coste anual de $99.
- Cuenta Enterprise: permite crear aplicaciones propietarias a una empresa que sólo se distribuyen entre los empleados de la misma. Tiene un coste anual de $299.

	Sign in with Apple ID	Individual	Organization	Enterprise Program
Xcode Developer Tools	•	•	•	•
Xcode Beta Releases	•	•	•	•
Developer Forums	•	•	•	•
Bug Reporter	•	•	•	•
Test on Device	•	•	•	•
Beta OS Releases		•	•	•
Advanced App Capabilities		•	•	•
App Store Distribution		•	•	
In-house App Distribution				•
Safari Extensions		•	•	
Developer ID		•	•	•
Technical Support Incidents		•	•	•
Team Management			•	•
TestFlight Beta Testing		•	•	
App Analytics		•	•	
Cost	Free	99 USD*	99 USD*	299 USD**
Requirement	13+	18+	DUNS Number	DUNS Number

Ilustración 15: Comparativa de las diferentes cuentas de desarrollador de Apple disponibles (Apple, 2015)

3.1.3.3 Windows Phone

Ilustración 16: Aspecto de la pantalla de inicio en un dispositivo Windows Phone

Windows Phone (abreviado WP) es un sistema operativo móvil desarrollado por Microsoft, como sucesor de Windows Mobile. A diferencia de su predecesor está enfocado en el mercado de consumo en lugar de en el mercado empresarial. Con Windows Phone; Microsoft ofrece una nueva interfaz de usuario que integra varios de sus servicios propios como OneDrive, Skype y Xbox Live en el sistema operativo. Compite directamente contra Android de Google e iOS de Apple.

La última versión del sistema es Windows 10, un sistema versátil disponible para todo tipo de plataformas (teléfonos móviles, tabletas y ordenadores). El hecho de que la versión móvil y la de ordenador compartan el núcleo del sistema hace que sus modelos de aplicación tengan similitudes y sea sencillo compartir código entre ambas plataformas, e incluso llegar a desarrollar una única aplicación que sea compatible con dispositivos móviles y ordenadores.

Para poder desarrollar en este sistema operativo, no es necesario tener una cuenta propia para descargarse el SDK, pero sí se requiere para desbloquear nuestro terminal móvil y para publicar nuestras apps en el Store de Windows Phone. Existen las siguientes opciones para adquirir una cuenta de desarrollador (Microsoft, 2016):

- Incluida si tienes una suscripción MSDN.
- Gratis para estudiantes con la suscripción Dreamspark.
- $99 al año para desarrolladores individuales.

3.1.3.4 Blackberry RIM

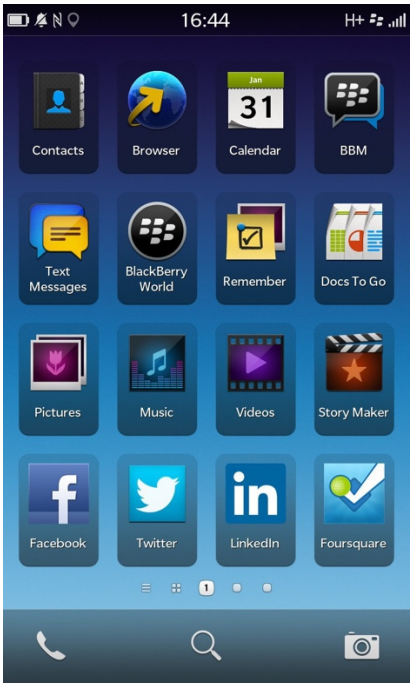


Ilustración 17: Aspecto de la pantalla de inicio en un dispositivo Blackberry

El BlackBerry OS es un sistema operativo móvil de código cerrado desarrollado por BlackBerry, antigua Research In Motion (RIM); para los dispositivos BlackBerry. El sistema permite multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la trackwheel, TrackBall, touchpad y pantallas táctiles. Su desarrollo se remonta la aparición de los primeros handheld en 1999.

El SO BlackBerry está claramente orientado a su uso profesional como gestor de correo electrónico y agenda. Desde la cuarta versión se puede sincronizar el dispositivo con el correo electrónico, el calendario, tareas, notas y contactos de Microsoft Exchange Server además es compatible también con Novell GroupWise.

Se caracteriza por ser un sistema operativo atractivo y algo diferente a Android e iOS, además, sus iconos son sencillos. Casi todas sus aplicaciones piden permisos para sincronizarse con BlackBerry Messenger, y es que esta aplicación es el centro del teléfono. El kernel del sistema operativo Blackberry está basado en Java y posee una arquitectura ARM. Se caracteriza por ser bastante ligero y funcionar con fluidez. Mediante dos pestañas permita separar la vida personal de la laboral, la seguridad en este dispositivo es una prioridad.

Uno de los inconvenientes que actualmente ha hecho que el uso de este sistema operativo haya disminuido tanto es que su tienda de aplicaciones, BlackBerry World, no cuenta con todas las aplicaciones que a los usuarios les

gustaría. Además de esto, sus escasas actualizaciones y el retraso en sacar nuevos dispositivos móviles, en comparación con otras tecnologías ha hecho que hoy en día, porcentaje de dispositivos que usan Blackberry sea inferior a un 1%.

3.1.3.5 Comparativa entre sistemas

Se ha observado que iOS goza de una presencia preferente entre los usuarios de Apple, debido en gran medida a la baja fragmentación que tiene los dispositivos de esta compañía, facilitando en gran medida la distribución de las actualizaciones a todos los dispositivos.

Sin embargo, en el caso de Android la disparidad de versiones del sistema operativo es mucho mayor, siendo el porcentaje de uso de la última versión disponible muy bajo.

En el caso de Windows Phone, Windows 10 ha supuesto una gran actualización del sistema operativo móvil, gracias a mejoras en diseño y funcionalidad, unidas a la gran ventaja de la convergencia entre el sistema operativo para dispositivos de escritorio y el sistema operativo móvil, que pueden llegar a compartir aplicaciones. Sin embargo, las funcionalidades que ofrece este sistema, aún se quedan por detrás de Android o iOS.

Teniendo en cuenta los requisitos para el desarrollo y distribución de aplicaciones, Android da más facilidades y es mucho más abierto que iOS o Windows Phone, los cuales requieren de una cuenta de desarrollador (con el coste asociado que eso conlleva) para tal efecto.

3.2 Elección a partir de comparativa entre tipos de aplicaciones

Ilustración 18: Comparativa entre aplicaciones móviles web, híbridas y nativas

La decisión tomada para la realización de la aplicación móvil para el uso de la plataforma eLiza ha sido la de implementarla como una aplicación nativa desarrollándola para el sistema operativo iOS.

Esta elección se ha tomado tras hacer una comparación de las diferentes tecnologías que existen actualmente en el mercado del diseño e implementación de aplicaciones móviles y comprobar que las aplicaciones nativas ofrecen una experiencia de usuario muy superior y una mayor visibilidad con la posibilidad de publicar la aplicación en una tienda oficial.

Las ventajas a destacar para la elección de una aplicación nativa que han llevado a tomar la decisión son:

- Mayor rendimiento.
- Mejor experiencia de usuario.
- Aprovechamiento de las funcionalidades del dispositivo.
- Posibilidad de distribución a través de la tienda de la plataforma.

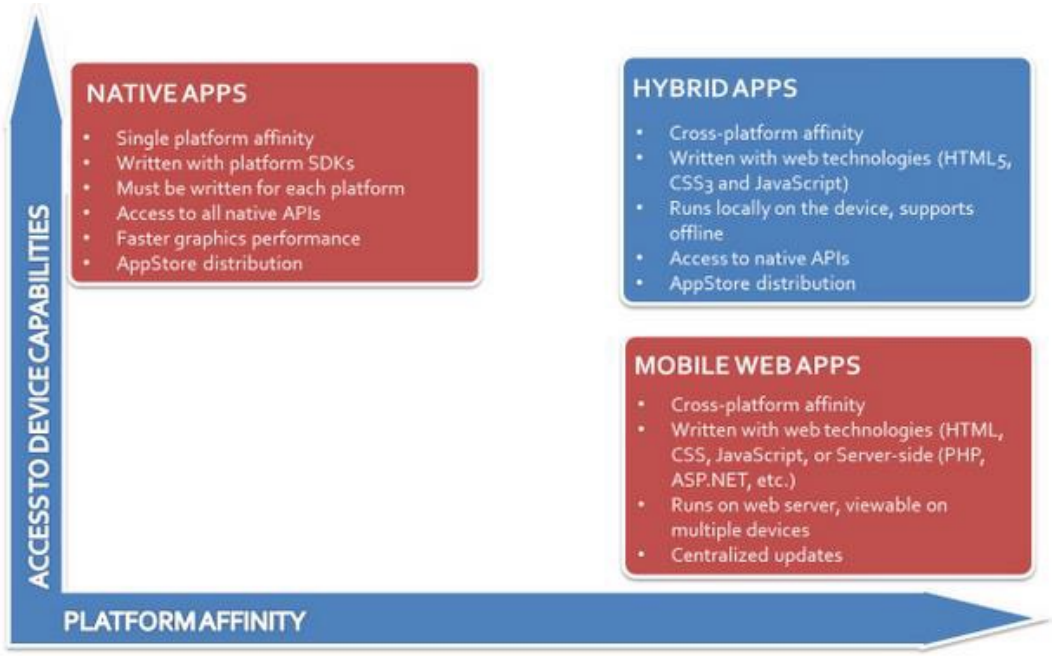


Ilustración 19: Ventajas de aplicaciones móviles web, híbridas y nativas

Respecto al sistema operativo a utilizar para desarrollar la aplicación, decidí hacerlo para dispositivos iOS. Entre los motivos que me llevaron a elegir esta opción destacaría el crecimiento que está experimentando esta plataforma entre los desarrolladores de aplicaciones móviles, debido en gran parte a la mayor visibilidad y seguridad que da la tienda oficial de la plataforma al no permitir instalar aplicaciones externamente. También tuve en cuenta las preferencias del cliente, que ya contaba con una aplicación para la plataforma eLiza para dispositivos Android.

3.3 Lenguajes en el cliente

En este apartado se contempla el análisis de los distintos lenguajes de programación que pueden utilizarse para desarrollar una aplicación nativa para un dispositivo con sistema operativo iOS.

3.3.1 Objective-C

Objective-C es un lenguaje de programación dinámico orientado a objetos creado como un superconjunto del lenguaje C. Objective-C hereda la sintaxis, los tipos primitivos y el control de flujo de C y añade sintaxis para la definición de clases y métodos (Kochan, 2011).

Originalmente fue creado por Brad Cox y la corportación StepStone en 1980. En 1988 fue adoptado como lenguaje de programación del sistema operativo orientado a objetos NEXTSTEP y en 1992 fue liberado bajo licencia GPL para el compilador GCC. Tras adquirir Apple la empresa NeXT en 1996, comenzó a emplear este lenguaje en sus nuevo sistema operativo, Mac OS X. Este incluía Objective-C y la herramienta de desarrollo Project Builder, que más tarde se convirtió en Xcode, así como la herramienta de diseño de interfaz, Interface Builder.

Objective-C es un superconjunto de C, por lo que es posible compilar cualquier programa escrito en C con un compilador de Objective-C, y puede incluirse código de C en una clase de Objective-C.

Toda la sintaxis de Objective-C de operaciones no orientadas a objetos, como variables primitivas, pre-procesamiento, declaración de funciones, expresiones, es idéntica a la de C, mientras que la sintaxis de características orientadas a objetos es propia.

El modelo de programación orientada a objetos de Objective-C se basa en enviar mensajes a instancias de objetos. Esto es diferente al modelo de programación utilizado por C++, el otro gran lenguaje orientado a objetos basado en C. En Objective-C no se llama a un método, si no que se envía un mensaje, que es simplemente un nombre que es resuelto en tiempo de ejecución. El objeto receptor tiene la tarea de interpretar el mensaje, esto hace que no se compruebe que el objeto receptor puede interpretar y responder al mensaje. Si no es capaz de hacerlo, simplemente ignora el mensaje y devuelve un puntero nulo.

Objective-C requiere una definición separada de una interfaz y de la implementación de la clase. La interfaz se define en un fichero de cabecera, que al igual que en C son ficheros .h, y la implementación se define en ficheros de tipo .m.

Una vez que se ha definido una clase en Objective-C, esta puede ser instanciada. Esto se lleva a cabo reservando una cantidad de memoria para el nuevo objeto, y luego se inicializa el objeto.

Objective-C puede utilizar tipado dinámico, es decir, un objeto puede recibir un mensaje que no está especificado en su interfaz. Esto aumenta la flexibilidad al permitir a un objeto recibir un mensaje y reenviarlo a otro objeto diferente que pueda procesar y responder a ese mensaje correctamente. Este comportamiento se conoce como delegación.

3.3.2 Swift

Ilustración 20: Logo de Swift

Swift es un lenguaje de programación orientado a objetos creado por Apple. Está diseñado para integrarse con los frameworks Cocoa y Cocoa Touch de la arquitectura de los sistemas operativos de Apple. Puede utilizar cualquier biblioteca desarrollada en Objective-C y llamar a cualquier función de C.

Apple presentó Swift en su congreso para desarrolladores de 2014, como evolución del antiguo Objective-C. En 2015, se lanzó la versión 2.0 de Swift, con la que pasó a ser un lenguaje de código abierto.

Swift es un lenguaje fuertemente tipado, aunque su declaración no es siempre necesaria gracias a su capacidad para inferir tipos. Los tipos de datos se dividen en dos grupos (Apple, 2015):

- Tipo de valor: se guarda una copia de su contenido, de modo que si se modifica el valor de esta copia el valor original no se modifica.
- Tipo por referencia: se asigna una instancia compartida, si se modifica el valor cualquier variable que apunte a él se ve modificada

Como evolución de Objective-C, Swift soporta bloques y módulos, adoptando tecnologías de lenguajes modernos.

La sintaxis de Swift es similar a la de C y Objective-C, aunque ha sido simplificada. Se elimina la necesidad de definir una función principal o de cerrar cada línea con ; .

Por defecto, Swift no utiliza punteros y otros métodos de acceso inseguros, en contraste con Objective-C. También se modifica el modo de llamar a métodos con una notación con puntos y un sistema de espacio de nombres más similar a otros lenguajes orientados a objetos como Java.

Una gran ventaja de Swift es su gestión de memoria, la cual se realiza de forma automática mediante el sistema ARC (Automatic Reference Counting).

Swift permite crear variables o constantes, en función de si el valor que va a adoptar va a cambiar o se va a mantener constante.

Debido a su mayor potencial a pesar de su mayor sencillez, y a que se establece como el lenguaje de programación oficial para aplicaciones iOS en la actualidad, he tomado la decisión de utilizar Swift en detrimento de Objective-C para el desarrollo de la aplicación eLiza.

3.4 Tecnologías en el servidor

3.4.1 Lenguajes de programación

Se denominan lenguajes de programación del lado del servidor a aquellos que se ejecutan en el servidor produciendo una salida que es la que le llega al cliente. Existen diferentes opciones a utilizar en esta categoría como son PHP (PHP: Hypertext Preprocessor), ASP (Microsoft Active Server Pages) o JSP (Java Server Pages).

Los lenguajes más utilizados en servidores son:

	Lenguaje	Porcentaje de uso
1	PHP	82.1%
2	ASP.NET	15.8%
3	Java	2.7%
4	Ficheros estáticos	1.5%
5	ColdFusion	0.7%

Tabla 5: Lenguajes de programación de servidores más utilizados

En este apartado se contempla el análisis de los distintos lenguajes de programación que pueden utilizarse para desarro

3.4.1.1 PHP

PHP (PHP Hypertext Preprocesor) es un lenguaje de programación interpretado, cuyos comandos se ejecutan en el servidor y permiten la creación de documentos HTML dinámicos. Su sintaxis es similar a la de otros lenguajes como C, Perl, Java o JavaScript (Antón Rodríguez & Pérez Juárez, Introducción a PHP, 2015).

PHP es un producto de código abierto, lo que quiere decir que se puede acceder a su código, usarlo, modificarlo y distribuirlo de forma gratuita sin que suponga coste alguno, al contrario de lo que ocurre con los productos comerciales. Esta es, sin duda, una de las principales ventajas de PHP, ya que la comunidad de usuarios que desarrollan mejoras para el sistema es enorme, lo que proporciona a PHP una vitalidad y capacidad de adaptación que no poseen otros lenguajes de

programación. Toda la evolución de PHP puede seguirse en el sitio web www.php.net.

PHP se encuentra disponible para muchas plataformas incluyendo Windows, Unix o Linux y con la ventaja de que las aplicaciones desarrolladas en PHP se pueden transportar de una plataforma a otra sin necesidad de modificaciones, es decir que PHP presenta una portabilidad elevada. A pesar de ser en la actualidad un lenguaje multiplataforma, debe comentarse que su entorno nativo es Unix/Linux. Con respecto al servidor web también existirían diferentes opciones, ya que podría usarse por ejemplo IIS (Internet Information Server) de Microsoft o Apache, este último con la ventaja de que, al igual que PHP, es también un producto de libre distribución.

Una de las características más potentes y destacables de PHP es su soporte para una gran cantidad de bases de datos. Escribir una página web con acceso habilitado a una base de datos es increíblemente simple utilizando una de las extensiones específicas (por ejemplo, para MySQL), o utilizar una capa de abstracción como PDO (PHP Data Objects), o conectarse a cualquier base de datos que soporte el estándar de Conexión Abierta a Bases de Datos por medio de la extensión ODBC (Open DataBase Connectivity) (Gutierrez Gallardo, 2009).

PHP fue concebido en 1994 por Rasmus Lerdorf con la intención de crear un contador para averiguar el número de visitas que recibía su CV virtual. Sin embargo, con el tiempo ha sido adoptado por otros desarrolladores que lo han transformado y convertido en la herramienta que es hoy en la actualidad. Una herramienta utilizada ya en más de 200 millones de dominios de Internet, número que sigue creciendo día a día.

PHP es el lenguaje más utilizado, con un 82.1% de utilización en los sitios web que se conocen según los datos de Junio de 2016 dados por W3techs, web especialista en encuestas sobre tecnologías web.

Algunos de los ejemplos de sitios web que utilizan PHP son:

- phpMyAdmin
- SquirrelMail
- Moodle
- Sourceforge
- Facebook
- Tuenti

Por último, comentar el aspecto funcional del lenguaje. El código PHP se puede insertar en un documento HTML, siendo así el proceso de generación de una página web dinámica con PHP el que se describe a continuación:

- Cuando un usuario hace clic desde su navegador en un enlace correspondiente a un documento HTML que contiene código PHP, el navegador realiza la solicitud al servidor web correspondiente.

- El servidor web localiza entonces el documento, detecta que contiene código PHP y pone en funcionamiento el intérprete del lenguaje.
- Dicho intérprete ejecuta el código PHP y genera un resultado, generalmente en forma de página web, que se devuelve al navegador para que éste se encargue de su visualización.

A modo de resumen de PHP, se puede concluir que es un lenguaje de programación gratuito, interpretado (necesita intérprete), sencillo de aprender y que permite realizar diseños escalables y orientados a objetos.

Dado que la plataforma Moodle, sobre la que corre el módulo eLiza para el que se desarrolla la aplicación objeto de este Trabajo Fin de Grado, está desarrollada en PHP, el desarrollo necesario por la parte del servidor se realizará en este lenguaje.

Las grandes ventajas que esto supone son:

- Alto rendimiento: PHP es muy eficiente, con un único servidor es capaz de servir millones de accesos al día.
- Integración con bases de datos: PHP dispone de una conexión propia a todos los sistemas de base de datos. Puede conectarse directamente a las bases de datos de MySQL, PostgreSQL, mSQL, Oracle, dbm, filePro, Hyperwave, Informix, Internase y Sybase, entre otras. Esto se debe a que PHP utiliza ODBC (Open Database Connectivity Standard).
- Coste: PHP es un lenguaje gratuito y puede utilizarse sin ningún coste.
- Sencillez: se trata de un lenguaje relativamente sencillo de aprender.
- Portabilidad: el intérprete PHP puede trabajar sobre una gran cantidad de sistemas operativos diferentes, bien sea basados en Unix o en Windows. El código funcionará del mismo modo sin necesidad de aplicar modificaciones.
- Código fuente: el código fuente de PHP es de libre acceso, a diferencia de productos comerciales de código cerrado. Este hecho da la posibilidad d modificar o agregar elementos al programa para personalizarlo a las necesidades del producto.

3.4.2 Servicios web

Un servicio *web* es un sistema de *software* diseñado para permitir la interoperabilidad máquina a máquina en una red (W3C, 2015). Se trata de APIs que son publicadas, localizadas e invocadas a través de la *web*. Es decir, estas APIs se instalan en un servidor de Internet y otras aplicaciones o servicios *web* pueden invocar uno de sus servicios.

Una de las principales motivaciones del uso de servicios web es la independencia de plataforma. Podemos utilizar un servicio web con independencia del sistema operativo o el lenguaje de programación en el que se haya desarrollado el dispositivo que invoca el servicio. Al estar apoyado sobre el protocolo HTTP

hace uso de sus sistemas de seguridad (https). Además facilitan la interoperabilidad entre sistemas, permitiendo disponer de componentes software independientes que pueden compartir sus funcionalidades con otros servicios y aplicaciones.

El esquema de funcionamiento de los servicios web requiere tres agentes fundamentales:

- Un proveedor del servicio web: se trata de quien lo diseña, desarrolla, implementa y pone a disposición para su uso.
- Un solicitante del servicio: se trata de quien accede al componente para utilizar los servicios que presta.
- Un directorio: sirve de enlace entre el proveedor y el solicitante, a efectos de publicación, búsqueda y localización del servicio.

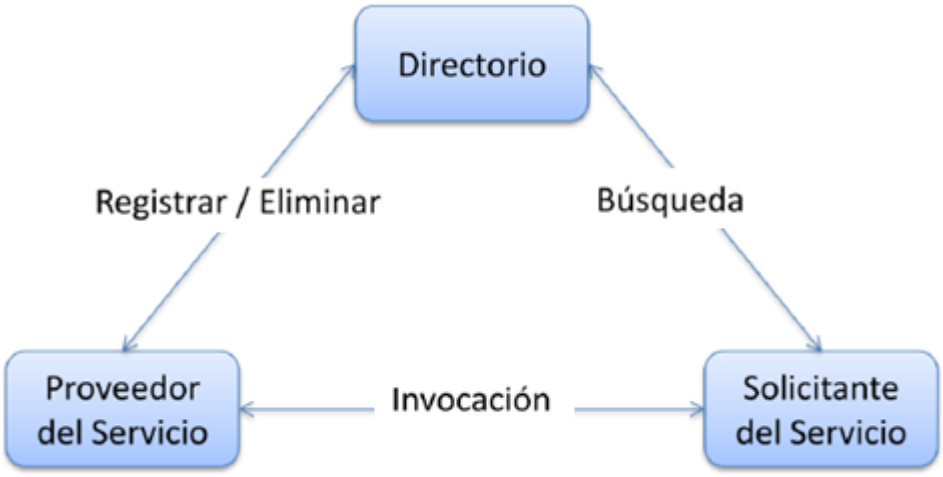


Ilustración 21: Esquema de funcionamiento de un servicio web

Para poner a disposición un servicio web se deben seguir los siguientes pasos:

- Definir el servicio web: para que pueda interactuar con otros se debe utilizar una codificación común que permita estructurar los datos que intercambia el servicio. Para ello lo más común es utilizar las tecnologías XML o JSON.
- Publicar el servicio: para conseguir que otros servicios y aplicaciones puedan interactuar con él. Esto implica ubicar el servicio en un determinado servidor y realizar una descripción del mismo, para que los clientes puedan saber qué funciones implementa y cómo se debe emplear el servicio para hacer uso de esas funciones.

El lenguaje utilizado para elaborar la descripción del servicio es WSDL (Web Service Description Language) y la publicación del mismo se realiza

mediante UDDI (Universal Description, Discovery and Integration), independientemente del servidor utilizado.

Para intercambiar información entre el proveedor y el solicitante se debe hacer uso de un protocolo de comunicaciones, como SOAP (Simple Object Access Protocol), que transmitirá los datos sobre HTTP, FTP o SMTP en formato XML, o REST, que intercambiará la información sobre HTTP en formato XML o JSON (Mateos Diaz, 2005). En función de ello, podemos distinguir diversos tipos de servicios web:

- SOAP: fue creado por Microsoft, IBM y otros y en la actualidad opera bajo el auspicio de W3C. Utiliza HTTP como protocolo de transporte y los mensajes usan un formato XML. Aunque SOAP está muy extendido, no es adecuado para su uso en Android debido a su complejidad que provoca un menor rendimiento que otras alternativas.
- XML-RPC: se trata de un protocolo muy simple que usa el formato XML como codificación de los datos y HTTP como protocolo de transmisión de mensajes. Su simplicidad hace que sólo se definan unos pocos tipos de datos y comandos.
- AMF: se trata de un formato de envío de mensajes utilizado para serializar objetos codificados en XML.
- REST: El término REST se refiere a una arquitectura y no a un protocolo (como en SOAP). Hace uso directo del protocolo HTTP. Este enfoque de comunicación implica el seguimiento de las normas web, que implican transporte de datos mediante http, haciendo uso de las operaciones propias de este protocolo: GET, POST, PUSH y DELETE, la invocación de servicios mediante un formato de direcciones URI, que identifica de manera unívoca un recurso de Internet a través de un nombre, y la codificación de datos identificada mediante tipos MIME, por ejemplo formatos JSON o XML. Se trata de la tecnología más utilizada, gracias en gran medida a su simplicidad.

Las ventajas de la utilización de REST para su uso como Servicio Web radican en la simplicidad, lo que repercute en un menor tiempo de respuesta y en una disminución de la sobrecarga.

El gran inconveniente es la volatilidad del estado, ya que no se guarda relación entre las distintas peticiones, cada petición se trata de forma independiente.

La plataforma Moodle incluyó el soporte para servicios web en su versión 2.0, por lo que se aprovechará esta funcionalidad para simplificar el desarrollo de la parte servidora, así como la comunicación entre cliente y servidor. Los protocolos soportados son los descritos anteriormente: SOAP, XML-RPC, AMF, REST. Dada su simplicidad y gran rendimiento, he decidido utilizar el formato REST para el servicio web a desarrollar para la comunicación entre cliente y servidor. Respecto

a la codificación de los datos, he decidido realizarla en formato JSON, dada su gran flexibilidad.

Esta combinación tiene un alto grado de utilización entre los servicios web, lo que supone la estandarización del servicio web a desarrollar.

3.4.3 Base de datos

Un SGBD (Sistema Gestor de Bases de Datos) es un sistema computacional que facilita la gestión de las bases de datos.

Una base de datos podría definirse como una colección de datos interrelacionados que son almacenados en un soporte informático. Algunas razones que justifican su uso son su capacidad para almacenar grandes volúmenes de información, la optimización de su gestión, la facilidad para realizar consultas y la exactitud, rapidez y fiabilidad en su administración (Cobo, Gómez, Pérez, & Rocha, 2011). Existen diferentes tipos de bases de datos, entre las que se encuentran las de XML, las orientadas a objetos y las relaciones.

3.4.3.1 Bases de datos orientadas a objetos

Un sistema gestor de bases de datos orientadas a objetos (SGBDO), del inglés OBDMS (Objetc DataBase Management System) permite trabajar con objetos complejos, herencias y otras características propias de los lenguajes de programación orientados a objetos (Ramakrishnan & Gehrke, 2003).

Utiliza un modelo de datos orientado a objetos ODM (Object Data Model) y un lenguaje de consulta orientado a objetos denominado OQL (Object Query Language).

En la actualidad, el modelo de base de datos orientado a objetos no es la opción más utilizada en aplicaciones de procesado de datos debido a su complejidad frente a otros modelos.

3.4.3.2 Bases de datos relacionales

El modelo de datos relacional organiza y representa los datos en forma de tablas o relaciones. El término relación representa una tabla de dos dimensiones formada por filas y columnas de datos. Cada fila de esta tabla contiene un conjunto de valores relacionados entre sí. Dicha tabla tiene un nombre, así como cada una de las columnas que la forman. Estos nombres clarifican el significado de los valores contenidos en la tabla (Rivero Cornelio, Martínez Fuentes, & Reina Julia, 2002).

Adicionalmente, una base de datos relacional tiene los siguientes componentes:

- Estructura de datos: se trata de una colección de objetos abstractos formados por datos, dominios, tuplas, atributos y relaciones.
- Operadores: permiten manipular las estructuras de datos a partir de unas reglas bien definidas. Estos operadores, además del cambio de esquema, son la unión, diferencia, producto cartesiano, proyección y selección, es decir, los primitivos del álgebra relacional para manipulación de datos.
- Definiciones de integridad: conjunto de reglas y conceptos que posibilita expresar qué valores de datos pueden aparecer de forma válida en el esquema.

Se incluyen las claves, la posibilidad de tener valores nulos y la reglas de integridad de claves primarias y de integridad referencial (Rivero Cornelio et al., 2002).

- Integridad de claves primarias: las claves se utilizan como identificadores de las tuplas en una relación dada, ya que a cada valor de una clave le corresponde únicamente una tupla y viceversa. En el modelo relacional, solo se puede encontrar una tupla determinada si se conoce el valor de una clave. Una relación puede disponer de varias claves, aunque se suele emplear siempre la misma como identificador. A esta clave, empleada para identificar una relación, se la denomina clave primaria. El resto de claves se llaman claves secundarias o alternativas.
- Integridad referencial: unas relaciones pueden hacer referencia a otras mediante claves primarias de estas. Adicionalmente al concepto de clave primaria, existe el de clave ajena, que se da en un atributo A de una relación R cuando se requiere que todos los valores de A no nulos existan en la clave primaria de alguna relación que no tiene por qué ser necesariamente distinta de R. Es decir, si A es un conjunto de atributos (A1, A2,... An), la definición anterior es válida si A toma valor nulo cuando alguno de sus componentes sea nulo. Al identificar como clave ajena a un atributo, es recomendable definir las acciones a realizar en caso de intentar actualizar dicho atributo con valores no válidos. Estas acciones dependerán del significado de los datos.

Las bases de datos relacionales presentan unas características diferenciadoras frente al resto de modelos como son atomicidad de los valores de los atributos, la no repetición de tuplas, la no ordenación de tuplas y la no ordenación de los atributos. Debido a esto, la forma en que se almacenan los datos en el modelo de datos relacional no importa, contribuyendo esto a una mayor facilidad en el uso de la base de datos por parte del usuario. Al contrario de lo que ocurría en otros modelos de bases de datos vistos anteriormente, se pueden realizar

consultas para recuperar o almacenar información de forma flexible y con poder sobra la administración de la información. Adicionalmente, durante la fase de diseño de una base de datos relacional, se realiza un proceso de normalización mediante el cual cada relación queda descrita en términos de dependencia. Dicho proceso evita la redundancia de datos, lo que a su vez evita problemas al actualizar los datos y permite proteger la integridad de los mismos.

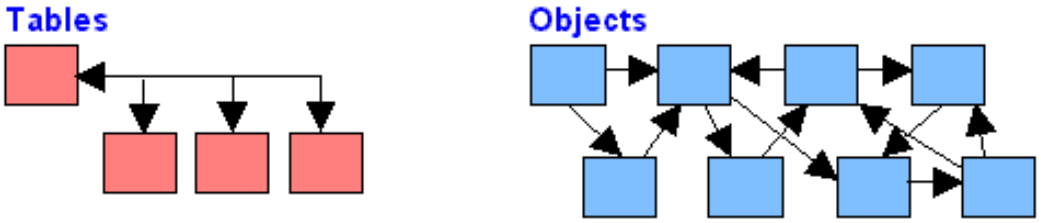


Ilustración 22: Comparativa de relaciones en bases de datos orientadas a objetos y bases de datos relacionales

3.4.3.2.1 MySQL

MySQL es un sistema gestor de bases de datos relacionales, que además ofrece compatibilidad con PHP, Perl, C y HTML, y funciones avanzadas de administración y optimización de bases de datos para facilitar las tareas habituales. Implementa funcionalidades Web, permitiendo un acceso seguro y sencillo a los datos a través de Internet. Este sistema gestor de base de datos incluye capacidades de análisis integradas, servicios de transformación y duplicación de datos y funciones de programación mejoradas.

Se puede decir que MySQL es un sistema cliente servidor de administración de bases de datos relacionales diseñado para el trabajo tanto en los sistemas operativos Windows como en los sistemas UNIX/LINUX. Además, determinadas sentencias de MySQL pueden ser embebidas en código PHP y HTML para diseñar aplicaciones Web dinámicas que incorporan la información de las tablas de MySQL a páginas Web (Pérez López, 2007).

Entre los competidores principales de MySQL, se puede citar a PostgreSQL, Microsoft SQL Server y Oracle. Sin embargo, MySQL dispone de una serie de ventajas que le hacen fuerte frente a sus competidores:

- Alto rendimiento. MySQL es muy rápido.
- Bajo coste. MySQL está disponible de manera gratuita.
- Facilidad de uso.
- Portabilidad. MySQL se puede utilizar en una gran cantidad de sistemas Unix diferentes, así como bajo Microsoft Windows.

- Código fuente accesible. Al igual que en el caso de PHP, se puede obtener y modificar el código fuente de MySQL.

Para el desarrollo de este Trabajo Fin de Grado, se parte de la utilización de tecnologías ya existentes, como la plataforma Moodle y el módulo eLiza. Estas tecnologías utilizan para almacenar la información, una base de datos de modelo relacional MySQL. Por tanto, en la implementación de la parte servidora para la aplicación, se hará uso de dicha base de datos.

Capítulo 4: Método de resolución del problema

Para desarrollar esta aplicación he utilizado el entorno de desarrollo Xcode, el cual incluye el SDK de iOS, entre otros, y con ello todas las librerías necesarias para el desarrollo de aplicaciones sobre el sistema operativo. Además, incluye de forma nativa simuladores de un gran número de dispositivos y versiones del sistema operativo para realizar las pruebas pertinentes. Esta herramienta también ofrece la posibilidad de incluir plug-ins para crear un entorno personalizado al gusto del desarrollador.

Debido a que el módulo de Moodle eLiza utiliza un servidor web, para su desarrollo he elegido el servidor Apache, el cual viene incluido en el núcleo del sistema operativo Mac OS X en sus últimas versiones. El servidor incluye el módulo PHP para interpretar este tipo de ficheros. Para incluir la base de datos, he instalado un servidor MySQL en su versión community.

Para la gestión de bases de datos he utilizado la herramienta phpMyAdmin, escrita en lenguaje PHP para la administración de bases de datos MySQL a través de un navegador. Con phpMyAdmin podemos crear o eliminar bases de datos y tablas, insertar datos, hacer búsquedas, exportar e importar bases de datos.

Para la edición de los documentos del servidor que definen las funciones del servicio web necesario para la comunicación entre el servidor y la aplicación he utilizado el editor de texto enfocado al desarrollo Sublime Text. Su principal característica es la sencillez y facilidad para la edición de archivos, además de su completo marcado de sintaxis y autocompletado. Los ficheros de definición del servicio web están escritos en lenguaje PHP y su función pasa por definir las funciones que recogen los parámetros que se pasan desde la aplicación y a partir de ellos realizan operaciones de inserción y/o consulta en la base de datos para retornar los datos que requiere la aplicación.

Para la depuración de las funciones provistas por el servicio web he utilizado la herramienta Paw, un cliente HTTP y REST que permite realizar peticiones a un servicio REST introduciendo los parámetros necesarios y examinar la respuesta obtenida de manera muy sencilla y rápida.

4.1 Descripción técnica

Este apartado se dividirá en funcionalidades que la aplicación puede realizar. En cada uno de ellos se analizará el funcionamiento de la aplicación y las vistas involucradas.

4.1.1 Acceso de usuarios

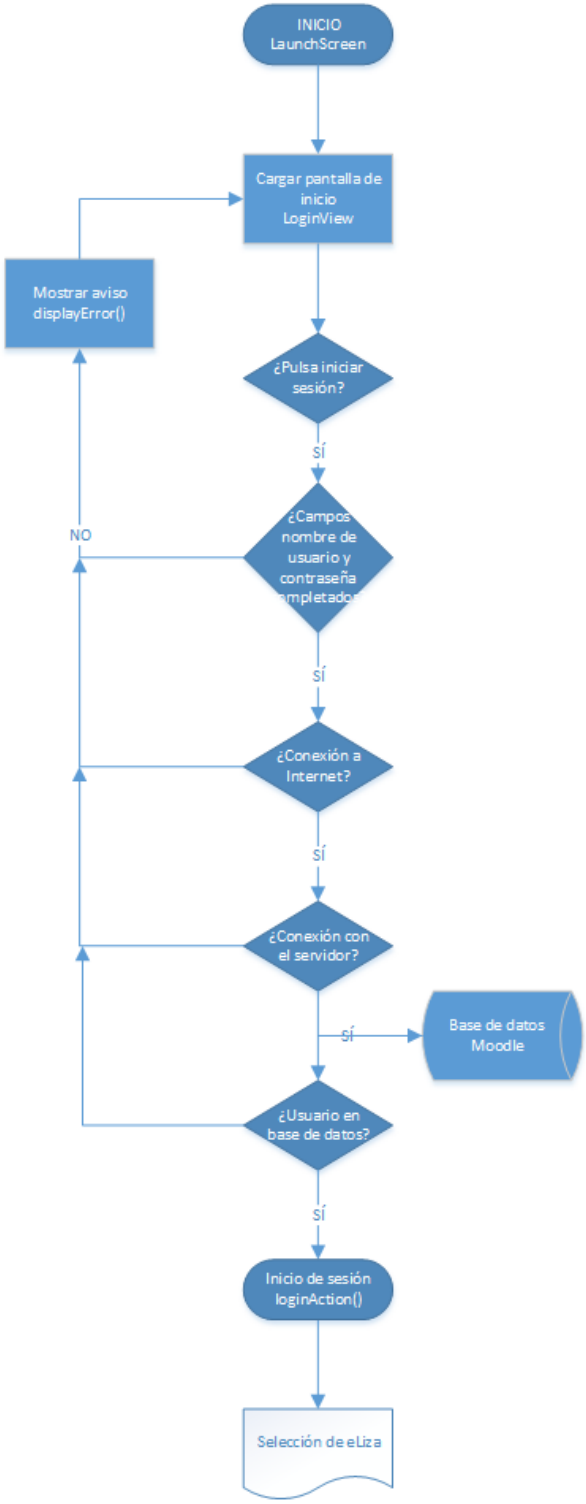


Ilustración 23: Diagrama de flujo de inicio de sesión

Al arrancar la aplicación se ejecuta como primera actividad la vista LaunchScreen. Esta vista muestra al usuario una imagen con el logo de eLiza y el logo de la Universidad de Valladolid. Esta imagen se muestra durante unos instantes durante los cuales se carga e inicia la aplicación.

Tras ello se carga la vista LoginView, la cual muestra una imagen de fondo con el logo de eLiza, además de dos TextField, uno para introducir el nombre de usuario y otro la contraseña. Además se muestra un Button para iniciar las comprobaciones y el inicio de sesión.

Si el usuario pulsa el botón de Iniciar sesión, se ejecuta la función vinculada loginAction(), la cual comienza las comprobaciones. Si alguno de los dos campos de texto no tiene contenido, se muestra un aviso de tipo UIAlertView indicando al usuario que no ha introducido los campos y se sale de la función. Si los campos tienen contenido, se comprueba que hay conexión a Internet mediante la función isConnectedtoNetwork() de la clase auxiliar Reachability, si no la hay se muestra un aviso y se sale de la función. Si hay conexión, se llama a la función retreiveToken(), la cual llama a una función del servicio web del servidor para obtener un token a partir de las credenciales introducidas. Si vence un temporizador de tipo NSTimer, se llama a la función timeout() para lanzar una alerta de tipo UIAlertView avisando de que no hay conexión con el servidor.

4.1.2 Selección de eLiza

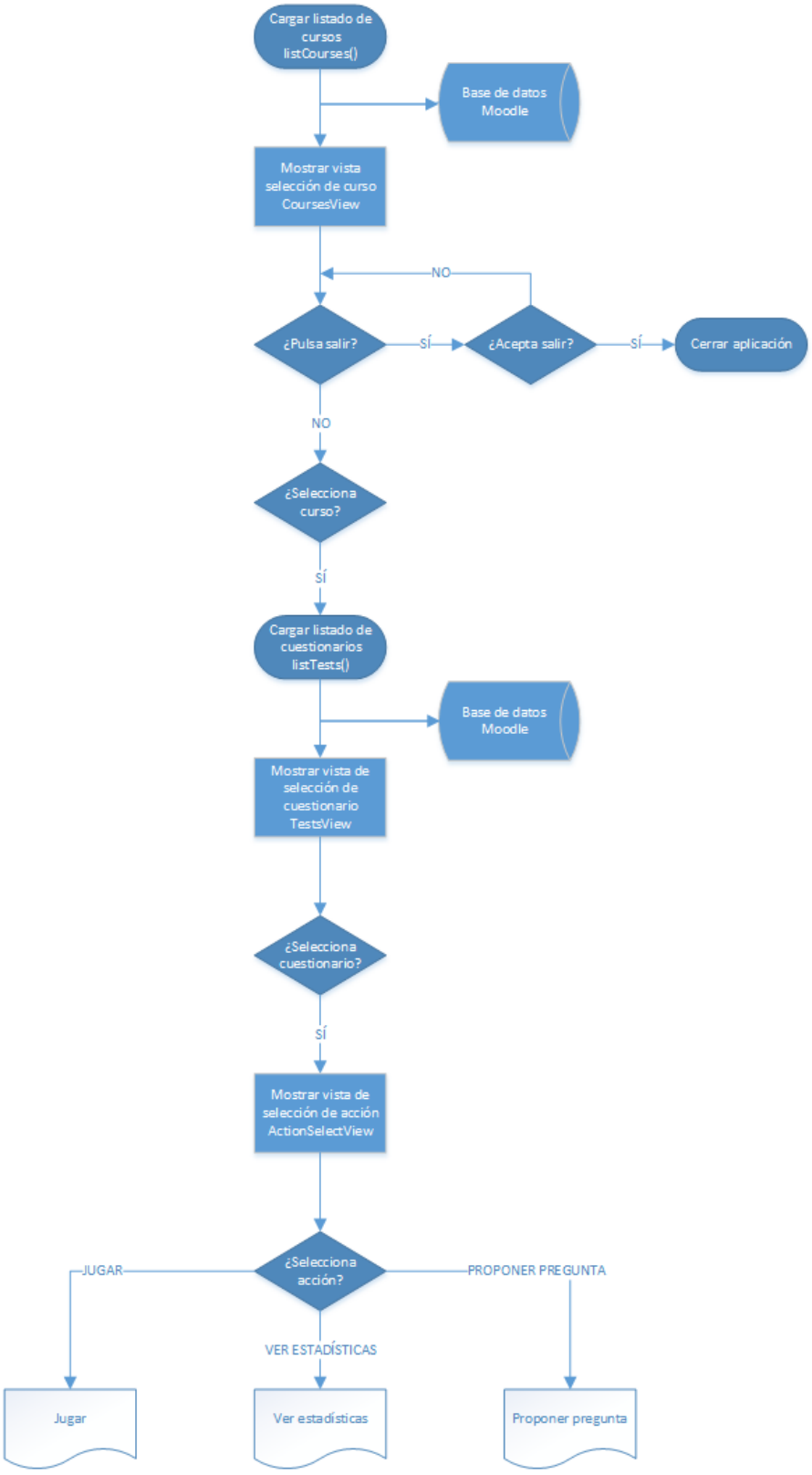


Ilustración 24: Diagrama de flujo de selección de eLiza

Si se recibe correctamente el token, se llama a la función listCourses(), la cual llama a una función del servicio web la cual devuelve una lista con las asignaturas en las que está matriculado el usuario y que a su vez tienen actividades de tipo eLiza.

Estos datos se pasan a la siguiente vista mediante una clase auxiliar. A continuación se ejecuta el segue para pasar a la vista CoursesView, de tipo UITableView. En esta vista se muestran los datos obtenidos en modo de tabla seleccionable, cargando el contenido a partir de una de las definiciones de la función tableView(). También se muestra en la barra de navegación un Button que permite cerrar sesión y salir de la aplicación, si se pulsa este botón se mostrará una alerta que solicita confirmación por parte del usuario, de modo que si pulsa Cancelar pueda volver a la vista y si pulsa Aceptar se cierre la aplicación.

Si se pulsa en una de las filas de la tabla, se selecciona la asignatura que muestra, y se retorna el valor de su identificador mediante otra definición de la función tableView(), este identificador está almacenado en un array de valores.

Con el identificador de curso, se llama a la función listTests(), la cual conecta con el servidor para, mediante una función del servicio web, obtener la lista de actividades de tipo eLiza para esa asignatura. Con esos datos se crea una clase auxiliar para pasar al ejecutar un segue los datos a la siguiente vista.

La siguiente vista, TestsView es también de tipo UITableView, en la que en este caso se muestran la lista de cuestionarios eLiza disponibles.

Si se pulsa una de las filas de la tabla, mediante la función tableView() se retorna el identificador de cuestionario, y ese identificador se pasa a la siguiente vista al ejecutar un segue hacia la vista ActionSelectView.

En la vista ActionSelectView se muestran tres botones de tipo Button, los cuales se muestran como imágenes en lugar de texto. Cada uno de ellos ejecuta una función distinta: un botón para jugar llama a la función play(), un botón para mostrar estadísticas llama a la función viewStatistics() y un botón para la sugerencia de preguntas llama a la función setSuggestion(). Estas funciones llevan a distintas vistas y realizan distintas acciones que se detallan a continuación.

4.1.3 Jugar

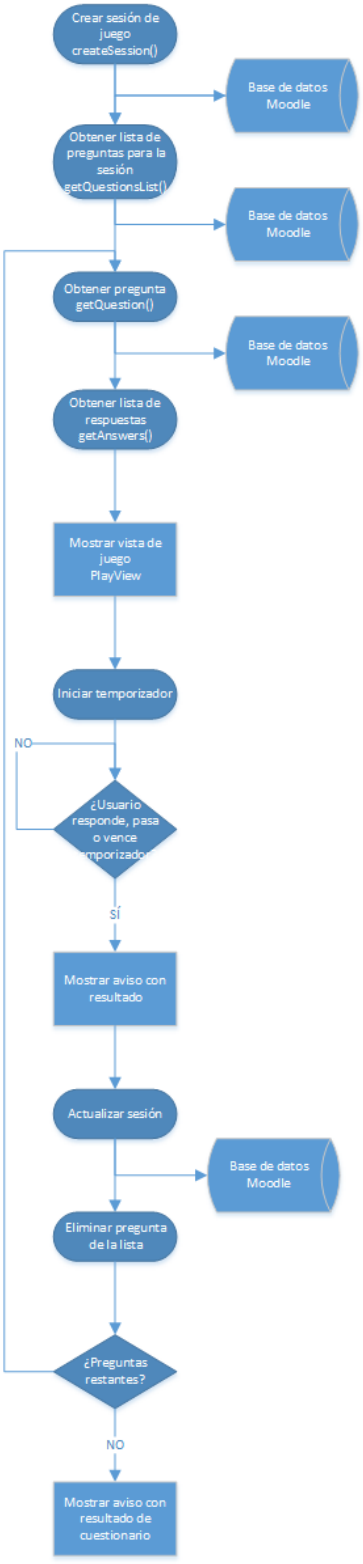


Ilustración 25: Diagrama de flujo de función jugar

Si se llama a la función play() en la vista ActionSelectView, se realiza una conexión con el servidor para crear una sesión mediante la llamada a la función createSession(). De ella se obtiene el identificador de sesión, el cual se pasa como parámetro a la función getQuestionsList(), esta función obtiene del servidor la lista de preguntas que se van a realizar en el cuestionario, en función de la cantidad de preguntas que deben hacerse y el orden que se debe seguir.

Con esa lista de preguntas, se sigue con cada una el siguiente proceso: se llama a la función getQuestion(), la cual obtiene los valores de contenido, puntuación, penalización y tiempo de respuesta para la pregunta actual. Esos valores se muestran en un TextView, y tres Label respectivamente. Después se llama a la función getAnswers(), la cual obtiene a través del servicio web una lista con las respuestas que se aplican a esa pregunta y cuál de ellas es correcta. La información de las respuestas se muestra rellenando una vista UITableView.

A la vez que se muestra todo este contenido, se inicia un temporizador NSTimer con el valor de tiempo de respuesta para la pregunta actual, en el caso de que el tiempo esté limitado. Si este temporizador llega a vencer, se llama a la función timeExpired(), la cual pasa el dato al servidor para almacenar esa pregunta en la base de datos con los valores de pregunta pasada. Se muestra una vista UIAlertView indicando al usuario que el tiempo se ha agotado.

Para que el usuario sea consciente del tiempo restante, al mismo tiempo que el anterior, se activa otro temporizador, este de un segundo y con repeticiones, que con cada vencimiento llama a la función refreshTime(), que se encarga de actualizar el valor de tiempo restante que se muestra en el Label.

Si el usuario pulsa en el botón para pasar pregunta, se llama a la función discardQuestion(), encargada de conectar con el servidor y llamar a la función de pasar pregunta pasando como parámetro el identificador de la pregunta para que se almacene en la base de datos que esa pregunta no se ha acertado ni fallado. En este caso, también se muestra una vista UIAlertView informando de que se ha pasado la pregunta.

Si por el contrario, el usuario contesta a la pregunta pulsando sobre la respuesta, se llama a la función answerQuestion(), pasando como parámetro la respuesta seleccionada. Esta función se encarga de comprobar según los datos obtenidos inicialmente si la respuesta ha sido correcta o no, y enviar los datos al servidor para registrar lo que se ha contestado. La función del servicio web al que se llama devuelve el valor de puntuación de la sesión actualizado, y ese valor se actualiza en el Label que lo muestra. Según se haya acertado o no la pregunta, se muestra una vista UIAlertView informando al usuario del resultado.

Una vez terminado el proceso, se elimina la pregunta de la lista y se comprueba si hay alguna más restante. Si es así se repite el proceso con la siguiente pregunta, actualizando todos los campos de la vista a los datos de la nueva pregunta. Si la pregunta era la última y no queda ninguna en la lista, se llama a la función closeSession(), que se conecta al servidor para que se actualicen los datos de la

sesión en la base de datos y se dé por cerrada. Además, se muestra al usuario una alerta UIAlertView informando de que ha terminado el cuestionario y el resultado final que ha obtenido. Cuando el usuario acepta esta alerta, se lanza un segue para pasar a la vista ActionSelectView, de modo que el usuario pueda volver a jugar o ver las estadísticas actualizadas.

4.1.4 Vista de estadísticas

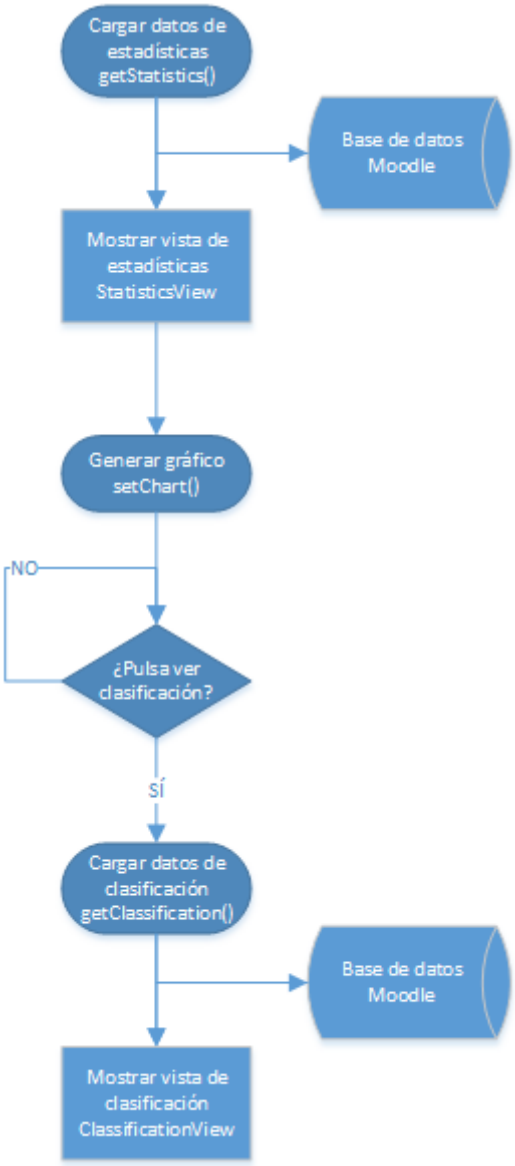


Ilustración 26: Diagrama de flujo de función de vista de estadísticas

Al llamar a la función viewStatistics() en la vista ActionSelectView, se realiza una conexión al servidor, y mediante una función del servicio web se obtienen los datos de preguntas acertadas, falladas, de sesiones iniciadas y terminadas, del tiempo utilizado para la realización de cuestionarios y de las

puntuaciones obtenidas. Para que los datos se obtengan sólo del cuestionario que nos interesa, se pasa como parámetro el identificador del cuestionario.

Con esos datos, se crea una clase auxiliar de tipo StatisticsSenderClass y se ejecuta con ella un segue hacia la vista StatisticsView, de tipo UIView. En ella se muestra una vista de tipo Chart, y varios Labels. En la vista de tipo Chart se genera un gráfico de tipo pieChart, mediante la llamada a la función setChart(), pasándole como parámetros los datos de las respuestas acertadas, falladas y sin contestar, así como cadenas de texto con las etiquetas de cada dato. La llamada a esta función se realiza en la función viewDidLoad(), el método que carga la vista y sus componentes. También en la función viewDidLoad() se calcula el total de preguntas y los porcentajes que suponen las preguntas acertadas, falladas y sin contestar. Estos datos se muestran en los distintos Label de la vista. Por último, se muestra un botón para mostrar la clasificación de ese cuestionario. Si se pulsa este botón, se llama a la función viewClassification(), esta función se conecta al servidor y obtiene un listado de los alumnos que han jugado al cuestionario y su puntuación máxima obtenida. Estos datos se incluyen en un segue hacia la vista ClassificationView, de tipo UIView.

Al cargar la vista ClassificationView, en la función viewDidLoad(), se generan de forma dinámica tantos Label como alumnos con puntuación tenga el cuestionario, y se completan con los datos obtenidos del servidor.

Mediante el botón hacia atrás de la barra de navegación, el usuario puede retornar a la vista de estadísticas o a la de selección de acción.

4.1.5 Envío de sugerencias

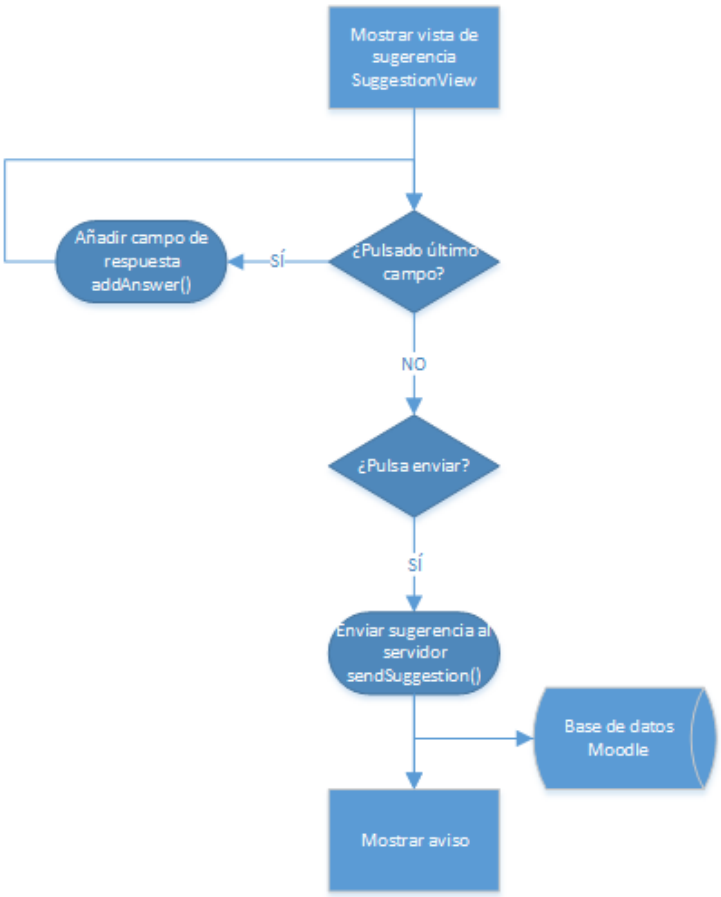


Ilustración 27: Diagrama de flujo de envío de sugerencias

Al llamar a la función setSuggestion() de la vista ActionSelectView, se pasa mediante un segue a la vista SuggestionView, de tipo UIView. Esta vista contiene un campo TextField para incluir el título de la pregunta, un campo TextField para incluir el contenido de la pregunta, y un campo TextField y un Switch para incluir la respuesta y si esta es correcta. Al pulsar sobre este último campo, se incluyen dinámicamente un nuevo TextField y un Switch mediante la función addAnswer(), de modo que se puedan añadir tantas respuestas como se quiera. En la parte inferior de la vista se muestra un botón que con su pulsación invoca a la función sendSuggestion(), la cual se comunica con el servidor pasando como parámetros los datos de la sugerencia para almacenarlo en la base de datos. Si se recibe una respuesta correcta del servidor, se muestra una vista UIAlertView indicando que se ha registrado la sugerencia correctamente y se retorna a la vista de ActionSelectView.

Capítulo 5: Manual de usuario

El objetivo de este capítulo es dar a conocer la aplicación eLiza implementada en este TFG. Se describirá la aplicación y sus funcionalidades.

Mediante capturas de pantalla realizadas en un móvil iPhone 6 con versión de iOS 9.3, se seguirán una serie de pasos descriptivos para que cualquier usuario que quiera hacer uso de la aplicación pueda realizarlo siguiendo este manual.

A lo largo del manual se incluirán también algunas fotografías de la aplicación instaladas en el dispositivo, así podrá observarse cómo se ve la aplicación instalada en un terminal real.

Como introducción, la aplicación eLiza ha sido diseñada para hacer uso de la herramienta de cuestionarios de evaluación eLiza, módulo de la plataforma de aprendizaje en línea Moodle, desarrollada por el Grupo de Telemática e Imagen de la Universidad de Valladolid. Con esta aplicación se pretende dar mayor flexibilidad a los usuarios de la herramienta al poder utilizarla desde su dispositivo móvil sin necesidad de un ordenador.

Para la utilización de esta aplicación es necesario estar dado de alta en la base de datos de la plataforma Moodle. Una vez que el usuario es dado de alta, se le proporciona un nombre de usuario y una contraseña para que pueda acceder a las funcionalidades de esta aplicación.

5.1 Escritorio: Icono de la aplicación

Una vez instalada la aplicación en el terminal, aparecerá en el escritorio del móvil el icono de la aplicación eLiza que ha sido implementada en este TFG, como se aprecia en Ilustración 28

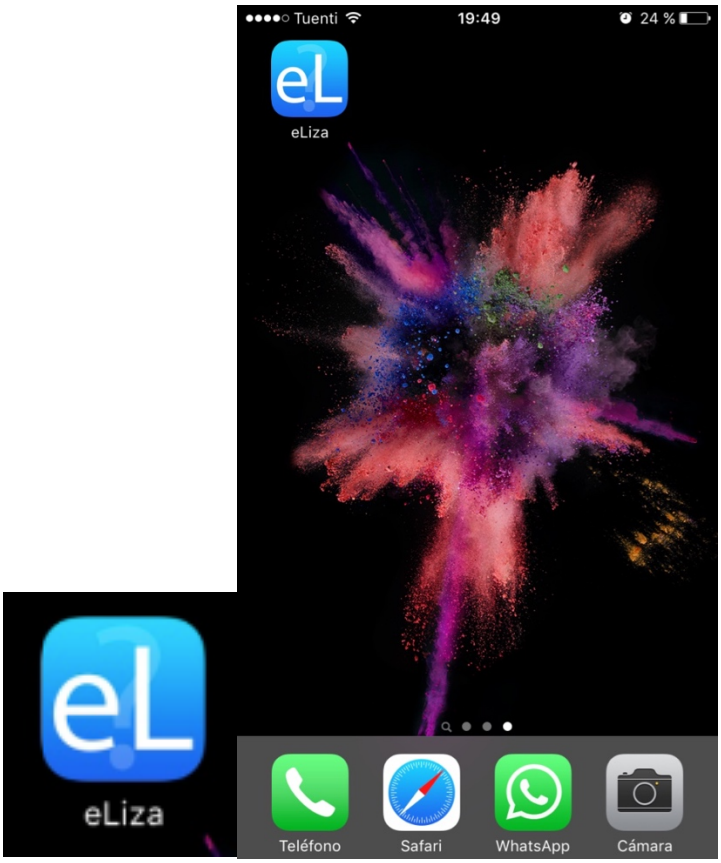


Ilustración 28: Icono de la aplicación eLiza en la pantalla de inicio del dispositivo

Si pulsamos sobre el icono de la aplicación, aparecerá durante unos instantes una pantalla de inicio que incluye los logos de eLiza y de la Universidad de Valladolid, como se puede ver en Ilustración 29

Ilustración 29: Pantalla de bienvenida de la aplicación

5.2 Pantalla de inicio de la aplicación: inicio de sesión

Tras la imagen de carga anterior, se mostrará la pantalla inicial de la aplicación desde la que se podrá acceder a la plataforma. En ella hay, como se puede apreciar en Ilustración 30:

- Un campo de texto para introducir el nombre de usuario.
- Un campo de texto para introducir la contraseña.
- Un botón para iniciar sesión

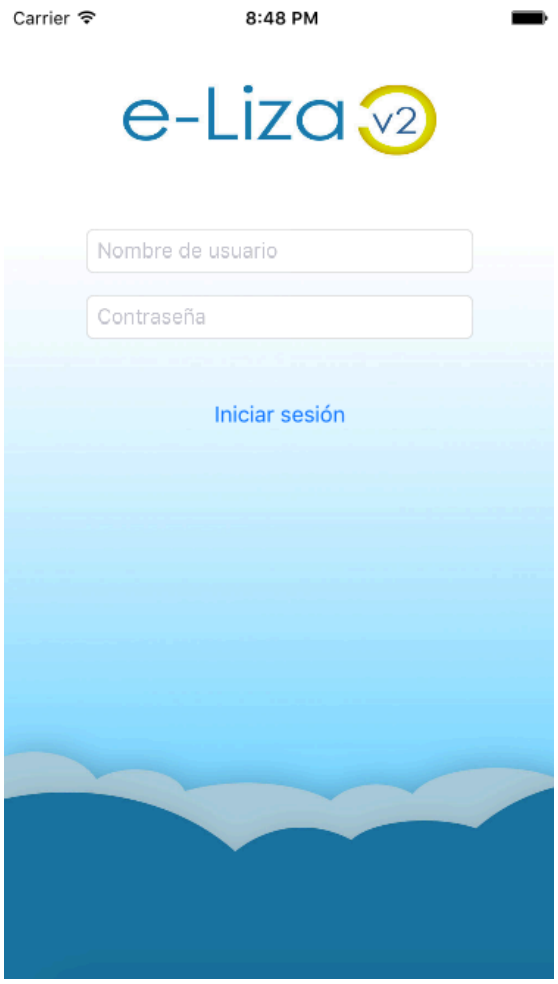


Ilustración 30: Pantalla de inicio de sesión

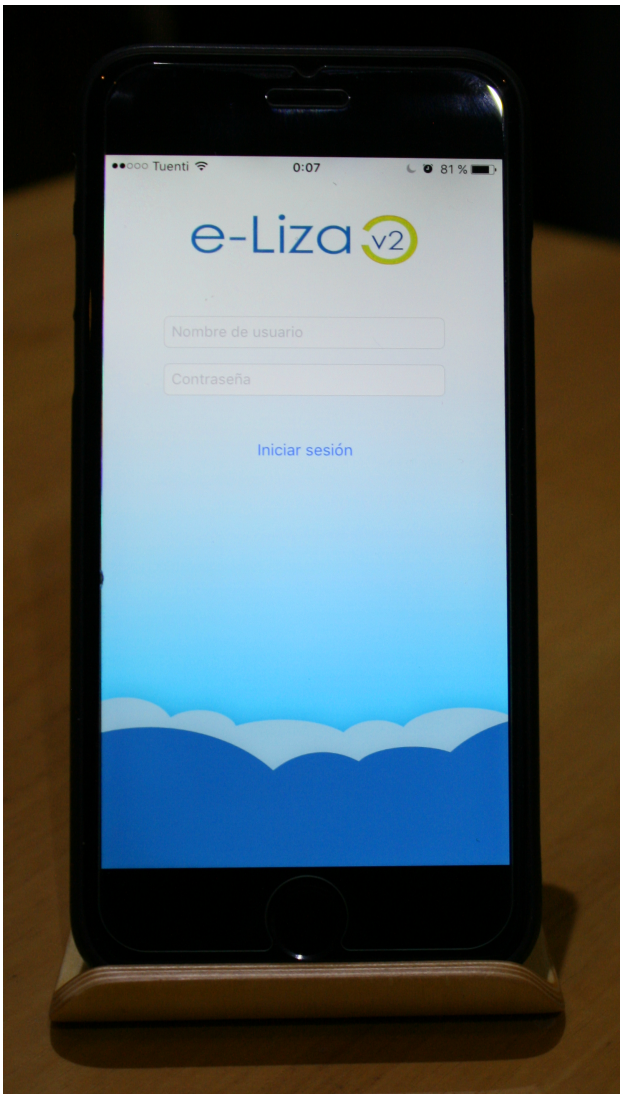


Ilustración 31: Pantalla de inicio de sesión en terminal real

Una vez que el usuario pulsa el botón de iniciar sesión, se realizan varias comprobaciones:

- Si no se ha introducido alguno de los dos campos obligatorios, se mostrará un error informando al usuario de que no ha introducido los datos necesarios. Si se pulsa el botón Aceptar se vuelve a la pantalla de inicio de sesión.
- Si al pulsar iniciar sesión no está activada la conexión a Internet, se mostrará un error informando al usuario de que es necesario estar conectado a Internet para utilizar la aplicación. Si se pulsa el botón Aceptar se vuelve a la pantalla de inicio de sesión.
- Si tras un pequeño periodo de tiempo no se recibe respuesta del servidor, se notifica al usuario para que compruebe si la conexión de su

dispositivo es correcta. Si se pulsa el botón Aceptar se vuelve a la pantalla de inicio de sesión.

- Si tras consultar la base de datos del servidor, se comprueba que las credenciales de acceso no son correctas, se muestra un aviso. Si se pulsa el botón Aceptar se vuelve a la pantalla de inicio de sesión.

Los errores que podrían aparecer se muestran en Ilustración 32

Ilustración 32: Posibles mensajes de error al intentar iniciar sesión

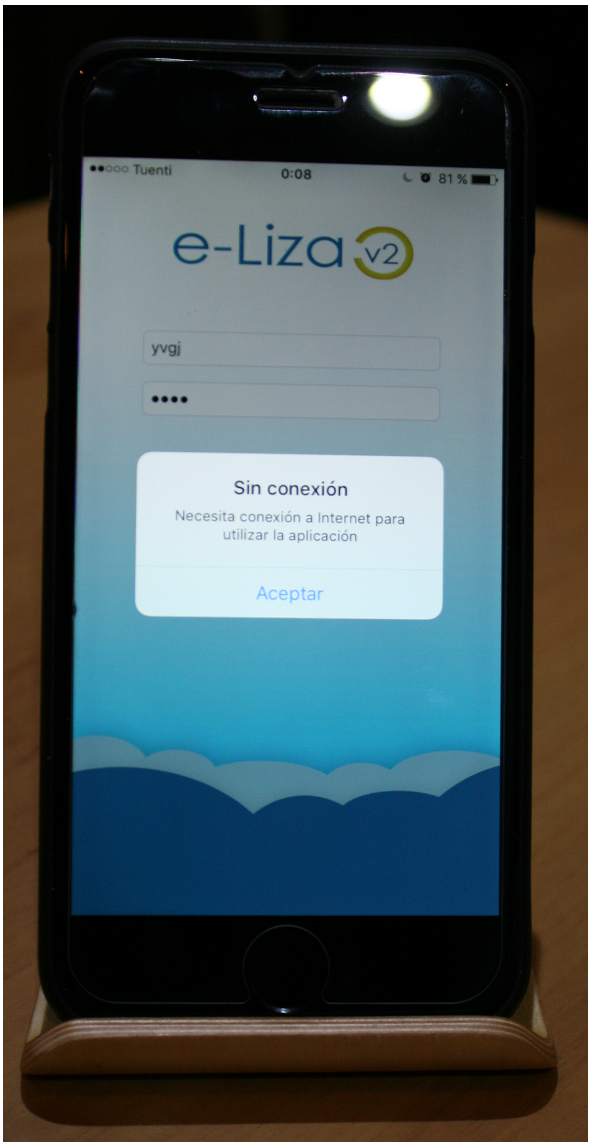


Ilustración 33: Mensaje de error en terminal real

5.3 Selección de cuestionario

Si todo es correcto, se iniciará sesión con las credenciales del usuario. Una vez hecho esto, el usuario puede seleccionar el cuestionario sobre el que quiere actuar. Para ello se muestra una pantalla con un listado de los cursos en que está matriculado el usuario y que tienen algún cuestionario eLiza. También se muestra un botón con el que el usuario puede cerrar sesión y salir de la aplicación.

Si se selecciona una asignatura, se muestra una pantalla en que se puede decidir cuál de los cuestionarios del curso seleccionado se desea elegir.

Ambas pantallas de selección se muestran en Ilustración 34

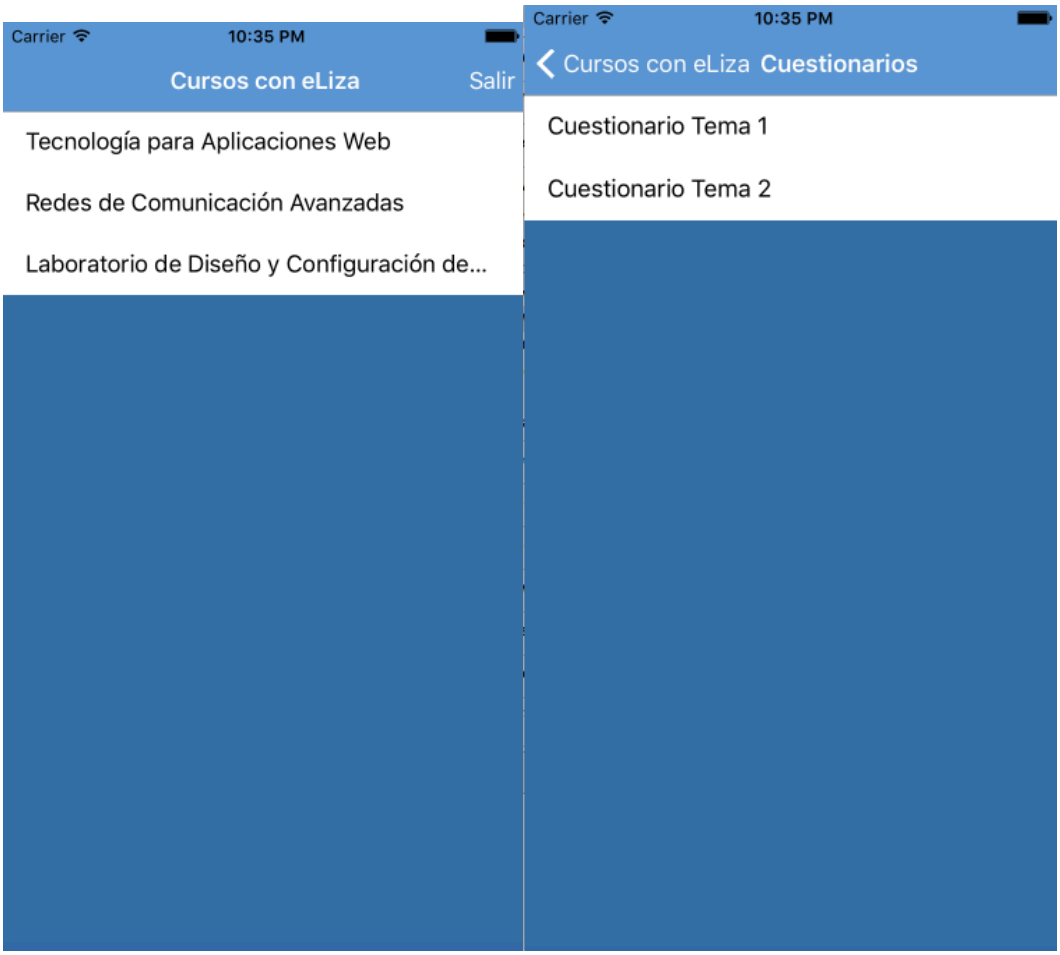


Ilustración 34: Pantallas de selección de asignatura y de cuestionario

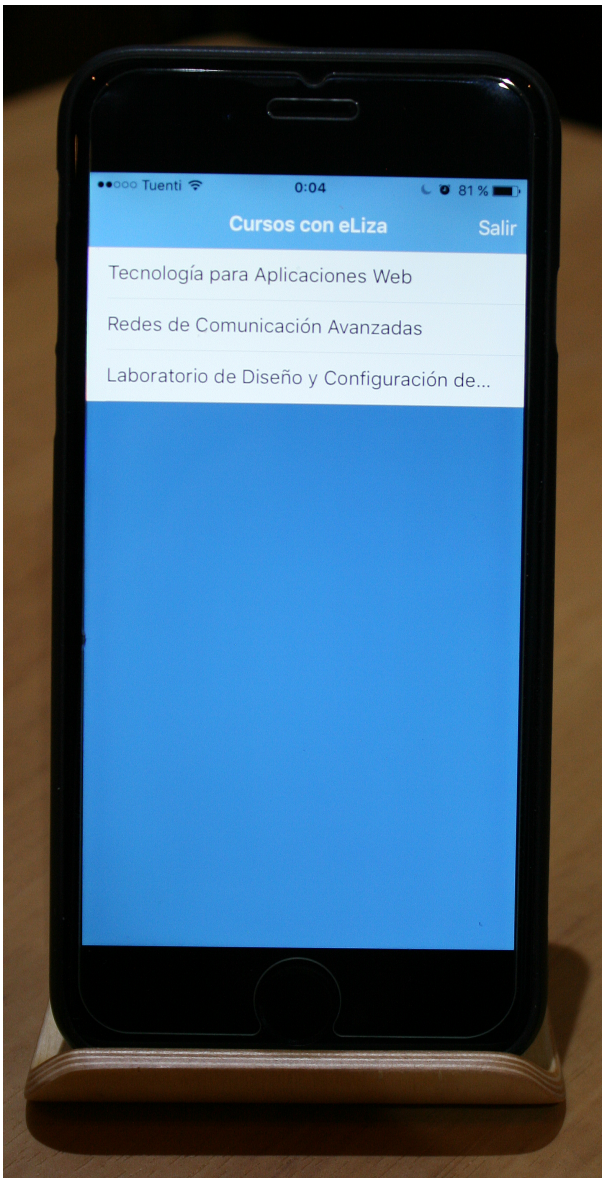


Ilustración 35: Pantalla de selección de asignatura en terminal real

5.4 Selección de acción

Tras elegir uno de los cuestionarios disponibles, el usuario puede seleccionar qué desea hacer: jugar, ver las estadísticas de juego o sugerir preguntas. Para ello dispone de tres botones en la pantalla, como se aprecia en Ilustración 36.

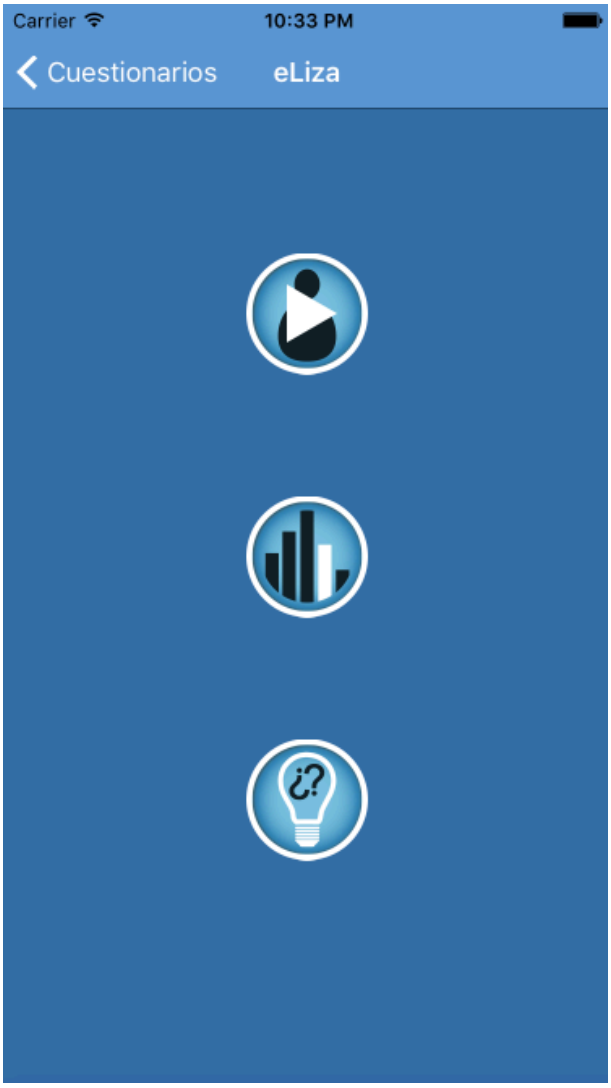


Ilustración 36: Pantalla de selección de acción para un cuestionario

Pulsando en uno de los botones se accede a la acción correspondiente.

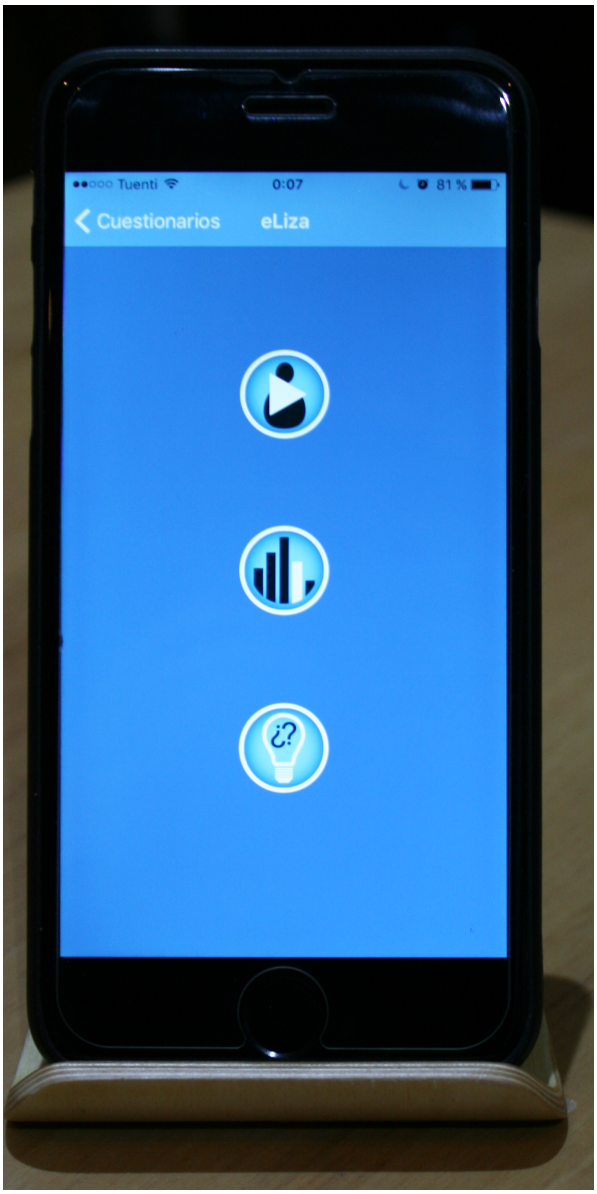


Ilustración 37: Pantalla de selección de acción en terminal real

5.5 Jugar

Al pulsar en jugar, se muestra la pantalla de juego (Ilustración 38). En ella, el usuario puede ver toda la información relativa a la pregunta actual.

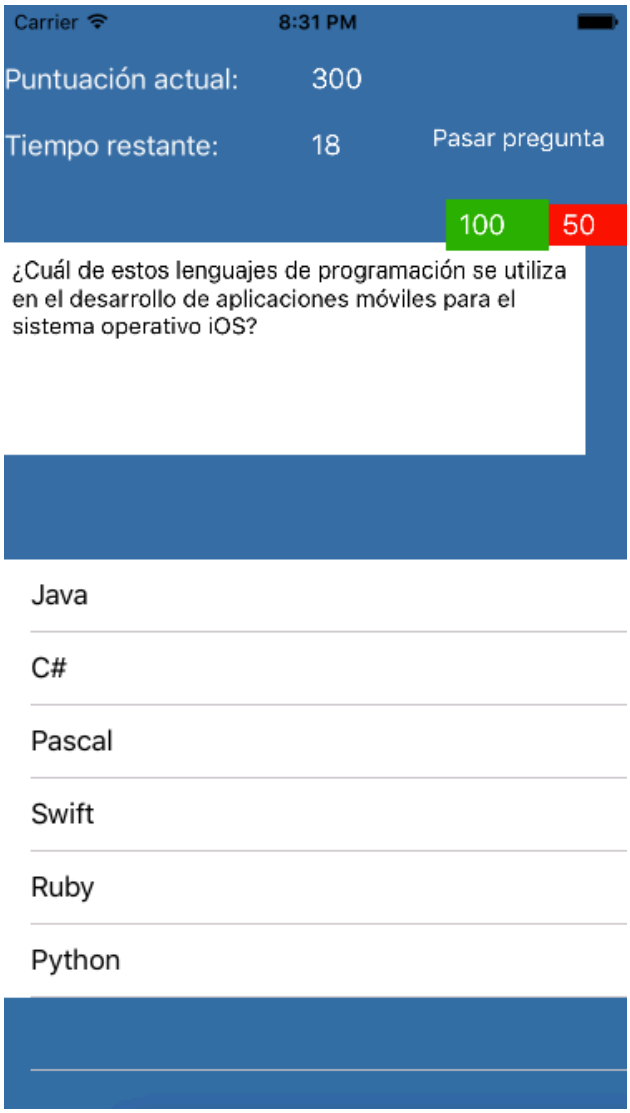


Ilustración 38: Pantalla de juego eLiza

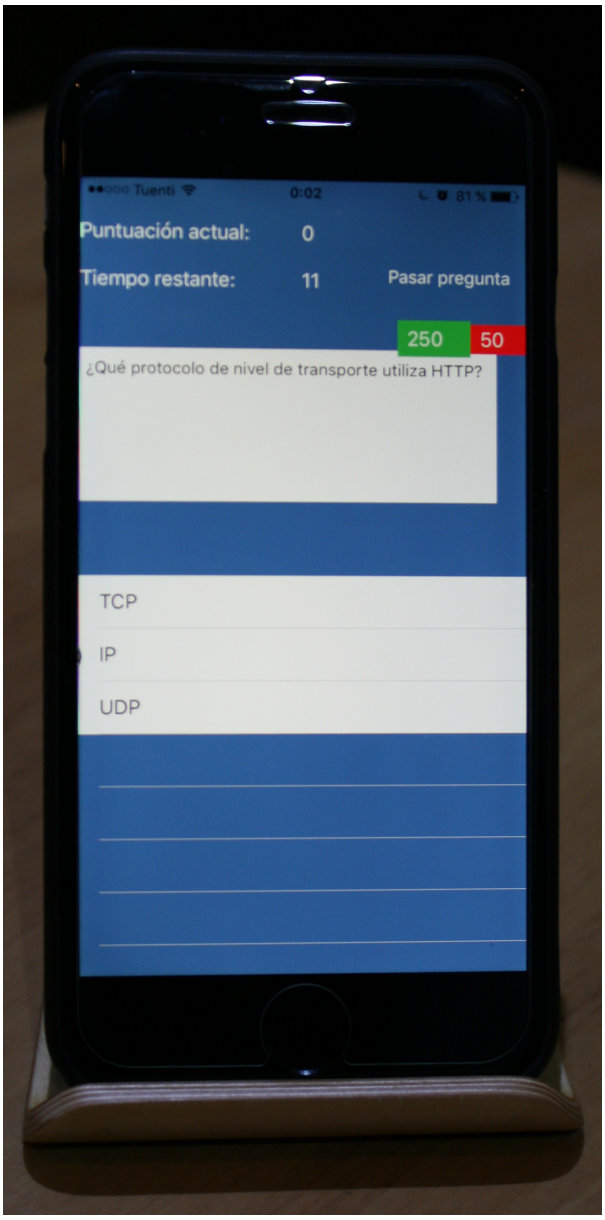


Ilustración 39: Pantalla de juego en terminal real

En la parte superior se muestra la puntuación actual en el cuestionario que se está realizando, además del tiempo restante para contestar a la pregunta actual, el cual irá disminuyendo progresivamente. También se da la opción al usuario de pasar la pregunta actual si no desea contestarla.

Debajo, se muestra el texto de la pregunta actual, y debajo las opciones de respuesta a esa pregunta. Pulsando sobre una de ellas se contesta a la pregunta. Una vez contestada, se muestra al usuario un aviso indicando si la respuesta era acertada o errónea.

Ilustración 40: Posibles mensajes de resultado al responder una pregunta

Si se agota el tiempo marcado para la respuesta, automáticamente se muestra un aviso indicándolo y no se puntúa dicha pregunta.

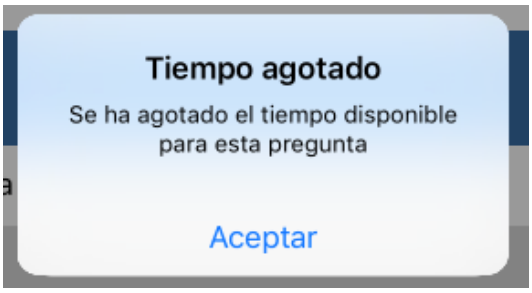


Ilustración 41: Aviso de tiempo agotado

Los posibles mensajes para la resolución de una pregunta se muestran en Ilustración 40 e Ilustración 41

Tras completar una pregunta, bien sea por contestarla, por pasarlo o porque se haya agotado el tiempo, si hay más preguntas en el cuestionario se mostrará la siguiente, actualizando los campos de la pantalla a los nuevos valores. Si por el contrario, la pregunta era la última del cuestionario, se muestra un aviso al usuario indicando que lo ha finalizado y la puntuación obtenida, como se muestra en Ilustración 42.

Ilustración 42: Aviso al completar un cuestionario

Tras terminar el cuestionario, se retorna a la pantalla de toma de decisión de acción a ejecutar del cuestionario, para que el usuario pueda ver la información de sus estadísticas actualizada o volver a jugar.

5.6 Estadísticas

Al seleccionar la acción de estadísticas, se muestra una pantalla con las estadísticas conseguidas por el usuario en el eLiza seleccionado.

Un ejemplo de la ventana de estadísticas se puede apreciar en Ilustración 43

En la parte superior se muestra un gráfico con la proporción de respuestas que el usuario ha acertado, ha fallado y ha dejado sin contestar en las diferentes sesiones en que ha jugado.

En la parte inferior se muestras los datos exactos de respuestas acertadas, respuestas falladas, preguntas que se han dejado sin contestar, el número de veces que se ha jugado, las sesiones que no se han cerrado, el tiempo medio invertido en contestar a las preguntas, la puntuación media obtenida en las diferentes sesiones y la puntuación máxima en una sesión.

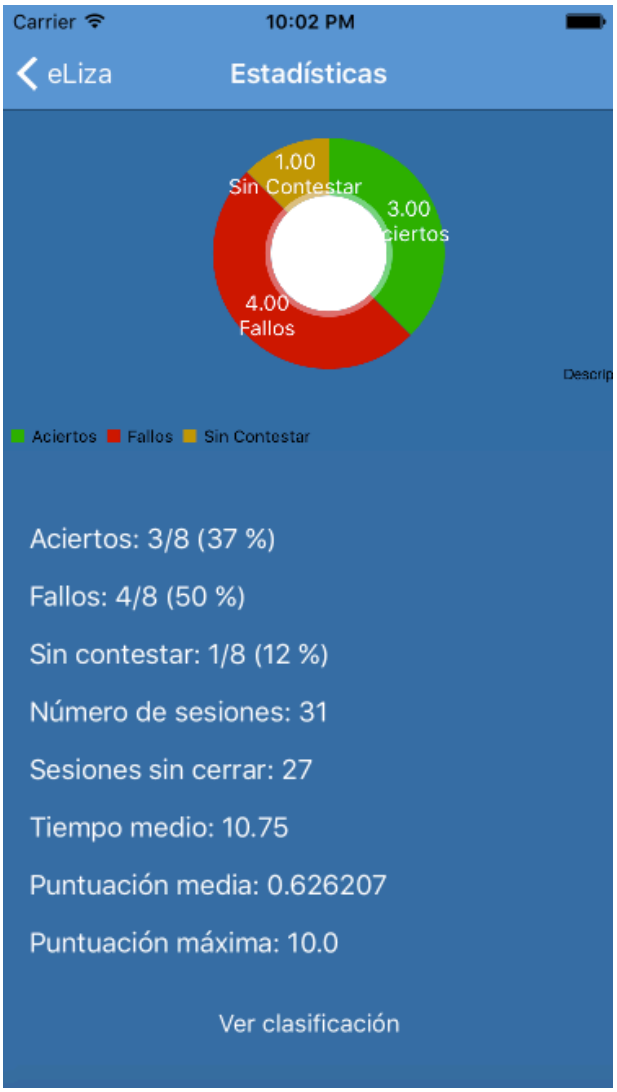


Ilustración 43: Pantalla de estadísticas eLiza para un usuario y cuestionario

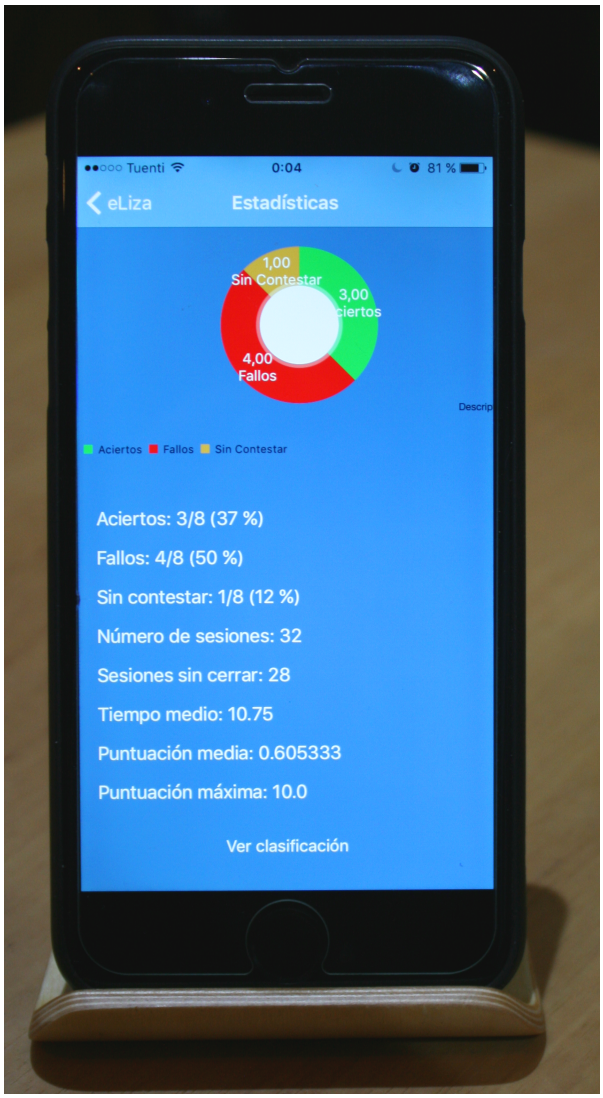


Ilustración 44: Pantalla de estadísticas en terminal real

El usuario también tiene la opción de pulsar el botón de ver clasificación. Esto le lleva a una pantalla (Ilustración 45) en que se muestran las máximas puntuaciones obtenidas por los usuarios que han jugado al mismo cuestionario.

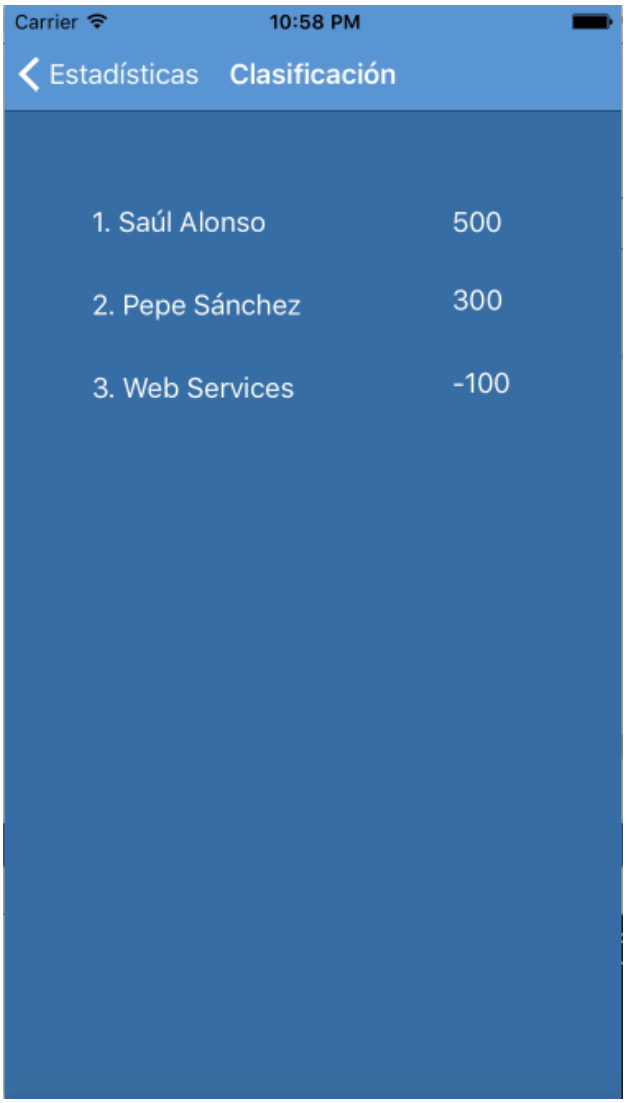


Ilustración 45: Pantalla de clasificación de mejores puntuaciones en un cuestionario

5.7 Sugerencias

Al pulsar en el botón de sugerencias, el usuario accede a una pantalla en la que puede rellenar los campos necesarios para sugerir una nueva pregunta para el cuestionario. Estos campos son el título de la pregunta, el contenido de la pregunta y una serie de respuestas, tantas como quiera, indicando cuál de ellas es correcta.

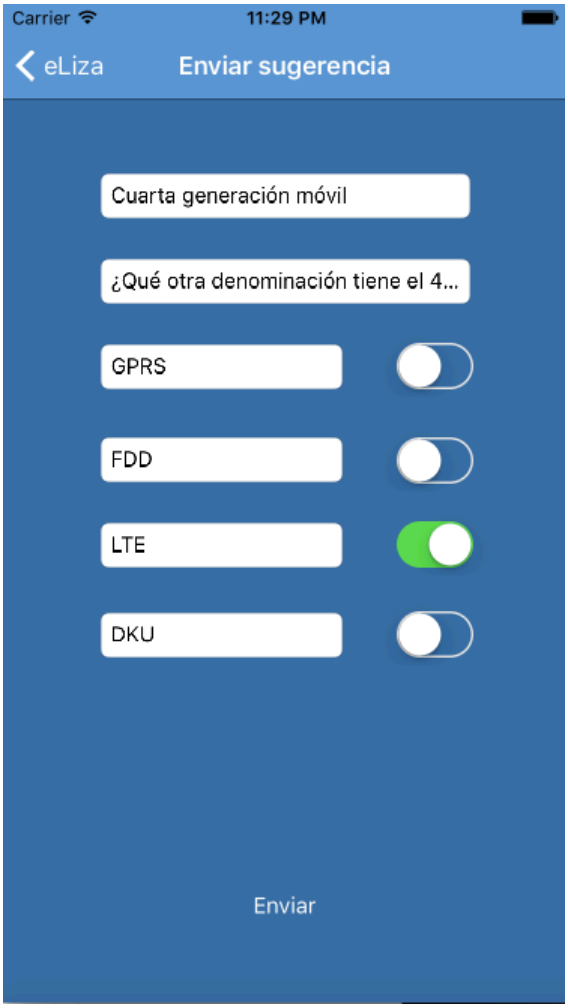


Ilustración 46: Pantalla de envío de sugerencias

Una vez que se han completado los campos, el usuario puede guardar la pregunta pulsando el botón Enviar. Si no ha completado los campos obligatorios, se mostrará un aviso en el que se indica que debe rellenarlos. Si los campos son correctos y se ha guardado correctamente la pregunta, se informará al usuario de este hecho.

Capítulo 6: Estudio económico

Para lograr los objetivos propuestos siguiendo las diferentes fases y métodos ya mencionados, es necesario hacer uso de elementos hardware:

- Un ordenador portátil MacBook Pro para la fase de desarrollo de la aplicación, con un mínimo de 4GB de memoria RAM y un procesador de doble núcleo. La valoración de estos equipos parte de aproximadamente 1300 euros.
- Un teléfono iPhone con soporte para la versión del sistema operativo iOS 9, lo que equivale a un iPhone 4S o superior. La valoración de estos terminales varía en torno a 500 euros.

Dado que la dedicación de los equipos no es exclusiva para este proyecto, si no que pueden utilizarse en otros o como dispositivos de uso personal, computaremos una quinta parte de su valor como gastos asociados a este proyecto.

No se computan costes asociados a software y licencias, ni a la cuenta de desarrollador de la plataforma, ya que es posible realizar el desarrollo de la aplicación con software gratuito y con la cuenta de desarrollador básica, también gratuita.

Se estima que la realización de este proyecto ha supuesto aproximadamente 200 horas, excluyendo de este cálculo las horas que he invertido en aprendizaje. Solamente computo las horas que hubiese necesitado de haber sabido cómo desarrollar la aplicación en cada momento. Teniendo en cuenta el salario medio de un programador Junior, dada mi poca experiencia al inicio del proyecto, valoro las horas trabajadas en 10 euros.

Por tanto valoro el coste económico del desarrollo de la aplicación (únicamente cubriendo gastos asociados al tiempo de realización de la misma) en 2400 euros.

.

Capítulo 7: Conclusiones y líneas futuras

En este capítulo se abordan las conclusiones obtenidas del desarrollo de este Trabajo Fin de Grado, así como algunas orientaciones para su perfeccionamiento en un futuro.

7.1 Conclusiones

Este Trabajo Fin de Grado surge de la actual necesidad de portar las plataformas Web o aplicaciones de escritorio a aplicaciones adaptadas a dispositivos móviles. Esta necesidad nace ante la gran cantidad de personas que hoy en día tienen un smartphone como dispositivo móvil. La mentalidad de la población ha cambiado, actualmente todo el mundo precisa un acceso a la información y servicios de Internet fácil y rápido. De este modo el desarrollo de aplicaciones móviles responde a la demanda de los usuarios satisfaciendo sus necesidades.

La aplicación desarrollada se basa en las ideas del aprendizaje en línea, el aprendizaje a través de dispositivos móviles y el aprendizaje basado en juegos.

En este Trabajo Fin de Grado se han analizado diferentes alternativas en cuanto a tecnologías del lado del cliente y servidor que permiten desarrollar la aplicación deseada, comparándolas y seleccionando las más apropiadas. Se ha utilizado MySQL para la gestión de bases de datos y PHP como lenguaje de programación del lado del servidor por ser las opciones utilizadas por la plataforma de la que se partía. Se ha decidido desarrollar los servicios web de los que hace uso la plataforma con la tecnología REST, y utilizar el lenguaje de programación Swift en el lado del cliente para la creación de una aplicación nativa.

Se ha desarrollado la aplicación incluyendo las funcionalidades que ofrecía la plataforma web de eLiza ya existente en la plataforma de aprendizaje Moodle en una aplicación nativa para sistemas iOS, conservando el método de juego y los recursos accedidos. De este modo, un mismo usuario puede utilizar ambas plataformas indistintamente dotando a eLiza de un mayor acceso y flexibilidad de uso.

Por último, se ha realizado un manual de usuario de la aplicación desarrollada, destinado a los alumnos que vayan a hacer uso de ella, con el fin de que queden patentes todas las posibilidades que permite y se pueda asegurar que todos los usuarios saben explotarlas o disponer de una ayuda adicional en caso de duda.

Con la realización de la aplicación se ha conseguido que ambas plataformas convivan, compartiendo los recursos de las bases de datos que *Moodle* ya usaba. De este modo, la aplicación eLiza para iOS queda lista para ser utilizada por el personal docente de la Universidad de Valladolid.

7.2 Líneas futuras

A lo largo del proceso de desarrollo de este Trabajo Fin de Grado han ido surgiendo nuevas ideas y mejoras que pueden ser incluidas en un futuro, en una nueva versión de la aplicación para mejorar y aumentar sus funcionalidades.

En un principio se planteó un modelo de aplicación caracterizado por la funcionalidad de realizar cuestionarios de eLiza, pero a medida que se fue trabajando en el desarrollo de la aplicación se encontró interesante incluir las funcionalidades de estadísticas y de sugerencia de nuevas preguntas.

Se proponen las siguientes funcionalidades para su implementación en futuras versiones de la aplicación:

- Añadir la posibilidad de juego en grupo para que un usuario pueda crear un grupo de alumnos que compitan entre ellos por conseguir la máxima puntuación en un único intento de contestar el cuestionario. Supondría una mejora respecto al sistema actual de clasificación al limitarse a un grupo seleccionado por el propio alumno, e incitaría a los alumnos a contestar a los cuestionarios para competir entre ellos.
- Incluir la obtención de medallas por la consecución de logros definidos a partir de los resultados conseguidos por el usuario en la realización de cuestionarios. Supondría también una incitación al uso de la aplicación para competir con el resto de usuarios.
- Implementar un sistema de comentarios que permita al alumno indicar objeciones a las preguntas que le son realizadas, si considera que la pregunta no es adecuada, la respuesta marcada como correcta no lo es, etc. Esto permitiría al profesor consultar los comentarios para comprobar si hay algún inconveniente en la pregunta.
- Diseñar una interfaz para el uso por parte del profesorado que permita unas opciones distintas a las del alumno al comprobar que el usuario que ha iniciado sesión tiene el rol de profesor. Las funcionalidades pasarían por una visualización más completa de las estadísticas que suponga una visión general del estado del cuestionario, la posibilidad de aprobar sugerencias enviadas por los alumnos o incluir preguntas definidas por el profesor.
- Dar la posibilidad de compartir los resultados obtenidos en redes sociales, lo que impulsaría la promoción de la aplicación. Se podría publicar la obtención de una puntuación, la consecución de un logro, la posición en clasificación.
- Traducir la aplicación a diversos idiomas expandiría el rango de usuario que podría hacer uso de ella, de modo que pudiera exportarse su utilización a otros ámbitos.

- Implementar un sistema de notificaciones que informe al usuario del lanzamiento de nuevos cuestionarios, nuevos logros, cambios en la clasificación o aprobación de sus sugerencias.

7.3 Experiencia personal

Como experiencia personal la realización de este Trabajo Fin de Grado ha sido muy positiva en varios aspectos.

El inicio del desarrollo fue complicado debido a mi nula experiencia en programación para sistema operativo iOS y en lenguaje Swift, aunque partía de una base teórica adquirida en la asignatura de Desarrollo de Aplicaciones para Dispositivos Móviles. Sin embargo, esto ha supuesto que mi evolución en este aspecto haya sido muy grande a lo largo de la realización de este proyecto. A la conclusión del mismo, considero que ha sido mucha la experiencia adquirida y me considero capaz de enfrentarme a la realización de una aplicación iOS más compleja.

Durante el desarrollo de este trabajo, han sido numerosos los errores cometidos que he tenido que corregir sobre mi propio código, así como las modificaciones para realizar mejoras detectadas. Esto ha mejorado mis habilidades de auto aprendizaje, y me ha hecho aprender a diseñar la estructura de la aplicación antes de comenzar a desarrollar el código. Es una metodología que no seguía antes de comenzar a desarrollar este trabajo y que, tras ello, me resulta imprescindible.

En definitiva, creo que he madurado como programador tanto técnicamente como en labores de planificación.

Capítulo 8: Bibliografía

Antón Rodríguez, M. (2014). *Introducción a iOS.* Valladolid: Universidad de Valladolid.

Antón Rodríguez, M., & Pérez Juárez, M. Á. (2015). *Introducción a PHP*. Valladolid: Universidad de Valladolid.

appfigures. (15 de Abril de 2015). *appfigures*. Recuperado el 2016, de appfigures: http://www.appfigures.com

Apple. (2015). *Choosing a membership*. Obtenido de Apple Developer: https://developer.apple.com/support/compare-memberships/

Apple. (2015). *The Swift Programming Language.* California.

Apple. (9 de Mayo de 2016). *App Store Support*. Obtenido de Apple Developers: https://developer.apple.com/support/app-store/

Barberá, E. (2008). *Aprender e-Learning.* Barcelona: Paidós Ibérica.

Cañellas Mayor, A. (2010). CMS, LMS y LCMS. *Comunicación y Pedagogía*, 12-18.

Google. (2016). *Platform Version*. Obtenido de Android Developers: https://developer.android.com

Kochan, S. G. (2011). *Programming in Objective-C.* Estados Unidos: Pearson Education.

Macprogramadores. (2013). *Macprogramadores*. Obtenido de Macprogramadores: http://www.macprogramadores.com

Microsoft. (2016). *Centro de desarrollo de Windows*. Obtenido de Recursos para desarrolladores Windows: https://developer.microsoft.com/es-es/windows

Moodle. (11 de Julio de 2016). *Moodle*. Recuperado el 2016, de Moodle: http://www.moodle.org

Ortiz, A. (10 de Enero de 2013). *Juego de tronos entre fabricantes y plataformas móviles*. Recuperado el 2016, de Xataka: http://www.xataka.com/moviles/juego-de-tronos-entre-fabricantes-y-plataformas-moviles-tecnologia-2013

Peter Vorderer, U. R. (2009). *Serious Games: Mechanisms and Effects.* Londres: Routledge.

Primeau, L. (14 de Septiembre de 2012). *Definición de los términos LCMS y LMS: LCMS y LMS, ¿en qué se diferencian?* Recuperado el 2016, de e-doceo: http://es.e-doceo.net

Rivero Cornelio, E., Martínez Fuentes, L., & Reina Julia, L. (2002). *Introducción al SQL para usuarios y programadores.* Madrid: Paraninfo.

Rodríguez Cayetano, M. (2014). *Introducción al sistema operativo Android.* Valladolid: Universidad de Valladolid.

Romero Oraá, R. (2014). *Trabajo Fin de Grado: Desarrollo del nuevo módulo para Moodle e-Liza v2 para la autoevaluación de conocimientos en apoyo al Grupo de Telemática e Imagen.* Valladolid: Universidad de Valladolid.

Vision Mobile. (Marzo de 2016). Developer Economics: State of Developer Nation Q1 2016.

W3C. (2015). *W3C Consortium*. Recuperado el Abril de 2015, de http://www.w3c.es/

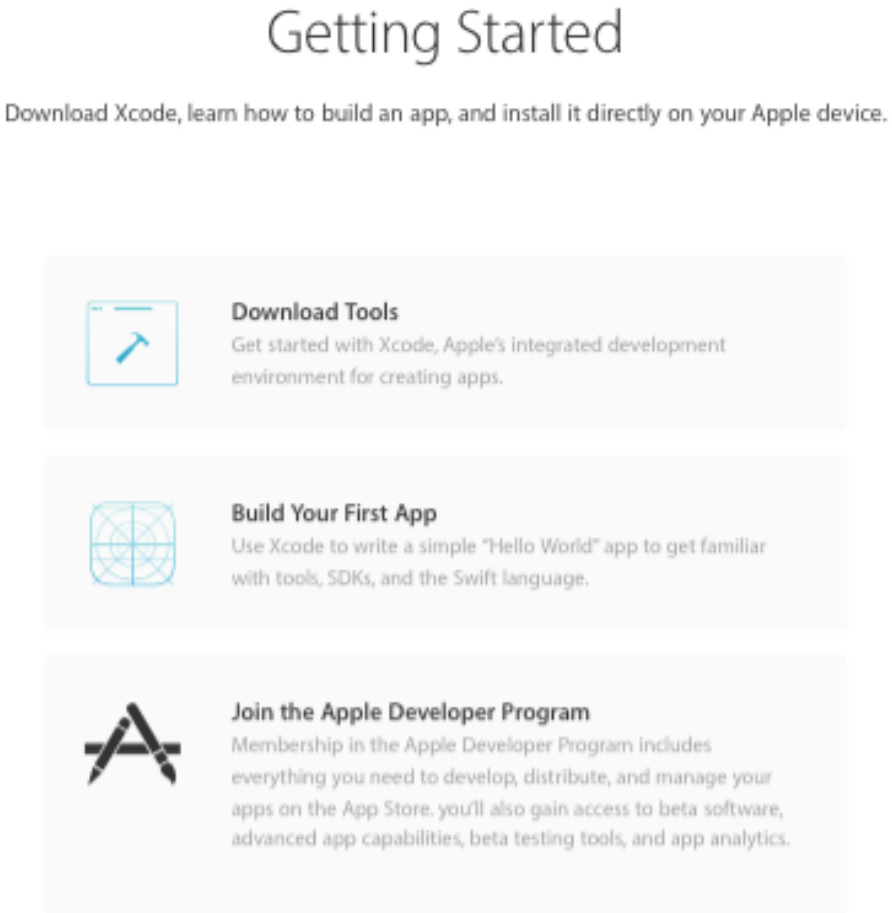
ANEXO: Despliegue de aplicación en terminal

En 2015, Apple renovó su oferta para desarrolladores con el lanzamiento de Swift 2, incluyendo la posibilidad de probar aplicaciones en terminales físicos con la cuenta de desarrollador gratuita. Cualquier persona con un ID de Apple puede empezar a desarrollar para sus plataformas y puede correr aplicaciones en dispositivos físicos.

Para poder realizar estas pruebas, necesitamos seguir los siguientes pasos:

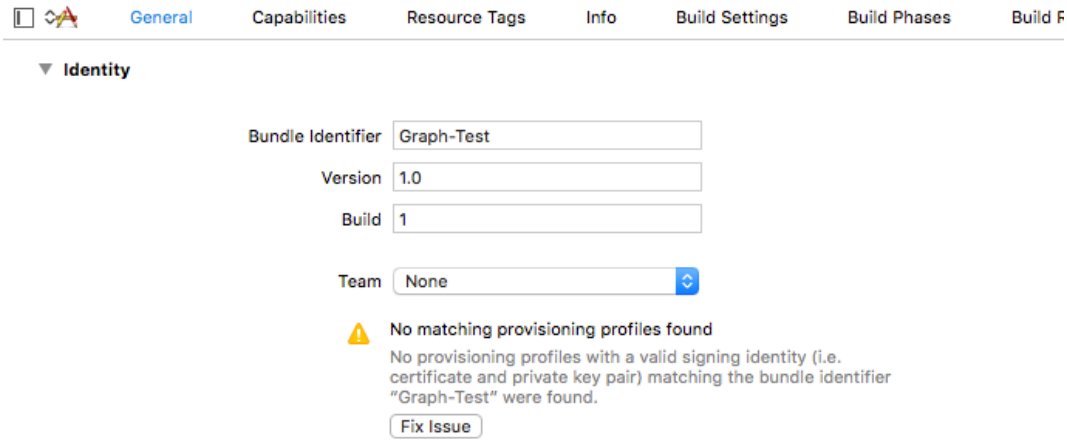
Paso 1: Registrarse como desarrollador

En el sitio web de desarrolladores de Apple (http://developer.apple.com), acudiendo al apartado de cuentas, se nos permite registrarnos como desarrolladores, tras aceptar los términos y condiciones.

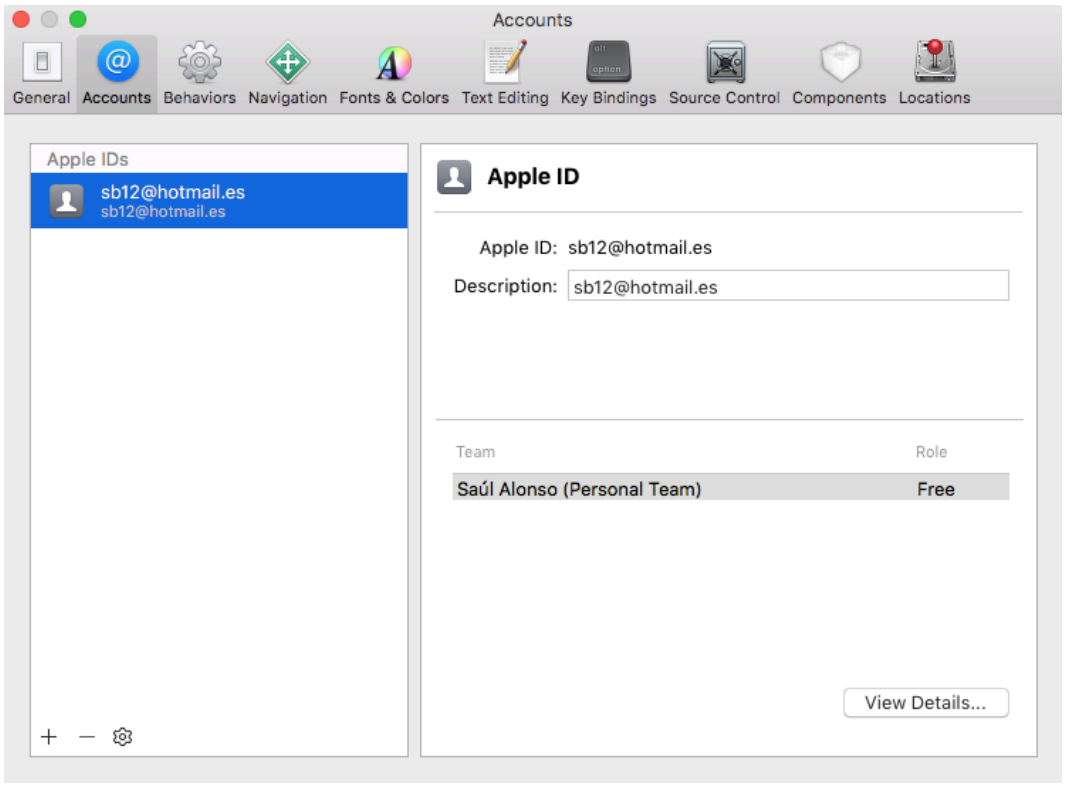


Paso 2: Asignar cuenta al proyecto

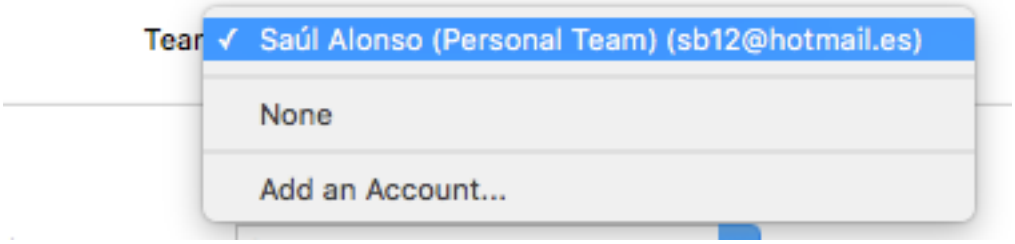
Tras haber completado el registro como desarrollador y tener un proyecto creado, debemos asignar el Apple ID a ese proyecto. Para ello, nos dirigimos a la pestaña General del proyecto en Xcode. Se mostrará una advertencia indicando que no hay ninguna cuenta de desarrollador introducida.



Para solucionar este problema, entramos en la configuración de Xcode, en el apartado de cuentas y pulsamos en añadir Apple ID. Introducimos los datos de la cuenta de desarrollador y quedará registrada esa cuenta.

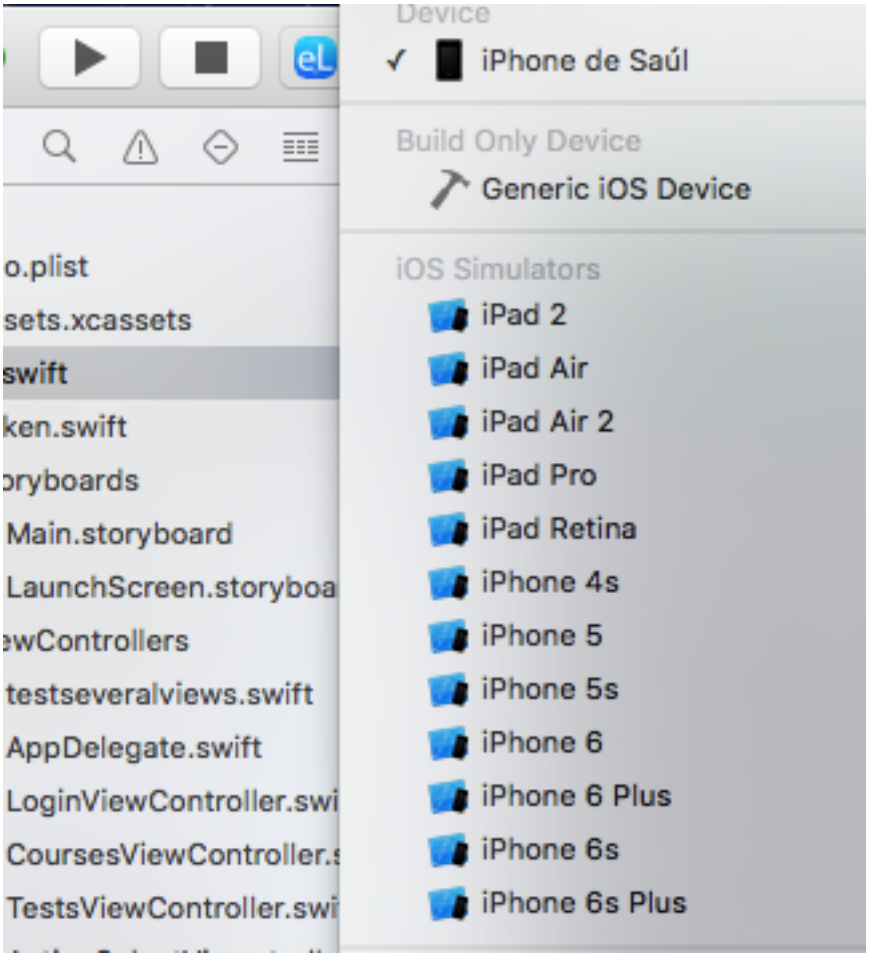


Si volvemos a la pestaña General del proyecto, ahora nos permitirá asignar la cuenta introducida a ese proyecto.



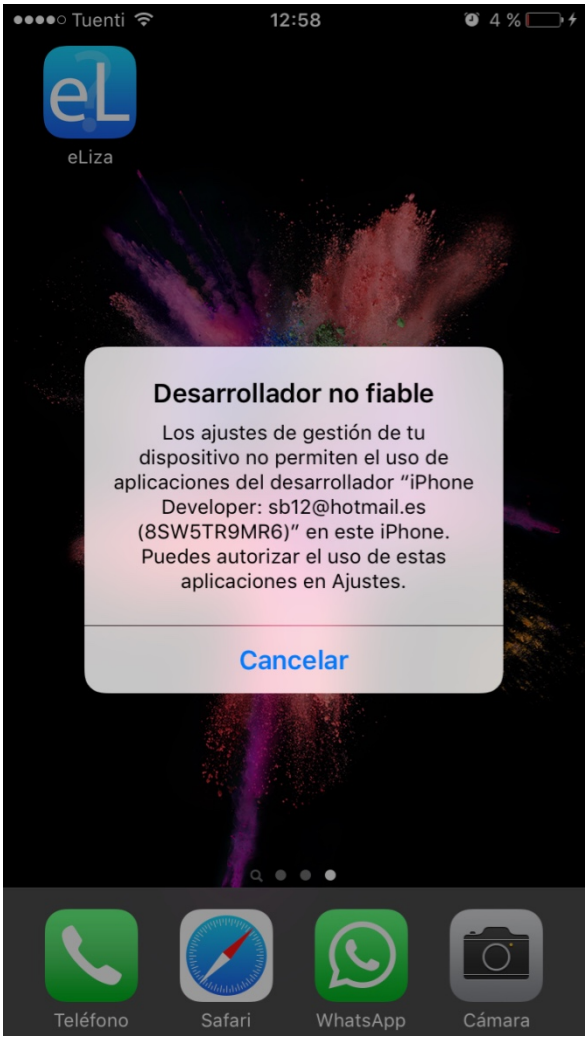
Paso 3: Lanzar aplicación en dispositivo

Entre los posibles destinos del despliegue de la aplicación, junto a numerosos emuladores, si conectamos el dispositivo al ordenador, se nos dará la opción de lanzar la aplicación en él.

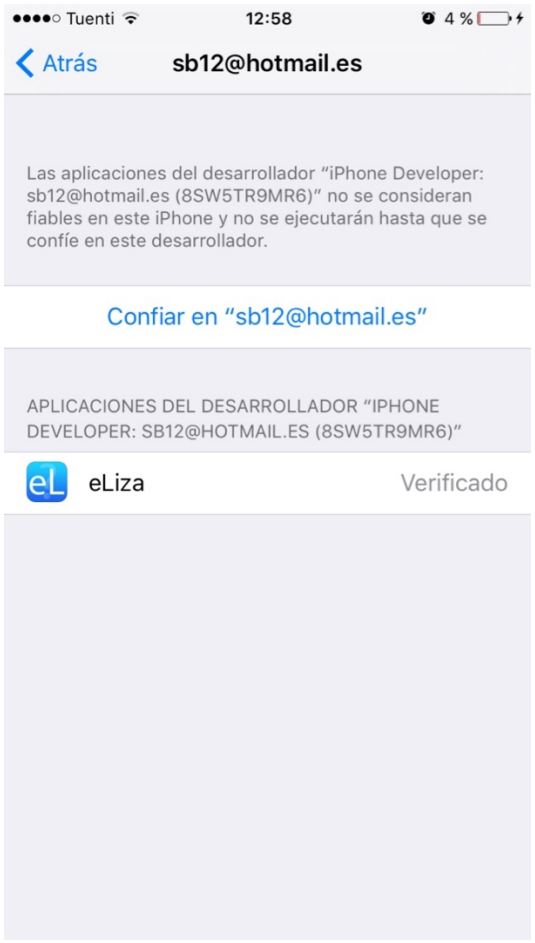


Paso 4: Confiar en desarrollador

Al lanzar la aplicación en nuestro dispositivo, se nos mostrará un mensaje de alerta en el que se indica que el desarrollador de la aplicación no es de confianza.



Para confiar en la cuenta de desarrollador y poder lanzar las aplicaciones firmadas con ella, debemos acudir a los ajustes del dispositivo, al apartado General y dentro de él a Gestión de perfiles y dispositivos. En él, nos aparecerá un nuevo perfil de desarrollador, el cual podemos marcar como de confianza.



Una vez realizados estos pasos, ya podremos lanzar la aplicación desarrollada en nuestro dispositivo.

ANEXO: Instalación de servicio web eLiza en Moodle

Moodle incluye la compatibilidad con servicios web desde su versión 2.0. Para hacer uso de un servicio web desarrollado, es necesario seguir los siguientes pasos:

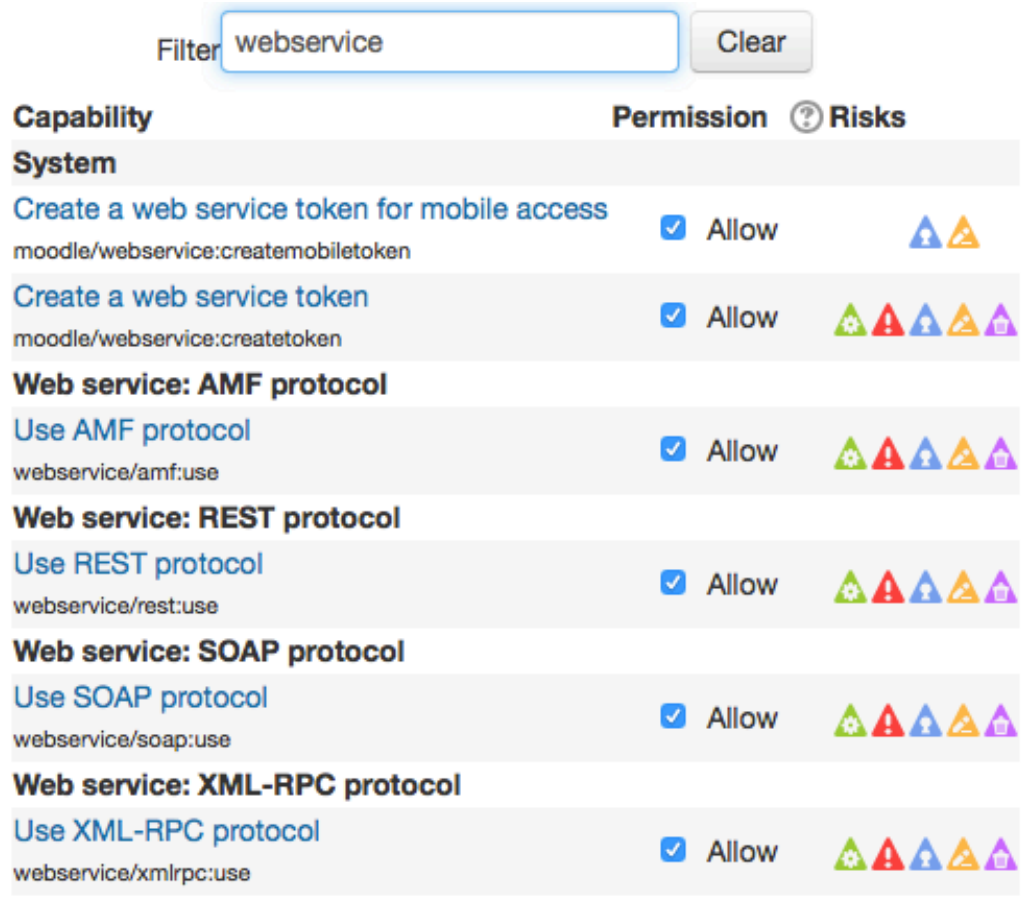
Paso 1: Habilitar servicios web

Para ello, debemos iniciar sesión con la cuenta de administración en Moodle. Desde ella, nos dirigimos a la sección Administración del sitio, Extensiones, Servicios Web, Vista general. En ella se nos muestran los pasos a seguir para habilitar los servicios web. Debemos activar los servicios web y los protocolos que vayamos a utilizar.

Step	Status	Description
1. Enable web services	Yes	Web services must be enabled in Advanced features.
2. Enable protocols	rest soap xmlrpc amf	At least one protocol should be enabled. For security reasons, only protocols that are to be used should be enabled.

Paso 2: Otorgar permisos a los usuarios

Para poder hacer uso de un servicio web, el usuario necesita tener permisos específicos. Para asignar los permisos a un grupo de usuarios, debemos asignárselos al rol que cumplen todos ellos. Para ello nos dirigimos desde la cuenta de administrador a la sección Administración del sitio, Usuarios, Permisos, Definir Roles, seleccionamos el rol al que queremos dar permisos y pulsamos en editar. Debemos asignar los siguientes permisos al rol:

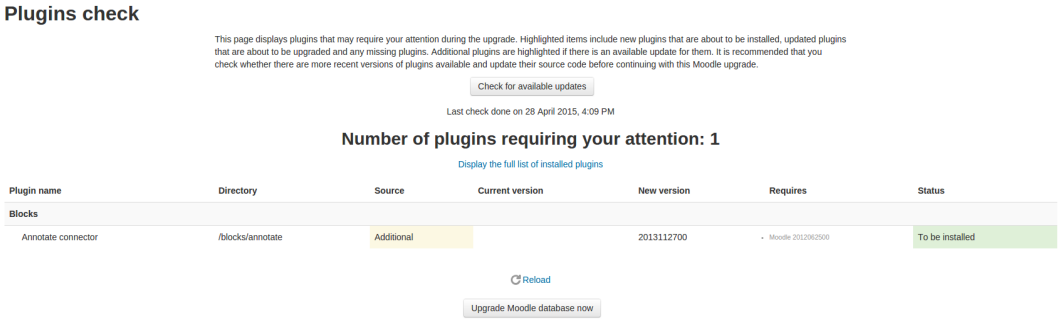


Paso 3: Copia de ficheros en servidor web

Se deben copiar los ficheros de definición del servicio web en el directorio del servidor web. La ruta a este directorio se puede consultar en la definición de la variable DocumentRoot del fichero de configuración del servidor web. Dentro del directorio raíz del servidor, debemos acceder a la carpeta "local" y copiar en ella la carpeta wseliza que contiene todos los ficheros del servicio web.

Paso 4: Activación del servicio web

Al acceder a la plataforma con la cuenta de administrador tras incluir una nueva extensión, se muestra una pantalla solicitando permiso para actualizar la base de datos e incluir la nueva extensión.



Pulsando en actualizar la base de datos la extensión quedará registrada y podremos hacer uso de ella.