

**ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD DE MÁLAGA**



PROYECTO FIN DE CARRERA

*APLICACIÓN DE CONSULTA Y GESTIÓN DEL
CATÁLOGO DE LA BIBLIOTECA DEL DEPARTAMENTO
DE TECNOLOGÍA ELECTRÓNICA DE LA UNIVERSIDAD
DE MÁLAGA*

**INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN
SISTEMAS ELECTRÓNICOS**

MÁLAGA, 2013

JESÚS PÉREZ SERRANO

**ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD DE MÁLAGA**

**Titulación: Ingeniería Técnica de Telecomunicación
*Sistemas Electrónicos***

Reunido el tribunal examinador en el día de la fecha, constituido por:

D./D^a. _____

D./D^a. _____

D./D^a. _____

para juzgar el Proyecto Fin de Carrera titulado:

***APLICACIÓN DE CONSULTA Y GESTIÓN DEL CATÁLOGO DE LA
BIBLIOTECA DEL DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA DE
LA UNIVERSIDAD DE MÁLAGA***

del alumno/a D. *Jesús Pérez Serrano*

dirigido por D. Francisco Javier González Cañete

ACORDÓ POR _____ OTORGAR LA
CALIFICACIÓN DE _____

Y, para que conste, se extiende firmada por los componentes del tribunal, la presente diligencia

Málaga, a _____ de _____ de _____

El/La Presidente/a

El/La Vocal

El/La Secretario/a

Fdo.: _____ Fdo.: _____ Fdo.: _____

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD DE MÁLAGA

**APLICACIÓN DE CONSULTA Y GESTIÓN DEL CATÁLOGO DE LA
BIBLIOTECA DEL DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA DE
LA UNIVERSIDAD DE MÁLAGA**

REALIZADO POR:

Jesús Pérez Serrano

DIRIGIDO POR:

Francisco Javier González Cañete

DEPARTAMENTO DE: *Tecnología Electrónica*

TITULACIÓN: **Ingeniería Técnica de Telecomunicación**
Sistemas Electrónicos

PALABRAS CLAVE: Aplicaciones móviles, Android, PhoneGap, PHP,
Symfony, Biblioteca.

RESUMEN:

Debido al auge que está experimentando la implantación de *smartphones*, terminales móviles con mayores prestaciones y numerosas posibilidades de conectividad, así como la necesidad de adaptar los sistemas para su gestión a través de estos dispositivos, se plantea el presente proyecto, cuyo objetivo es desarrollar una aplicación móvil para la consulta y gestión de todos los libros de la biblioteca del departamento de Tecnología Electrónica de la Universidad de Málaga. El proyecto consiste en el desarrollo de dos aplicaciones: una aplicación móvil de consulta de los libros y una aplicación Web para su gestión. La aplicación que reside en los terminales se ha realizado con tecnología Web (HTML, CSS, JavaScript) para dispositivos con sistema operativo Android, con ayuda del *framework* PhoneGap. La aplicación que reside en el servidor se ha realizado en lenguaje PHP con ayuda del *framework* Symfony.

Málaga, *Febrero de 2013*

Agradecimientos

Para mí es obligado agradecer en primer lugar a mi hermano, Juan Manuel, ya que sin su ayuda no podría haber llegado hasta este punto. De la misma manera, tengo que agradecer a mis padres, Carmen y Prudencio, por ayudarme en todo en esta vida, por ser un ejemplo a seguir de trabajo, sacrificio, humildad y honradez y por creer en mí en todo momento.

También quiero agradecer a Esperanza, por dejarme entrar en tu vida, por apoyarme y confiar en mí siempre. No me puedo olvidar de Josefina, Pedro y Sofía, mi segunda familia, por hacerme sentir como en mi propia casa.

A todos mis amigos y compañeros, tantos a los de mi tierra como a los que he conocido durante el transcurso de mis estudios, en especial a Alberto, por dejarme ser parte de tus proyectos futuros.

A mis compañeros de trabajo, que me han enseñado mucho.

Y a Francis, por ayudarme en todo lo posible en la realización de este proyecto fin de carrera, por su apoyo y disponibilidad.

A todos, muchas gracias.

" Soy vanidoso, pero no en lo que se refiere a mi apariencia y sí en cuanto a mi trabajo."

Harrison Ford

ÍNDICE DE CONTENIDOS

ÍNDICE DE CONTENIDOS.....	i
RELACIÓN DE ACRÓNIMOS	v
ÍNDICE DE FIGURAS.....	vii
CAPÍTULO 1: Introducción	1
CAPÍTULO 2: Tecnologías y herramientas empleadas	5
2.1 ANDROID.....	5
2.1.1 Android 1.0, Septiembre 2008.....	6
2.1.2 Android 1.1, Febrero 2009	6
2.1.3 Android 1.5, <i>Cupcake</i> , Abril 2009	7
2.1.4 Android 1.6, <i>Donut</i> , Septiembre 2009.....	7
2.1.5 Android 2.0, <i>Eclair</i> , Octubre 2009	8
2.1.6 Android 2.2, <i>Froyo</i> , Mayo 2010.....	8
2.1.7 Android 2.3, <i>Gingerbread</i> , Diciembre 2010	9
2.1.8 Android 3.0, <i>Honeycomb</i> , Mayo 2011	9
2.1.9 Android 4.0, <i>Ice Cream Sandwich</i> , Octubre 2011.....	10
2.2 PHONEGAP	11
2.3 ECLIPSE	14
2.4 HTML.....	15
2.5 CSS	15
2.6 JAVASCRIPT	15
2.7 FRAMEWORKS PHP.....	16
2.8 PHP.....	17
2.9 BASES DE DATOS	18
2.10 APACHE	18
2.11 PHPMYADMIN.....	19
2.12 XML	19
2.13 UML	20
2.13.1 DIAGRAMAS DE CLASES.....	21
2.13.2 DIAGRAMAS DE ESTADOS.....	22

2.13.3 DIAGRAMAS DE SECUENCIA	23
CAPÍTULO 3: Descripción de la aplicación. Cliente y servidor.....	25
3.1 CLIENTE	25
3.1.1 Pantallas del cliente	26
3.1.1.1 Pantalla de inicio	26
3.1.1.2 Pantalla de elección de búsqueda.....	26
3.1.1.3 Pantalla de búsqueda de libros	27
3.1.1.4 Pantalla de búsqueda de PFC.....	28
3.1.1.5 Pantalla de búsqueda de TFM.....	28
3.1.1.6 Pantalla de búsqueda de tesis.....	29
3.1.1.7 Pantalla listado de resultados	30
3.1.1.8 Pantalla ficha de datos	31
3.1.2 Recepción y procesado de datos del servidor.....	32
3.2 SERVIDOR	36
3.2.1 Base de datos	37
3.2.2 Frontend de la aplicación.....	43
3.2.2.1 Módulo datos	43
3.2.3 Backend de la aplicación	49
CAPÍTULO 4: Plan de pruebas.....	53
4.1 PRUEBAS DE LA APLICACIÓN MÓVIL	53
4.1.1 Pruebas con navegadores Web	53
4.1.2 Pruebas con emuladores	54
4.1.3 Pruebas con dispositivos móviles reales.....	55
4.1.4 Análisis de los resultados.	56
4.2 PRUEBAS DE LA APLICACIÓN EN EL SERVIDOR	57
4.2.1 Pruebas del frontend	57
4.2.2 Pruebas del backend	58
4.2.3 Análisis de resultados	59
CAPÍTULO 5: Conclusiones y líneas futuras.....	61
5.1 CONCLUSIONES FINALES	61
5.2 LÍNEAS FUTURAS.....	62
ANEXO A. Manual de usuario	65

A.1 MANUAL DE USUARIO DE LA APLICACIÓN MÓVIL.....	65
A.2 MANUAL DE USUARIO ADMINISTRADOR.....	67
BIBLIOGRAFÍA Y REFERENCIAS	69
BIBLIOGRAFÍA.....	69
REFERENCIAS	70

RELACIÓN DE ACRÓNIMOS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AVD	Android Virtual Devices
CDMA	Code Division Multiple Access
CSS	Cascade Style Sheet
ECJ	Eclipse Compiler for Java
EDVO	Evolution-Data Optimized o Evolution-Data Only
GPL	General Public License
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	Hyper Text Mark-up Language
IDE	Integrated Development Environment
Inc.	Incorporation
iOS	iPhone Operating System
IP	Internet Protocol
ISBN	International Standard Book Number
NFC	Near Field Communication
OHA	Open Handset Alliance
PDF	Portable Document Format
PFC	Proyecto Fin de Carrera
PHP	Hypertext Preprocessor

SDK	Software Development Kit
SGML	Standard Generalized Markup Language
TFM	Trabajo Fin de Máster
VPN	Virtual Private Networks
W3C	World Wide Web Consortium
XHTML	eXtensible Hyper Text Mark-up Language
XML	eXtensible Markup Language

ÍNDICE DE FIGURAS

Figura 2.1 Logotipo de la versión de Android 1.0	6
Figura 2.2 Logotipo de la versión de Android CupCake	7
Figura 2.3 Logotipo de la versión de Android Donut	7
Figura 2.4 Logotipo de la versión de Android Eclair	8
Figura 2.5 Logotipo de la versión de Android Froyo	8
Figura 2.6 Logotipo de la versión de Android Gingerbread	9
Figura 2.7 Logotipo de la versión de Android Honeycomb.....	9
Figura 2.8 Logotipo de la versión de Android Ice Cream Sandwich.....	10
Figura 2.9 Distribución de las versiones de Android del 21 de Agosto al 4 de Septiembre de 2012.....	11
Figura 2.10 Lista de las plataformas de los navegadores en las distintas plataformas móviles.....	12
Figura 2.11 Arquitectura de una aplicación PhoneGap	13
Figura 2.12 Tabla comparativa de los cinco frameworks principales.....	16
Figura 2.13 Tipos de diagramas UML	21
Figura 2.14 Ejemplo de diagrama de clases de una universidad	21
Figura 2.15 Ejemplo de diagrama de estados	22
Figura 2.16 Ejemplo de diagrama de secuencia	23
Figura 3.1 Esqueleto de navegación entre pantallas de la aplicación móvil cliente	25
Figura 3.2 Pantalla de inicio de la aplicación móvil.....	26
Figura 3.3 Pantalla de elección de búsqueda de la aplicación móvil	27
Figura 3.4 Pantalla de búsqueda de libros de la aplicación móvil	27
Figura 3.5 Pantalla de búsqueda de PFC de la aplicación móvil	28
Figura 3.6 Pantalla de búsqueda de TFM de la aplicación móvil.....	29

Figura 3.7 Pantalla de búsqueda de tesis de la aplicación móvil	29
Figura 3.8 Pantalla del listado de libros.....	30
Figura 3.9 Pantalla del listado de PFC.....	30
Figura 3.10 Pantalla del listado de TFM.....	30
Figura 3.11 Pantalla del listado de tesis	30
Figura 3.12 Ficha de datos de un libro.....	31
Figura 3.13 Ficha de datos de un PFC	31
Figura 3.14 Ficha de datos de un TFM	31
Figura 3.15 Ficha de datos de una tesis.....	31
Figura 3.16 Diagrama de secuencia de la búsqueda de un libro.....	32
Figura 3.17 Diagrama de secuencia de la búsqueda de un PFC	33
Figura 3.18 Diagrama de secuencia de la búsqueda de un TFM	33
Figura 3.19 Diagrama de secuencia de la búsqueda de una tesis.....	34
Figura 3.20 XML obtenido al realizar la petición de las titulaciones en un PFC.....	35
Figura 3.21 Función AJAX en JavaScript que procesa el documento XML de las titulaciones	36
Figura 3.22 Estructura de la aplicación Symfony	37
Figura 3.23 Esquema de las tablas de la base de datos realizado con MysqlWorkbench...	38
Figura 3.24 Definición de la tabla "alumnos" en Doctrine	40
Figura 3.25 Definición de la tabla "profesores" en Doctrine	40
Figura 3.26 Definición de la tabla "libros" en Doctrine	41
Figura 3.27 Definición de la tabla "pfc" en Doctrine	41
Figura 3.28 Definición de la tabla "tfm" en Doctrine.....	42
Figura 3.29 Definición de la tabla "Tesis" en Doctrine	42
Figura 3.30 Estructura de archivos del módulo frontend.....	43

Figura 3.31	Tabla de las funciones del archivo <i>actions.class.php</i> del módulo datos	44
Figura 3.32	Ejemplo de consulta Doctrine	45
Figura 3.33	Código PHP del template <i>imprimirxmlcamposSuccess.php</i>.....	46
Figura 3.34	XML devuelto por el servidor al ejecutar la función <i>executeListadoLibros()</i>..	46
Figura 3.35	Código PHP de la función <i>executeListadoLibros()</i>	47
Figura 3.36	Código PHP del template <i>imprimirxmlSuccess.php</i>.....	47
Figura 3.37	XML generado al ejecutar la función <i>executeListadoMateres()</i>	49
Figura 3.38	Aspecto de la pantalla de elección del módulo administrar	50
Figura 3.39	Cuadro de búsqueda de un libro del módulo <i>adminl</i>.....	51
Figura 3.40	Tabla de resultados de un libro del módulo <i>adminl</i>.....	51
Figura 4.1	Tabla de dispositivos móviles empleados en las pruebas, versión de Android y tamaño de la pantalla de cada uno de ellos.....	55
Figura A.1	Icono de la aplicación móvil.....	66
Figura A.2	Botones de navegación de un dispositivo móvil Android	66
Figura A.3	Ventana emergente de alerta sin conexión	67
Figura A.4	Ejemplo de pantalla de edición de un elemento de la tabla "libros"	68

CAPÍTULO 1: Introducción

En los últimos años, el uso de telefonía móvil está creciendo. Desde los 700 millones de teléfonos móviles que había en el año 2000, hasta los 6200 millones en el segundo trimestre del año 2012 [1].

Actualmente, está en aumento el uso de *smartphones*, teléfonos móviles que pueden comunicarse a través de Wi-Fi, bluetooth, conexión a Internet y permiten a los usuarios la instalación de programas. Esto último se ha traducido en un aumento del desarrollo de aplicaciones estos dispositivos.

Son varios los sistemas operativos para *smartphones*, siendo el principal Android, ya que en el año 2012, las ventas de estos dispositivos suponen un 53%, frente al 20% de *smartphones* con iOS (*iPhone Operating System*) o el 8% con BlackBerry. Otros sistemas operativos como Symbian, Bada o Windows Phone tienen porcentajes de implantación menores (11%, 3% y 2% respectivamente)[2].

Además, OVUM, una empresa de consultoría en los campos de la tecnología y las telecomunicaciones, ha presentado unas estadísticas respecto a la proyección en el mercado de *smartphones*. Según este estudio, Android será el líder en el año 2016, dejando al iOS de Apple y Windows Phone de Microsoft muy por detrás. Mientras Android tendrá un 38% de cuota de mercado, iOS tendrá un 17.5%, Windows Phone un 17.2% y BlackBerry un 16,5% [3].

Otro aspecto importante es el uso del Internet móvil, el cual se duplica año a año desde el 2009. De esta manera, mientras en el año 2009 sólo el 0.7% de las consultas a la Web eran mediante navegadores de dispositivos móviles, en el año 2010 este porcentaje subió al 1.6%, al 4.3% en el 2011, hasta llegar al 8.5% en enero del año 2012 [4].

Principalmente, los usos de Internet a través del móvil son: navegar por la red, realizar búsquedas, visualización de vídeos y el acceso a las redes sociales. Los usuarios consultan de manera más frecuente Internet en el móvil que en un ordenador, aunque éstas consultas suelen ser por lo general más breves [5].

Otro uso que no ha sido mencionado es el de la geolocalización, es decir, la búsqueda de lugares con respecto a la posición del usuario. La mayoría de *smartphones* tienen integrado un GPS, con la ayuda del cual pueden desplazarse hacia lugares concretos con tan solo buscarlo en un mapa.

Ante el panorama actual, en el que el teléfono móvil es una herramienta de estudio o trabajo más, se tiene la obligación de adaptar los sistemas para su gestión a través de estos dispositivos. Un ejemplo de esto se encuentra en la biblioteca del departamento de Tecnología Electrónica de la Universidad de Málaga, en la que la gestión se lleva a cabo en la secretaría de dicho departamento, mediante una base de datos local, por lo que si alguna persona necesita conocer la ubicación de un libro, un PFC (Proyecto Final de Carrera), un TFM (Trabajo Fin de Máster) o una Tesis, no tiene más remedio que desplazarse hasta la secretaría en horario de atención al público para obtener la información necesaria.

El objeto del presente proyecto es el desarrollo de una aplicación móvil en la que poder consultar la información relacionada con libros (autores, título, editorial, tejuelo, ISBN: *International Standard Book Number*, ubicación, ...), PFC (autor, título, tutor, ...), TFM (autor, título, tutor, ...), Tesis (autor, título, director, ...).

El presente proyecto se compone de dos aplicaciones bien diferenciadas:

- Por un lado se encuentra la aplicación del cliente, que sería la aplicación móvil propiamente dicha. Esta aplicación se basa en la navegación entre pantallas, la inicial en la que se puede elegir el tipo de libro del que se va a proceder la búsqueda (libros, PFC, TFM, Tesis), las pantallas de búsqueda, una pantalla en la que se listan los resultados obtenidos y por último, una pantalla en la que se muestran todos los datos disponibles de uno de los resultados de la búsqueda.
- Por otra parte, la aplicación del servidor, en el que se tiene una aplicación Web que es la que va a proporcionar los datos en formato XML (*eXtensible Markup Language*) a la aplicación móvil cuando ésta los solicite. En el servidor también se implementarán funciones de administración, para facilitar la inserción, modificación y eliminación de los datos. Esto será posible gracias a un interfaz bastante intuitivo y fácil de usar, similar al interfaz de la aplicación móvil descrita anteriormente. En la pantalla principal de la

administración, al igual que en la parte del cliente, se puede elegir el tipo de libro con el que se va a trabajar. Una vez hecha la elección, se pasa a otra página en la que aparecen tres elementos principales: un cuadro de búsqueda, un listado con todos los elementos almacenados en la base de datos, y un botón con el que se puede añadir un nuevo elemento.

Una vez decidido realizar la aplicación para el sistema operativo Android, hay dos posibilidades en cuanto a su implementación: realizarla de forma nativa en Java o usando PhoneGap, un *framework* de libre distribución que permite programar las aplicaciones usando HTML (*Hyper Text Mark-up Language*), CSS (*Cascade Style Sheets*) y Javascript. Cada uno de los métodos tienen sus ventajas y desventajas.

La programación nativa de aplicaciones tiene como ventajas principales el poder acceder a todos los recursos del dispositivo y el ejecutarse mucho más rápidamente. El problema consiste en que si se quiere portar la aplicación a otras plataformas habría que reprogramar la aplicación completamente para cada una de ellas [6].

Como ventajas del uso de PhoneGap destaca que se puede usar el mismo código base para las distintas plataformas soportadas. Otra ventaja significativa durante el desarrollo de la aplicación consiste en que se puede ver el resultado del diseño con la ayuda de un navegador Web, disminuyendo de esta manera el tiempo de desarrollo [7].

El presente documento está estructurado en diversos capítulos para poder explicar con detalle todas las partes del proyecto:

1. Introducción: es el presente capítulo, en el que se hace una somera descripción y justificación del proyecto, así como de los objetivos a conseguir.
2. Tecnologías y herramientas empleadas: en este capítulo se detallan las herramientas software y hardware utilizadas, comparándolas con otras similares y justificando su elección en base a sus ventajas e inconvenientes.
3. Descripción de la aplicación: aquí se describe en profundidad la lógica de la aplicación, tanto la parte del cliente como la parte del servidor. Se aclara la navegación entre ventanas de la aplicación móvil y se detallan aspectos de la implementación.

4. Plan de pruebas: en este apartado se detallan cada una de las pruebas realizadas al sistema para comprobar su correcto funcionamiento.
5. Conclusiones y líneas futuras: en el último capítulo se hace un balance del proyecto, así como el estudio de posibles mejoras o ampliaciones.

CAPÍTULO 2: Tecnologías y herramientas empleadas

En este capítulo se detallan las tecnologías que se han empleado para la realización de este proyecto, así como las herramientas *software* utilizadas, su elección frente a otras alternativas y la labor para la que fueron utilizadas en el ámbito del proyecto.

2.1 ANDROID

Todo el mundo cree que Android empezó con Google pero nada más lejos de la realidad. La empresa Android Incorporation (la cual ya no existe como tal, sino que se trata de una parte de la empresa Google) fue fundada en Palo Alto, California, en Octubre del 2003 por Andy Rubin (cofundador de Danger Incorporation, una empresa que trabajaba exclusivamente en plataformas, *software*, diseño y servicios para dispositivos móviles), Rich Miner (cofundador de Wildfire Communications Incorporation), Nick Sears (fue vicepresidente de T-Mobile) y Chris White (encabezó el desarrollo del diseño y la interfaz en Web TV). En el inicio de Android Incorporation sólo se sabía que trabajaban en el desarrollo de *software* para dispositivos móviles [8].

Google adquirió Android Incorporation en verano del año 2005, incorporándose Andy Rubin, Rich Miner y Chris White a su plantilla. No se conoce mucho más de la adquisición de Android Incorporation por parte de Google, pero se rumoreó con que Google estaba planeando entrar en el mercado de la telefonía móvil [9].

En Noviembre del 2007 se publicó el primer SDK de Android. Un SDK (*Software Development Kit*) es un kit de desarrollo *software* que permite hacer nuevas aplicaciones. En ese mismo mes, la OHA (*Open Handset Alliance*) [10], un consorcio de varias compañías (Broadcom Corporation, Google, HTC, Intel, LG, Marvell TechnologyGroup, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile and Texas Instruments) surgió con el propósito de desarrollar estándares abiertos para dispositivos móviles. Al mismo tiempo, la OHA sacó a la luz su primer producto, Android, un sistema operativo construido sobre el kernel de Linux en su

versión 2.6. En diciembre del año 2008, se unieron 14 miembros nuevos (ARM Holdings, Asustek Computer Incorporation, Garmin Ltd, Huawei Technologies, Packet Video, Atheros Communications, Vodafone, Sony Ericsson, Toshiba Corporation).

Desde entonces, han surgido varias versiones del S.O. Android, que arreglan errores y añaden nuevas funcionalidades con respecto a las versiones anteriores. Generalmente, los nombre de las distintas versiones (excepto las versiones 1.0 y 1.1) son desarrolladas bajo un nombre relacionado con postres[11][12].

2.1.1 ANDROID 1.0, SEPTIEMBRE 2008

El primer dispositivo con S.O. Android 1.0 fue el HTC Dream (G1) [13]. Esta versión incorpora el *Android Market*, una aplicación para descarga y actualizaciones de aplicaciones. Está integrado con los servicios de Google, dispone también de un navegador Web capaz de mostrar páginas Web HTML y XHTML (*eXtensible Hyper Text Mark-up Language*), y varias ventanas a la vez. Tiene multitarea, mensajería instantánea, Wi-Fi y Bluetooth. En la Figura 2.1 se puede ver el logotipo de esta versión de Android.



Figura 2.1 Logotipo de la versión de Android 1.0

2.1.2 ANDROID 1.1, FEBRERO 2009

Esta versión fue publicada solamente para el T-1 Mobile G1. La actualización corrige errores, cambia la *API (Application Programming Interface)* y añade algunas características nuevas como la posibilidad de mandar mensajes con archivos adjuntos, la disponibilidad de detalles y opiniones cuando un usuario busca en los mapas de Google un negocio, la capacidad de mostrar y ocultar el teclado durante una llamada y el

aumento del tiempo de espera por defecto de la pantalla de llamada cuando se usa el altavoz.

2.1.3 ANDROID 1.5, *CUPCAKE*, ABRIL 2009

La velocidad de arranque y la captura de imágenes de la cámara aumenta. La adquisición de la posición GPS es mucho más rápida (usa tecnología de *SUPL AGPS*). Introduce el teclado en pantalla, la capacidad de subir vídeos directamente a *YouTube* y *Picassa*. El logotipo de la versión de Android 1.5 se muestra en la Figura 2.2.



Figura 2.2 Logotipo de la versión de Android *CupCake*

2.1.4 ANDROID 1.6, *DONUT*, SEPTIEMBRE 2009

Introduce un cuadro de búsqueda rápida y la búsqueda por voz. Se puede alternar entre el modo de vídeo y foto de la cámara. Incorpora el indicador de uso de la batería y la función de síntesis de voz multi-lenguaje. Soporta *CDMA/EVDO*, *802.1x* y *VPN*. Se puede observar el logotipo de Android Donut en la Figura 2.3.



Figura 2.3 Logotipo de la versión de Android *Donut*

2.1.5 ANDROID 2.0, *ECLAIR*, OCTUBRE 2009

En esta versión se da soporte a *Bluetooth 2.1* y se añade el *Microsoft Exchange* para la sincronización del e-mail. Cambia el interfaz de usuario del navegador Web, que soporta HTML5, y se agregan nuevas características al calendario. El logotipo de la versión 2.0 se muestra en la Figura 2.4.



Figura 2.4 Logotipo de la versión de Android *Eclair*

2.1.6 ANDROID 2.2, *FROYO*, MAYO 2010

Las principales novedades que presentaba esta versión son la incorporación del *Adobe Flash 10.1* y teclado en múltiples lenguajes. En la Figura 2.5 se puede observar el logotipo de la versión de Android 2.2.



Figura 2.5 Logotipo de la versión de Android *Froyo*

2.1.7 ANDROID 2.3, *GINGERBREAD*, DICIEMBRE 2010

Interfaz de usuario renovado con mayor simplicidad y velocidad, junto con un teclado nuevo para una rápida introducción de texto. Se integra la selección de palabras con un toque y la función copiar/pegar. Se da soporte a *NFC* (*Near Field Communication*) y las llamadas vía Internet. El logotipo de Android *Gingerbread* se muestra en la Figura 2.6.



Figura 2.6 Logotipo de la versión de Android *Gingerbread*

2.1.8 ANDROID 3.0, *HONEYCOMB*, MAYO 2011

Específicamente optimizado para *tablets* y dispositivos con pantallas de grandes dimensiones. Se redefine la multitarea, se añaden notificaciones y la posibilidad de personalización de la pantalla principal. Se soporta protocolo de transferencia para vídeos e imágenes. En la Figura 2.7 se puede observar el logotipo de la versión de Android 3.0.



Figura 2.7 Logotipo de la versión de Android *Honeycomb*

2.1.9 ANDROID 4.0, *ICE CREAM SANDWICH*, OCTUBRE 2011

Esta versión es la unificación del sistema operativo tanto para *tablets* como para móviles. Se tiene un nuevo tipo de letra llamado *Roboto* y el sistema de desbloqueo del dispositivo cambia. Añade botones virtuales en la barra del sistema (retroceso, inicio, aplicaciones recientes y opciones). Una característica innovadora es la posibilidad de desbloqueo facial usando la cámara frontal (aunque ha sido el objeto del principal error de seguridad). Otras novedades con respecto a versiones anteriores son la introducción de texto y revisión ortográfica, la mejora del reconocimiento de voz, una mayor accesibilidad destinada a facilitar el uso del dispositivo a personas con mayores dificultades, junto con otras características algo menos importantes. En la Figura 2.8 se muestra el logotipo de Android *Ice Cream Sandwich*.



Figura 2.8 Logotipo de la versión de Android *Ice Cream Sandwich*

En la Figura 2.9 se puede observar la distribución en Septiembre del año 2012 de los dispositivos con las diferentes versiones del sistema operativo mencionado que visitaron la tienda de aplicaciones de Android, *Google Play* [14].



Figura 2.9 Distribución de las versiones Android del 21 de Agosto al 4 de Septiembre de 2012 [15]

2.2 PHONEGAP

PhoneGap es un *framework* de aplicaciones HTML5 que se usa para desarrollar aplicaciones nativas a través de tecnologías Web. Se va explicar el desarrollo de aplicaciones móviles en los distintos tipos de dispositivos existentes actualmente. Hay muchas plataformas de *smartphones* en el mercado: Android, iPhone, BlackBerry, Symbian, Windows Phone, Windows 8 y WebOS. El gran número de plataformas disponibles en los dispositivos es un punto importante a tener en mente a la hora de desarrollar una aplicación móvil.

La fragmentación en los sistemas operativos móviles es muy grande debido a que no hay especificaciones ni estándares en el área de desarrollo. En el año 2007, Apple y Google lanzaron sus plataformas móviles (iOS y Android respectivamente), y un año más tarde, en 2008, ambas compañías lanzaron sus tiendas de aplicaciones para permitir a los usuarios de *smartphones* descargar aplicaciones. Desde entonces, el número de *smartphones*, y obviamente el número de aplicaciones que se desarrollan para estos dispositivos, ha crecido exponencialmente.

En el libro *Beginning PhoneGap. Mobile Web Framework for JavaScript and HTML* [15] se clasifican las aplicaciones móviles en dos grupos: aplicaciones móviles independientes y aplicaciones móviles basadas en servicios Web. Las primeras son

aplicaciones como alarmas, juegos *offline* y el marcador del teléfono. Las segundas son aplicaciones como el correo electrónico, calendarios, juegos *online* y aplicaciones que interactúan con servicios Web. Aunque se puede usar PhoneGap para implementar aplicaciones móviles independientes, se adapta mejor a las aplicaciones basadas en servicios Web, debido a que las aplicaciones PhoneGap son principalmente aplicaciones Web con funciones para controlar las características del dispositivo (GPS, acelerómetro, cámara, ...).

PhoneGap surgió debido a la concordancia entre la mayoría de las plataformas móviles en el uso del navegador. Los navegadores tenían muchas diferencias entre sí hasta hace unos pocos años. Desde entonces, los navegadores se han ido adaptando a los estándares W3C (*World Wide Web Consortium*). Los primeros navegadores adheridos al estándar fueron Firefox y Safari y actualmente, los navegadores (sobre todos los de las plataformas móviles) siguen los estándares. Esto es debido también a que las plataformas móviles más modernas tienen el mismo navegador basado en *Webkit*, un motor de navegadores Web de código abierto [16]. Además, los nuevos navegadores, tanto de escritorio como de *smartphones*, se adhieren a los nuevos estándares como HTML5 y CSS3. Esto añade nuevas características en el mundo del navegador y disminuye la fragmentación.

En la Figura 2.10 se puede observar una lista con las plataformas móviles y sus correspondientes plataformas del navegador. Como se puede ver, todas las plataformas excepto Windows Phone7 usan un navegador basado en *Webkit*.

SISTEMA OPERATIVO MÓVIL	NAVEGADOR
Android	Basado en <i>Webkit</i>
iPhone	Basado en <i>Webkit</i>
Blackberry 6.0 +	Basado en <i>Webkit</i>
Windows Phone 7	Basado en Internet Explorer 7
WebOS	Basado en <i>Webkit</i>
Nokia	Basado en <i>Webkit</i>
BADA	Basado en <i>Webkit</i>

Figura 2. 10 Lista de las plataformas de los navegadores en las distintas plataformas móviles.

PhoneGap usa estos navegadores como plataforma para construir aplicaciones basados en los estándares HTML5/CSS3. Todas las aplicaciones PhoneGap tienen navegadores embebidos, por lo que una pantalla de una aplicación puede ser un navegador que muestra una página HTML. A estos navegadores embebidos se les llama a veces *Webviews*. Se tienen dos opciones: que la *Webview* cargue la página desde la Web o que cargue las páginas HTML almacenadas localmente en el dispositivo. La segunda opción es la que se usa cuando se desarrolla una aplicación PhoneGap.

Por tanto, las aplicaciones que son desarrolladas usando PhoneGap son híbridas, es decir, no son puramente aplicaciones basadas en HTML y JavaScript, ni tampoco nativas. Algunas partes de la aplicación, principalmente el interfaz de usuario, la lógica de la aplicación y la comunicación con el servidor, están basadas en HTML/JavaScript. Las otras partes de la aplicación que comunican y controlan el dispositivo están basadas en el lenguaje nativo para cada plataforma concreta. PhoneGap provee un puente desde la parte de JavaScript a la parte nativa de la plataforma que permite a la *API* de JavaScript el acceso y control de las características del dispositivo.

Como se puede observar en la Figura 2.11, una aplicación desarrollada con PhoneGap consta principalmente de dos partes: la parte lógica de JavaScript que maneja el interfaz de usuario y su funcionalidad, y la *API* de PhoneGap, que accede y controla el dispositivo.

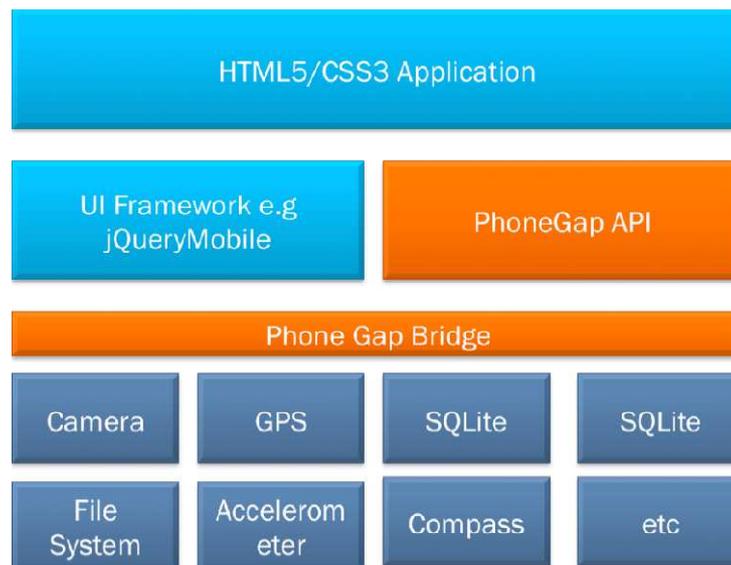


Figura 2. 11 Arquitectura de una aplicación PhoneGap

Como ya se ha comentado anteriormente en la introducción, el uso de PhoneGap tiene como principal ventaja a la hora de desarrollar para múltiples plataformas en que se puede reutilizar el mismo código base, ahorrando así tiempo de desarrollo.

2.3 ECLIPSE

Eclipse es un entorno de desarrollo multiplataforma basado en Java, de código abierto y extensible mediante *plugins*. Es el IDE (*Integrated Development Enviroment*) oficial del proyecto Android. Un IDE es un programa compuesto por un conjunto de herramientas para desarrolladores de software que cuenta como elementos básicos, con un editor de código, un compilador/intérprete y un depurador. Eclipse da soporte a multitud de lenguajes, entre los que se encuentran C/C++, Cobol, Fortran, PHP o Python. Esta plataforma ha sido usada para desarrollar otros IDE como el de Java (JDT, *Java Development Toolkit*) y el compilador ECJ (*Eclipse Compiler for Java*), que ya se encuentra integrado con Eclipse [17].

Una gran parte de la programación inicial de Eclipse fue realizada por IBM como sucesor de la familia de herramientas para *Visual Age*. En Noviembre del 2001, se formó un consorcio para el desarrollo del proyecto Eclipse como código abierto, con IBM y Borland a la cabeza. A finales del 2003, este consorcio había crecido hasta los 80 miembros. En Febrero del 2004, se anunció la reorganización de Eclipse a una fundación sin ánimo de lucro [18].

Para poder desarrollar aplicaciones para el sistema operativo Android con Eclipse es necesaria la instalación del *ADT Plugin (Android Development Tools)*. Dicho plugin extiende las capacidades de Eclipse para desarrollar nuevos proyectos Android, crear interfaces de usuario de las aplicaciones, añadir paquetes basados en la *API* de Android y depurar las aplicaciones usando el SDK de Android. También nos permite exportar los archivos de aplicación con extensión *.apk* con el objeto de distribuir las aplicaciones desarrolladas.

Para poder comprobar el correcto funcionamiento de las aplicaciones se hace necesario el simulador AVD (*Android Virtual Devices*), disponible en las herramientas del SDK de Android. Se pueden crear tantos dispositivos virtuales AVD como se tengan instalados, de manera que no existe la necesidad de disponer de los dispositivos móviles reales con todas las versiones del sistema operativo.

2.4 HTML

HTML son las siglas del inglés de lenguaje de marcado de hipertexto (*Hyper Text Markup Language*). HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas Web en Internet de modo hipertexto, es decir, permite escribir texto de forma estructurada, y está compuesto por etiquetas (palabras que van entre corchetes angulares <,>), que marcan el inicio y el fin de cada elemento del documento. Por ejemplo, el inicio de un documento HTML se marca con la etiqueta <html> y el final de dicho documento se marcaría con </html> [19][20].

La extensión de un documento HTML debe ser *.htm* o *.html* para que puedan ser visualizados en los navegadores. Los navegadores interpretan el código HTML de los documentos y muestran a los usuarios el resultado. Cabe destacar que en el presente proyecto se utiliza el navegador Web Mozilla Firefox para visualizar las páginas de las que consta la aplicación durante la fase de diseño.

2.5 CSS

CSS son las siglas del inglés de hojas de estilo en cascada (*Cascading Style Sheets*). CSS es un lenguaje usado para definir el formato y el estilo de un documento HTML o XML. De esta manera, se separa el contenido de la presentación. CSS permite controlar múltiples páginas Web al mismo tiempo: cualquier cambio en el estilo marcado para un elemento en la CSS afecta a todas las páginas vinculadas a esa CSS en la que aparezca el elemento. CSS funciona a base de declaraciones sobre el estilo de uno (a través de los identificadores) o más elementos (a través de las clases). La sintaxis es muy sencilla, usa una serie de palabras clave para especificar los nombres de sus selectores, propiedades y atributos [21].

2.6 JAVASCRIPT

JavaScript es una de las múltiples maneras que existen para extender las capacidades del lenguaje HTML. No es un lenguaje de programación propiamente dicho como C, C++, Java, etc. Es un lenguaje orientado al documento, por lo que no se puede desarrollar un programa con JavaScript que se ejecute fuera de un navegador Web. Las instrucciones de un programa Javascript las analiza y las procesa el navegador en el

momento que deben ser ejecutadas, es decir, es un lenguaje interpretado, por lo que no necesita compilar el programa para ejecutarlo. A pesar del nombre, JavaScript no tiene nada que ver con el lenguaje Java [22][23].

2.7 FRAMEWORKS PHP

Un *framework* (su traducción literal del inglés sería "marco de trabajo") es un esquema para el desarrollo y/o implementación de una aplicación [24]. Hay varias razones por las que usar un *framework* PHP (Hypertext Preprocessor). Una de ellas es para acelerar el proceso de desarrollo. Otra razón es la estabilidad de la aplicación. Si no se usa puede ser que la aplicación funcione, pero se hayan cometido errores en la programación introduciendo "código malo" sin darnos cuenta. Se dispone de muchos *framework* PHP entre los que elegir. Que se elija uno u otro depende de lo que se quiera desarrollar.

En la Figura 2.12 se muestra una tabla comparativa de 5 de los principales *framework* disponibles [25]:

Compare PHP frameworks												
Framework	PHP4	PHP5	MVC	Modules	ORM	EDP	Auth	Cache	Validator	License	Docs	Website
Zend Framework	✗	✓	✓	✓	✓	✗	✓	✓	✓	New BSD	#	🌐
symfony	✗	✓	✓	✓	✓	✗	✓	✓	✓	MIT	#	🌐
CakePHP	✗	✓	✓	✓	✓	✗	✓	✓	✓	MIT	#	🌐
CodeIgniter	✓	✓	✓	✓	✓	✗	✗	✓	✓	Apache/BSD	#	🌐
Seagull	✓	✓	✓	✓	✓	✗	✓	✓	✓	BSD	#	🌐

Figura 2. 12 Tabla comparativa de los cinco frameworks principales

A continuación se explican brevemente algunas características de estos *frameworks*[26]:

- *The Zend Framework* [27] es desarrollado por la empresa que respalda comercialmente a PHP. Posee propiedades importantes que están construidas para el desarrollo a nivel corporativo como soporte avanzado para la internacionalización y módulos para manejar archivos PDF (*Portable*

Document Format), canales RSS (*Really Symple Syndication*¹), Servicios Web (Amazon, Flickr, Yahoo). Requiere un gran conocimiento de PHP.

- *Symfony* [28] se puede utilizar tanto en Linux como en Windows. Es un *framework* de desarrollo rápido, con una estructura de librerías y clases para programar aplicaciones Web.
- *CakePHP* [29] nos permite programar más rápido evitándonos escribir código tedioso de tareas muy comunes. Su sistema de soporte creciente, simplicidad y escalabilidad hacen que CakePHP sea una de los *frameworks* PHP más populares hoy en día[30].
- *CodeIgniter* [31] es utilizado por una gran comunidad de usuarios. Está construido para codificadores PHP que necesitan una herramienta de desarrollo fácil para crear aplicaciones Web simples y elegantes.
- *Seagull* [32] es un *framework* PHP muy reconocido utilizado para la construcción de Webs, líneas de comando y aplicaciones GUI. Es fácil de usar, tanto para principiantes, porque posee una librería de aplicaciones de muestra que pueden ser personalizadas, como para los expertos, ya que ofrece muchas opciones para construir aplicaciones Web rápida y fácilmente.

Para este proyecto, cualquiera de los *frameworks* expuestos nos sirve, ya que todos se adaptan a nuestras necesidades. Se ha elegido por el *framework* Symfony, ya que dispone de manuales y guías en abundancia y cuenta además, con varios foros donde la comunidad resuelve dudas. Otra razón por la que elegir este *framework* y no otro es la posibilidad de trabajar con el motor de base de datos que se desee, lo que proporciona una enorme flexibilidad. Además, Symfony dispone de *plugins* para aumentar su funcionalidad, mientras que los otros no.

2.8 PHP

PHP es un lenguaje de programación de código abierto especialmente indicado para el desarrollo Web con contenido dinámico. Es un lenguaje de *script* del lado del servidor, que se puede incorporar directamente a un documento HTML, con lo que no es necesario llamar a un archivo externo que procese los datos.

¹ Anteriormente RSS eran las siglas de *Rich Site Summary* y *RDF Site Summary*.

PHP soporta muchos tipos de bases de datos (*MySQL*, *SQLite*, *Oracle*, *Solid*, *PostgreSQL*, etc.). Para el presente proyecto se va a usar las bases de datos MySQL.

Un archivo PHP puede contener texto, etiquetas HTML y *scripts*. Todo esto, lo procesa el servidor (que debe tener instalado un intérprete PHP) y devuelve una página HTML al navegador que le hace la petición. En el caso del presente proyecto, el cliente (la aplicación Android) realiza una petición al servidor y éste le devuelve un documento XML. Los archivos PHP pueden tener como extensiones *.php*, *.php3*, *.phtml*. [33][34]

2.9 BASES DE DATOS

Una base de datos es un conjunto de información relacionada que se encuentra agrupada o estructurada. Desde el punto de vista informático, una base de datos es el sistema formado por el conjunto de datos almacenados y el conjunto de programas que manipulan esos datos.

Los sistemas de gestión de bases de datos (en inglés *Data Base Management System*) son programas que sirven de interfaz entre las bases de datos, los usuarios y las aplicaciones que la utilizan. En el presente proyecto se ha elegido MySQL [35] como sistema de gestión de base de datos. MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiplataforma. Se ha elegido MySQL porque se adapta a pequeñas y grandes aplicaciones, soporta el estándar SQL, compila en numerosas plataformas y es gratuita su descarga y uso. Además, es muy rápido y fácil de usar [36] [37].

2.10 APACHE

Un servidor Web es un programa que se ejecuta continuamente en un ordenador que se mantiene a la espera de peticiones y las gestiona de forma adecuada, entregando como resultado una página Web o información de todo tipo. Hay que distinguir entre servidor refiriéndose al equipo con el software servidor instalado y entre ese software. Estrictamente, el servidor es el software que realiza las funciones descritas.

Apache es el servidor Web que más se utiliza debido a su robustez y estabilidad. Apache es un proyecto de código abierto, uso gratuito, multiplataforma y existen versiones para los sistemas operativos más importantes. Otras ventajas de Apache son la

eficiencia, el alto grado de personalización, la capacidad de administrarlo remotamente vía línea de comandos de manera fácil y la posibilidad de transferir las configuraciones de un servidor Apache a otro [38].

2.11 PHPMYADMIN

PHPMyAdmin es una herramienta escrita en PHP, con licencia GPL (*General Public License*), que permite administrar bases de datos MySQL empleando un navegador. Se pueden administrar tanto de forma local como remotamente. Es multiplataforma, multilenguaje y permite acceder a todas las funciones típicas de MySQL a través de una interfaz Web muy intuitiva. Además permite ejecutar sentencias SQL y hacer copia de seguridad de la base de datos [39][40].

2.12 XML

XML (*eXtensible Markup Language* - lenguaje extensible de etiquetas), fue desarrollado por el W3C[41] y es una adaptación del SGML (*Standard Generalized Markup Language*). XML no es realmente un lenguaje en particular, sino que define la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de HTML, en el que las etiquetas presentan el contenido, las etiquetas XML se limitan a describirlo.

Como aproximación, se podría decir que XML es un subconjunto de SGML especializado en la gestión de información para la Web, mientras que HTML es un subconjunto de XML especializado en presentación de documentos para la Web. Siendo estrictos, XML contiene a HTML, pero no en su totalidad. XHTML es la definición de HTML dentro de XML, y por lo tanto, XHTML queda incluida en SGML.

XML es extensible, por lo que se permite la creación de etiquetas y atributos propios para describir los datos, siempre atendiendo a unas normas de sintaxis para garantizar la consistencia de dichos datos.

La elección de XML como lenguaje para la comunicación con el cliente por parte del servidor en el presente proyecto se debe a la fácil organización y flexibilidad en la definición de etiquetas, unido a que se trata de un estándar abierto [42].

2.13 UML

UML (Lenguaje Unificado de Modelado - *Unified Modeling Language*). Un modelo, de forma sencilla, es una simplificación de la realidad, y se construyen modelos para comprender mejor el sistema que se está desarrollando. Con el modelado, se consiguen cuatro objetivos: visualizar cómo es o cómo se quiere que sea un sistema, especificar la estructura o el comportamiento de un sistema, proporcionar plantillas que nos guían en la construcción del sistema y documentar las decisiones que se han tomado.

Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema. UML, como lenguaje de modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Un modelo UML está compuesto por tres elementos principales:

- Elementos: son los bloques básicos de construcción UML, abstracciones de cosas reales o ficticias (objetos, acciones, etc.).
- Relaciones: son las reglas que dictan cómo se pueden combinar los elementos
- Diagramas: son la representación gráfica de un conjunto de elementos relacionados entre sí.

De esta forma, un diagrama es la proyección de un sistema. En la mayoría de los sistemas, excepto en los más triviales, un diagrama representa una vista resumida de los elementos que constituyen un sistema. En la Figura 2.13 se pueden ver los diversos tipos de diagrama que proporciona UML.

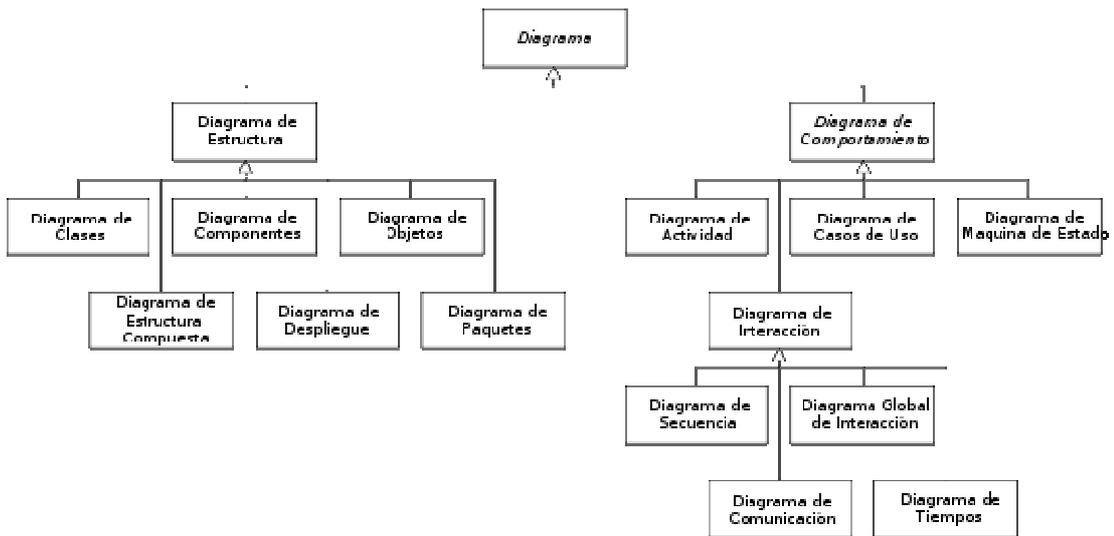


Figura 2. 13 Tipos de diagramas UML.

Los diagramas más usados son los de clases, de estados y de secuencia. Debido a sus características y a los puntos de vista que ofrecen, son los diagramas elegidos para realizar el diseño del presente proyecto.

2.13.1 DIAGRAMAS DE CLASES

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Son los diagramas más usados en el modelado de sistemas orientados a objetos, y abarcan la vista de diseño estática de un sistema, como se puede ver en el ejemplo de una universidad de la Figura 2.14.

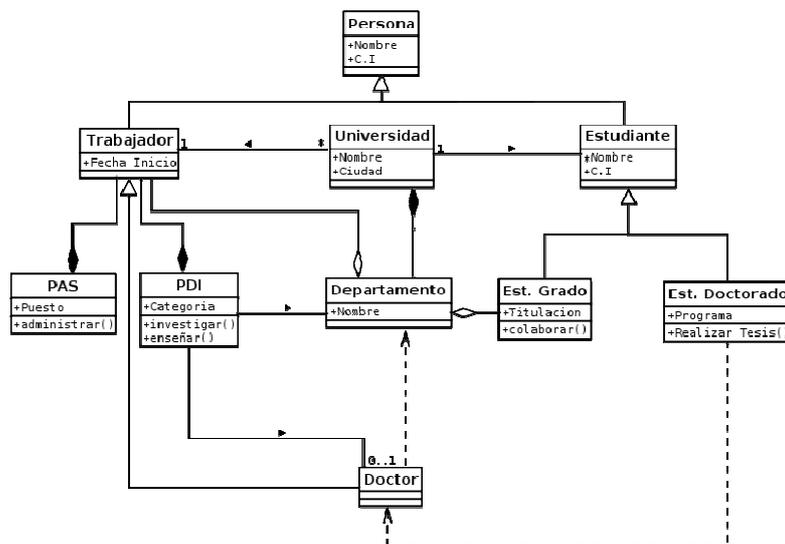


Figura 2. 14 Ejemplo de diagrama de clases de una universidad.

2.13.2 DIAGRAMAS DE ESTADOS

Los diagramas de estados muestran una máquina de estados, que consta de estados, transiciones, eventos y actividades. Se utilizan para modelar los aspectos dinámicos de un sistema y son útiles para modelar la vida de un objeto. Lo habitual es utilizar los diagramas de estados en el contexto del sistema global, de un subsistema o de una clase. Un ejemplo de diagrama de estados de un proceso se puede ver en la Figura 2.15.

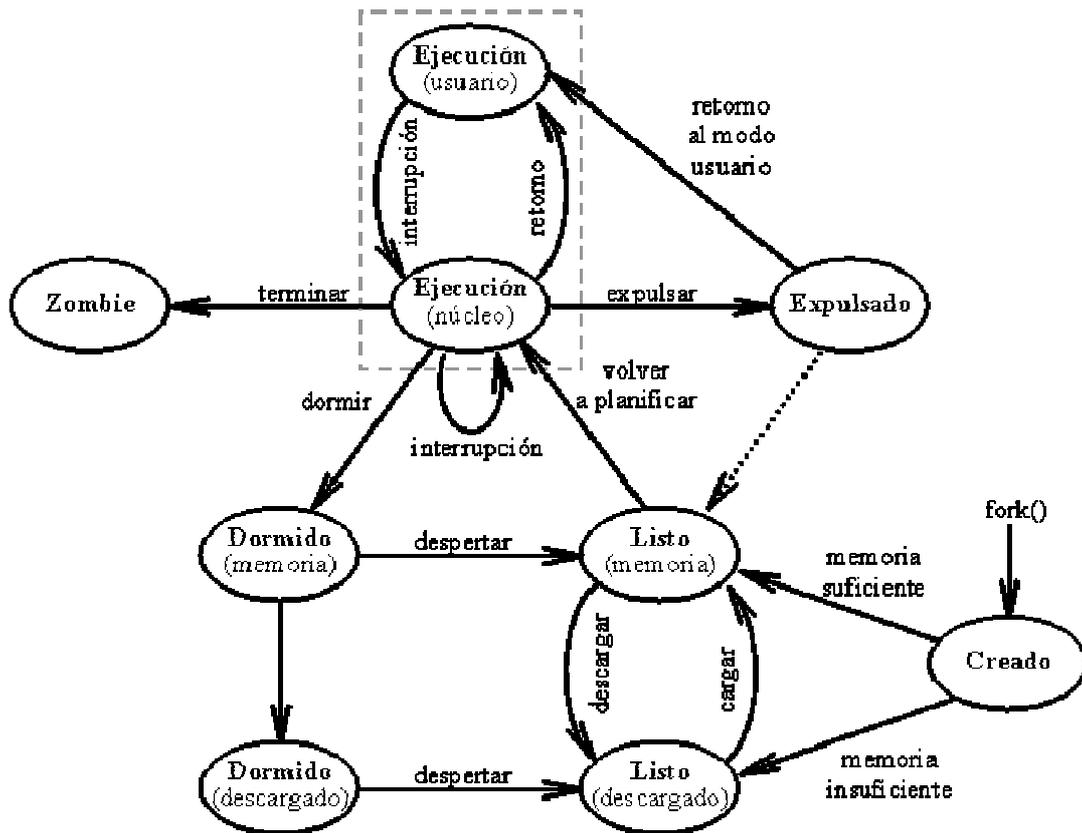


Figura 2. 15 Ejemplo de diagrama de estados.

La mayoría de las veces, al utilizarse los diagramas de estados para modelar los aspectos dinámicos de un sistema, esto supone el modelado del comportamiento de objetos reactivos. Un objeto reactivo es aquel para el que la mejor forma de caracterizar su comportamiento es señalar cuál es su respuesta a los eventos lanzados desde fuera de su contexto. Estos objetos tienen un ciclo de vida bien definido, cuyo comportamiento se ve afectado por su pasado.

Los diagramas de estados no sólo son importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables a través de ingeniería directa e inversa.

2.13.3 DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia son tipos de diagramas de interacción. Un diagrama de interacción muestra un conjunto de objetos o roles y los mensajes que pueden ser enviados entre ellos. Los diagramas de interacción cubren la vista dinámica de un sistema.

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes. Gráficamente, un diagrama de secuencia es una tabla que representa objetos, dispuestos a lo largo del eje X, y mensajes, ordenados temporalmente, a lo largo del eje Y. En la Figura 2.16 se puede observar un ejemplo de un diagrama de secuencia del préstamo de libros de una biblioteca.

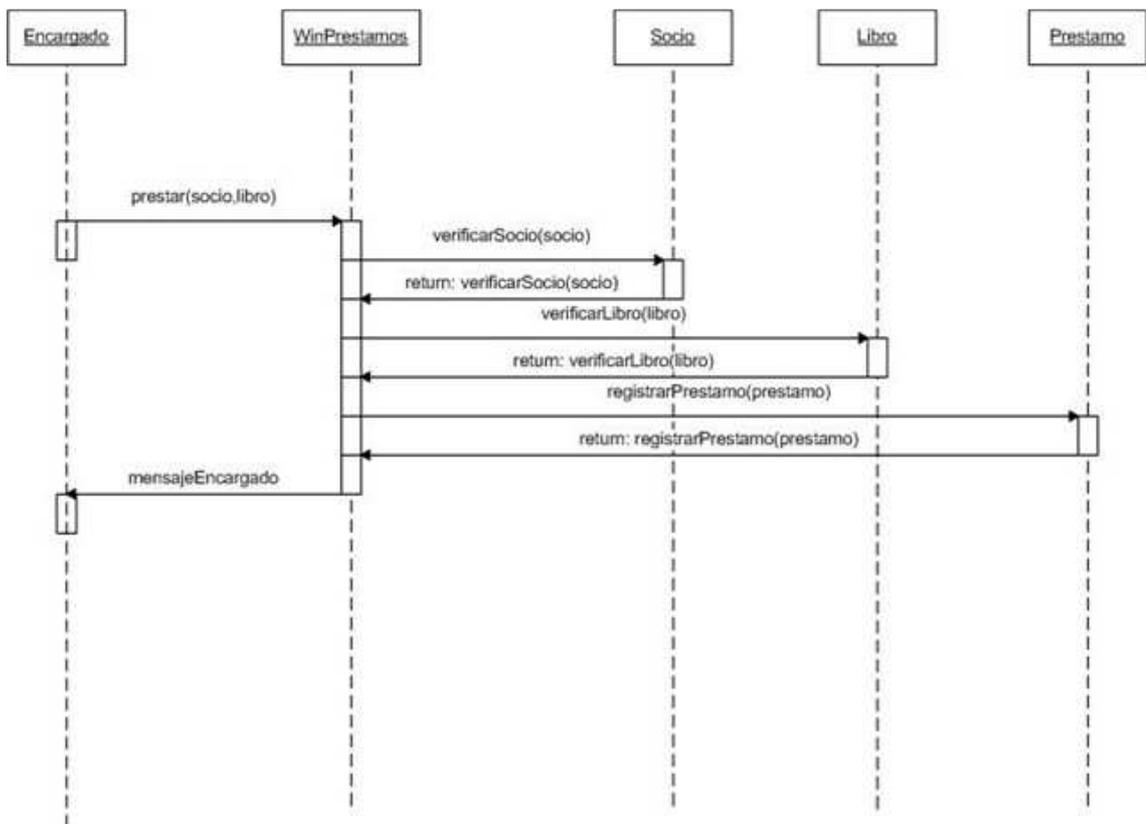


Figura 2. 16 Ejemplo de diagrama de secuencia.

CAPÍTULO 3: Descripción de la aplicación. Cliente y servidor.

La aplicación desarrollada en el presente proyecto está compuesta por dos partes fundamentales, una que reside en un servidor remoto y que se encarga de almacenar los datos y proporcionar los contenidos a la otra parte, la de los dispositivos móviles, que interpretan dichos datos y los muestran a los usuarios de una forma fácil y clara.

3.1 CLIENTE

Este apartado se centra en la parte referida al cliente móvil de la aplicación. Este cliente debe proporcionar el interfaz gráfico con el que los usuarios puedan obtener los datos deseados. Para ello es necesario un sistema de navegación entre pantallas que permita a los usuarios seleccionar un elemento determinado y obtener todos sus datos, como pueden ser el título, autor o la ubicación actual.

Como ya se ha comentado en el Capítulo 1, para la implementación del cliente se ha elegido usar el *framework* PhoneGap, ya que, al permitir programar en HTML5, la división de la aplicación en pantallas es relativamente sencilla. Se puede observar en la Figura 3.1 el esqueleto de navegación entre pantallas del cliente.

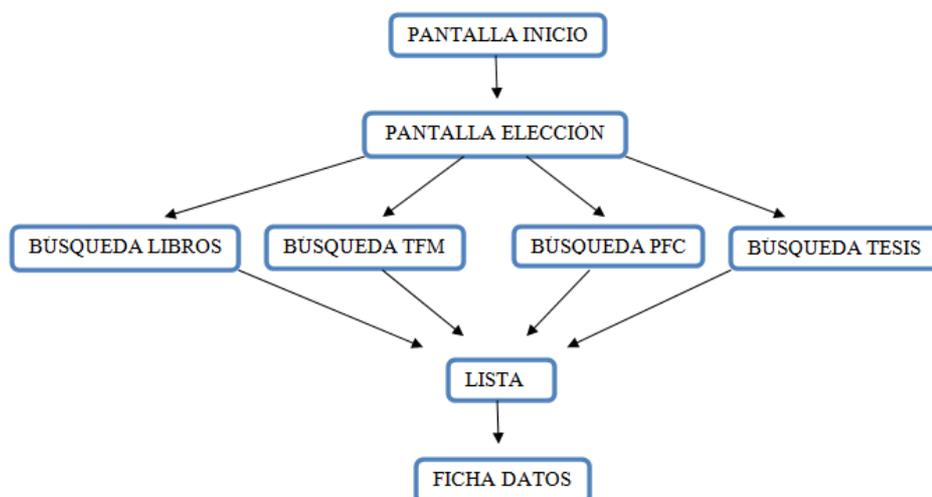


Figura 3.1 Esqueleto de navegación entre pantallas de la aplicación móvil cliente.

3.1.1 PANTALLAS DEL CLIENTE

La aplicación móvil dispone de una pantalla de inicio, una pantalla de elección de búsqueda, cuatro pantallas de búsqueda diferentes (libros, PFC, TFM, tesis), un pantalla que lista los elementos y una pantalla que muestra los datos de un elemento en concreto. A continuación se procede a explicar cada una de las pantallas de las que se compone la aplicación, mostrando además una captura de pantalla del dispositivo móvil en el que se ha probado, un Samsung Galaxy Note. En cada una de ellas se va a explicar qué partes de la página HTML5 son estáticas, es decir, el código está escrito directamente en el archivo HTML, y cuáles son dinámicas, las generan las funciones Javascript dependiendo de los datos que se reciben del servidor.

3.1.1.1 Pantalla de inicio

La pantalla de inicio, mostrada en la Figura 3.2, es la que aparece al iniciar la aplicación. Transcurridos tres segundos la aplicación muestra la pantalla de elección de búsqueda. En esta pantalla todo el contenido HTML es estático.



Figura 3.2 Pantalla de inicio de la aplicación móvil.

3.1.1.2 Pantalla de elección de búsqueda

En la pantalla de elección de búsqueda, Figura 3.3, el usuario elige el tipo de elemento del que desea obtener información. Dichos elementos pueden ser libros, proyectos final de carrera, trabajos final de máster o tesis. Pulsando en cada uno de los

botones se muestran las respectivas pantallas de búsqueda. En esta pantalla todo el contenido HTML es estático.



Figura 3.3 Pantalla de elección de búsqueda de la aplicación móvil.

3.1.1.3 Pantalla de búsqueda de libros

En esta pantalla (Figura 3.4) se muestran los parámetros por los que se pueden buscar los libros. Dichos parámetros son el título, el autor, el ISBN, el código de la biblioteca, la referencia de la biblioteca, el tejuelo, la editorial y el año de publicación del libro. No es necesario rellenarlos todos para realizar la búsqueda. Una vez elegidos los parámetros, al pulsar el botón "BUSCAR" la aplicación muestra una pantalla con el listado de los resultados obtenidos. Todo el contenido HTML de la página es estático.



Figura 3.4 Pantalla de búsqueda de libros de la aplicación móvil.

3.1.1.4 Pantalla de búsqueda de PFC

Como se puede observar en la Figura 3.5, un PFC permite definir como parámetros de búsqueda el nombre del autor, la titulación a la que pertenece, el título, el nombre del tutor y cotutor (en caso de que este último exista), el tejuelo y la calificación. Tanto el campo de titulación como el campo de calificación son desplegables con las distintas opciones disponibles. El contenido HTML es estático a excepción del desplegable de la titulación, que se genera dinámicamente a partir de los datos obtenidos del servidor.



The screenshot shows a mobile application interface for searching a Final Project (PFC). At the top, there are logos for 'uma' and 'TE'. Below the logos, the title 'Búsqueda Proyecto Fin de Carrera' is displayed. The search form includes the following fields: 'Autor:' (text input), 'Titulación:' (dropdown menu with 'INGENIERÍA TELECOMUNICACIÓN' selected), 'Título:' (text input), 'Tutor:' (text input), 'Cotutor:' (text input), 'Tejuelo:' (text input), and 'Calificación:' (dropdown menu with 'MATRÍCULA DE HONOR' selected). A blue button labeled 'BUSCAR' is positioned at the bottom of the form.

Figura 3.5 Pantalla de búsqueda de PFC de la aplicación móvil.

3.1.1.5 Pantalla de búsqueda de TFM

En la pantalla de búsqueda de TFM los parámetros de búsqueda son prácticamente iguales a los de un PFC, a excepción de la titulación que se sustituye por el nombre del máster. De este modo, el contenido estático es el mismo y sólo cambia dinámicamente el nombre del máster. Su captura de pantalla se puede ver en la Figura 3.6.



Figura 3.6 Pantalla de búsqueda de TFM de la aplicación móvil.

3.1.1.6 Pantalla de búsqueda de tesis

La búsqueda de tesis tiene como parámetros el nombre del autor, el título de la tesis, el director, el codirector y un desplegable con la calificación. En esta pantalla todo el contenido HTML es estático. En la Figura 3.7 se puede ver una captura de esta pantalla.



Figura 3.7 Pantalla de búsqueda de tesis de la aplicación móvil.

3.1.1.7 Pantalla listado de resultados

La pantalla listado de resultados es común a todas las búsquedas, por lo que su contenido se genera dinámicamente. En las Figuras 3.8, 3.9, 3.10 y 3.11 se muestran capturas de pantalla de los listados obtenidos en las búsquedas de libros, PFC, TFM y tesis respectivamente. Cada uno de los elementos del listado están compuestos por el título del libro y su autor. Pulsando en uno de ellos se muestra la pantalla con la ficha de datos del elemento.



Figura 3.8 Pantalla del listado de libros



Figura 3.9 Pantalla del listado de PFC



Figura 3.10 Pantalla del listado de TFM



Figura 3.11 Pantalla del listado de tesis

3.1.1.8 Pantalla ficha de datos

Al igual que ocurre con la pantalla del listado, la pantalla ficha de datos, también es común a todas las búsquedas, aunque los datos mostrados dependen del tipo de libro, como se puede ver en las Figuras 3.12, 3.13, 3.14 y 3.15. Debido a esto, el contenido HTML de la pantalla ficha de datos es dinámico, se genera conforme a los datos obtenidos del servidor. Si uno de los campos de la base de datos no existe, se filtra y no se muestra el campo vacío.



Figura 3.12 Ficha de datos de un libro



Figura 3.13 Ficha de datos de un PFC



Figura 3.14 Ficha de datos de un TFM



Figura 3.15 Ficha de datos de una tesis

3.1.2 RECEPCIÓN Y PROCESADO DE DATOS DEL SERVIDOR EN EL CLIENTE

Como ya se ha comentado en el Capítulo 1, la aplicación móvil realiza peticiones al servidor, que devuelve los datos en forma de documentos XML. Dependiendo de la elección del usuario en la pantalla de elección (libros, PFC, TFM o tesis) la aplicación realizará una búsqueda u otra. En las Figuras 3.16, 3.17, 3.18 y 3.19 respectivamente se pueden observar los diagramas de secuencia para cada una de estas búsquedas.

En el diagrama de secuencia de la Figura 3.16 se muestra la secuencia de comunicación entre cliente y servidor que se produce en la búsqueda de un libro. En primer lugar, en el cliente, se pasa de la página de inicio a la de elección de búsqueda, como se ha visto anteriormente en el diagrama de estados de la Figura 3.1. En la página de búsqueda de libros se definen los parámetros y con la función *\$.obtenerLibros()* se le envían los parámetros al servidor y éste devuelve un documento XML con el listado de libros. A continuación, se muestra el listado de los libros y se realiza una petición al servidor con el identificador del elemento seleccionado. El servidor devuelve otro documento XML con los datos del elemento y el cliente los muestra en pantalla.

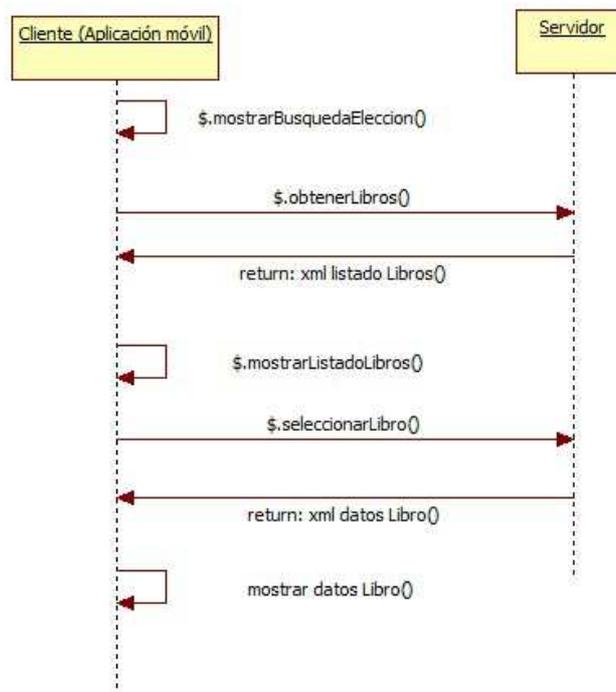


Figura 3.16 Diagrama de secuencia de la búsqueda de un libro

Como se puede observar en las Figuras 3.17, 3.18 y 3.19, la secuencia que sigue la aplicación es prácticamente la misma a la vista en la Figura 3.16. La única diferencia en la Figura 3.17 es la petición de las titulaciones al servidor y la recepción de su correspondiente documento XML. En el caso de la Figura 3.18 la diferencia consiste en la petición del nombre de los másteres disponibles en los TFM.

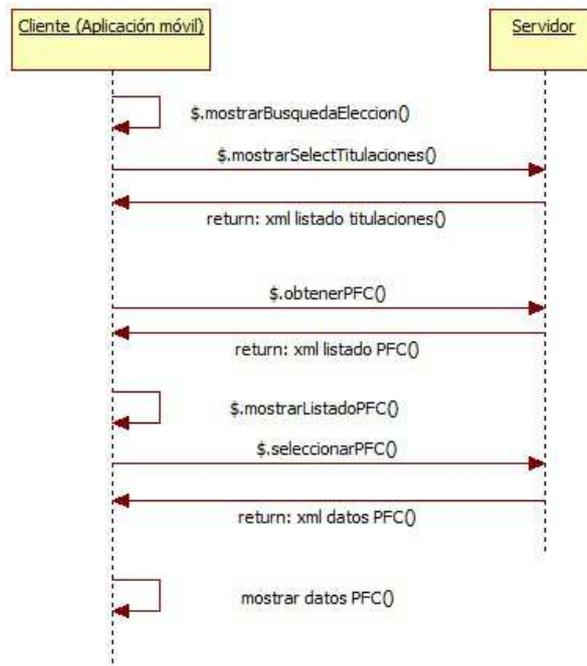


Figura 3.17 Diagrama de secuencia de la búsqueda de un PFC

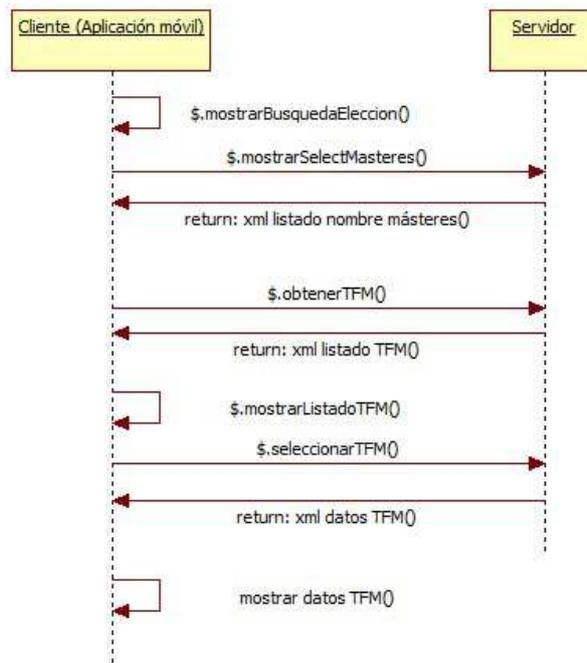


Figura 3.18 Diagrama de secuencia de la búsqueda de un TFM

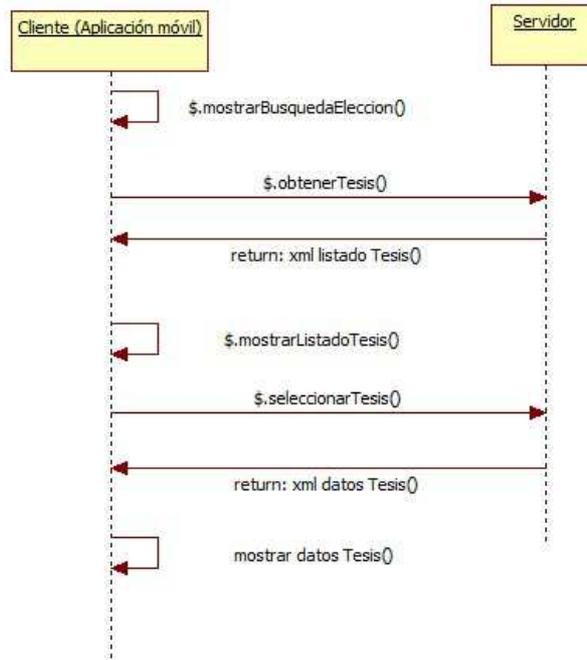


Figura 3.19 Diagrama de secuencia de la búsqueda de una tesis

Los documentos XML se procesan en el cliente con funciones AJAX (*Asynchronous JavaScript and XML*). En la Figura 3.20 se muestran los datos recibidos al realizar la petición de las titulaciones al servidor. Como se puede observar, todos los datos están bajo una etiqueta principal `<datos>` y cada uno de los elementos del listado bajo la etiqueta `<titulacion>`. La función AJAX correspondiente para el procesado de este código XML se puede ver en la Figura 3.21.

La sintaxis de una petición AJAX es muy sencilla: `"$.ajax(opciones)"`. En el ejemplo de dicha Figura, las opciones son las siguientes:

- **URL:** La dirección del servidor a la que se realiza la petición.
- **data:** Información que se incluye en la petición, es decir, los parámetros. En este caso se trata de una cadena de texto, por lo que su formato es "parametro1=valor1¶metro2=valor2".
- **type:** Tipo de petición que se realiza, en nuestro caso "GET".
- **beforeSend:** Permite indicar que se ejecute una función antes de realizar la petición. En este caso no se utiliza.
- **complete:** Establece la función que se ejecuta cuando la petición se ha completado (y después de ejecutar, si se han establecido, las funciones de *success* o *error*). En este caso se utiliza para verificar que los datos recibidos son correctos.

- **error**: Establece la función a ejecutar cuando se produce un error.
- **ifModified**: Permite considerar como correcta la petición solamente si la respuesta recibida es diferente a la anterior respuesta.
- **processData**: Indica si se transforman los datos de la opción *data* para convertirlos en una cadena de texto.
- **async**: Indica si la petición es asíncrona.
- **success**: Permite establecer la función que se ejecuta cuando la petición se ha completado de forma correcta. Esta función recibe como primer parámetro los datos recibidos del servidor. Es en este caso, cuando se procesan los datos obtenidos y se crea dinámicamente el contenido de las páginas de código HTML. En este caso, se lee la etiqueta `<datos>` y se realiza un recorrido por todo el documento leyendo el contenido de todas las etiquetas `<titulacion>` y se añade cada una de ellas como opción al desplegable de titulaciones con ayuda de *jQuery*, un *framework* Javascript.

```

<datos>
  <titulacion>"INGENIERÍA TÉCNICA TELECOMUNICACIÓN"</titulacion>
  <titulacion>
    "INGENIERÍA TÉCNICA TELECOMUNICACIÓN, SISTEMAS ELECTRÓNICOS"
  </titulacion>
  <titulacion>"INGENIERÍA TELECOMUNICACIÓN"</titulacion>
  <titulacion>"INGENIERÍA TÉCNICA INFORMÁTICA, SISTEMAS"</titulacion>
  <titulacion>"INGENIERÍA ORGANIZACIÓN INDUSTRIAL"</titulacion>
  <titulacion>"INGENIERÍA TÉCNICA INDUSTRIAL"</titulacion>
  <titulacion>
    "INGENIERÍA TÉCNICA INDUSTRIAL, ELECTRÓNICA INDUSTRIAL"
  </titulacion>
  <titulacion>"INGENIERÍA INFORMÁTICA"</titulacion>
  <titulacion>"INGENIERÍA TÉCNICA INDUSTRIAL, MECÁNICA"</titulacion>
  <titulacion>"INGENIERÍA TÉCNICA INDUSTRIAL, DISEÑO INDUSTRIAL"</titulacion>
  <titulacion>"INGENIERÍA TÉCNICA INFORMÁTICA, GESTIÓN"</titulacion>
  <titulacion>"INGENIERÍA ELECTRÓNICA"</titulacion>
  <titulacion>"INGENIERÍA TÉCNICA INDUSTRIAL, ELECTRICIDAD"</titulacion>
  <titulacion>"INGENIERÍA INDUSTRIAL"</titulacion>
  <titulacion>"INGENIERÍA AUTOMÁTICA Y ELECTRÓNICA INDUSTRIAL"</titulacion>
</datos>

```

Figura 3.20 XML obtenido al realizar la petición de las titulaciones en un PFC.

```

$.ajax({
  url: "http://194.140.146.17:8080/index.php/datos/listadoTitulaciones",
  data: "tabla="+tabla,
  type: "GET",
  beforeSend: function(objeto){
  },
  complete: function(objeto, exito){
    //alert(objeto.responseText);
  },
  error: function(objeto, quepaso, otroobj){
    alert('error de conexion');
    $.cancelarCargando();
  },
  ifModified: false, processData:true, async:false,
  success:function(datos){
    try{
      var aux='<option selected value="todas">TODAS</option>';
      $(datos).find('datos').each(function(){
        $(this).find('titulacion').each(function(){
          aux+='<option
value="'+$(this).text().replace(/['"]/g, '')+'">'+$(this).text().replace(/['"]/g, '')
+'</option>';
        });
      });
      $('#titulacion-'+tabla+' > *').remove();
      $('#titulacion-'+tabla).append(aux);
    }catch(e){
      alert(e);
    }
  },
  timeout: 60000
});

```

Figura 3.21 Función AJAX en JavaScript que procesa el documento XML de las titulaciones.

3.2 SERVIDOR

Este apartado se centra en la parte del servidor. El servidor proporciona por un lado respuestas a las peticiones que le realiza el cliente (devolviendo documentos en lenguaje XML con los datos) y por otro, proporciona un interfaz de usuario para la administración de los datos almacenados.

Como ya se ha comentado anteriormente en el Capítulo 2, las funciones del servidor se han escrito en lenguaje PHP con la ayuda del *framework* Symfony en su versión 1.4. En la Figura 3.22 se puede observar la estructura de la aplicación Symfony de este proyecto.

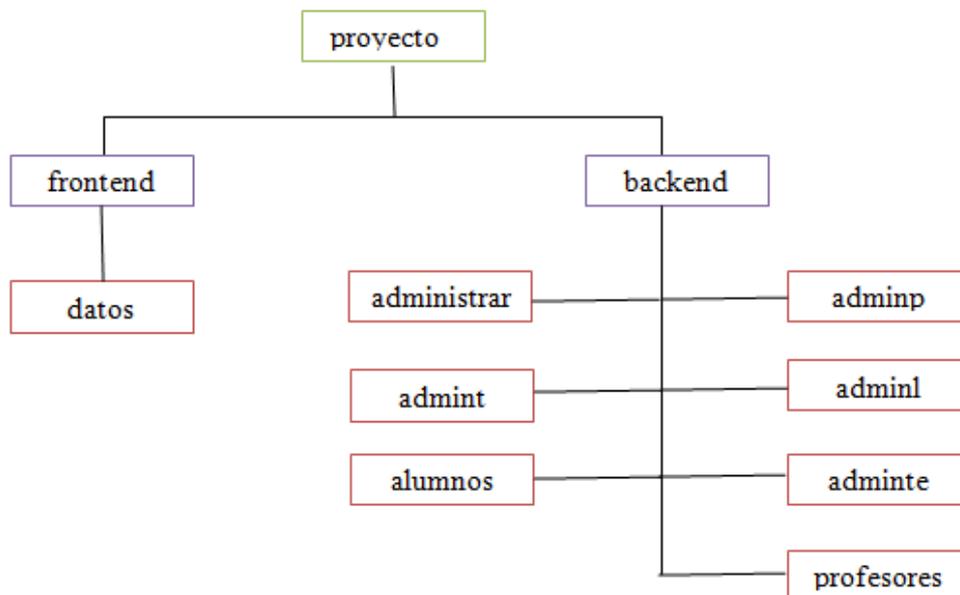


Figura 3.22 Estructura de la aplicación Symfony

Una aplicación Symfony está dividida en dos partes fundamentales, el *frontend* que se trata de la parte pública de la aplicación y el *backend*, que se refiere a la parte de administración. Además, cada una de estas partes están divididas en módulos. En este caso el *frontend* consta de un único módulo (*datos*) y el *backend* consta de siete módulos (*administrar*, *adminp*, *adminl*, *adminte*, *alumnos*, *profesores*).

3.2.1 BASE DE DATOS

Toda aplicación Symfony tiene detrás una base de datos, que puede estar escrita en lenguaje *Propel* o en lenguaje *Doctrine*. Una vez definidas las tablas de la base de datos y las relaciones entre ellas en un lenguaje u otro, al construir el proyecto, Symfony crea la base de datos MySQL de forma transparente al desarrollador. En este caso se ha elegido *Doctrine*, pero igualmente se podría haber elegido *Propel*. En la Figura 3.23 se pueden observar el esquema con las distintas tablas necesarias en la aplicación.

En Symfony la descripción de las tablas y sus relaciones se pueden hacer de dos formas diferentes: por ingeniería inversa de una base de datos existente o creando la definición a mano editando el archivo *schema.yml*. Dicha definición consta del nombre de la tabla, y a continuación se definen el tipo de conexión, el nombre de la tabla (si se quiere cambiar con respecto al nombre citado anteriormente) y las columnas. Cada columna se puede describir con la siguiente información:

- **type:** El tipo de columna (*boolean, integer, float, decimal, string, array, object, blob, clob, timestamp, time, date, enum, gzip*).
- **notnull:** Es *true* si se desea que la columna sea obligatoria.
- **unique:** Es *true* si se desea crear un índice único para la columna.

Cada nivel de indentación en el esquema se realiza con dos espacios en blanco, nunca con tabulaciones.

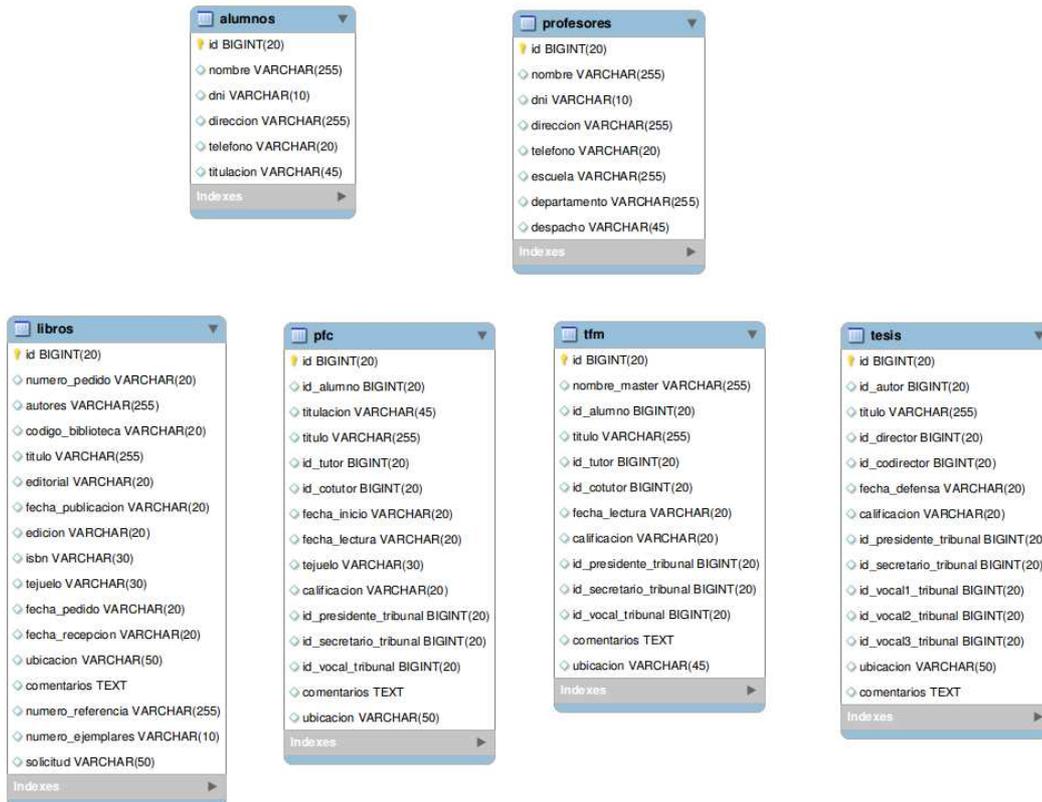


Figura 3.23 Esquema de las tablas de la base de datos realizado con MySQLWorkbench.

A continuación, se explican cada una de las tablas y se muestra su definición en *Doctrine*:

- Tabla *alumnos*: Esta tabla va a almacenar los datos de los alumnos. Dichos datos son el identificador, que se autogenera al insertar un nuevo elemento en la base de datos, el nombre, el dni, la dirección, el teléfono y la titulación a la que pertenece. Todos los campos excepto el identificador son de tipo *VARCHAR* aunque de diferente tamaño dependiendo del tamaño de la información del campo. Así por ejemplo, el dni tiene tamaño diez y el teléfono veinte. Todos los identificadores de todas las tablas son de tipo

INTEGER de tamaño ocho. En la Figura 3.24 se puede ver la definición de la tabla *alumnos* en doctrine.

- Tabla *profesores*: En esta tabla se almacena la información relativa a los profesores. Es muy similar a la tabla *alumnos* y consta del identificador, el nombre, el dni, la dirección, el teléfono, la escuela, el departamento y el número de despacho. En la Figura 3.25 se muestra la definición de esta tabla en el esquema.
- Tabla *libros*: Almacena la información relativa a los libros. Sus campos son el identificador, el numero de pedido, los autores, el código de la biblioteca, el título, la editorial, la fecha de publicación (el año), el número de edición, el código ISBN, el tejuelo, la fecha de pedido, la fecha de recepción, la ubicación, algún comentario referente al libro, el número de referencia, el número de ejemplares y quien solicitó el libro. El esquema de esta tabla se puede ver en la Figura 3.26.
- Tabla *pfc*: En esta tabla se almacenan todos los datos de los proyectos fin de carrera. Estos datos están compuestos por el identificador, el identificador del autor (un identificador de alumno), el identificador del tutor y cotutor en caso de que exista, el título, la titulación, la fecha de inicio, la fecha de lectura, el tejuelo, la clasificación, el identificador de los miembros del tribunal que lo evaluó (presidente, secretario y vocal), la ubicación y un campo de comentarios. Los identificadores del tutor, cotutor y de los miembros del tribunal son identificadores de profesores. Su definición se muestra en la Figura 3.27.
- Tabla *tfm*: Esta tabla almacena los datos de los trabajos fin de máster. Es muy similar a la tabla *pfc*. La diferencia consiste en que esta tabla carece de los campos fecha de inicio y tejuelo y que el campo titulación se sustituye por el campo nombre del máster. Su definición se muestra en la Figura 3.28.
- Tabla *tesis*: Esta tabla es similar a las tablas *pfc* y *tfm*. Su definición se muestra en la Figura 3.29.

```
alumnos:
connection: doctrine
tableName: alumnos
columns:
id:{type: integer(8), fixed: false, unsigned: false, primary: true,
    autoincrement:true}
nombre: { type: varchar(255), notnull: true }
dni:{ type: varchar(10), notnull: true}
direccion: { type: varchar(255), notnull: true}
telefono: { type: varchar(20)}
titulacion: { type: varchar(45), notnull: true}
curso_finalizacion_estudios: { type: varchar(255), notnull: true}
```

Figura 3.24 Definición de la tabla "alumnos" en Doctrine

```
profesores:
connection: doctrine
tableName: profesores
columns:
id:{type: integer(8), fixed: false, unsigned: false, primary: true,
    autoincrement: true}
nombre: { type: varchar(255)}
dni:{ type: varchar(10)}
direccion: { type: varchar(255)}
telefono: { type: varchar(20)}
escuela: { type: varchar(255)}
departamento: { type: varchar(255)}
despacho: { type: varchar(45)}
```

Figura 3.25 Definición de la tabla "profesores" en Doctrine

```

pfc:
connection: doctrine
tableName: pfc
columns:
id:{type: integer(8), fixed: false, unsigned: false, primary: true,
autoincrement: true}
id_alumno:{ type: integer(8)}
titulacion: { type: varchar(255)}
titulo: { type: varchar(255)}
id_tutor:{ type: integer(8)}
id_cotutor:{ type: integer(8)}
fecha_inicio: { type: varchar(20)}
fecha_lectura: { type: varchar(20)}
tejuelo: { type: varchar(255) }
calificacion: { type: varchar(255) }
id_presidente_tribunal:{ type: integer(8)}
id_secretario_tribunal:{ type: integer(8)}
id_vocal_tribunal:{ type: integer(8)}
comentarios: { type: varchar(1024) }
ubicacion: { type: varchar(50)}

```

Figura 3.26 Definición de la tabla "libros" en Doctrine

```

libros:
connection: doctrine
tableName: libros
columns:
id:{type: integer(8), fixed: false, unsigned: false, primary: true,
autoincrement: true}
numero_pedido: { type: varchar(20)}
autores: { type: varchar(255)}
codigo_biblioteca: { type: varchar(20)}
titulo: { type: varchar(255)}
editorial: { type: varchar(20)}
fecha_publicacion: { type: varchar(20)}
edicion: { type: varchar(20)}
isbn: { type: varchar(30)}
tejuelo: { type: varchar(30)}
fecha_pedido: { type: varchar(20)}
fecha_recepcion: { type: varchar(20)}
ubicacion: { type: varchar(50)}
comentarios: { type: varchar(1024)}
numero_referencia: { type: varchar(255)}
numero_ejemplares: { type: varchar(255)}
solicitud: { type: varchar(50)}

```

Figura 3.27 Definición de la tabla "pfc" en Doctrine

```
tfm:
connection: doctrine
tableName: tfm
columns:
id:{type: integer(8), fixed: false, unsigned: false, primary: true,
autoincrement: true}
nombre_master: { type: varchar(255)}
id_alumno:{ type: integer}
titulo: { type: varchar(255)}
id_tutor:{ type: integer}
id_cotutor:{ type: integer}
fecha_lectura: { type: varchar(20)}
calificacion: { type: varchar(20)}
id_presidente_tribunal:{ type: integer, notnull: true}
id_secretario_tribunal:{ type: integer, notnull: true}
id_vocal_tribunal:{ type: integer, notnull: true}
comentarios: { type: varchar(1024) }
ubicacion: { type: varchar(50)}
```

Figura 3.28 Definición de la tabla "tfm" en Doctrine

```
tesis:
connection: doctrine
tableName: tesis
columns:
id:{type: integer(8), fixed: false, unsigned: false, primary: true,
autoincrement: true}
id_autor:{ type: integer}
titulo: { type: string(255)}
id_director:{ type: integer}
id_codirector:{ type: integer}
fecha_defensa: { type: varchar(20)}
calificacion: { type: string(255)}
id_presidente_tribunal:{ type: integer}
id_secretario_tribunal:{ type: integer}
id_vocal1_tribunal:{ type: integer}
id_vocal2_tribunal:{ type: integer}
id_vocal3_tribunal:{ type: integer}
ubicacion: { type: varchar(50)}
```

Figura 3.29 Definición de la tabla "Tesis" en Doctrine

3.2.2 FRONTEND DE LA APLICACIÓN

El *frontend* de la aplicación Symfony es la parte del usuario y, en este caso, consta solamente de un módulo, el módulo *datos*, que se encarga de crear los documentos XML con los datos que se envían a la aplicación móvil. Un módulo consta de *actions*, los archivos que indican la funcionalidad, *templates*, los archivos que indican la vista de la aplicación y archivos de configuración.

3.2.2.1 Módulo datos

Como se puede observar en la Figura 3.30, el módulo *datos* consta de los subdirectorios *actions*, *config* y *templates*. En el archivo *actions.class.php* se especifican las funciones que crean los documentos XML anteriormente comentados. Bajo el directorio *config* se encuentran los archivos de configuración. En un proyecto Symfony los archivos de configuración siguen una estructura piramidal, de forma que la configuración final de cada parte del proyecto se realiza tomando la configuración principal y complementándola con las configuraciones que se añaden en los siguientes niveles. De esta manera, si en un nivel inferior se sobrescribe una determinada configuración, se toma esta última, pero solo para la parte afectada. Por ejemplo, en el directorio *config* del módulo *datos* se encuentra el archivo *view.yml*, por lo que dicho módulo tomará la configuración del *frontend*, sustituyendo la configuración del fichero *view.yml* por la suya propia. Bajo el directorio *templates* se encuentran las plantillas que forman la vista final. En este caso se tienen el archivo por defecto, *indexSuccess.php* y las plantillas que se han creado para generar los documentos XML con los datos.

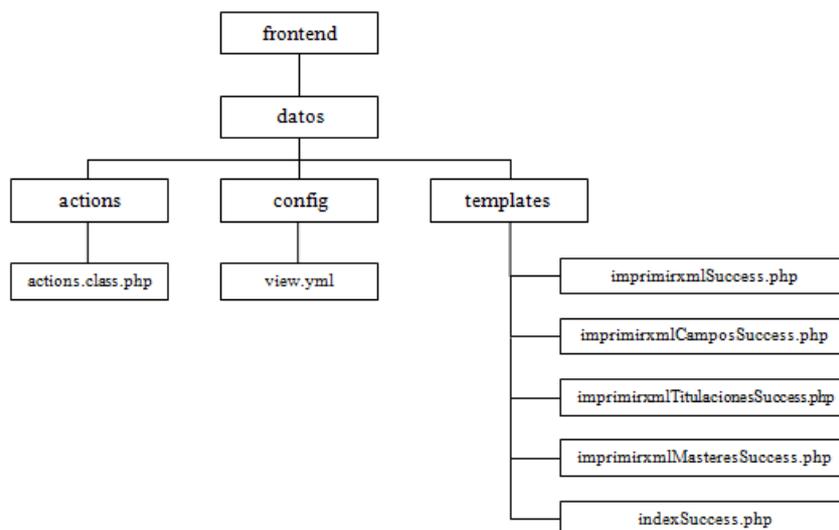


Figura 3.30 Estructura de archivos del módulo *frontend*.

action.class.php
<pre> executeIndex(sfWebRequest \$request); executeListadoLibros (sfWebRequest \$request); executeDatosLibro (sfWebRequest \$request); executeListadoPFC (sfWebRequest \$request); executeDatosPFC (sfWebRequest \$request); executeListadoTFM (sfWebRequest \$request); executeDatosTFM (sfWebRequest \$request); executeListadoTesis (sfWebRequest \$request); executeDatosTesis (sfWebRequest \$request); executeListadoTitulaciones (sfWebRequest \$request); executeListadoMasteres (); </pre>

Figura 3.31 Tabla de las funciones del archivo *actions.class.php* del módulo datos.

En la Figura 3.31 se muestran las funciones del archivo *actions.class.php*. A continuación se explican dichas funciones junto con los *templates* a los que llaman:

- *executeIndex(sfWebRequest \$request)*: Esta función se crea por defecto al crear el módulo. Llama al *template indexSuccess.php*. En este caso no se usa.
- *executeListadoLibros(sfWebRequest \$request)*: Al ejecutarse esta función se realiza una búsqueda con los parámetros que se le pasan y se crea un *array* de resultados. Dicha búsqueda se realiza con una consulta Doctrine a la base de datos. En el cuadro de texto de la Figura 3.32 se puede observar una de las consultas que se realizan en esta función. En el ejemplo, en la variable *\$libros* se almacenan los resultados obtenidos al realizar la consulta a la tabla de la base de datos *libros* en los que el campo título sea similar al del parámetro "título" que se le ha pasado a la función. A continuación, para cada resultado se crea un *array* en el que se almacena el identificador, el título y el autor o autores y se le añade al *array* que se va a enviar con los datos al *template imprimirxmlcamposSuccess.php*.

```

$titulo=$request->getParameter('titulo');
...
if($titulo!='' && $titulo!=null){
    $libros=Doctrine_Query::create()
        ->from('libros l')
        ->where('l.titulo LIKE ?', '%'.$titulo.'%')
        ->fetchArray();
    foreach ($libros as $libro){
        $arraylibro = array(
            'idlibro' => $libro['id'],
            'titulo' => $libro['titulo'],
            'autores'=> $libro['autores']
        );
        if(!in_array($arraylibro,$arrayenviar)){
            array_push($arrayenviar, $arraylibro);
        }
    }
}
...
$this->variables=$arrayenviar;
$this->setTemplate('imprimirxmlcampos');

```

Figura 3.32 Ejemplo de consulta Doctrine.

- En la Figura 3.33 se tiene el código del *template imprimirxmlcamposSuccess.php*. Este *template* se utiliza para todas las funciones implementadas en las que se requiere listar los elementos. Es el encargado de generar el documento XML con el listado de resultados. En la Figura 3.34 se puede ver un ejemplo de documento XML que se quiere obtener como resultado. En el ejemplo se le ha pasado como parámetro "autor=porat". Para ello, en el *template*, se recorren todos los elementos del *array \$variables* y por cada uno de ellos se genera una nueva etiqueta "<datos>" dentro de la cual se generan nuevas etiquetas según los pares clave-valor, donde los nombres de las etiquetas son las claves y el contenido los valores.
- executeDatosLibro(sfWebRequest \$request)*: A esta función se le pasa el identificador de un libro en concreto, obtiene los datos almacenados en la base de datos relativos a ese libro y genera un *array* con ellos. Para ello, se utiliza una consulta Doctrine que crea un objeto *\$libro* con todos los datos del libro que tiene por identificador el que se le ha pasado como parámetro a la función. Al

tratarse de un objeto, el valor de los campos se obtienen mediante la función "getNombreDelCampo()". Una vez creado el *array* se llama al *template imprimirxmlSuccess.php* y éste genera el documento XML que será devuelto por el servidor. En los cuadros de texto de las Figuras 3.35 y 3.36 respectivamente se muestra el código de la función y del *template*.

```
<?php if (isset ($variables)){?>
    <datos>
        <?phpforeach ($variables as $key => $value) :?>
            <dato>
                <?phpforeach ($value as $key2 => $value2) :?>
                    <<?php echo $key2 ?>>
                    <?php echo $value2?>
                </<?php echo $key2?>>
            <?phpendforeach; ?>
        </dato>
    <?phpendforeach; ?>
</datos>
<? }?>
```

Figura 3.33 Código PHP del *templateimprimirxmlcamposSuccess.php*.

```
<datos>
    <dato>
        <id>2</id>
        <titulo>INTRODUCTION TO DIGITAL TECHNIQUES</titulo>
        <autores>PORAT, D.I. Y OTRO</autores>
    </dato>
    <dato>
        <id>26</id>
        <titulo>DESIGNING WITH TTL INTEGRATED CIRCUITS</titulo>
        <autores>TEXAS INSTRUMENTS INCORPORATED</autores>
    </dato>
</datos>
```

Figura 3.34 XML devuelto por el servidor al ejecutar la función *executelistadoLibros()*.

```

public function executeDatosLibro(sfWebRequest $request){
    $libro=Doctrine::getTable('libros')->find($request->getParameter('id'));
    $arrayenviar=array(
        'idlibro' => $libro->getId(),
        'titulo' => $libro->getTitulo(),
        'autores'=> $libro->getAutores(),
        'numeropedido' => $libro->getNumeroPedido(),
        'edicion' => $libro->getNumeroPedido(),
        'isbn' => $libro->getIsbn(),
        'codigobiblioteca' => $libro->getCodigoBiblioteca(),
        'numeroejemplares' => $libro->getNumeroEjemplares(),
        'ubicacion' => $libro->getUbicacion(),
        'solicitud' => $libro->getSolicitud()
    );
    $this->variables=$arrayenviar;
    $this->setTemplate('imprimirxml');
}

```

Figura 3.35 Código PHP de la función *executeDatosLibro()*.

```

<?php if (isset ($variables)){?>
    <datos>
        <?phpforeach ($variables as $key => $value) :?>
            <<?php echo $key?>><?php echo $value?></<?php echo key?>>
        <?phpendforeach; ?>
    </datos>
<? }?>

```

Figura 3.36 Código PHP del *template imprimirxmlSuccess.php*.

- *executeListadoPFC(sfWebRequest \$request)*: Esta función realiza una búsqueda en la tabla PFC de la base de datos, genera un *array* con los resultados y llama al *template imprimirxmlcamposSuccess.php*. Es muy similar a la función *executeListadoLibros()*.
- *executeDatosPFC(sfWebRequest \$request)*: De la misma manera que la función *executeDatosLibro()* busca los datos de un libro, esta función buscar los datos de un PFC y llama al *template imprimirxmlSuccess.php*.

- *executeListadoTFM(sfWebRequest \$request)*: Como se puede suponer, esta función realiza una búsqueda similar a la realizada en las funciones *executeListadoLibros()* o *executeListadoPFC()*, sólo que en este caso se buscan TFM. De la misma forma que en dichas funciones llama al *template imprimirxmlcamposSuccess.php*.
- *executeDatosTFM(sfWebRequest \$request)*: Cuando se ejecuta esta función se realiza una búsqueda en la tabla "tfm" y se obtienen los datos para un TFM en concreto según el identificador que se facilita como parámetro. Una vez obtenidos los datos se llama al *template imprimirxmlSuccess.php* y éste genera el XML correspondiente.
- *executeListadoTesis(sfWebRequest \$request)*: En esta función se realiza una búsqueda en la tabla "tesis", se genera una *array* con los resultados y se llama al *template imprimirxmlcamposSuccess.php*.
- *executeDatosTesis(sfWebRequest \$request)*: Esta función obtiene los datos de una tesis y llama al *template imprimirxmlSuccess.php* para que devuelva el documento XML de los resultados.
- *executeListadoTitulaciones(sfWebRequest \$request)*: Con esta función se recorren todos los elementos almacenados en la tabla que se le pasa como parámetro (por defecto es la tabla "pfc") y obtiene todas las titulaciones, sin repetirse, y llama al *template imprimirxmltitulacionesSuccess.php*, obteniendo como resultado un documento XML como el que se mostró en la Figura 3.20.
- *executeListadoMateres()*: De la misma forma que en la función *executeListadoTitulaciones()* el resultado es un listado con las titulaciones, esta función obtiene de la tabla "tfm" de la base de datos el valor del campo "nombre_master" y lo guarda en un *array* sin repetirse. A continuación, llama al *template imprimirxmlmateresSuccess.php* y se genera un documento XML como el que se puede observar en la Figura 3.37.

```
<datos>
  <master>"TECNOLOGÍAS DE TELECOMUNICACIÓN"</master>
  <master>"SISTEMAS ELECTRÓNICOS PARA ENTORNOS INTELIGENTES"</master>
  <master>"INGENIERÍA DE FABRICACIÓN"</master>
</datos>
```

Figura 3.37 XML generado al ejecutar la función *executeListadoMateres()*.

3.2.3 BACKEND DE LA APLICACIÓN

El *backend* de la aplicación Symfony es la parte correspondiente a la administración de la base de datos. Si bien la base de datos se podría administrar con la ayuda de programas como PHPMyAdmin, resultaría una labor tediosa para alguien inexperto, por lo que el *backend* se hace necesario.

Como se ha mostrado anteriormente en el diagrama de la Figura 3.22, el *backend* consta de siete módulos, uno por cada tabla de la base de datos más uno que permite la elección entre ellos. A continuación se explican dichos módulos:

- Módulo *administrar*: Este módulo es el que permite seleccionar el tipo de libro del que se desea introducir un nuevo elemento o modificar uno existente (de las tablas "*libros*", "*pf*", "*tfm*", "*tesis*"). También permite introducir o modificar columnas en las tablas "*profesores*" o "*alumnos*". Al pulsar el botón correspondiente (Figura 3.38) este módulo redirige a los otros módulos.
- Módulo *adminl*: Este módulo es el encargado de la gestión de los elementos de la tabla "*libros*". Al haber usado el *framework* Symfony, la creación de los módulos de administración es relativamente sencilla, con tan solo la ejecución de unos comandos se generan automáticamente un cuadro de búsqueda de elementos (Figura 3.39) y una tabla con los elementos listados (Figura 3.40). El cuadro de búsqueda permite señalar si algún campo está vacío y vaciar todos los campos para generar una nueva búsqueda. La tabla que contiene a los elementos permite editar o borrar un elemento de la tabla. Además, se pueden seleccionar varios elementos y borrarlos todos o crear un nuevo elemento.

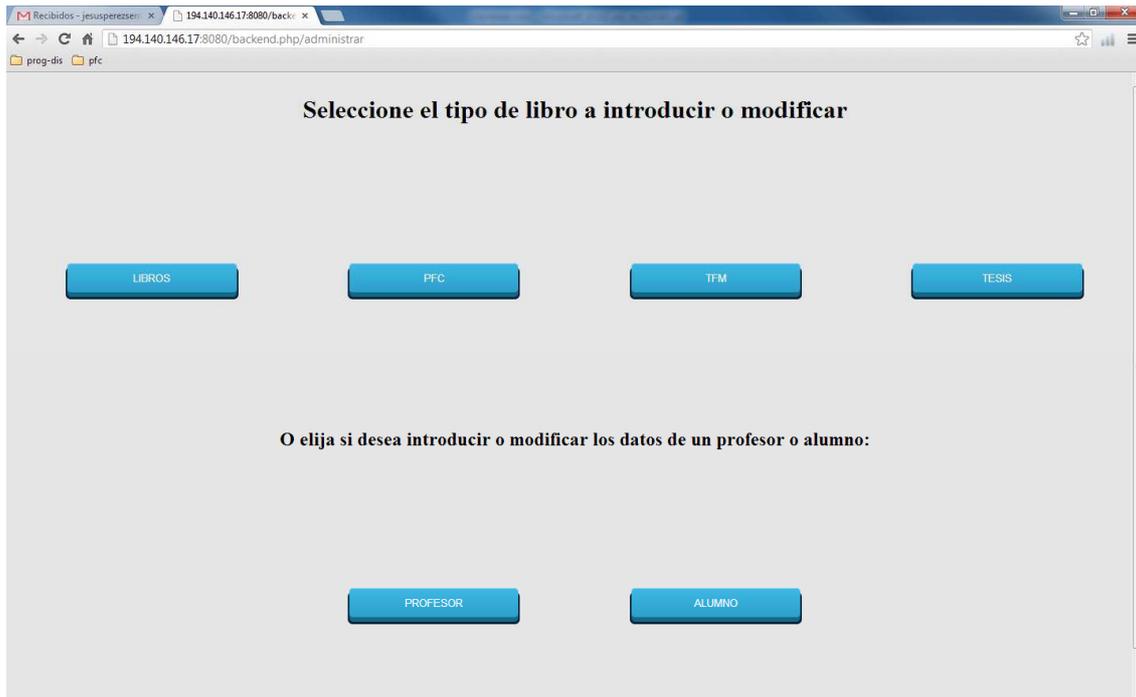


Figura 3.38 Aspecto de la pantalla de elección del módulo administrar.

- Módulo *adminp*: Con este módulo se realizan las tareas de administración de la tabla "*pfc*". Al igual que en el módulo *adminl* se tiene un cuadro de búsqueda y una tabla de resultados.
- Módulo *admint*: Con el módulo *admint* se administra la tabla "*tfm*" de la base de datos. Como los módulos *adminl* y *adminp* dispone de un cuadro de búsqueda y la tabla de resultados.
- Módulo *adminte*: Este módulo es muy similar a los módulos *adminl*, *adminp* y *admint*. Al igual que ellos dispone de un cuadro de búsqueda de elementos y una tabla con los resultados obtenidos.
- Módulo *alumnos*: El módulo alumnos se encarga de la administración de los elementos de la tabla "*alumnos*" de la base de datos. Dispone también de un cuadro de búsqueda y de una tabla con los resultados de dicha búsqueda.
- Módulo *profesores*: Con el módulo *profesores* se administra la tabla "*profesores*" de la base de datos. Al igual que los módulos anteriores se tiene de un cuadro de búsqueda y una tabla de resultados.

Admin List

Numero pedido	<input type="text"/>	<input type="checkbox"/> is empty
Autores	<input type="text" value="porat"/>	<input type="checkbox"/> is empty
Codigo biblioteca	<input type="text"/>	<input type="checkbox"/> is empty
Titulo	<input type="text"/>	<input type="checkbox"/> is empty
Editorial	<input type="text"/>	<input type="checkbox"/> is empty
Fecha publicacion	<input type="text"/>	<input type="checkbox"/> is empty
Edicion	<input type="text"/>	<input type="checkbox"/> is empty
Isbn	<input type="text"/>	<input type="checkbox"/> is empty
Tejuelo	<input type="text"/>	<input type="checkbox"/> is empty
Fecha pedido	<input type="text"/>	<input type="checkbox"/> is empty
Fecha recepcion	<input type="text"/>	<input type="checkbox"/> is empty
Ubicacion	<input type="text"/>	<input type="checkbox"/> is empty
Comentarios	<input type="text"/>	<input type="checkbox"/> is empty
Numero referencia	<input type="text"/>	<input type="checkbox"/> is empty
Numero ejemplares	<input type="text"/>	<input type="checkbox"/> is empty
Solicitud	<input type="text"/>	<input type="checkbox"/> is empty
Reset		<input type="button" value="Filter"/>

Figura 3.39 Cuadro de búsqueda de un libro del módulo adminl.

Id	Numero pedido	Autores	Codigo biblioteca	Titulo	Editorial	Fecha publicacion	Edicion	Isbn	Tejuelo	Fecha pedido	Fecha recepcion	Ubicacion	Comentarios	Numero referencia	Numero ejemplares	Solicitud	Acciones
2	2	PORAT, D I Y OIRO		INTRODUCTION TO DIGITAL TECHNIQUES	JOHN WILEY	1987	*	0-471- 09187-1	1/12/87	20/10/83	20/10/83	GADRIH VALENCIA MIRANDA		1435		GADRIH VALENCIA MIRANDA	Edit Delete
26	26	TEXAS INSTRUMENTS INCORPORATED		DESIGNING WITH TTL INTEGRATED CIRCUITS	MICRAW HILL	1975		0-07- Y85793- 8	5/27/70	14/02/84	14/02/84	GADRIH VALENCIA MIRANDA		0472		GADRIH VALENCIA MIRANDA	Edit Delete

2 results

Choose an action

Figura 3.40 Tabla de resultados de un libro del módulo adminl.

CAPÍTULO 4: Plan de pruebas.

En este capítulo se detallarán las pruebas realizadas a las aplicaciones desarrolladas para verificar su correcto funcionamiento, tanto de la aplicación móvil como la aplicación del servidor. Para efectuar dichas comprobaciones, se ha tomado un conjunto significativo de muestras, abarcando la aplicación móvil las pruebas con navegadores Web en el PC, las pruebas con emuladores y las pruebas con dispositivos móviles reales, y en la aplicación del servidor, abarcando las pruebas en distintos navegadores y en los dispositivos móviles reales.

4.1 PRUEBAS DE LA APLICACIÓN MÓVIL

En este apartado se explican las pruebas a las que se ha sometido la aplicación móvil. En el desarrollo de la aplicación se ha pasado por tres fases. En la primera de ellas, en la maquetación HTML, las pruebas se han realizado con un navegador Web. En la segunda, los emuladores de dispositivos móviles se han empleado para probarla y, en la última fase, dichas pruebas se han realizado en dispositivos reales.

4.1.1 PRUEBAS CON NAVEGADORES WEB

Como ya se ha comentado anteriormente, en la fase de maquetación del diseño HTML se han empleado los navegadores Web Internet Explorer, Google Chrome, pero principalmente el navegador más utilizado ha sido Mozilla Firefox, debido a que con la ayuda de su *plugin* Firebug, permite modificar el valor de los atributos al documento HTML y ver paralelamente en tiempo real el resultado de dichas modificaciones.

El conjunto de pruebas realizadas con el navegador abarca principalmente la parte estética de la aplicación, aunque también se incluye en parte el manejo de los menús. En esta fase, los datos con los que se prueba la aplicación son datos genéricos y estáticos. Las pruebas que se han realizado se enumeran a continuación:

- Comprobación de que el tiempo de transición entre la pantalla de inicio y la pantalla de elección de búsqueda es razonable.
- Comprobación de que los tamaños de todas las imágenes de las pantallas tienen tamaño y resolución adecuados.

- Comprobación de que el tamaño de los botones es suficiente para contener el texto requerido.
- Comprobación de que todas las características de fuente (tamaño, tipo de fuente, color, subrayado, negrita, cursiva, etcétera) son las adecuadas dependiendo de la función que desempeña cada texto.
- Comprobación de que los márgenes, expresados en porcentaje del tamaño de la pantalla, son los adecuados para cada elemento.
- Comprobación de que el tamaño de los elementos de la página del listado están proporcionados con respecto a la pantalla.
- Comprobación de que el *scroll* vertical funciona correctamente cuando la vista de una página de la aplicación es mayor que la pantalla.
- Comprobación de que la vista es correcta para distintos tamaños de pantalla.

4.1.2 PRUEBAS CON EMULADORES

En este tipo de pruebas se ha empleado el emulador incluido en el SDK de Android, configurándolo con distintos tamaños de pantalla y con distintas versiones de Android, desde la versión 2.1.3 (*Cupcake*) a la 4.0 (*Ice Cream*). En esta fase las pruebas se centran principalmente en la funcionalidad de la aplicación.

Además de las pruebas del apartado 4.1.1 se han realizado las pruebas que se detallan a continuación:

- Comprobación de que todos los botones internos de la aplicación funcionan correctamente.
- Comprobación de que el botón "volver" del dispositivo Android funciona correctamente.
- Comprobación de que la aplicación se cierra correctamente.
- Comprobación de que la navegación completa entre todas las ventanas es correcta.
- Comprobación de que en el cambio de una pantalla a otra se oculta completamente la primera y se muestra completa la segunda de ellas.
- Comprobación de que la codificación de caracteres en la aplicación es UTF-8.

- Comprobación de que la obtención de datos del servidor se produce correctamente.
- Comprobación de que los datos recibidos del servidor son correctos.
- Comprobación de que cuando se produce un error de conexión aparezca un aviso de error.
- Comprobación de que los contenidos creados dinámicamente son correctos.
- Comprobación de que si desde la ficha de datos se vuelve al listado y se elige otro elemento, los datos se actualizan con los del nuevo elemento.
- Comprobación de que si desde el listado se vuelve a la página de búsqueda (del tipo que sea) y se actualizan los parámetros, los elementos del listado se actualizan.

4.1.3 PRUEBAS CON DISPOSITIVOS MÓVILES REALES

Una vez pasadas satisfactoriamente cada uno de los test tanto en el navegador como en el simulador, se procedió a la ejecución de la aplicación en dispositivos móviles reales. Como es lógico, estos dispositivos necesitaban disponer de conexión a Internet, ya fuera una conexión móvil con un operador de telefonía o mediante una conexión Wi-Fi.

Para la mayor parte de las pruebas se han utilizado los terminales que aparecen en la tabla Figura 4.1, en la que se detalla también la versión del sistema operativo Android instalada en cada dispositivo y el tamaño en pulgadas de la pantalla.

Dispositivo	Versión del sistema operativo Android	Tamaño de la pantalla en pulgadas
Móvil <i>Huawei U8650</i>	2.3	3.5
Móvil <i>Samsung Galaxy Note</i>	2.3.6	5.3
Móvil <i>Huawei Selina (U8110)</i>	2.1	2.8
Móvil <i>Samsung Galaxy Ace</i>	2.2	3.5
Móvil <i>Samsung Galaxy Nexus</i>	4.2.1	4.65
Tablet PC <i>BQ Pascal 2</i>	4.0	7
Tablet PC <i>Asus Transformer TF101</i>	4.0.3	10.1

Figura 4.1 Tabla de dispositivos móviles empleados en las pruebas, versión de Android y tamaño de la pantalla de cada uno de ellos.

Además de las pruebas anteriormente descritas en los apartados 4.1.1 y 4.1.3 y que se necesitó validarlas en los dispositivos móviles reales, se realizaron las siguientes comprobaciones:

- Ejecución de la aplicación en dispositivos con distintas versiones del sistema operativo Android.
- Ejecución de la aplicación en dispositivos con distintas resoluciones de pantalla.
- Ejecución de la aplicación en varios dispositivos de distintos fabricantes.

En cada uno de estos escenarios, se han tenido en cuenta algunos test adicionales a los ya descritos anteriormente, exclusivos para los dispositivos reales, que se enumeran a continuación:

- Inicio satisfactorio de la aplicación.
- Comprobación de que el terminal responde adecuadamente a cada uno de los eventos del teclado.
- Comprobación de que se ejecutan correctamente la petición y recepción de datos del servidor.
- Comprobación de que la aplicación se ajusta proporcionalmente al tamaño de pantalla.

4.1.4 ANÁLISIS DE LOS RESULTADOS.

Las pruebas realizadas sobre los dispositivos móviles reales constataron que no todo lo que funciona correctamente en el emulador funciona también en los dispositivos físicos, como puede ser el tiempo de espera entre que se produce la pulsación de un botón y se ejecuta la función asociada.

A esto hay que añadir que el retardo introducido en la petición y recepción de datos entre el simulador y el servidor es prácticamente despreciable, mientras que en los dispositivos reales este lapso de tiempo es variable en función de la capacidad de la red a la que se encuentre conectado. No obstante, este tiempo es lo suficientemente pequeño como para que no afecte a la navegación normal de la aplicación.

Por otra parte, cabe destacar que al tratarse de una aplicación que se puede ejecutar en varias plataformas, como *smartphones* o *tablets*, es importante que el tamaño de

todos los elementos esté definido con porcentajes relativos a la resolución del dispositivo, de modo que la vista de la aplicación sea lo más agradable posible en todos ellos.

4.2 PRUEBAS DE LA APLICACIÓN EN EL SERVIDOR

En este apartado se van a explicar las pruebas realizadas a la aplicación Symfony del servidor, tanto al *frontend* que sirve los datos a la aplicación móvil, como al *backend*, la parte de administración de la base de datos. Al igual que en las pruebas de la aplicación móvil, las pruebas de la aplicación del servidor han pasado por varias fases. En la primera de ellas se ha comprobado el correcto funcionamiento del *frontend*. Esta fase se ha realizado previamente a las pruebas de la aplicación móvil explicadas en los apartados 4.1.2 y 4.1.3, debido a que el *frontend* de la aplicación del servidor es el encargado de recibir las peticiones y generar los documentos XML con los datos. La última fase se trata de las pruebas realizadas sobre el *backend*.

4.2.1 PRUEBAS DEL FRONTEND

Como se ha comentado anteriormente, esta serie de test se han validado previamente a las pruebas con simuladores y pruebas con dispositivos reales de la aplicación móvil, ya que era necesario asegurarse del correcto funcionamiento de esta parte de la aplicación Symfony antes de poder realizar dichas pruebas.

Para esta serie de pruebas se ha utilizado un navegador. Por un lado, se ha comprobado el funcionamiento en el equipo en el que se ha desarrollado la aplicación en local, y más tarde, en un servidor de prueba. A continuación se listan las pruebas realizadas:

- Comprobación de que los tiempos de respuesta son adecuados.
- Comprobación de que la URL es correcta.
- Comprobación de que el formato de salida de la aplicación es XML.
- Comprobación de que las tablas se han definido y creado correctamente.
- Comprobación de que los datos introducidos en la base de datos son correctos.
- Comprobación de que no existen datos repetidos en la base de datos.

- Comprobación de que los datos de acceso a la base de datos están definidos correctamente en la configuración.
- Comprobación de que los *templates* asociados a cada función son los correctos.
- Comprobación de que la generación de los documentos XML se hace correctamente.
- Comprobación de que las búsquedas son correctas.
- Comprobación de los *logs* de error en el modo desarrollo.
- Comprobación de que una vez pasados las pruebas anteriores se pasa la aplicación al modo de producción y se vuelven a validar las pruebas anteriores.

4.2.2 PRUEBAS DEL BACKEND

Esta serie de pruebas es independiente de las pruebas realizadas a la aplicación móvil y de las pruebas realizadas al *frontend* del servidor, excepto de las pruebas relativas a la base de datos, debido a que como ya se ha dicho en varias ocasiones, el *backend* es el encargado de la administración.

Al igual que ocurre con el *frontend*, las pruebas del *backend* se han realizado en primera instancia en local en el equipo de desarrollo y posteriormente en el servidor de prueba. Las pruebas realizadas sobre esta parte de la aplicación Symfony son las siguientes:

- Comprobación de que la URL es correcta.
- Comprobación de que el formato de salida de la aplicación es HTML.
- Comprobación de que los botones de la aplicación funcionan correctamente.
- Comprobación de que los tamaños de los botones son proporcionales a la resolución del navegador.
- Comprobación de que todas las características de fuente (tamaño, tipo de fuente, color, subrayado, negrita, cursiva, etcétera) son las adecuadas dependiendo de la función que desempeña cada texto.
- Comprobación de que los márgenes, expresados en porcentaje del tamaño de la pantalla, son los adecuados para cada elemento.

- Comprobación de que la barra de desplazamiento horizontal y vertical funciona correctamente cuando la vista de una página de la aplicación es mayor que la pantalla.
- Comprobación de que la vista es correcta para distintos tamaños de pantalla.
- Comprobación de que la búsqueda y listado de resultados es correcta en cada una de las pantallas de administración.
- Comprobación del funcionamiento correcto de los botones editar y eliminar de las tablas de elementos.

4.2.3 ANÁLISIS DE RESULTADOS

Las pruebas realizadas sobre la máquina local y posteriormente sobre el servidor de prueba han constatado que en modo local los tiempos de respuesta son inferiores a los que se tienen en el servidor Web. Esto es debido principalmente al acceder al servidor Web hay que realizar una conexión vía Internet, por lo que la velocidad de la línea y la potencia de procesado del servidor influyen de manera importante.

A la hora de probar las aplicaciones del servidor hay que tener en cuenta tener bien configurado el *router*, teniendo asignados los puertos correspondientes al servidor de prueba, ya que de otra manera se produce un error. Otro aspecto a tener en cuenta es la URL a la que se accede, ya que debemos añadir a la dirección IP (*Internet Protocol*) el puerto, quedando de esta forma: XXX.XXX.XXX.XX:PPPP, siendo las "X" los valores de la dirección IP y las "P" los valores del puerto.

Cabe destacar la ayuda que proporciona el poder usar el modo de desarrollo en las aplicaciones Symfony, ya que añade *logs* de error que en el modo de producción no se muestran, por lo que es mucho más fácil detectar y corregir los posibles errores en la programación a lo largo de todo el desarrollo.

CAPÍTULO 5: Conclusiones y líneas futuras

En el presente capítulo se exponen las conclusiones de este proyecto fin de carrera y se especifican algunas propuestas de posibles líneas futuras que añadan funcionalidad al sistema o lo mejoren en algún área.

5.1 CONCLUSIONES FINALES

Como se ha explicado en el capítulo introductorio de esta memoria (Capítulo 1), el uso de *smartphones* influye directamente en el uso de los ordenadores personales, debido a que con los primeros se tienen muchas de las funcionalidades de los segundos. Con un *smartphone* se puede consultar el correo, acceder a las redes sociales, navegar por Internet, ver vídeos, en definitiva, se han convertido en una herramienta necesaria.

Como consecuencia se han extendido las aplicaciones que dan servicio a dichos *smartphones*, lo que las convierte en un área interesante para los desarrolladores de *software*. Esto hace que se adapten los sistemas para gestionarlos mediante estos dispositivos. Otro aspecto importante es la programación de dichas aplicaciones, pudiéndose realizar de forma nativa en lenguaje Java, Objective C, C, C#, .NET, etcétera, o usando programación Web en CSS, HTML y Javascript. Depende del tipo de aplicación se elige una opción u otra. Si la aplicación va a ser exclusiva para un sistema operativo es lógico usar la programación nativa, mientras que si se pretende que la aplicación funcione en varios sistemas operativos es interesante usar la programación Web.

El objetivo de este proyecto fin de carrera ha sido desarrollar un sistema de consulta y gestión de los libros del departamento de Tecnología Electrónica de la Universidad de Málaga. Para ello, se ha seguido una arquitectura cliente-servidor. Por un lado se ha desarrollado un servicio Web encargado de almacenar, gestionar y ofrecer los datos, y por otro lado, se ha desarrollado una aplicación móvil para dispositivos con el sistema operativo Android que permita la búsqueda de los libros y muestre los datos almacenados para cada uno de ellos. La aplicación móvil es muy intuitiva y fácil de usar, de manera que cualquier usuario puede utilizarla sin necesitar ningún

conocimiento previo. La aplicación de administración de los datos del servidor tiene un interfaz que resulta también fácil de usar y que para su uso tan sólo se necesita un navegador Web.

5.2 LÍNEAS FUTURAS

El sistema desarrollado en este proyecto está abierto a posibles cambios y mejoras en el futuro, con el objetivo de conseguir una herramienta más completa. A continuación, se sugieren algunas posibles alternativas:

- Añadir un sistema de gestión de cuentas de usuario a la parte de administración, de manera que cualquier persona autorizada pueda modificar los datos de la base de datos con tan sólo identificarse con un usuario y contraseña.
- Añadir una parte de administración a la aplicación móvil mediante cuenta de usuario, para gestionar la base de datos desde los dispositivos móviles, sin necesitar el uso de un ordenador de escritorio.
- Extender la aplicación móvil a otros sistemas operativos, como iOS, Windows Phone, Windows 8, BlackBerry o Symbian, de manera que todos los usuarios de *smartphones* tengan acceso a la aplicación.
- Extender la aplicación a otros departamentos de la Universidad de Málaga, personalizando el diseño para cada uno de ellos.
- Crear un sistema de gestión de usuarios de cada uno de los libros, de manera que cualquier persona que tenga en su poder uno de los libros almacenados en la base de datos esté totalmente identificado por el administrador.
- Incorporación de un modo fuera de línea de la aplicación, sin acceso a Internet, de modo que habría que almacenar los datos en el dispositivo y añadir un sistema de actualización de dichos datos en caso de modificación en el servidor. La aplicación se conectaría en su primera ejecución al servidor para la descarga de los datos y de ahí en adelante, al iniciar la aplicación se compararía un código local en el dispositivo con otro en el servidor que indicaría la necesidad de una actualización o no.
- Añadir un menú de configuración, de forma que se permitiera por ejemplo cambiar el idioma de la aplicación, cambiar la dirección IP del servidor que proporciona los datos, aplicar un zoom, etcétera.

- Incorporación dentro de la aplicación móvil la posibilidad de enviar un correo electrónico al administrador para la modificación de la ubicación de un determinado libro, lo que evitaría tener que presentarse ante el administrador de forma física para tal fin.

ANEXO A. Manual de usuario

En este anexo se explica el funcionamiento de la aplicación desde el punto de vista del usuario final, tanto como usuario de la aplicación móvil, como de un usuario administrador. En el primer caso estará enfocado al uso normal de la aplicación por parte del usuario, y se explicará la navegación y el uso de las opciones que ofrece la plataforma. En cuanto al segundo caso, se comentarán los pasos a seguir para la gestión de la base de datos.

A.1 MANUAL DE USUARIO DE LA APLICACIÓN MÓVIL

En este apartado se explicarán los pasos a seguir por un usuario para instalar la aplicación en su móvil y se hará una somera descripción de los pasos a seguir para su uso correcto.

El usuario dispone de dos formas distintas de instalar la aplicación en su móvil:

- Descargando la aplicación, compuesta por un archivo *.apk*, desde Internet con un ordenador, tras lo cual tendrá que transferir la aplicación al teléfono móvil. Para ello se pueden usar distintos medios, como puede ser la conexión directa por cable USB eligiendo en el dispositivo el modo "Almacenamiento masivo", o el uso del entorno de desarrollo Eclipse, que permite realizar este proceso mediante su opción "*Deploy*".
- Realizando la descarga de la aplicación directamente desde el móvil. Para ello, dicha aplicación debe haber sido subida previamente a *Google Store*, la tienda de aplicaciones Android. Con este método, se debe tener conectado el dispositivo a Internet, ya sea por Wi-Fi, WAP, GPRS, etc; y realizar una búsqueda con el nombre de la aplicación en el *Google Store*. Una vez elegida la aplicación se instalará directamente al seleccionarla.

Una vez instalada la aplicación en el móvil, hay que ejecutarla, teniendo en cuenta de que es necesaria la conexión a Internet para su funcionamiento. Para ello, será necesario ir al menú de aplicaciones del teléfono, desde el que tendrá que aparecer la opción para

lanzar esta aplicación, que se mostrará bajo el nombre "BiblioDTE" y tendrá el icono que aparece en la Figura A.1.



Figura A.1 Icono de la aplicación móvil

Al pulsar sobre dicho icono, se inicia la aplicación, y se muestra la pantalla de inicio que se puede ver en la figura 3.2. En el apartado 3.1.1 del Capítulo 3 ya se han mostrado las distintas pantallas de las que consta la aplicación y se ha explicado la navegación entre ellas. Sólo cabe añadir que el botón para volver a la página anterior, según el esquema de la Figura 3.1, es el botón físico del dispositivo móvil. En la Figura A.2 se muestran los botones de navegación de un terminal Android. Al igual que en cualquier otra aplicación Android, al pulsar el botón "*HOME*" la aplicación queda en ejecución en segundo plano. El botón "*MENU*" no tiene funcionalidad en esta aplicación.



Figura A.2 Botones de navegación de un dispositivo móvil Android.

Si se da el caso de que el usuario no ha establecido la conexión antes de iniciar la aplicación, una vez iniciada ésta muestra una ventana emergente en la que nos avisa de que no existe conexión a Internet (Figura A.3). Para solucionarlo, basta con pulsar el botón "*HOME*", establecer la conexión a Internet y volver a iniciar la aplicación.



Figura A.3 Ventana emergente de alerta sin conexión.

A.2 MANUAL DE USUARIO ADMINISTRADOR

El objetivo de este apartado es proporcionar una guía básica a un usuario administrador sobre la gestión de la base de datos, incluyendo tanto los datos de los libros como los datos de alumnos y profesores. Esta guía servirá de manual de referencia en el que se definen los pasos y acciones a llevar a cabo para la edición e inserción de datos nuevos.

Para poder acceder a la aplicación administración es necesario poner en la URL del navegador la siguiente dirección IP: *194.140.146.17:8080/backend.php/administrar*. En la Figura 3.38 se ha mostrado la pantalla general de administración.

Como se comentado en el apartado 3.2.3, pulsando sobre los botones correspondientes de la pantalla general de administración, se redirige a las páginas de administración de libros, proyectos fin de carrera, trabajos fin de máster, tesis, profesores o alumnos. En cada una de estas páginas, aparece un formulario y una tabla similares a los que se muestran en las Figuras 3.39 y 3.40.

En el formulario se muestran entradas de texto con todos los campos almacenados para cada elemento en la base de datos. Además, se puede marcar un *checkbox* indicando que dicho campo está vacío. Una vez rellenados los campo deseados, al pulsar el botón "*Filter*", se realiza una búsqueda y se listan en la tabla los resultados

obtenidos. Por defecto, al iniciar cada pantalla de búsqueda, se muestran todos los elementos que se encuentran almacenados en la base de datos.

En la tabla, además de los datos de todos los campos de los elementos, consta de dos botones en cada una de la filas, "*Edit*" y "*Delete*". Si se pulsa "*Delete*" se borra la fila y si se pulsa "*Edit*" se redirige a otra pantalla de edición del elemento. Un ejemplo de dicha pantalla se puede observar en la Figura A.4, en la que se está editando el elemento con id=1 de la tabla "*libros*".

The screenshot shows a web form titled "Edit Admin" for editing a book record. The form contains the following fields and values:

Numero pedido	1
Autores	BOREMAN, G.D.
Codigo biblioteca	
Titulo	FUNDAMENTOS DE EL
Editorial	SPIE
Fecha publicacion	1999
Edicion	
Isbn	0-8194-3323-3
Tejuelo	TE/ 1645
Fecha pedido	18/12/01
Fecha recepcion	08/05/02
Ubicacion	ANTONIO JESÚS BAND
Comentarios	
Numero referencia	3350
Numero ejemplares	
Solicitud	ANTONIO JESÚS BAND

At the bottom of the form, there are three buttons: "Delete" (with a red X icon), "Back to list" (with a list icon), and "Save" (with a floppy disk icon).

Figura A.4 Ejemplo de pantalla de edición de un elemento de la tabla "*libros*"

La pantalla de edición consta de un formulario con todos los campos rellenos con los datos del elemento que se está editando, todos ellos modificables, y también de los botones "*Delete*", que permite borrar el elemento, "*Back to list*", que permite volver a la página de administración de la tabla y "*Save*", que permite guardar los cambios realizados en la información del elemento.

BIBLIOGRAFÍA Y REFERENCIAS

BIBLIOGRAFÍA

Jonathan Stark, Brian Jepson, "*Building Android Apps with HTML, CSS, and JavaScript*", O'Reilly, Enero 2012.

Steven Holzner, "*PHP 5: El lenguaje para los profesionales de la Web (Manual avanzado)*", Anaya Multimedia, 2005.

Página de referencia del tutorial del proyecto *Jobeet* que explica cómo desarrollar una aplicación Web con Symfony desde cero.

http://symfony.com/legacy/doc/jobee/1_4/es/01?orm=Doctrine

Página oficial de referencia para los desarrolladores Android:

<http://developer.android.com/indexHTML>

Página oficial de referencia para la descarga y uso de PhoneGap:

<http://phonegap.com>

REFERENCIAS

- [1] Teleco, Inteligencia en Telecomunicaciones:
http://www.teleco.com.br/es/pais/es_celular.asp

- [2] Estadísticas Mundiales de *Smartphones* y Celulares 2012:
<http://www.poderpda.com/investigacion-y-desarrollo/estadisticas-mundiales-de-smartphones-y-celulares-2012/>

- [3] Android sería el Sistema Operativo Líder en *Smartphones* en el 2016:
<http://www.tecnogizmo.com/android-sistema-operativo-lider-en-smartphones-en-2016/>

- [4] El uso de Internet móvil se duplica año tras año desde 2009:
http://www.tendencias21.net/El-uso-de-Internet-movil-se-duplica-ano-tras-ano-desde-2009_a10621HTML

- [5] El uso de los móviles en España. Tendencias y comportamientos de los usuarios:
<http://retelur.files.wordpress.com/2007/10/onepagerthinkmobile1-110927092026-phpapp01.pdf>

- [6] Desarrollo de aplicaciones móviles multiplataforma:
<http://albertovilches.com/desarrollo-de-aplicaciones-moviles-multiplataforma-phonegap-y-titanium-appcelerator>

- [7] PhoneGap vs. Código Nativo:
<http://rauldevilla-movilidad.blogspot.com.es/2012/04/phonegap-vs-codigo-nativoHTML>

- [8] Blog de Android en español:
<http://www.android.es/android-incHTML#axzz2IhY22GCm>

- [9] Google Buys Android for Its Mobile Arsenal:
<http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal>
- [10] Página oficial de la Open Handset Alliance:
<http://www.openhandsetalliance.com/>
- [11] History of Android:
<http://www.tech2crack.com/history-android/>
- [12] The History of Android:
<http://mashable.com/2011/07/26/android-history-infographic/>
- [13] Página oficial de soporte del móvil HTC T-Mobile-G1:
<http://www.htc.com/us/support/htc-g1-tmobile/>
- [14] Así se distribuyen las distintas versiones de Android entre los terminales:
<http://gizmologia.com/2012/09/android-distribucion-versiones>
- [15] Rohit Ghatol, Yogesh Patel, "*Beginning PhoneGap. Mobile Web Framework for JavaScript and HTML5*", Apress, 2012.
- [16] Página oficial de *The WebKit Open Source Project*:
<http://www.Webkit.org/>
- [17] Artículo sobre el entorno de desarrollo Eclipse:
http://www.ecured.cu/index.php/Eclipse_entorno_de_desarrollo_integrado
- [18] Página oficial del IDE Eclipse:
<http://www.eclipse.org>
- [19] Curso online de HTML:
http://www.aulaclie.es/html/t_1_1.htm

- [20] Página de referencia de tecnologías Web:
http://www.w3schools.com/html/html_intro.asp
- [21] Guía Breve de CSS:
<http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo>
- [22] Tutorial online "JavaScript Ya":
<http://www.javascriptya.com.ar/temarios/descripcion.php?cod=2>
- [23] Introducción a JavaScript:
<http://www.librosWeb.es/javascript>
- [24] Qué es un *framework*:
<http://jordisan.net/blog/2006/que-es-un-framework>
- [25] Tabla comparativa de los distintos *frameworks*:
<http://www.bestWebframeworks.com/compare-Web-frameworks/php/>
- [26] *Frameworks* para el desarrollo de aplicaciones:
<http://www.monografias.com/trabajos70/frameworks-desarrollo-aplicaciones-php/frameworks-desarrollo-aplicaciones-php2.shtml>
- [27] Página oficial del *framework* Zend:
<http://framework.zend.com/>
- [28] Página oficial del *framework* Symfony:
<http://symfony.com/>
- [29] Página oficial del *framework* CakePHP:
<http://cakephp.org/>
- [30] *Frameworks* PHP recomendados, guía para principiantes:
<http://www.elWebmaster.com/articulos/frameworks-php-recomendados-guia-para-principiantes>

- [31] Página oficial del *framework* CodeIgniter:
<http://ellislab.com/codeigniter>
- [32] Página oficial del *framework* Seagull:
<http://seagullproject.org/>
- [33] Manual oficial de referencia de PHP en español:
<http://es1.php.net/manual/es/>
- [34] PHP página de referencia de tecnologías Web:
http://www.w3schools.com/php/php_intro.asp
- [35] Página oficial de la base de datos de código abierto MySQL:
<http://www.mysql.com/>
- [36] Libro online, Yhony Aguilar Quenta, "Programación Web / AppServ-PHPMySQL":
<http://es.scribd.com/doc/94998619/39/QUE-ES-MYSQL>
- [37] ¿Qué son las bases de datos?:
<http://www.maestrosdelWeb.com/principiantes/%C2%BFque-son-las-bases-de-datos/>
- [38] ¿Qué hace un Servidor Web como Apache? Configuración:
<http://www.digitallearning.es/blog/apache-servidor-Web-conFiguracion-apache2-conf/>
- [39] Artículo sobre PHPMyAdmin:
<http://www.desarrolloWeb.com/articulos/844.php>
- [40] Diccionario de informática de alegsa.com.ar:
<http://www.alegsa.com.ar/Dic/phpmyadmin.php>

[41] Página oficial en España del estándar W3C:

<http://www.w3c.es/>

[42] Manual de XML de la página oficial del W3C:

<http://www.w3.org/XML/>