

2012-09

Framework de desarrollo de aplicaciones web multiplataforma

Espinosa Alfonso, Javier

<http://hdl.handle.net/10016/17139>

Descargado de e-Archivo, repositorio institucional de la Universidad Carlos III de Madrid



Universidad
Carlos III de Madrid

TRABAJO FIN DE GRADO

Grado de Ingeniería de Informática

FRAMEWORK DE DESARROLLO DE APLICACIONES WEB MULTIPLATAFORMA

Autor: Javier Espinosa Alfonso

Tutor: Francisco Javier García Blas

Leganés, a 5 de septiembre de 2012

Título:

Framework de desarrollo de aplicaciones web multiplataforma

Autor:

Javier Espinosa Alfonso

Tutor:

Francisco Javier García Blas

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo de Fin de Grado el día ____ de _____ de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

Índice General

1. Introducción.....	7
1.1 Motivación.....	7
1.2 Objetivos.....	8
1.3 Estructura de la memoria.....	8
1.4 Glosario de términos.....	9
2. Estado del arte.....	10
2.1 HTTP.....	10
2.2 HTML5.....	11
2.3 PHP.....	12
2.4 MySQL.....	13
2.5 REST Web Services.....	14
2.6 jQuery & jQuery Mobile.....	16
2.7 JSON.....	18
2.8 PhoneGap.....	21
2.9 Smarty 3.....	23
2.10 Justificación de tecnologías.....	24
3. Análisis.....	26
3.1 Descripción general.....	27
3.2 Definición de requisitos.....	28
3.2.1 Requisitos funcionales.....	29
3.2.2 Requisitos no funcionales.....	32
3.2.3 Requisitos de restricción.....	33
4. Diseño e implementación.....	35
4.1 Arquitectura del sistema.....	35
4.2 Diagrama de flujo.....	39
5. Evaluación y resultados.....	41
5.1 Entorno de evaluación.....	41
5.1.1 Tests de PHPUnit.....	41
5.1.2 Uso de la aplicación.....	42
5.2 Resultados.....	43
6. Ejemplo de uso: Aplicación móvil.....	44
7. Planificación y presupuesto.....	60

7.1 Planificación.....	60
7.1.1 Diagrama de Gantt.....	61
7.2 Presupuesto.....	62
7.2.1 Costes de personal.....	62
7.2.2 Costes de hardware.....	63
7.2.3 Costes de software.....	64
7.2.4 Coste total.....	65
8. Conclusiones y líneas futuras.....	66
8.1 Conclusiones.....	66
8.2 Líneas futuras.....	67
9. Bibliografía.....	68

Índice de tablas

1. Tabla: Plataformas móviles que soporta PhoneGap	22
2. Tabla: Requisito funcional RFU-0001.....	29
3. Tabla: Requisito funcional RFU-0002.....	29
4. Tabla: Requisito funcional RFU-0003.....	29
5. Tabla: Requisito funcional RFU-0004.....	30
6. Tabla: Requisito funcional RFU-0005.....	30
7. Tabla: Requisito funcional RFU-0006.....	30
8. Tabla: Requisito funcional RFU-0007.....	31
9. Tabla: Requisito funcional RFU-0006.....	31
10. Tabla: Requisito no funcional RNF-0001.....	32
11. Tabla: Requisito no funcional RNF-0002.....	32
12. Tabla: Requisito no funcional RNF-0003.....	32
13. Tabla: Requisito no funcional RNF-0004.....	33
14. Tabla: Requisito de restricción RRE-0001.....	33
15. Tabla: Requisito de restricción RRE-0002.....	33
16. Tabla: Requisito de restricción RRE-0003.....	34
17. Tabla: Planificación del trabajo por actividades.....	60
18. Tabla: Costes de personal.....	62
19. Tabla: Costes de hardware.....	63
20. Tabla: Costes de software.....	64
21. Tabla: Coste total.....	65

Índice de figuras

1. Ilustración: Logo HTML5.....	11
2. Ilustración: Logo PHP.....	12
3. Ilustración: Logo MySQL.....	13
4. Ilustración: Logo de jQuery.....	16
5. Ilustración: Logo de jQuery Mobile.....	17
6. Ilustración: Construcción de un JSON object.....	18
7. Ilustración: Construcción de un JSON array.....	18
8. Ilustración: Formación de un string en JSON.....	19
9. Ilustración: Opciones válidas para la formación de un value en JSON.....	20
10. Ilustración: Formato de un number en JSON.....	20
11. Ilustración: Logo de PhoneGap.....	21
12. Ilustración: Desarrollo aplicación PhoneGap.....	21
13. Ilustración: Plataformas soportadas.....	22
14. Ilustración: Logo de Smarty.....	23
15. Ilustración: Estructura principal del sistema.....	27
16. Ilustración: Estructura detallada del sistema.....	36
17. Ilustración: Diagrama de flujo de una petición HTTP.....	39
18. Ilustración: Pantalla de inicio.....	45
19. Ilustración: Pantalla de menú principal.....	46
20. Ilustración: Pantalla de formulario "Pide tu presupuesto".....	47
21. Ilustración: Pantalla con teclado numérico.....	48
22. Ilustración: Botón de opciones.....	49
23. Ilustración: Botón de opciones (II).....	49
24. Ilustración: Botón de opciones (III).....	50
25. Ilustración: Pantalla con teclado de correo electrónico.....	51
26. Ilustración: Pantalla de opciones "Localízate".....	52
27. Ilustración: Pantalla de preferencias.....	53
28. Ilustración: Pantalla de preferencias (II).....	53
29. Ilustración: Pantalla de preferencias (IV).....	54
30. Ilustración: Pantalla de preferencias (III).....	54
31. Ilustración: Pantalla "Formulario no válido".....	55
32. Ilustración: Uso de "Usar mis preferencias".....	55
33. Ilustración: Búsqueda de categoría.....	56
34. Ilustración: Pantalla "Buscar por categoría".....	56
35. Ilustración: Pantalla de formulario de "Mudanzas".....	57
36. Ilustración: Formulario con campo de fecha.....	58
37. Ilustración: Formulario con campo de fecha (II).....	58
38. Ilustración: Pantalla de éxito.....	59

1. Introducción

El objetivo de este apartado es presentar el proyecto de fin de grado y describir de forma breve cuáles han sido los motivos por los que se ha realizado el proyecto, los objetivos planteados y una estructura de la memoria que se ha llevado a cabo para describir el proyecto. Adicionalmente se añade un glosario de términos.

1.1 Motivación

El principal motivo de la realización de este proyecto es buscar de manera ágil un framework de desarrollo de aplicaciones web compatible con el máximo número de dispositivos. Esto incluye dispositivos tradicionales como el ordenador personal disponible en la mayoría de hogares desde hace años y dispositivos móviles con acceso a Internet con gran auge en los últimos años.

El problema surge a la hora de desarrollar para dispositivos móviles. Cada uno de ellos necesita una aplicación con un entorno de desarrollo distinto, un lenguaje de programación distinto, etc. Esto obliga a derivar ese desarrollo a una empresa especializada o a hacer una gran inversión de aprendizaje de cada una de las tecnologías para abarcar el mayor número de dispositivos móviles.

Gracias al proyecto, no se necesita desarrollar en cada dispositivo móvil. Un único desarrollo es suficiente para todos los dispositivos. Y lo más ventajoso de todo: se siguen utilizando los mismos lenguajes de programación que en los dispositivos tradicionales.

Con lo cual la curva de aprendizaje es mínima para desarrollar en estos dispositivos.

Por tanto, la necesidad de ser compatible con el mayor número de dispositivos y conseguir que la curva de aprendizaje sea mínima para que esto sea posible, son los dos grandes motivos por los cuales se ha llevado a cabo el trabajo de fin de grado que describe el presente documento.

1.2 Objetivos

El objetivo principal que persigue el trabajo es el desarrollo del framework para aplicaciones web de forma ágil. Para llevar a cabo este objetivo principal, hay otros objetivos secundarios a conseguir:

- Desarrollo ágil de una aplicación web.
- Compatible en mayoría de dispositivos móviles con un único desarrollo.
- Uso de tecnologías dominantes en el mercado laboral.
- Curva de aprendizaje mínima.

1.3 Estructura de la memoria

El presente documento está dividido en varios capítulos, agrupados de forma lógica de tal modo que a través de su lectura ordenada se consiga una comprensión paulatina y global de la aplicación y su dominio:

En el capítulo 1, **Introducción**, se presenta la motivación del trabajo, los objetivos que se pretenden alcanzar, y la estructura del presente documento. En resumen, se pretende dar una visión inicial y global del trabajo de fin de grado.

En el capítulo 2, **Estado del arte**, se presentan y analizan a fondo las tecnologías que se utilizarán en la realización del trabajo, así como una justificación de su uso frente a otras alternativas disponibles.

En el capítulo 3, **Análisis del sistema**, se estudia detenidamente la funcionalidad requerida por la aplicación, dando lugar a los casos de uso y a los requisitos del sistema, herramienta fundamental para tener presente hacia dónde se pretende llegar con el trabajo, y que servirán también a la finalización del mismo, para verificar que el trabajo se ha llevado a cabo de manera exitosa.

En el capítulo 4, **Diseño e implementación**, se define con precisión la aplicación a todos los niveles, desde su arquitectura general, hasta llegar al nivel de detalle de clase, exponiendo de manera precisa la disposición de todos los elementos del sistema y la comunicación entre los mismos. Se presentan también, tanto el diseño detallado de la base de datos utilizada, como algunos aspectos interesantes del proceso de implementación y consideraciones a tener en cuenta.

En el capítulo 5, **Evaluación y resultados**, se muestra detalladamente la batería de pruebas realizada al sistema en su totalidad y que permitirá validar los requisitos identificados en la fase de análisis, y concluir el trabajo con éxito.

En el capítulo 6, **Ejemplo de uso**, se enseñará la aplicación móvil de ejemplo, mostrando gráficamente todas las opciones disponibles que se han aplicado, y explicando cómo hay que solicitar un presupuesto de manera satisfactoria en el sistema.

En el capítulo 7, **Planificación y presupuesto**, se expondrá la planificación detallada que se seguirá en el desarrollo del trabajo, así como el presupuesto, que se explicará por partes, justificando los costes de personal, hardware y software, e informando del coste total del trabajo, incluyendo beneficios y margen de riesgo.

En el capítulo 8, **Conclusiones** y líneas futuras, se recogerán las conclusiones posteriores a la realización del trabajo, así como una lista de posibles aspectos ampliables en versiones posteriores, o líneas de investigación en las que se puede trabajar para la mejora constante de la aplicación.

Finalmente, en el capítulo 9, **Bibliografía**, se mostrarán todas las fuentes de información consultadas durante el proceso de desarrollo del trabajo.

1.4 Glosario de términos

En este apartado se recopilan los acrónimos más importantes que se hayan utilizado a lo largo del documento.

Término	Descripción
HTTP	Protocolo de transferencia de hipertexto.
TCP	Protocolo de Control de Transmisión.
UDP	Protocolo de Datagramas de Usuario.
W3C	El Consorcio World Wide Web es una comunidad internacional donde las organizaciones Miembro, personal a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web.
XML	Es un lenguaje de marcas desarrollado por el W3C.
XPath	Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML.
DOM	Modelo de Objetos del Documento. Su responsable es W3C.
CSS	Hoja de Estilos en Cascada. Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.
SSL	Capa de conexión segura (Secure Sockets Layer)

2. Estado del arte

2.1 HTTP

El HTTP (Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto) es el protocolo utilizado en las transacciones de internet. Su versión 1.0 surgió en mayo de 1996, y entre sus características podemos encontrar:

- Sigue una arquitectura cliente-servidor, lo más frecuente es que los clientes sean los navegadores de Internet y los servidores sean servidores web.
- Funciona sobre el protocolo de nivel de transporte TCP en el puerto 80.
- No sólo soporta el envío de hipertexto, también trabaja con otros recursos web como imágenes.
- Es un sistema orientado a transacciones, y cada petición/respuesta va acompañada de unas cabeceras que incluyen metadatos sobre la misma.

Este protocolo va a ser importante para la aplicación web a la hora de comunicarse, y para ello será necesario implementar el framework para que permita hacer peticiones HTTP. Se trata de un requisito imprescindible, ya que el cliente para la aplicación web obtiene toda la información a través de Internet, y todas las acciones que se pueden realizar desde la interfaz requieren llamadas al servidor. Existen dos tipos de peticiones necesarias:

GET: petición de la representación de un recurso concreto. El servidor que va a recibir las peticiones del dispositivo cuenta con un API por URL, y en las peticiones de

este tipo se encuentran pares clave-valor separados por & que contienen los parámetros. La mayor parte de las peticiones de la aplicación se realizarán con este sistema.

POST: al igual que con GET, con este tipo de petición se pueden pedir recursos, pero tiene otras funciones. Las peticiones POST se suelen emplear para enviar los resultados de un formulario, ya que no tiene la limitación del tamaño máximo por argumento que tiene GET (256 caracteres) y además así no se muestran en la URL los datos enviados, ya que van incluidos en el cuerpo del mensaje.

2.2 HTML5

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML (XHTML) (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Todavía se encuentra en modo experimental, lo cual indica la misma W3C; aunque ya es usado por múltiples desarrolladores web por sus avances, mejoras y ventajas.



1. Ilustración: Logo HTML5

Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se le recomienda al usuario común actualizar a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML5.

El desarrollo de este lenguaje de marcado es regulado por el Consorcio W3C.

Nuevos elementos

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas <div> y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada, como los elementos <audio> y <video>. Mejoras en el elemento <canvas>, capaz de renderizar en los navegadores más importantes (Mozilla, Chrome, Opera, Safari e IE) elementos 3D. Algunos elementos de HTML 4.01 han quedado obsoletos, incluyendo elementos puramente de presentación, como y <center>, cuyos efectos son manejados por el CSS. También hay un renovado énfasis en la importancia del scripting DOM para el comportamiento de la web.

Las novedades frente a HTML 4.01 son las siguientes:

- Incorpora etiquetas (canvas 2D y 3D, audio, video) con codecs para mostrar los contenidos multimedia. Actualmente hay una lucha entre imponer codecs libres (WebM + VP8) o privados (H.264/MPEG-4 AVC).

- Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime ...) y facilidades para validar el contenido sin Javascript.
- Visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales). En general se deja abierto a poder interpretar otros lenguajes XML.
- Drag & Drop. Nueva funcionalidad para arrastrar objetos como imágenes.

2.3 PHP

PHP (PHP Hypertext Pre-processor) es un lenguaje de programación interpretado, diseñado en 1994 para la creación de páginas web dinámicas. Siendo multiplataforma y teniendo un gran parecido con los lenguajes más comunes de programación estructurada permiten que el aprendizaje sea muy corto. Todo esto junto con la facilidad de instalación hizo que PHP cogiera fama muy rápidamente.



2. Ilustración: Logo PHP

Cuando el cliente realiza una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP procesando el script solicitado y generando el contenido de manera dinámica.

Las características más importantes que hacen que PHP destaque entre todas ellas son:

- Lenguaje multiplataforma.
- Orientado al desarrollo de web dinámicas con acceso a información de base de datos.
- Soporte con diferentes tipos de servidores de bases de datos, como por ejemplo MySQL.
- Integración con librerías externas hace que la funcionalidad aumente bastante.
- Código invisible al usuario que lo solicita, programación segura y confiable.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos, sino que decide en periodo de ejecución.
- Manejo de excepciones.
- Uso libre.

Por todo esto se ha elegido su uso como lenguaje para la programación de las distintas funciones y procedimientos. Se han explotado al máximo las posibilidades de este lenguaje creando una programación orientada a objetos para la gestión de la base de datos.

De cara al proyecto PHP será el encargado de ejecutar toda la lógica del programa, siendo el controlador que medie entre las vistas y el modelo de bases de datos.

2.4 MySQL

MySQL es un sistema de gestión relacional, multihilo y multiusuario con más de seis millones de instalaciones. El objetivo de la empresa en el momento de su creación en 1995 fue seguir con el estándar *SQL*, sin sacrificar la velocidad, fiabilidad o usabilidad. A lo largo de los años ha ido convirtiéndose en el gestor más popular del mundo por su rendimiento, eficiencia y facilidad de uso.



Sus características más importantes son:

- Implementación multihilo.
- Soporte en el tipo de datos.
- Portabilidad entre sistemas.
- Buen nivel de seguridad.

Su punto fuerte es una gran aceptación, debido a la cual existen infinidad de librerías para una gran cantidad de lenguajes de programación. En cuanto a la licencia, MySQL es de uso libre y código abierto pero Oracle tiene el copyright sobre ese código.

Hay varios tipos de gestión dentro del propio MySQL, entre los que destacan dos, MyISAM y InnoDB. En el caso de MyISAM, tecnología usada por defecto, se da una rápida lectura al utilizar el motor no transaccional ya que no tiene que hacer comprobaciones de integridad referencial ni bloquear las tablas por ausencia de atomicidad, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación.

En segundo caso, InnoDB, es la tecnología opuesta, soporta las transacciones tipo ACID y bloqueo de registros junto con la integridad referencial, ofreciendo una fiabilidad y consistencia a cambio de un rendimiento menor.

2.5 REST Web Services

La Transferencia de Estado Representacional (Representational State Transfer) o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST de Fielding y también es posible diseñar interfaces XMLHTTP de acuerdo con el estilo de llamada a procedimiento remoto pero sin usar SOAP. Estos dos usos diferentes del término REST causan cierta confusión en las discusiones técnicas, aunque RPC no es un ejemplo de REST.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful; los defensores más acérrimos de REST se llaman a sí mismos RESTafaris. REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST).
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
- Una sintaxis universal para identificar los recursos: En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El uso de hipermedias, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Recursos

Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso o URI, de sus siglas en inglés). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

La petición puede ser tramitada por cualquier número de conectores (por ejemplo clientes, servidores, cachés, túneles, etc.) pero cada uno lo hace sin "ver más allá" de su propia petición (lo que se conoce como separación en capas, otra restricción de REST, que es un principio común con muchas otras partes de la arquitectura de redes y de la información).

Así, una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida, no necesitando conocer si existen cachés, proxys, cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. La aplicación, sin embargo, debe comprender el formato de la información devuelta (la representación), que es por lo general un documento HTML o XML, aunque también puede ser una imagen o cualquier otro contenido.

2.6 jQuery & jQuery Mobile

jQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC.



4. Ilustración: Logo de jQuery

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Las empresas Microsoft y Nokia anunciaron que incluirán la biblioteca en sus plataformas. Microsoft la añadirá en su IDE Visual Studio y la usará junto con los frameworks ASP.NET AJAX y ASP.NET MVC, mientras que Nokia los integrará con su plataforma Web Run-Time.

Sus características más importantes son:

- Selección de elementos DOM. Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath.
- Eventos, manipulación de la hoja de estilos CSS.
- Efectos y animaciones. Animaciones personalizadas.
- AJAX. Soporta extensiones.
- Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

jQuery Mobile

jQuery Mobile es una librería de Javascript optimizada para web en móviles que actualmente está siendo desarrollada por el equipo del proyecto jQuery. El desarrollo se centra en crear un framework compatible con el mayor número de smartphones o tablets debido a la gran heterogeneidad que hay en el mercado de estos dispositivos móviles. Es compatible con otros frameworks de aplicaciones para móvil como PhoneGap, Worklight y otros.



5. Ilustración: Logo de jQuery Mobile

Sus características más importantes son:

- Compatible con la mayoría de las plataformas móviles como la mayoría de los navegadores: iOS, Android, Blackberry, WebOS, Symbian, Windows Phone 7, y más.
- Construido sobre el núcleo jQuery, para que la curva de aprendizaje sea mínima para desarrolladores familiarizados con la sintaxis de jQuery.
- Un framework de temas que permite personalizar uno rápidamente.
- Dependencias limitadas y ligero que optimizan su velocidad..
- El mismo código de base escalará el tamaño automáticamente a cualquier pantalla.
- Posibilidad de navegación entre páginas por AJAX con animaciones en las transiciones de página
- UI widgets optimizados para táctil y para quien desconozca la plataforma.



2.7 JSON

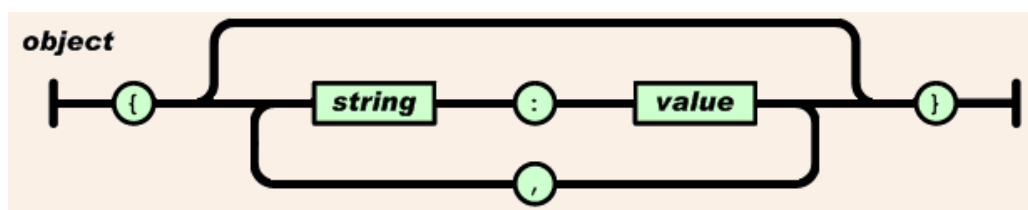
JSON se corresponde con las siglas *JavaScript Object Notation* y es un formato ligero para el intercambio de datos entre distintas partes. Es un lenguaje muy sencillo de leer y escribir por las personas, y fácil de ser analizado sintácticamente y ser generado por máquinas. Aunque es un subconjunto de la notación de JavaScript, JSON es un formato de texto completamente independiente de JavaScript o cualquier otro lenguaje de programación.

Los mensajes JSON se construyen siguiendo dos estructuras:

- Una colección de pares nombre/valor. En varios lenguajes, esto equivale a un *object*, *record*, *struct*, *dictionary*, *hash table*, *keyed list* o *associative array*.
- Una lista de valores ordenados. En muchos lenguajes, equivale a un *array*, *vector*, *list* o *sequence*.

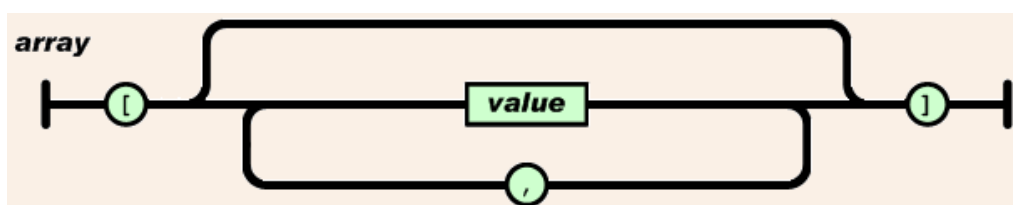
En JSON, los mensajes toman la forma de *object* y *array* respectivamente, y se describen del siguiente modo:

Un *object* es un conjunto no ordenado de pares nombre/valor. Un *object* comienza con { (llave izquierda) y termina con } (llave derecha). Cada nombre es seguido por : (dos puntos) y el par nombre/valor separados por , (coma).



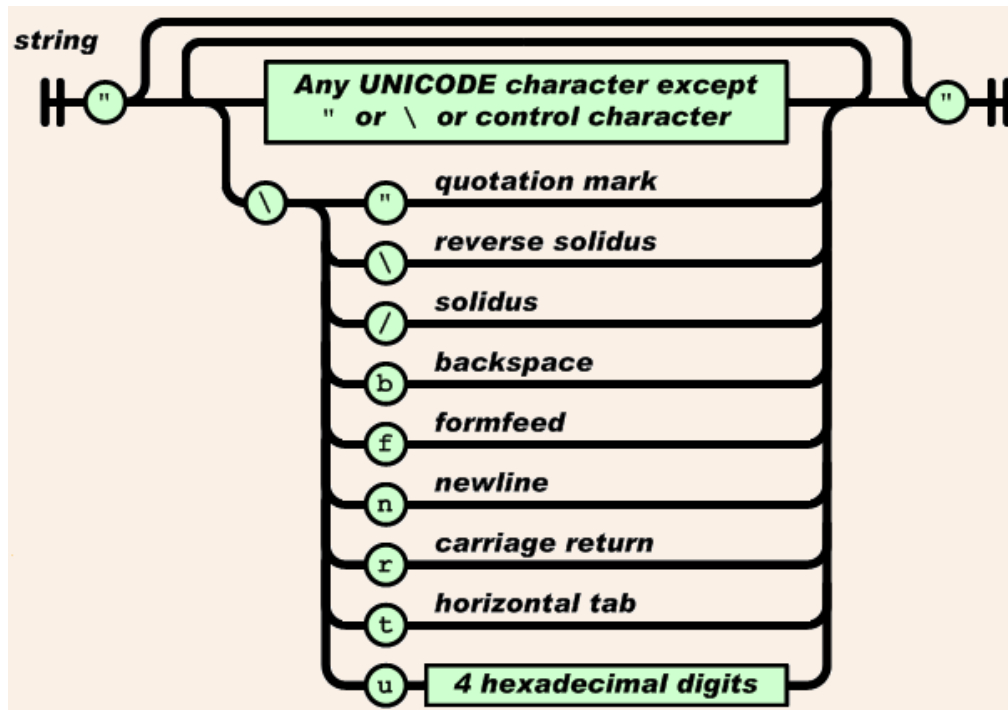
6. Ilustración: Construcción de un JSON object

Un *array* es una colección ordenada de valores. Un *array* empieza con el carácter [(corchete izquierdo) y termina con] (corchete derecho). Los valores están separados entre sí por , (coma).



7. Ilustración: Construcción de un JSON array

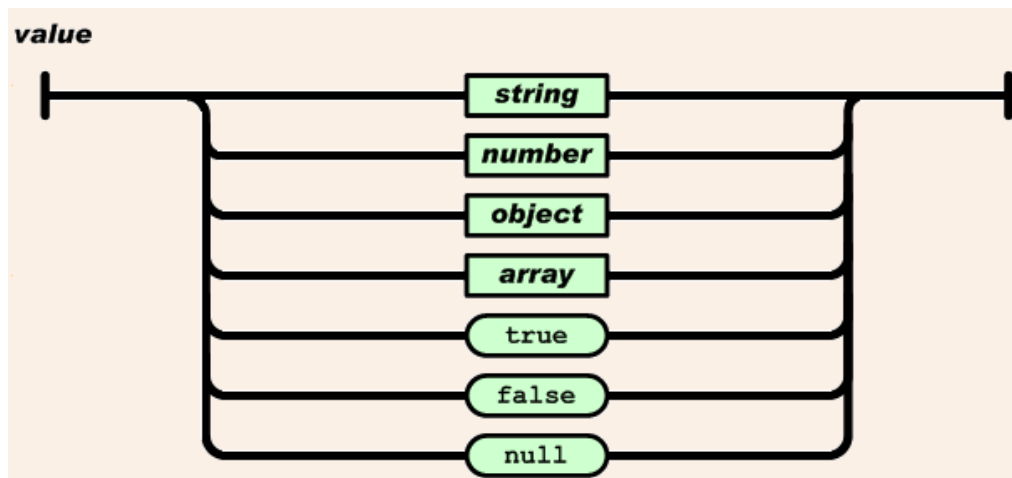
Los *objects* y *arrays* necesitan de *strings* y *values* para contener información útil. Se forman del siguiente modo:



8. Ilustración: Formación de un string en JSON

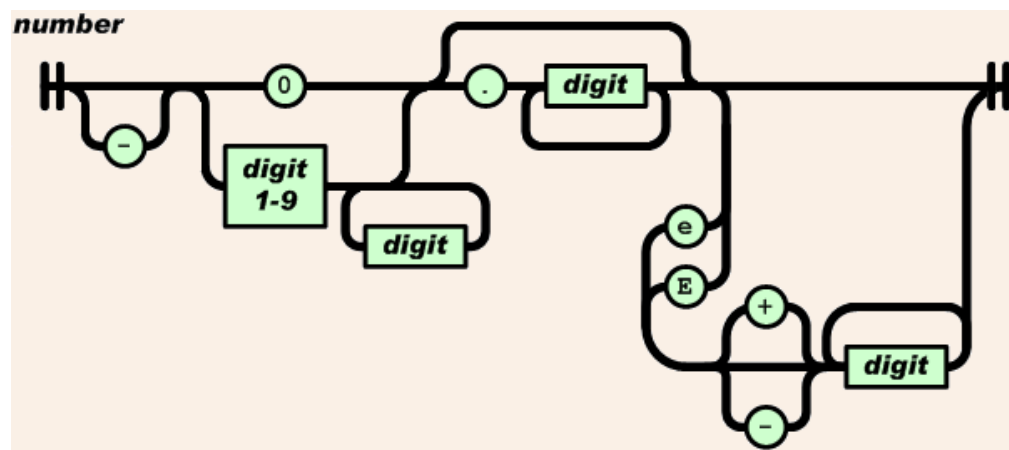
Un *string* es una colección de cero o más caracteres Unicode, envueltos entre " (comillas dobles). Un *string* de JSON es muy similar a uno de C o Java, pudiéndose utilizar escapes de barra invertida (\).

Un *value* puede ser un *string* entre " (comillas dobles), un *number*, *true* o *false* o *null*, un *object* o un *array*. Estas estructuras pueden anidarse.



9. Ilustración: Opciones válidas para la formación de un *value* en JSON

Un *number* es muy similar a un número de Java o C, exceptuando que el formato octal y hexadecimal no se utilizan.



10. Ilustración: Formato de un *number* en JSON

2.8 PhoneGap

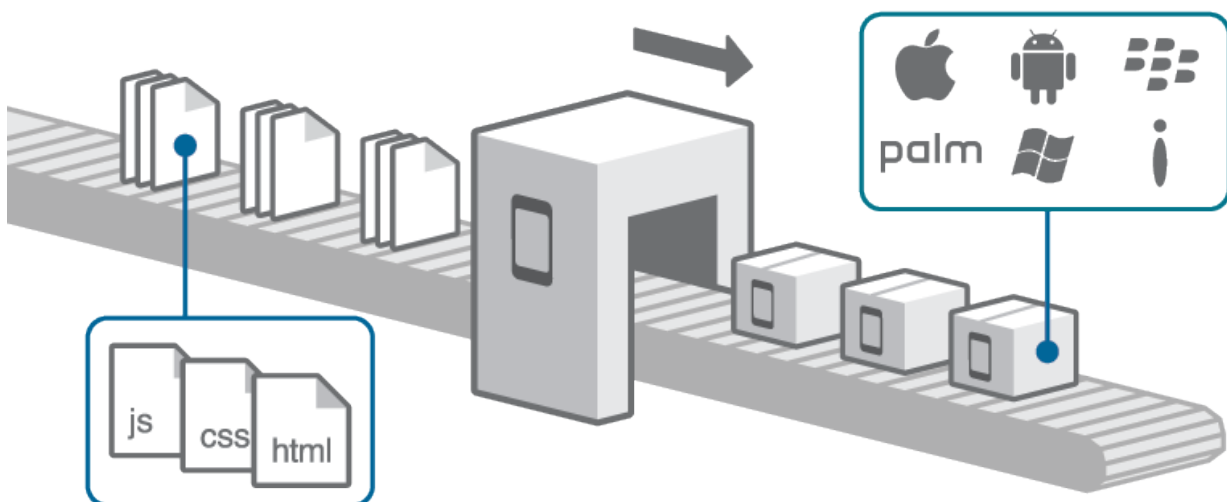
PhoneGap (anteriormente llamado Apache Callback, pero actualmente Apache Cordova) es un framework de desarrollo de código abierto para móvil producido por Nitobi, y actualmente comprado por Adobe Systems.

Permite a los programadores de software desarrollar aplicaciones para dispositivos móviles usando JavaScript, HTML5 y CSS3, en vez de lenguajes de bajo nivel como Objective-C.

Las aplicaciones resultantes son híbridas, lo que significa que no son ni verdaderamente nativas (todo el renderizado de diseño se realiza a través del navegador web de la plataforma nativa de la interfaz de usuario) ni puramente basadas en la web (que no son aplicaciones sólo web pero empaquetados para la distribución de la AppStore, Android o cualquier otra y tener acceso a una parte de las funciones del dispositivo).



11. Ilustración: Logo de PhoneGap



12. Ilustración: Desarrollo aplicación PhoneGap

Plataformas soportadas

13. Ilustración: Plataformas soportadas

PhoneGap actualmente soporta para desarrollar en sistemas operativos como Apple iOS, Google Android, HP webOS, Microsoft Windows Phone, Nokia Symbian OS RIM BlackBerry. Soporta versiones recientes como BlackBerry 5 y 6 y Windows Phone 7, para la cual está siendo implementado actualmente. Bada (el sistema operativo usado por Samsung Wave S85000) tendrá soporte muy pronto. La siguiente tabla muestra las características que soporta en cada uno de los sistemas operativos (los más importantes):

Características	iPhone 3GS+	Android 1.0+	Windows Phone 7	BlackBerry (5.x y 6.0+)
Acelerómetro	Sí	Sí	Sí	Sí
Cámara	Sí	Sí	Sí	Sí
Brújula	Sí	Sí	Sí	Sí
Contactos	Sí	Sí	Sí	Sí
Archivo	Sí	Sí	Sí	Sí
Geolocalización	Sí	Sí	Sí	Sí
Media	Sí	Sí	Sí	N/A
Red 3G / WiFi	Sí	Sí	Sí	Sí
Notificación de alerta	Sí	Sí	Sí	Sí
Notificación de sonido	Sí	Sí	Sí	Sí
Notificación de vibración	Sí	Sí	Sí	Sí
Almacenamiento	Sí	Sí	Sí	Sí
Escáner Barcode	Sí	Sí	N/A	Sí

1. Tabla: Plataformas móviles que soporta PhoneGap

2.9 Smarty 3

Smarty es un motor de plantillas para PHP, es decir, separa el código PHP, como lógica de negocios, del código HTML, como lógica de presentación, y genera contenidos web mediante la colocación de etiquetas Smarty en un documento. Se encuentra bajo la Licencia Pública General Reducida de GNU.



14. Ilustración: Logo de Smarty

Es común que en grandes proyectos el rol de diseñador gráfico y el de programador sean cubiertos por personas distintas, sin embargo la programación en PHP tiene la tendencia de combinar estas dos labores en una persona y dentro del mismo código, lo que trae consigo grandes dificultades a la hora de cambiar alguna parte del diseño de la página, pues se tiene que escarbar entre los scripts para modificar la presentación del contenido, Smarty tiene como objetivo solucionar este problema.

Sus características principales son:

- Expresiones regulares
- Bucles foreach, while
- Sentencias condicionales if, elseif, else
- Modificadores de variables (por ejemplo: `{ $\$$ variable|nl2br}`)
- Funciones creadas por el usuario
- Evaluación de expresiones matemáticas en la plantilla

Existen más sistemas de plantillas para PHP, pero éste es de los más avanzados y con más frecuencia de desarrollo. También hay detractores de estas técnicas que alegan que las mismas hacen en cierta medida un grado más complejo el desarrollo web, por la necesidad de aprender un (pseudo) lenguaje más.

Los detractores de esta idea se basan en el hecho de que, precisamente, el lenguaje PHP nació como un lenguaje rápido para hacer desarrollos web a pequeña escala. A medida van surgiendo sistemas de separación en capas que intentan disciplinar un poco las metodologías de programación envueltas en el desarrollo con PHP, pero que no hacen otra cosa que acercarse más y más a otras herramientas ya existentes en otros entornos de desarrollo más complejos y pensados desde sus orígenes para proyectos más grandes, como pueden ser J2EE (Java), .NET (C#) o Django (Python).

2.10 Justificación de tecnologías

Debido a que el principal objetivo del trabajo es realizar un framework para crear aplicaciones web multiplataforma de forma ágil, se ha optado por elegir unas tecnologías libres, de código abierto y gratuitas; y con una posición importante entre sus alternativas, intentando evitar nuevas tecnologías poco probadas y poco estables, y así, conseguir un equipo de desarrolladores mucho más experimentados y de forma más fácil para llevar a cabo cualquier aplicación web, minimizando así la curva de aprendizaje.

El protocolo por excelencia para realizar aplicaciones web desde los inicios de Internet es **HTTP**. Como novedad, se ha optado por usar la quinta revisión de lenguaje básico para la web, **HTML5**. Tras la experiencia de HTML4 se ha aprendido mucho y se ha mejorado la experiencia de usuario (sobre todo en dispositivos móviles), aunque aún está en fase experimental y no todos los navegadores web implementan todas sus funcionalidades definidas en el estándar. Aún así, no deja de ser una sintaxis muy potente, sobre todo a la hora de realizar formularios web, la cual es una funcionalidad muy usada en la aplicación de ejemplo de este trabajo de fin de grado.

La utilización de **MySQL** como gestor de bases de datos viene determinada principalmente por ser un gestor de código abierto, libre y gratuito muy potente y con un fuerte soporte detrás ya que es el software más usado en el mundo en bases de datos, y además, ha sido adquirido recientemente por la empresa multinacional Oracle. Esto garantiza que a la hora de desarrollar para la base de datos sea fácil encontrar documentación y personal adecuado con gran experiencia en el gestor.

Para el acceso a los recursos de la aplicación, se ha optado por **REST**, ya que el enfoque de diseño es diferente por ejemplo a RPC (llamadas a procedimientos remotos). En REST se focaliza en la diversidad de recursos, a diferencia de RPC que se pone énfasis en la diversidad de operaciones del protocolo.

Para el desarrollo de aplicaciones web en dispositivos móviles hemos elegido un framework compatible con la mayoría de ellos, de tal forma que sólo sea necesario un único desarrollo para todos los dispositivos (Android, iPhone, BlackBerry...). Estas características las cumple a la perfección **jQuery Mobile**, el cual está basado en **jQuery** y cómo no, es de código abierto y gratuito. Debido a esta familiarización con jQuery nos ha resultado mejor opción que una alternativa como Sencha Touch, de esta forma se reducía una vez más la curva de aprendizaje.

Al usarse dispositivos móviles, la representación de información transmitida es un punto importante para reducir el volumen de datos de la mejor forma posible, y por ello, se optó por **JSON**, en vez de, por ejemplo, representación de datos en XML. Además, es muy fácil de aprender y muy relacionado con Javascript.

Para que la aplicación web pueda ser nativa del teléfono, y además, se pueda disponer de funciones propias del dispositivo móvil, se ha elegido **PhoneGap**. Una alternativa sería MoSync.

Como motor de plantillas para PHP se eligió **Smarty 3**, el cual es de los más usados y de los más veteranos. Quizá no es el motor más rápido de todos, pero en esta ocasión se ha valorado más su estabilidad y su veteranía frente a velocidad de proceso. Además, su sintaxis es muy familiar a HTML o PHP, lo cual la curva de aprendizaje es mínima y tiene muchísimas más funciones ya definidas, a diferencia de otros motores de plantillas como Savant.

3. Análisis

A la hora de realizar un trabajo de fin de grado hay que tener muy presente cual es el fin de este, así como haber profundizado en el dominio sobre el que versa. Estos dos factores son los que generan los requisitos del sistema, que son el comienzo de todo proyecto.

Para comenzar se presentará una descripción general de la aplicación web diseñada e implementada en este Trabajo de Fin de Grado, en la cual, ya se establecen los requisitos básicos de la misma, para seguir con una más detallada. Estas descripciones son el resultado del estudio realizado sobre el dominio e incluye las primeras decisiones, restricciones y limitaciones tomadas, todas ellas con el fin de definir distintos requisitos, lo más detallados posible.

A continuación se describirán de manera formal los requisitos de la aplicación, sin olvidar que estos juegan un papel muy importante en el desarrollo del resto del trabajo, puesto que son el punto de referencia para alcanzar de la forma más satisfactoria posible el resultado deseado.

Finalmente mencionar que en el desarrollo del presente trabajo se ha gozado de una gran libertad, por lo que ciertos requerimientos han surgido durante fases más avanzadas del desarrollo del trabajo, así como otros se han visto modificados o han sido adaptados.

3.1 Descripción general

El trabajo desarrollado tiene por objetivo fundamental crear aplicaciones web multiplataforma de manera ágil sobre tecnologías dominantes para así reducir el coste de aprendizaje de manera notable. De esta forma resultará mucho más sencillo poder llevarlo a cabo en un periodo de tiempo reducido.

Como la aplicación web se va a ejecutar en dispositivos móviles y en ordenadores personales, en distintas localizaciones y de forma frecuente, es necesario que la aplicación sea rápida y eficiente con los datos. En el caso de los dispositivos se hará un esfuerzo mayor de eficiencia ya que son dispositivos con poca capacidad de almacenamiento y de proceso.

Es necesario disponer de un servidor web el cual sirva todo el contenido a los distintos dispositivos, ya sean ordenadores o dispositivos móviles. Este servidor dispondrá acceso a la base de datos y contará con los mecanismos necesarios para para recibir y enviar información desde y hacia los dispositivos que ejecuten la aplicación.

En el siguiente esquema se puede ver la estructura principal del sistema:



15. Ilustración: Estructura principal del sistema

3.2 Definición de requisitos

Este apartado tiene como objetivo especificar todas las funcionalidades y restricciones asociados a la aplicación a desarrollar.

Los requisitos serán plasmados de modo que se complete por cada uno de ellos una plantilla, compuesta por los siguientes campos:

- **Identificador (ID):** Cadena identificadora única, asociada a un único requisito al cual hace referencia unívocamente. El ID seguirá la siguiente nomenclatura, R<tipo>-<número>, donde:
 - **Tipo:** Hace referencia a la clasificación del requisito, pudiendo ser este:
 - **FU:** Requisito funcional.
 - **NF:** Requisito no funcional.
 - **RE:** Requisito de restricción.
 - **Número:** Será un conjunto de cifras que comenzarán desde el 0001 hasta el que sea necesario para identificar a todos los requisitos.
- **Descripción:** Breve comentario textual que define al requisito.
- **Necesidad:** Campo que indica si el requisito es imprescindible o no lo es. Toma uno de los siguientes valores:
 - Esencial.
 - No esencial.
- **Prioridad:** Indica el nivel de preferencia que se le debe dar al requisito durante el desarrollo del sistema. Puede tomar los siguientes valores:
 - Baja.
 - Media.
 - Alta.
- **Estabilidad:** se refiere a la probabilidad de que el requisito se vea alterado en un futuro por el cliente. Toma los valores:
 - Sin cambios.
 - Con cambios.
- **Fuente:** indica el origen del requisito. En el presente documento, todos los requisitos tienen como fuente al "Tutor", "Alumno" o "Empresa".

3.2.1 Requisitos funcionales

ID	RFU-0001
Descripción	El framework debe implementar geolocalización tanto en dispositivos móviles como en navegadores web con coordenadas de latitud y de longitud.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Empresa

2. Tabla: Requisito funcional RFU-0001

ID	RFU-0002
Descripción	El framework debe geolocalizar con calle, ciudad, provincia y país al usuario en todos los dispositivos en los que disponga sus coordenadas de latitud y longitud.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Empresa

3. Tabla: Requisito funcional RFU-0002

ID	RFU-0003
Descripción	El framework debe implementar notificaciones en dispositivo móvil y alertas en navegador web.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Alumno

4. Tabla: Requisito funcional RFU-0003

ID	RFU-0004
Descripción	El framework debe implementar diálogos de confirmación en dispositivo móvil y en navegador web.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Alumno

5. Tabla: Requisito funcional RFU-0004

ID	RFU-0005
Descripción	El framework debe permitir crear formularios adaptados a dispositivos móviles de forma ágil.
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Empresa

6. Tabla: Requisito funcional RFU-0005

ID	RFU-0006
Descripción	El framework debe implementar validación de datos en los formularios (texto, teléfono, código postal, etc.) a través de la funcionalidad del estándar HTML5 y JavaScript. También se validarán estos datos en el servidor.
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Empresa

7. Tabla: Requisito funcional RFU-0006

ID	RFU-0007
Descripción	El framework debe permitir almacenamiento interno en los dispositivos, para guardar información del tipo clave-valor. Ejemplo: correo electrónico, número de teléfono, ubicación...
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Empresa

8. Tabla: Requisito funcional RFU-0007

ID	RFU-0008
Descripción	La aplicación móvil podrá mostrar formularios de todas las secciones
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Empresa

9. Tabla: Requisito funcional RFU-0006

3.2.2 Requisitos no funcionales

ID	RNF-0001
Descripción	La aplicación ha de ajustarse a la resolución por defecto del dispositivo.
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Alumno

10. Tabla: Requisito no funcional RNF-0001

ID	RNF-0002
Descripción	El framework ha de estar desarrollado en tecnologías libres de código abierto y con gran impacto en sus competencias con el fin de minimizar la curva de aprendizaje.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Empresa

11. Tabla: Requisito no funcional RNF-0002

ID	RNF-0003
Descripción	El framework ha de tener una interfaz de usuario poco cargada y fácil de manejar, siendo un aspecto fundamental su alta usabilidad.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Alumno

12. Tabla: Requisito no funcional RNF-0003

ID	RNF-0004
Descripción	La aplicación móvil minimizará la inserción de información ya existente en el sistema por parte del usuario. Cargando las preferencias introducidas, rellenando campos con geolocalización, etc.
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Empresa

13. Tabla: Requisito no funcional RNF-0004

3.2.3 Requisitos de restricción

ID	RRE-0001
Descripción	La aplicación web deberá ejecutarse en dispositivos con una conexión funcional a Internet, para poder realizar las comunicaciones con el servidor.
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Alumno

14. Tabla: Requisito de restricción RRE-0001

ID	RRE-0002
Descripción	La aplicación móvil será ejecutable en el sistema operativo Android (+2.2) y en el sistema operativo iOS (+4). Opcionalmente en otros sistemas operativos como BlackBerry o Windows Phone 7.
Necesidad	Esencial
Prioridad	Alta
Estabilidad	Sin cambios
Fuente	Empresa

15. Tabla: Requisito de restricción RRE-0002

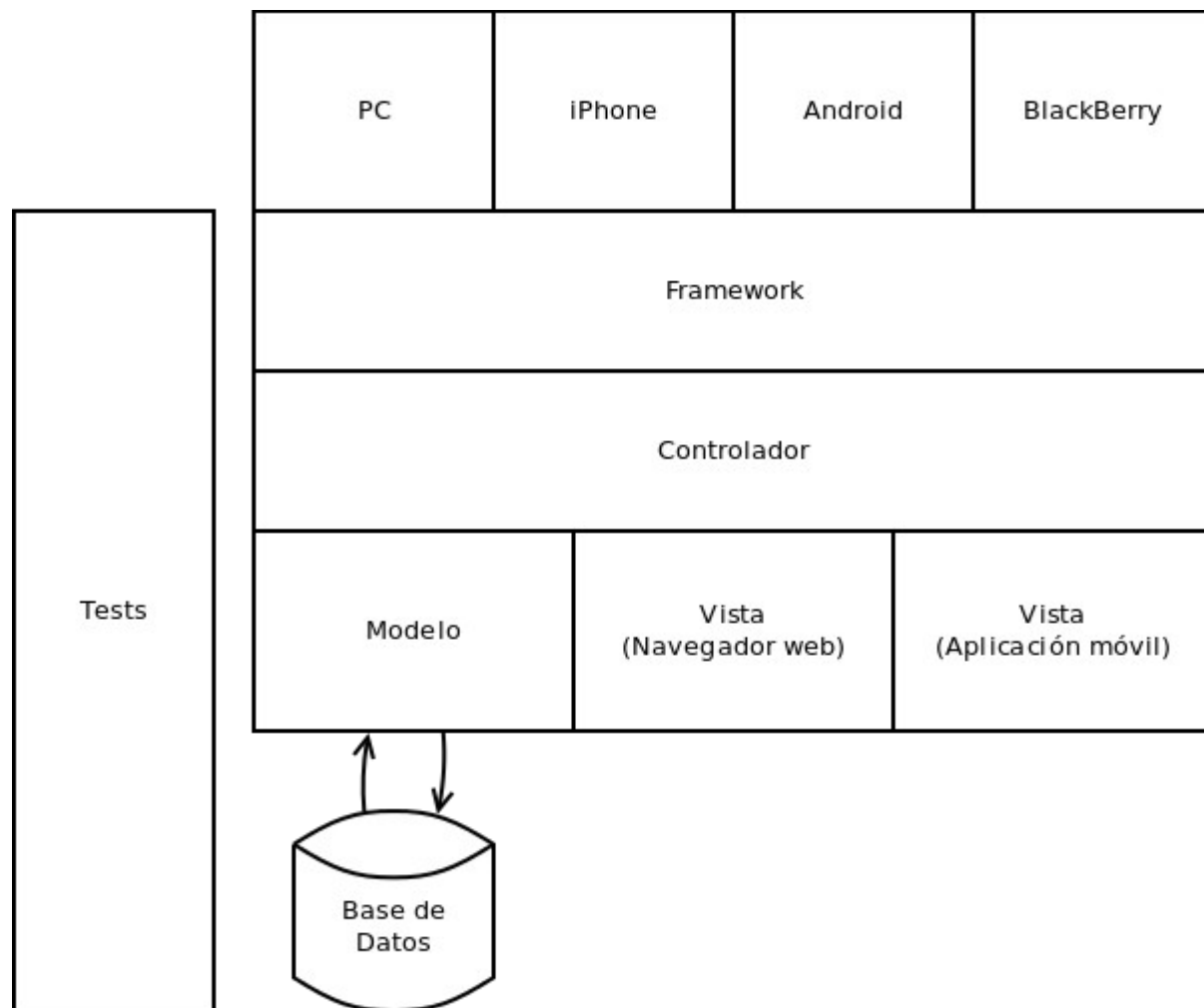
ID	RRE-0003
Descripción	La aplicación web será funcional en los siguientes navegadores: Internet Explorer 10+, Google Chrome 10+, Firefox 4+, Safari 5+ y Opera 11+.
Necesidad	Esencial
Prioridad	Media
Estabilidad	Sin cambios
Fuente	Empresa

16. Tabla: Requisito de restricción RRE-0003

4. Diseño e implementación

4.1 Arquitectura del sistema

Como la mayoría de las aplicaciones web de cliente-servidor, el sistema está basado en el patrón Modelo – Vista – Controlador siendo compatible con el mayor número de dispositivos posibles (multiplataforma) teniendo una estructura como la siguiente ilustración representa:



16. Ilustración: Estructura detallada del sistema

Como se aprecia en la estructura, se divide en las siguientes partes:

PC (navegador web), iPhone, Android, BlackBerry

Dispositivos controlados por los usuarios que acceden al sistema y realizan una petición. Dependiendo de qué plataforma realice la petición, tendrá una respuesta adaptada a su dispositivo de forma transparente, de tal forma que el usuario no tenga que informar al sistema cómo accede al sistema.

En el caso de los dispositivos móviles, a través del framework PhoneGap, se instala una aplicación la cual solicita un primer contenido al servidor dependiendo de dónde se localice el usuario.

Framework

El núcleo del framework que tramita la comunicación con esa petición al sistema e instancia los servicios necesarios para poder realizar una respuesta adecuada al dispositivo.

No solo se encarga de una respuesta adecuada sino que también realiza comprobaciones de seguridad, como por ejemplo, que el usuario tenga permisos para acceder a un recurso concreto o una acción concreta.

En caso de que la petición sea incorrecta o que el usuario no disponga de los permisos necesarios, es el framework el encargado de procesar una respuesta concreta según el error al dispositivo. Para este tipo de respuestas se respeta el estándar HTTP con los códigos de respuesta:

- 3XX:
 - Redirecciones. Ocurre cuando una página ha sido movida a otra dirección. Puede ser de forma temporal (código 302) o de forma permanente (código 301)
- 4XX:
 - Errores del cliente. Ocurre cuando una página no existe (código 404), un método no está permitido ya que se realiza un GET a un recurso que sólo permite POST (código 405); o que al recurso en concreto el usuario no tenga permisos para acceder (código 403 y 401).
- 5XX:
 - Errores del servidor. Ocurre cuando el servidor tiene un error interno (ej: un error fatal de PHP).

El framework está programado en lenguaje PHP.

Controlador

Responde a los eventos y a las acciones de los usuarios, e invoca las peticiones al modelo y a la vista. Para cada recurso o acción del sistema hay un controlador.

El controlador envía la información necesaria que necesita la "vista" para poder generar una interfaz al usuario. Es quien decide desde qué dispositivo se solicita la petición, y por tanto, cómo se generará el contenido adecuado al dispositivo.

Modelo

Representación de la información del sistema. Se compone de muchos patrones de diseño:

- Modelo:
 - Define las propiedades y acciones propias de un objeto en concreto en sí.
- Factory:
 - Instancia nuevos objetos de un modelo concreto. Además, se han añadido métodos que no son estrictamente de creación de objetos, como encontrar un objeto mediante su id único o por ejemplo una colección de objetos que cumplan unas propiedades concretas.
- Persistence:
 - Capa de persistencia que realiza las consultas a la base de datos. Ya sea para cargar los datos de un objeto concreto, para modificarlo, para crear uno nuevo o para eliminarlo.

- **Helper:**
 - Colección de métodos que tienen una relación directa con el modelo pero no es propio de una instancia concreta. Por ejemplo, para cambiar un estado de una colección de objetos.

Todos estos patrones de diseño están programados en lenguaje PHP.

Base de datos

Donde se almacena la información de cada objeto. Como hay muchas entidades y mucha información almacenada en la aplicación web es necesario disponer de varias bases de datos para tener una organización óptima de todos los datos. La base de datos está bajo MySQL.

Test

Ya que hay muchos modelos de objetos y muchas acciones sobre ellos, y además, entre ellos interactúan modificándose una y otra vez es importante tener un entorno de test para poder realizar pruebas.

Este entorno se compone de pruebas unitarias y funcionales, donde se evalúa cada método del modelo (incluyendo cada patrón de diseño como Helper, Factory, Persistence, etc), cada método del framework, cada controlador y cada consulta a la base de datos.

Las pruebas funcionales están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el modelo. Las pruebas funcionales se hacen mediante el diseño de prueba que buscan evaluar cada una de las opciones con las que cuenta el objeto.

Por otro lado, las pruebas unitarias son una forma de probar el correcto funcionamiento de un método de código en concreto. Esto sirve para asegurar que cada uno de los métodos funcionen correctamente por separado. Al estar todo en lenguaje PHP se han realizado estos test sobre el framework PHP Unit.

Vista

Muestra una respuesta en función del dispositivo que la solicita, ya sea dispositivo móvil o un ordenador personal. Como su nombre indica, es lo que se ve. Recibe los datos del controlador y "pinta" según éstos.

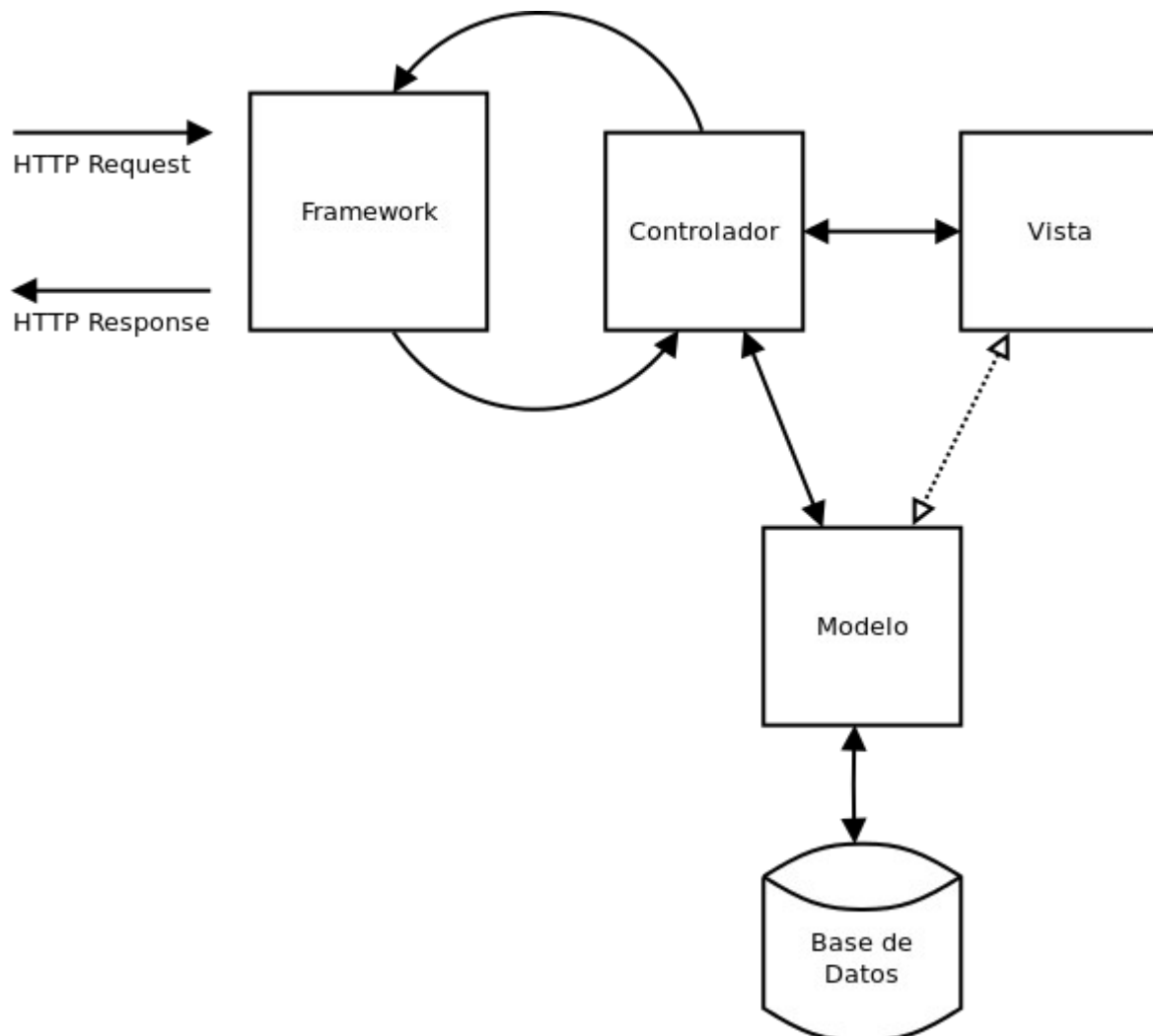
En el caso de un dispositivo móvil, pintará el contenido gracias al framework jQuery Mobile UI y en el caso de un ordenador personal simplemente será HTML. Aunque para ambos casos, se apoyará en el motor de plantillas Smarty 3.

En el caso de los formularios web se ha diseñado una plantilla de Smarty genérica para que se pueda obtener un formulario con los elementos necesarios (campos de texto, opciones, tipo radio, botones...) de forma rápida y sencilla para minimizar el coste

de desarrollo a la hora de programar.

4.2 Diagrama de flujo

Un diagrama de flujo de una petición HTTP al sistema se correspondería con este gráfico:



17. Ilustración: Diagrama de flujo de una petición HTTP

1. El usuario desde su dispositivo realiza una petición HTTP, la cual la procesa y la enruta el framework al controlador correspondiente, llamando al recurso y acción concreta.
2. El controlador procesa la acción solicitada instanciando los objetos necesarios a través del modelo, y una vez realizada la acción, envía la información a la vista para que este la muestre al usuario.
3. El modelo realiza una consulta a la base de datos para instanciar el objeto

concreto pedido, y si no es el caso, instancia uno nuevo.

4. El controlador devuelve al framework la vista ya generada y es éste quien se la devuelve al usuario final como respuesta HTTP.

5. Evaluación y resultados

Durante el siguiente apartado se describe cómo se ha llevado a cabo la evaluación del trabajo y los resultados de los mismos.

5.1 Entorno de evaluación

Los dos entornos en los que se ha trabajado de evaluación del trabajo son los siguientes: tests de PHP Unit y uso de la aplicación.

5.1.1 Tests de PHPUnit

El sistema tiene a su disposición una colección de tests hechos en **PHPUnit** los cuales evalúan una funcionalidad en concreto o método a método de cada modelo (u otro patrón) esperando un resultado en concreto.

En el caso de los test funcionales, como se ha explicado anteriormente, se trata de pruebas que están basadas en la ejecución de las funcionalidades previamente diseñadas en el modelo. Para realizar estas pruebas se parte de una base de datos con la estructura definida pero sin datos. A continuación se instancian los objetos que sean necesarios y con las propiedades adecuadas para justo después evaluar que la funcionalidad sea correcta.

Para ello se pueden evaluar desde consultas a la base de datos comprobando que

se han hecho las modificaciones correctamente, como igualdades de propiedades de objetos a un valor concreto. Las pruebas funcionales tienen la ventaja de que todo lo que evalúas es del sistema, no supones ningún valor ni se falsea ninguna otra funcionalidad. Son las pruebas más complicadas de programar ya que se evalúa un proceso complejo y se tiene que tener bien claro el punto de partida y el resultado que se ha de obtener.

En el caso de las pruebas unitarias se busca evaluar un método concreto de un objeto. Para ello, gracias a la herramienta PHPUnit y sus objetos "Mock" se esperan llamadas a objetos "mockeados". Un objeto "mockeado" es un objeto que interviene en el método (ya sea como argumento o no) y se esperan una serie de llamadas sobre él. "Mockeando" el objeto se consigue hacer expectativas sobre métodos que se ejecuten sobre sí mismo y devolver resultados forzados. De esta forma, evalúas únicamente el método que se está probando y no otros métodos que se supone están probados en otro test, no una funcionalidad completa. Al igual que en el las pruebas funcionales, se pueden comparar también los resultados de la prueba con los esperados.

En ambos casos se pueden realizar pruebas donde se fuerce un error. Como por ejemplo que un objeto no pueda cambiar una propiedad en un momento dado. Son casos que nunca deben ocurrir en producción pero que también se prueba que los errores hayan saltado correctamente y hayan sido "cazados" adecuadamente, y lo más importante, no se ha realizado los cambios en la propiedad que por definición no está permitido por su estado.

De esta forma de evaluación, cuando se cambia algún modelo de objeto la mayoría de tests que tienen relación con este objeto han de ser adaptados al nuevo concepto de modelo. Así, en todo momento, el sistema adquiere una seguridad ante cambios futuros.

5.1.2 Uso de la aplicación

Otra forma de evaluación del trabajo ha sido directamente el uso de la aplicación web. Como la aplicación es multiplataforma se han realizado dos pruebas de uso:

- Desde los navegadores web del ordenador como Internet Explorer, Google Chrome o Firefox
- Desde la aplicación móvil mediante un dispositivo Android.

Adoptando un rol de usuario y mediante herramientas de desarrollo web abiertas (consola Javascript en el navegador, inspector de elementos HTML del DOM de la página, analizador de tráfico HTTP, etc.) como la consola de Google Chrome o el plug-in Firebug de Firefox se ha podido evaluar que todo funcionara correctamente.

En el caso de la aplicación móvil se veía el comportamiento de la aplicación se hacía a través de la herramienta Logcat que proporciona Android SDK en Eclipse y se ha comprobado que el comportamiento del sistema sea el esperado.

5.2 Resultados

PHPUnit realiza las pruebas una a una y muestra en cuáles ha ido correctamente los tests, así hasta más de 3000 pruebas distintas.

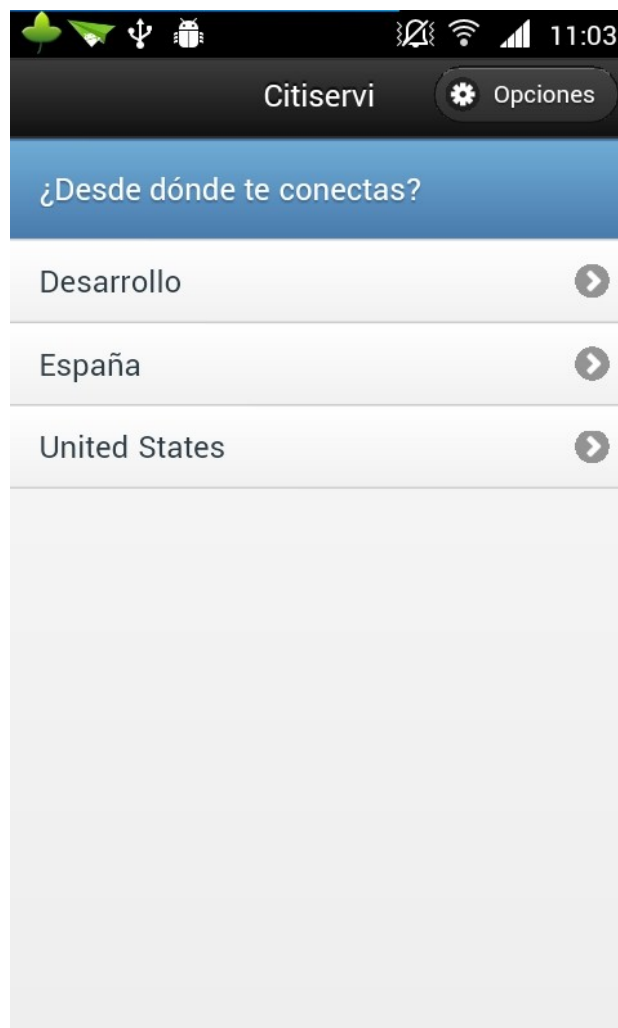
En el caso de las pruebas funcionales, mostraría algún error de resultados inesperados, en cambio, en las pruebas unitarias mostrarías errores de resultados inesperados e incluso de métodos que han ejecutado y no se han esperado, o justo al contrario, métodos que esperaban ser llamados y no se ha dado el caso. Hasta la fecha, todas las pruebas se realizan con éxito.

Los resultados tras el uso de la aplicación han sido satisfactorios y los resultados obtenidos y funcionamiento del sistema ha sido el esperado en todo momento.

6. Ejemplo de uso: Aplicación móvil

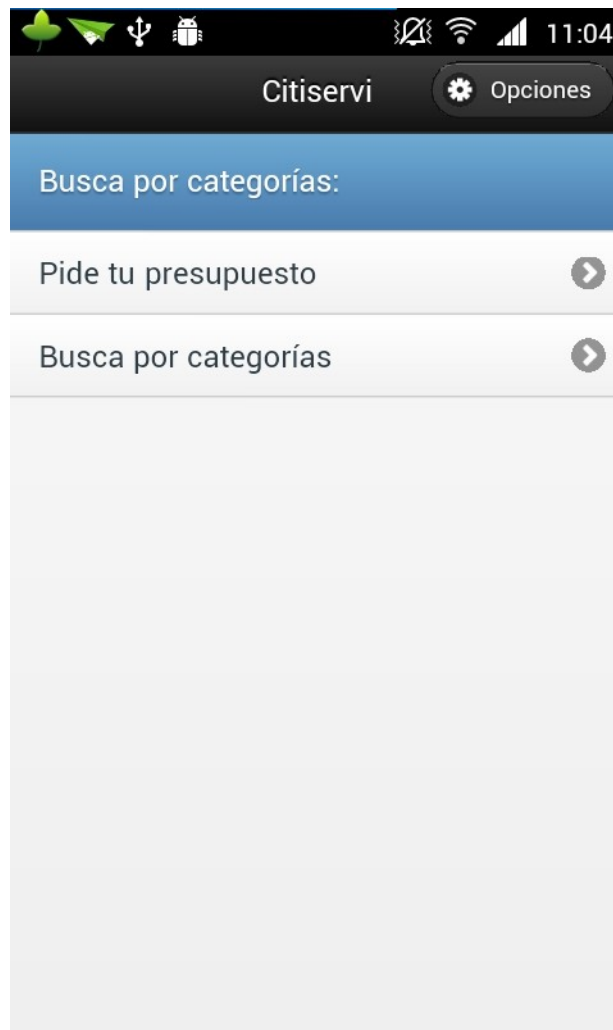
En este apartado se va a mostrar el uso de la aplicación móvil realizado durante el proyecto fin de grado el cual usa el framework multiplataforma. En este caso, se ha hecho sobre un dispositivo Android.

La estructura del apartado es la siguiente: se irán mostrando capturas de pantalla del dispositivo móvil, y debajo una breve descripción para entender el uso de la misma.



18. Ilustración: Pantalla de inicio

En primer lugar aparece esta pantalla (ilustración anterior) para preguntar al usuario desde qué país se conecta. Esta página es provisional ya que es la que determina a qué servidor conectarse, si al de “Desarrollo”, al servidor de “España” o al servidor de “United States”. En este caso, elegimos el servidor de desarrollo.



19. Ilustración: Pantalla de menú principal

La ilustración anterior muestra cómo es el menú principal de la aplicación donde se puede elegir entre dos opciones:

- “Pide tu presupuesto” que sirve para pedir un presupuesto directamente se trate del sector o categoría que sea.
- “Busca por categorías” que sirve para buscar entre alguna categoría o sector y definir con más detalle qué es lo que necesita el usuario.

Arriba a la derecha hay un botón de “Opciones”, pantalla la cual se mostrará más hacia delante, ya que, es un botón que aparece durante casi toda la aplicación para poder editarlas cuando el usuario así lo deseé.

Si elegimos la opción "Pide tu presupuesto" nos aparece el siguiente formulario:



The screenshot shows a mobile application interface for 'Citiservi'. At the top, there is a navigation bar with three buttons: 'Inicio' (Home), 'Citiservi' (the app name), and 'Localízate' (Locate yourself). Below the navigation bar is a blue header with the text 'Solicita un presupuesto'. The main form area contains several input fields and dropdown menus, all outlined with a red border. The first field is a text input with the placeholder '¿Qué es lo que necesitas?'. Below it is a larger text input with the placeholder 'Describe el trabajo a realizar'. The next field is a text input for 'Código postal' with the example 'Ej: 28000'. Below that is a dropdown menu for 'Provincia' with the placeholder 'Selecciona una provincia'. The final field is a dropdown menu for 'Area' with the placeholder 'Selecciona un área'. The status bar at the top of the screen shows various icons including a green leaf, a USB symbol, a bug, a signal strength indicator, and the time '11:10'.

**20. Ilustración: Pantalla de formulario
"Pide tu presupuesto"**

Como se puede observar en la ilustración anterior se pide al usuario una serie de datos para que solicite su presupuesto. Arriba dispone de dos opciones, o localizarse mediante su posición de geolocalización del dispositivo y así rellenar varios datos del formulario o ir hacia atrás, a la pantalla de inicio.



The screenshot shows a mobile application interface with a status bar at the top displaying various icons and the time 11:11. The form contains three input fields: a text field labeled 'Describe el trabajo a realizar' with a red border, a text field labeled 'Código postal' containing the value '28025' with a blue border, and a dropdown menu labeled 'Provincia' with the text 'Selecciona una provincia' and a heart icon. A numeric keypad is overlaid on the bottom half of the screen, featuring buttons for digits 1-9, 0, *, #, and symbols like parentheses, plus/minus, slash, and a right arrow. The digit '5' is highlighted in a dark box.

21. Ilustración: Pantalla con teclado numérico

Cada campo del formulario es de un tipo. En el caso de los campos de tipo numéricos, se ofrece mayor comodidad al usuario mostrando un teclado numérico. De esta forma, como se muestra en la ilustración, es más sencillo y fácil insertar un código postal.



280233

Provincia

Selecciona una provincia

Area

Selecciona un área

Población

Donde se prestará servicio

Nombre

Tu nombre para contactar

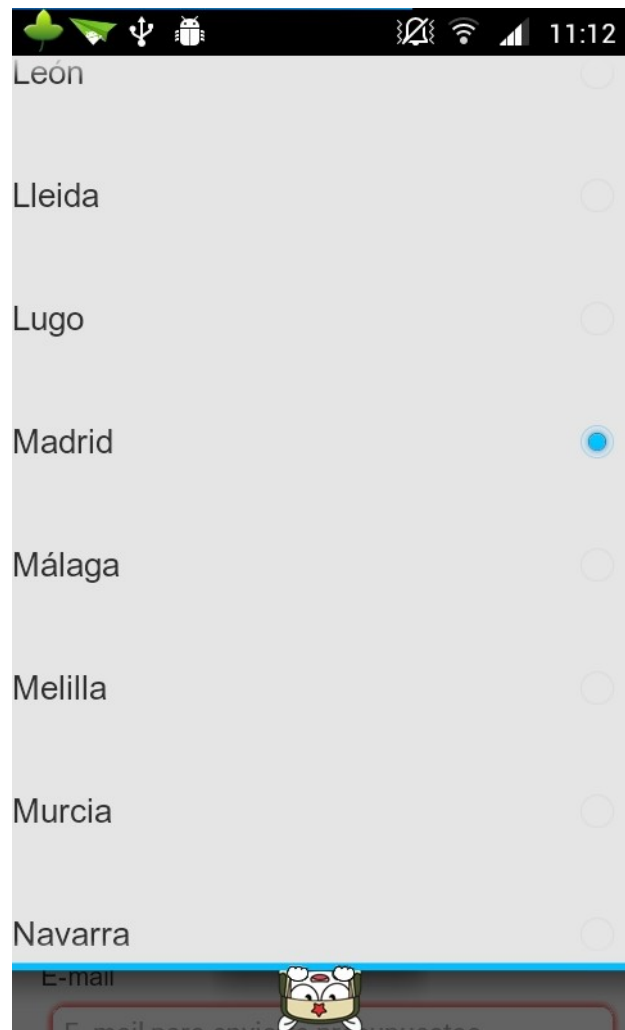
Teléfono

Móvil o fijo para contactar

E-mail

E-mail para enviar propuestas

22. Ilustración: Botón de opciones



León

Lleida

Lugo

Madrid

Málaga

Melilla

Murcia

Navarra

E-mail

E-mail para enviar propuestas

23. Ilustración: Botón de opciones (II)

En estas dos ilustraciones se puede observar el funcionamiento de un botón de opciones en el formulario.



24. Ilustración: Botón de opciones (III)

Cuando se elija una provincia, automáticamente el dispositivo hará una consulta al servidor y cargará las áreas correspondientes a esa provincia en el siguiente botón. El resultado se puede observar en la anterior ilustración.



The screenshot shows a mobile application interface with a status bar at the top displaying various icons and the time 11:12. The main content area has a light gray background and contains three input fields, each with a red border. The first field is labeled 'Teléfono' and contains the placeholder text 'Móvil o fijo para contactar'. The second field is labeled 'E-mail' and is empty. The third field is labeled 'Repita e-mail' and contains the placeholder text 'Para verificarlo'. Below these fields is a button labeled 'Enviar formulario'. At the bottom of the screen is a virtual keyboard with a light gray background. The keyboard has four rows of keys. The first row contains numbers 1-0. The second row contains letters q-p. The third row contains symbols @, #, &, *, -, +, =, (,), and ñ. The fourth row contains a backspace key, letters z-m, and a forward slash key. The bottom row contains a globe icon, a microphone icon, a spacebar, an @ symbol, a comma/semicolon icon, and a checkmark icon.

25. Ilustración: Pantalla con teclado de correo electrónico

En el campo “E-mail” (correo electrónico) el teclado se adapta poniendo más accesible el carácter '@', como se puede observar en la ilustración anterior.



26. Ilustración: Pantalla de opciones
"Localízate"

Si el usuario pulsa la opción "Localízate" anteriormente mostrada, aparece este diálogo que se ve en la ilustración anterior con las tres opciones: "Usar mis preferencias", la cual carga en el formulario los datos almacenados en las preferencias de la aplicación, "Geolocalizar ahora", directamente geolocaliza al usuario y "Ir a preferencias", el cual es un enlace a la página de preferencias.

Siendo el supuesto que el usuario ha pulsado “Ir a preferencias” aparece la siguiente pantalla:



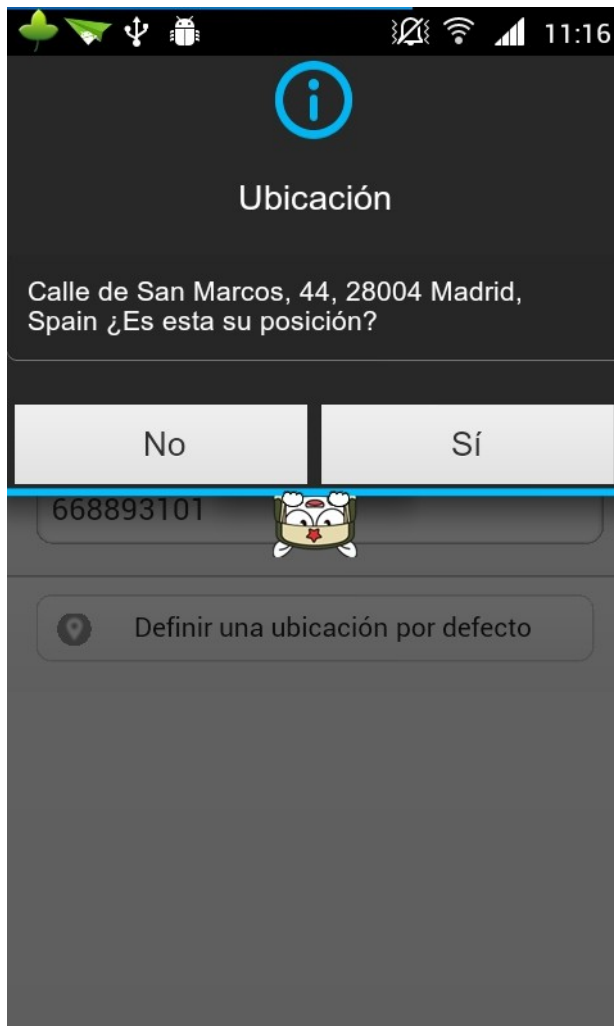
27. Ilustración: Pantalla de preferencias



28. Ilustración: Pantalla de preferencias (II)

En ella, puede guardar su correo electrónico personal donde quiera recibir los presupuestos, su número de teléfono y su ubicación por defecto, para así, no tener que geolocalizar en todo momento el dispositivo (ya que es un proceso algo lento).

En el caso de querer guardar una localización por defecto, el comportamiento es el siguiente:




30. Ilustración: Pantalla de preferencias (III)



29. Ilustración: Pantalla de preferencias (IV)

Se geolocaliza al usuario con sumo detalle y se le pregunta si esa es su posición. En caso afirmativo rellena el formulario de las preferencias, en caso negativo desecha la geolocalización. Si el usuario desear guardar sus preferencias pulsa el botón "Guardar" y se le redirige a la página de donde procede.

Si en vez de ir a las preferencias, el usuario hubiera pulsado “Usar mis preferencias” los campos del formulario como la localización, el teléfono y el correo se hubieran rellenado automáticamente (como se puede observar en la ilustración siguiente de la izquierda):



Una vez que envíes la solicitud te llamaremos por teléfono para validar los datos antes de enviárselo a los profesionales

(*) Teléfono:
668893101

(*) Correo Electrónico:
jespinosa@citiservi.com

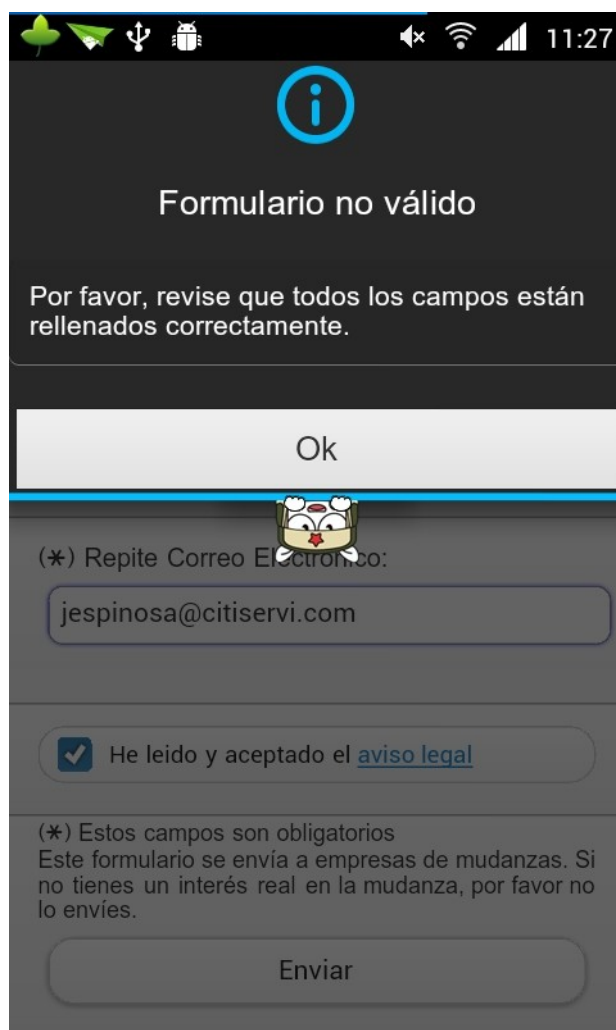
(*) Repite Correo Electronico:
jespinosa@citiservi.com

☒ He leído y aceptado el [aviso legal](#)

(*) Estos campos son obligatorios
Este formulario se envía a empresas de mudanzas. Si no tienes un interés real en la mudanza, por favor no lo envíes.

Enviar

32. Ilustración: Uso de “Usar mis preferencias”



Formulario no válido

Por favor, revise que todos los campos están rellenos correctamente.

Ok

(*) Repite Correo Electrónico:
jespinosa@citiservi.com

☒ He leído y aceptado el [aviso legal](#)

(*) Estos campos son obligatorios
Este formulario se envía a empresas de mudanzas. Si no tienes un interés real en la mudanza, por favor no lo envíes.

Enviar

31. Ilustración: Pantalla "Formulario no válido"

Al pulsar “Enviar” se validan los datos introducidos y se comprueba que no falte algún campo obligatorio por rellenar. En caso de que algún campo no se haya rellenado correctamente o algún campo obligatorio no se haya completado, aparece el aviso de la ilustración anterior de la derecha y el dispositivo móvil vibra.

Si se elige la opción del menú de inicio “Buscar por categoría” aparecería una lista de sectores donde el usuario podría elegir para solicitar presupuesto:



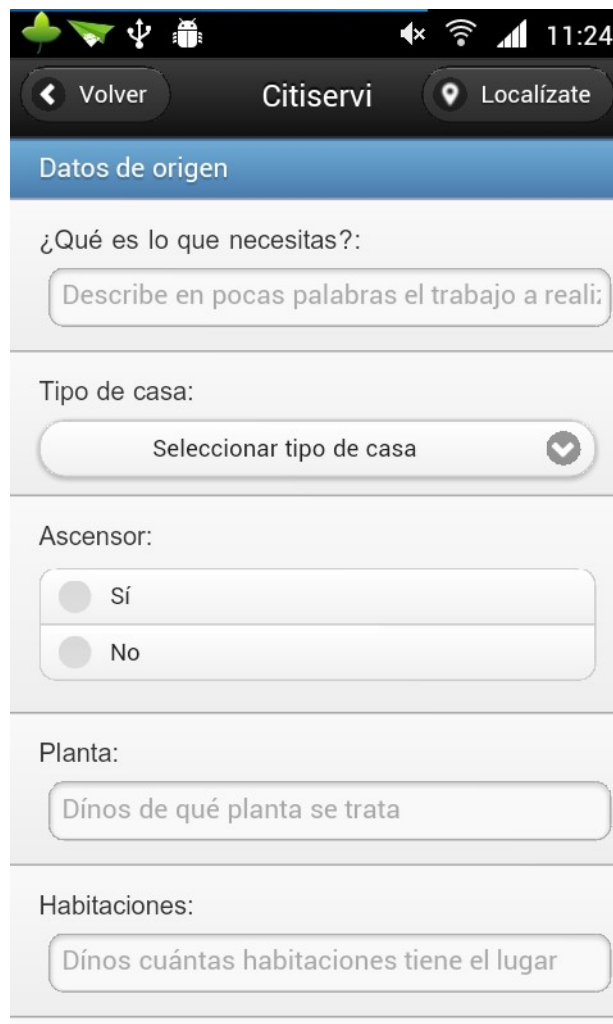
34. Ilustración: Pantalla "Buscar por categoría"



33. Ilustración: Búsqueda de categoría

La lista de categorías tiene más de 80 elementos, por tanto tiene un buscador “al vuelo” para que sea mucho más cómoda la elección, como se puede apreciar en la ilustración anterior de la derecha.

Si el usuario elige la opción "Mudanzas" aparece el siguiente formulario con campos mucho más específicos para la categoría, facilitando el servicio posterior a la hora de recibir el presupuesto:



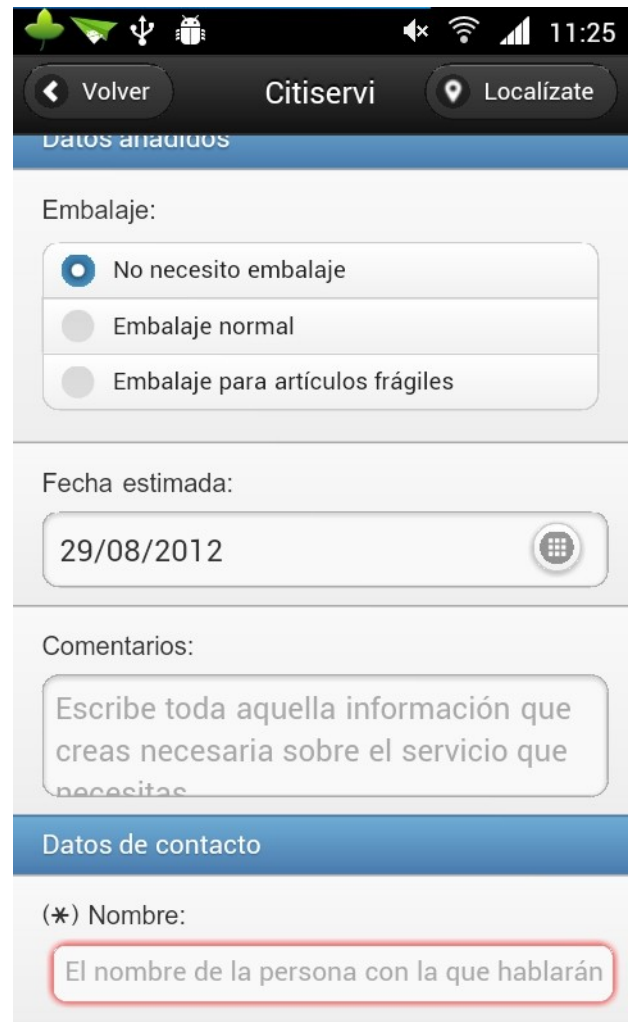
The screenshot shows a mobile application interface for 'Citiservi'. At the top, there is a status bar with icons for signal, Wi-Fi, and battery, and the time 11:24. Below the status bar is a navigation bar with a back arrow and 'Volver', the app name 'Citiservi', and a location icon with 'Localízate'. The main content area is titled 'Datos de origen' in a blue header. The form consists of several sections: 1. '¿Qué es lo que necesitas?:' with a text input field containing the placeholder 'Describe en pocas palabras el trabajo a reali:'. 2. 'Tipo de casa:' with a dropdown menu showing 'Seleccionar tipo de casa' and a downward arrow. 3. 'Ascensor:' with two radio button options: 'Sí' and 'No'. 4. 'Planta:' with a text input field containing the placeholder 'Dínos de qué planta se trata'. 5. 'Habitaciones:' with a text input field containing the placeholder 'Dínos cuántas habitaciones tiene el lugar'.

35. Ilustración: Pantalla de formulario de "Mudanzas"

Para los campos de fecha, se muestra un botón que ayuda al usuario a introducir una fecha en concreto, como se puede observar en las ilustraciones siguientes:

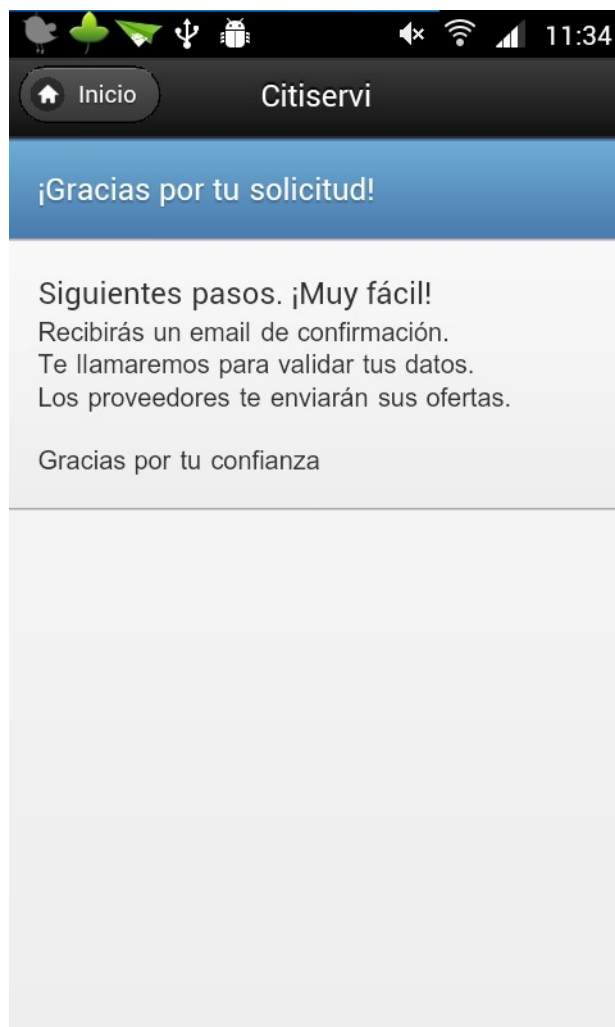


36. Ilustración: Formulario con campo de fecha



37. Ilustración: Formulario con campo de fecha (II)

Por último, tras pulsar “Enviar” (y que los datos sean correctos) se recibe el formulario en el servidor y muestra una página de éxito como esta:



38. Ilustración: Pantalla de éxito

7. Planificación y presupuesto

7.1 Planificación

En este apartado se detallará la planificación seguida para la elaboración del presente trabajo de fin de grado.

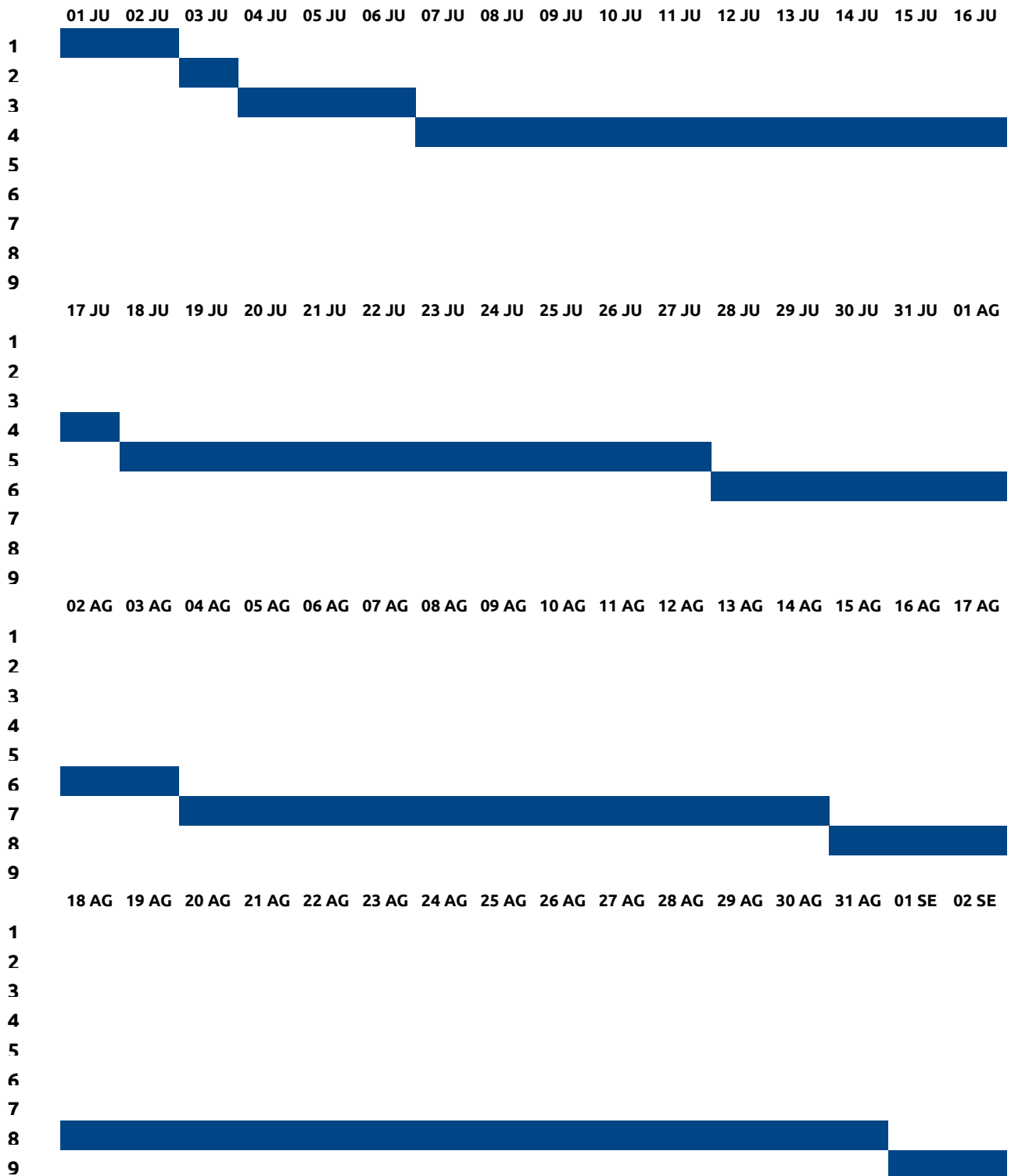
A continuación, en la siguiente tabla se puede observar las actividades realizadas durante el desarrollo del sistema del trabajo indicando la fecha de inicio y fin de cada una y la duración en días de cada actividad.

Actividad	Fecha inicio	Duración	Fecha fin
1. Elaboración requisitos	01/07/2012	2 días	02/07/2012
2. Selección de tecnologías	03/07/2012	1 día	03/07/2012
3. Diseño de framework	04/07/2012	3 días	06/07/2012
4. Implementación en el servidor	07/07/2012	10 días	17/07/2012
5. Implementación diseño web	18/07/2012	10 días	27/07/2012
6. Implementación aplicación móvil	28/07/2012	7 días	03/08/2012
7. Evaluación y pruebas	04/08/2012	10 días	14/08/2012
8. Elaboración de la memoria	15/08/2012	15 días	30/08/2012
9. Revisión	31/08/2012	2 días	01/09/2012

17. Tabla: Planificación del trabajo por actividades

7.1.1 Diagrama de Gantt

La planificación del trabajo en formato de Diagrama de Gantt (el número se corresponde con la anterior tabla de planificación del trabajo):



7.2 Presupuesto

En este apartado se va a detallar el presupuesto del proyecto de fin de grado.

Para la realización de los cálculos de costes se han tenido en cuenta las siguientes consideraciones:

- El inicio del proyecto fue el 1 de julio de 2012 y la fecha de finalización el 1 de septiembre de 2012. Por tanto, la duración total fueron dos meses.
- Tomando como referencia la duración de días de las actividades en el apartado de planificación queda un total de **60 días**. No se elimina ningún día por festivo, enfermedad o por otras razones.
- La jornada laboral ha sido variable, pero se estima una media de 7 horas diarias. Por tanto, hace un total de horas trabajadas de **420 horas** (60 días * 7 horas/día).

7.2.1 Costes de personal

En este apartado se va a detallar los costes asignados al personal del proyecto. En este caso, la única persona que ha participado en la realización del proyecto ha sido el autor de este documento. Por tanto, será quien asuma todos los roles especificados a continuación para la realización de las actividades antes definidas en la planificación:

Actividad	Rol	Coste (por hora)	Número de horas	Coste total
Elaboración requisitos	Analista	15 €	14 horas	210 €
Selección de tecnologías	Analista	15 €	7 horas	105 €
Diseño de framework	Programador	25 €	21 horas	525 €
Implementación en el servidor	Programador	20 €	70 horas	1.400 €
Implementación diseño web	Programador	25 €	70 horas	1.750 €
Implementación aplicación móvil	Programador	30 €	49 horas	1.470 €
Evaluación y pruebas	Analista	25 €	70 horas	1.750 €
Elaboración de la memoria	Analista	20 €	105 horas	2.100 €
Revisión	Analista	25 €	14 horas	350 €
TOTAL				9.310 €

18. Tabla: Costes de personal

7.2.2 Costes de hardware

En este apartado se detallarán los gastos del proyecto sobre elementos hardware imprescindibles para la realización del mismo. Estos elementos son el ordenador, los dispositivos móviles donde se han realizado las pruebas del sistema y un servidor donde desplegar el proyecto.

Para calcular el coste imputable se hace mediante la fórmula de amortización:

Coste imputable = (Dedicación / Período de depreciación) * Coste sin IVA * % Uso dedicado

La tabla de costes de hardware queda así:

Concepto	Coste sin IVA	% uso dedicado	Dedicación	Período de depreciación	Coste imputable
HTC Desire S	350 €	100 %	2 meses	48 meses	14,58 €
Ordenador Acer eMachines	250 €	100 %	2 meses	48 meses	10,41 €
Pantalla Acer 19"	70 €	100 %	2 meses	48 meses	2,91 €
Mini ratón Acer óptico	7 €	100 %	2 meses	12 meses	1,16 €
Teclado Acer Aspire	40 €	100 %	2 meses	12 meses	6,66 €
iPhone 3GS 8GB	230 €	50 %	1 mes	48 meses	2,39 €
Apple MacBook Pro	950 €	20 %	1 mes	48 meses	3,95 €
Servidor dedicado OVH	210 €	100 %	2 meses	3 meses	140 €
TOTAL					182,06 €

19. Tabla: Costes de hardware

7.2.3 Costes de software

En este apartado se detallarán los costes debidos a licencias de software necesario para el desarrollo del proyecto. El proyecto se ha desarrollado sobre las siguientes herramientas de software, la mayoría de ellas no tienen coste:

Concepto	Coste
Licencia de Ubuntu 12.04	0,00 €
Licencia de Apache Software	0,00 €
Licencia de PhoneGap	0,00 €
Licencia de Eclipse	0,00 €
Licencia de Smarty 3	0,00 €
Licencia de PHP Unit	0,00 €
Licencia iOS Developer Program	66,95 €
Licencia Google Play Market	21,20 €
TOTAL	88,15 €

20. Tabla: Costes de software

7.2.4 Coste total

Teniendo en cuenta lo anterior, tenemos como presupuesto total del proyecto la siguiente tabla:

Concepto	Coste
Coste de personal	9.310,00 €
Coste de hardware	182,06 €
Coste de software	88,15 €
Coste sin IVA	9.580,21 €
Beneficio 50 %	4.790,11 €
Margen de riesgo 25 %	2.395,05 €
IVA (18%)	3.017,77 €
COSTE TOTAL	19.783,14 €

21. Tabla: Coste total

Para calcular el beneficio se ha considerado el adecuado, ya que hay que tener en cuenta que la rama de las tecnologías es muy sensible a cambios. Se ha calculado sobre el coste sin IVA.

En cambio, para calcular el margen de riesgo, se ha considerado un 25% por tener cierta experiencia en el uso de las herramientas con las que se ha llevado a cabo el proyecto y reducen los riesgos de retraso de entrega. Se ha calculado sobre el coste sin IVA.

El IVA se ha calculado sobre la suma del coste sin iva, el beneficio y el margen de riesgo.

El coste total del proyecto es de 19.783,14 € (diecinueve mil setecientos ochenta y tres euros con catorce céntimos).

8. Conclusiones y líneas futuras

En este apartado se describirán las conclusiones posteriores a la realización del trabajo de fin de grado con el fin de revisar los objetivos planteados en el apartado de introducción, para comprobar el grado de satisfacción de cada uno de ellos. Tras este análisis, se comentan algunas líneas futuras sobre el trabajo fin de grado donde se podrían seguir avanzando funcionalidades del proyecto, en este caso, del framework.

8.1 Conclusiones

En el primer apartado de este documento se describen una serie de objetivos que se plantearon para la realización del trabajo de fin de grado, por tanto, a continuación se exponen las conclusiones sobre el cumplimiento de dichos objetivos:

- Se ha conseguido que el desarrollo de una aplicación web sea ágil, y que no se dedique el mínimo de tiempo en las etapas de inicio de una nueva aplicación. Además, se ha conseguido que la ampliación con nuevas páginas o funcionalidades de una aplicación web ya creada sea también desarrollado de forma ágil y sin afectar a lo anterior ya desarrollado.
- Se ha conseguido que las aplicaciones web creadas con el framework del trabajo sean compatible con la mayoría de dispositivos móviles desarrollo único para cada uno de ellos. Además, las aplicaciones web son compatibles con la mayoría de

navegadores web en los ordenadores personales.

- Se ha conseguido llevar a cabo todos los objetivos planteados mediante el uso de tecnologías dominantes en el mercado laboral. Facilitando de esta forma el desarrollo de las aplicaciones web.
- Por último, gracias a los anteriores objetivos planteados y al uso de tecnologías dominantes, se consigue que mediante este framework el desarrollo de aprendizaje sea ágil y sencillo, reduciendo así la curva de aprendizaje al mínimo.

Tras haber cumplido los objetivos principales planteados en un principio, se puede llegar a la conclusión de que se ha alcanzado satisfactoriamente los objetivos del trabajo de fin de grado.

Desde un punto de vista personal, el autor de este mismo documento ha podido comprobar cómo el trabajo ha llegado a buen recaudo y se va a dar un uso real en un futuro muy próximo, sentando unas bases importantes de desarrollo para aplicaciones web futuras sin distinción de dispositivos.

Tratándose de un trabajo realizado dentro de un entorno empresarial será posible poder ampliar alguna funcionalidad al sistema e incluso mejorar alguna de ellas. Teniendo por tanto la gratitud de que no es un trabajo que quede en el olvido. Se obtendrán frutos en un periodo breve de tiempo, ya sean beneficios económicos como beneficios a la hora de desarrollar nuevas aplicaciones.

8.2 Líneas futuras

En este apartado se van a presentar posibles líneas de futuro para el trabajo, las cuales con gran probabilidad serán estudiadas y planteadas para desarrollarse.

- Conexión segura por SSL. En el desarrollo para aplicaciones de dispositivo móvil no se ha tenido en cuenta el acceso mediante conexión segura SSL al servidor.
- Almacenamiento seguro. En el desarrollo para aplicaciones web de dispositivo móvil el almacenamiento no se realiza de forma segura y la información va en claro. En un principio, la información que se almacena actualmente no se considera muy sensible si llega a ser descubierta, pero sí sería importante en un futura solventar esta carencia.
- Mejora de la interfaz gráfica. La actual interfaz gráfica que proporciona el framework para el dispositivo móvil carece de personalidad y es algo complicado desarrollar para mejorarla. Por tanto, mejorando el framework en este aspecto, se podría conseguir una interfaz más flexible para su desarrollo.
- Sesiones de usuario. Permitir que en las aplicaciones web para dispositivos móviles se pueda mantener una sesión de usuario con el servidor. De tal forma que pueda gestionar su cuenta y realizar una serie de acciones en el sistema dependiendo de sus permisos. Todo ello, obviamente, de una forma segura.

9. Bibliografía

1. JSON. JavaScript Object Notation, 2008:
<http://www.json.org/>
2. MoSync, 2012:
<http://www.mosync.com/>
3. Sencha Touch, 2012:
<http://www.sencha.com/products/touch>
4. Savant, 2012:
<http://www.phpsavant.com/>
5. PHP Unit. Main page, 2012:
<http://www.phpunit.de/>
6. PHP Unit. Mock Objects, 2010:
<http://www.phpunit.de/manual/3.0/en/mock-objects.html>
7. HTML5 Rocks, 2012:
<http://www.html5rocks.com/en/>
8. PHP Documentation, 2012:

<http://www.php.net/manual/en/>

9. MySQL. The world's most popular open source database, 2012:

<http://www.mysql.com>

10. REST. Representational State Transfer, septiembre 2012:

http://en.wikipedia.org/wiki/Representational_State_Transfer

11. REST. How I explained REST to my wife, diciembre 2004:

<http://tomayko.com/writings/rest-to-my-wife>

12. PhoneGap. Apache Cordova Documentation, agosto 2012:

<http://docs.phonegap.com/en/2.0.0/index.html>

13. Smarty 3. Documentation, agosto 2012:

<http://www.smarty.net/docs/en/>

14. HTTP. List of HTTP status codes, agosto 2012:

http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

15. jQuery. Documentation, 2012:

http://docs.jquery.com/Main_Page

16. jQuery Mobile. Documentation, 2012:

<http://jquerymobile.com/demos/1.1.1/>

17. jQuery Mobile. Theme Roller, 2012:

<http://jquerymobile.com/themeroller/index.php>

18. jQuery Mobile. Andy Matthews. Creating and using custom icons, 2011:

<http://www.andymatthews.net/read/2011/02/13/Creating-and-using-custom-icons-in-jQuery-Mobile>

19. jQuery Mobile. DateBox, 2012:

<http://dev.jtsage.com/jQM-DateBox/>

20. jQuery. Custom Input Validation Error Messages, 2012:

<https://github.com/javespi/civem-jquery-plugin>

21. "Pro Javascript Techniques. Real-world Javascript techniques for the modern, professional web developer". John Resig, Apress, 2006

22. "Pro Javascript Design Patterns. The essentials of object-oriented Javascript programming" Ross Harmes and Dustin Diaz, Apress, 2008