Escuela Técnica Superior de Ingeniería de Telecomunicación

Ingeniería de Telecomunicación-ITIS

Proyecto Fin de Carrera

Aplicación de Ayuda a la Gestión de Actividades Universitarias orientada a Smartphones

Autor: Christian De La Fuente Barrio

Tutor: Gregorio Robles Martínez

Curso Académico 2012/2013

No progresas mejorando lo que ya está hecho,
sino esforzándote por lograr lo que aún queda por hacer.
Gibran Jalil Gibran

Agradecimientos

Primeramente me gustaría agradecer el interés del lector que va a proceder a leer esta memoria. Espero que a lo largo de la misma se pueda hacer una idea de lo que ha supuesto este Proyecto de Fin de Carrera desde sus inicios hasta el final del mismo.

A continuación agradecer la labor del tutor de este proyecto: Gregorio Robles Martínez, que desde que fui a su despacho me ofreció una idea y dedicó tiempo de su trabajo para lo que necesitase. De igual modo me gustaría agradecer la labor de todos y cada uno de los profesores que me han impartido clase a lo largo de la carrera, los cuáles me han ayudado además de aprender lo concerniente a las asignaturas, a ser más maduro y tener mente de ingeniero.

También agradecer a todos los alumnos y profesores que se han tomado el tiempo de probar este Proyecto de Fin de Carrera, algunos antes incluso de la fase de pruebas, mientras estaba en proceso de desarrollo.

Me gustaría agradecer con gran cariño a todos aquellos compañeros que pasaron a ser amigos a lo largo de este periplo estudiantil: a Marijose por su infinita paciencia y apoyo, a María por toda esa ayuda que siempre estaba dispuesta a dar, a Jaime por su gran nobleza, a Gonzalo en el cual encontré un apoyo inesperado, y a muchos más como a Mónica, Dani o Alex, entre otros.

Por último, pero no por ello menos importante, querría agradecer incondicionalmente a mi familia, la cual ha estado siempre apoyándome y sufriendo conmigo, desde el primer día que pisé la universidad e incluso mucho antes. Un gran agradecimiento a los que aún están conmigo, como mi tía que me inspira confianza cada día de mi vida, y otro mayor a los que ya no están, como mi tío que me sugirió meterme a esta carrera, y al que le dedico un infinito gracias, o a mi padre, que estoy seguro que se sienten orgullosos desde donde quiera que estén.

Resumen

Las redes sociales y los smartphones han supuesto un cambio en cómo se orienta la tecnología en los últimos años. La posibilidad de estar conectado a internet en todo momento ha abierto un nuevo mundo de posibilidades y de aplicaciones.

Aparte, las universidades, como punto tecnológico de referencia, tienen la necesidad de estar siempre a la última en cuanto a desarrollo tecnológico. Como la sociedad, y en particular los alumnos universitarios, tienen al alcance la posibilidad de hacer uso de los smartphones y están íntimamente relacionados con las redes sociales, era de esperar poder unir ambos mundos en una aplicación.

Este Proyecto de Fin de Carrera se centra precisamente en esa unión, de forma que las actividades o eventos de la universidad sean fácilmente accesibles tanto a profesores como a alumnos. Así, los profesores pueden publicar en una interfaz web los eventos que precisen y asociarlos a listas relativas al contenido de dichos eventos. Los alumnos por su parte pueden "seguir" dichas listas y compartir los eventos en las redes sociales, de modo que haya la mayor difusión de información posible de la manera más rápida y sencilla.

La implementación es de fácil comprensión pero usando las últimas tecnologías web del momento, para que la interfaz, aparte de ser visualmente atractiva, exprima la usabilidad y a la vez suponga un reto para el desarrollador: usando HTML5 y JQuery Mobile sobre un servidor web en Django.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivaciones

Una de las principales motivaciones ha sido el hecho de poder desarrollar un proyecto más allá de lo que son las prácticas realizadas a lo largo de la carrera. Un proyecto en el que las decisiones que se vayan tomando no partan de un guión preestablecido y que siga todo el proceso, desde el inicio hasta el final, con las repercusiones que éste pueda ocasionar.
Para esto era necesario también que fuera acorde con los conocimientos y aptitudes del desarrollador, así como la búsqueda de un reto al que poder enfrentarse.

Las posibilidades de hacer un calendario de eventos son muchas y variadas, a continuación se enumerarán unas cuantas:

- **Calendario de eventos de Moodle:** La principal ventaja de Moodle es que es una plataforma que ya se está usando, los alumnos se han habituado a ella y es bastante sencilla de usar. La principal desventaja es que cada calendario de cada asignatura o departamento es independiente, y hay que meterse en cada apartado de forma individual. Partiendo de este hecho, una herramienta que uniese los calendarios en una misma interfaz, y que el alumno pudiera elegir cuáles ver y cuáles no, podría ser una solución a esta desventaja. Aparte podríamos unir eventos externos a Moodle de lo que no tenemos posibilidad hasta ahora.
- **Calendario de eventos de Google (Google Calendar):** El calendario de eventos de Google es una herramienta muy potente, muy bien desarrollada, pero tiene una funcionalidad extra innecesaria y cierta complicación de uso. Además no está conectada de manera directa al ámbito universitario, y los alumnos y profesores podrían no sentirse cómodos si no están acostumbrados a ella. Por tanto tenemos

potencia pero descentralizada, una solución sería imitar esta herramienta adecuándonos a nuestro contexto o unir ambas interfaces.

Aparte de estas dos opciones tenemos otras, como calendarios personalizados, ya sea en el PC o en nuestro móvil o Tablet, pero perderían la globalidad. Los profesores no tendrían la posibilidad de publicar eventos en todos los calendarios personalizados de los alumnos a la vez. Es más, la posibilidad de que los profesores puedan publicar eventos, y que de manera inmediata el alumno pueda tener constancia de ello es algo que se puede buscar con este proyecto.
Por tanto los calendarios personalizados no son lo más adecuado para nuestros propósitos.

Estos últimos años son cada vez más los que navegan por internet y gestionan todo el mundo electrónico a través de un smartphone. Estos dispositivos, cada vez más potentes, ofrecen al usuario una experiencia global: el acceso al correo, la gestión de contactos, las redes sociales, etc., todo ello en un único dispositivo. Por tanto no es descabellado pensar que las interfaces web cada vez estén más orientadas al acceso a las mismas desde dispositivos móviles.
Entre ellos también destacan con mayor peso cada día las tablets. Éstas, con una pantalla más amplia, permiten exprimir las interfaces web de una manera más visual, cercana a la experiencia con un PC.
Por tanto vemos que aunque todavía un gran número de personas accede a internet mediante un PC o un portátil, el uso de smartphones y tablets crece diariamente.
Esto nos hace pensar que nuestra interfaz web debería considerar este hecho y tendríamos que buscar herramientas para aprovechar este potencial uso de la mejor manera posible.

Desde que en 2002 las redes sociales hicieron su aparición en internet, la vida de los internautas ha cambiado. Con cientos de redes sociales distintas, cada una orientada a un público determinado, cada persona ha podido elegir en cual o cuales se siente más cómodo. Esto ha revolucionado la forma de comunicarse, ya que mediante chats, menciones o publicaciones, la información fluye de manera continua y llega a más personas por el simple hecho de compartir una red social. Esto por supuesto supone centrar nuestra atención en las redes sociales y desarrollar productos que las complementen o que nos ayuden a aprovechar su potencial. De este modo, la conexión con las redes sociales se ha convertido en algo casi necesario para las nuevas aplicaciones web. Ya sea publicitándose o mediante el simple boca a boca, no cabe la menor duda de que la difusión de información es enorme. Esto supone una motivación suficiente para tener un producto unido a estos entornos a los que a su vez están más acostumbrados los usuarios.

Teniendo en cuenta todas estas necesidades, podemos pasar a ver las características principales que motivan la creación de este proyecto:

- **Multiplataforma:** La interfaz web debe ser accesible desde cualquier dispositivo, haciendo hincapié en el hecho de que pueda ser visitada desde dispositivos móviles. Para ello debemos buscar una herramienta que nos ayude a explotar el acceso.
- **Difusión de la información:** Debido a que queremos guardar la información de eventos universitarios, buscamos la forma de llegar a un gran número de personas. Esto se podría solucionar mediante la difusión en las redes sociales del momento.
- **Personalización de contenidos:** Está claro que gran cantidad de información puede ocasionar que los usuarios se pierdan o no lleguen con claridad al contenido que busquen. Para ello debemos disponer de un buscador y la posibilidad de guardar, de forma personalizada, la información que cada usuario desee.
- **Visibilidad de la información:** Conectado con el punto anterior está el hecho de cómo mostrar la información para que quede todo claro y se pueda usar la aplicación de manera intuitiva.

1.2. Objetivos

El objetivo primordial de este Proyecto de Fin de Carrera es el desarrollo de una interfaz web que permita a profesores publicar eventos o actividades de índole universitaria y la posibilidad de los alumnos de 'seguir' estos eventos organizados en listas de contenidos. Este objetivo es demasiado ambicioso como para hacerlo directamente luego es mejor desglosarlo en objetivos más pequeños para descomponer el problema y abordarlo de manera más efectiva:

- **Diseño e implementación de una base de datos que albergue toda la información disponible.**
  Es fundamental tener una base de datos sólida que almacene los distintos tipos de usuarios y la información personalizada de cada uno de ellos de forma que las relaciones entre ellos sean lo más solidas posibles. Además esta base de datos debe servir para estructurar la información total de eventos en listas y guardarla de manera óptima.
- **Diseño e implementación de una interfaz web que aproveche el uso desde toda clase de dispositivos.**

Nuestro proyecto debe tener una interfaz web acorde a las motivaciones que se han expuesto, y una de ellas fundamental es el uso de smartphones y tablets. Por esto debemos buscar herramientas que nos ayuden a explotar el potencial web orientado tanto a dispositivos móviles como a dispositivos más tradicionales.

- **Evaluación de la aplicación en una fase de pruebas con encuestas para determinar la experiencia por parte de los usuarios.**
  Como casi todo proyecto en el que intervenga una interfaz web y una base de datos de usuarios, se necesita de un proceso de prueba en el que se vea la aplicación activa y cómo los usuarios se desenvuelven en ella. Además, se necesita un control de la experiencia que hemos recogido mediante una encuesta libre de realizar por cualquiera que haya probado la aplicación web para poder comprobar si se cumplen los objetivos principales y recabar información para posibles ampliaciones sobre el proyecto.

En resumen, tenemos por delante que trabajar sobre una base de datos que almacene la información para su correcta importación y exportación, sobre una interfaz web orientada al uso desde dispositivos móviles y todo ello sobre un servidor web que soporte la aplicación.

1.3. Estructura de la Memoria

De manera breve aquí se facilita un listado de las secciones y capítulos de la memoria junto con una pequeña descripción de su contenido:

- ***Capítulo 1: Introducción***
  En este capítulo se introduce al lector en los inicios del por qué surge la necesidad de crear este Proyecto de Fin de Carrera, las motivaciones originales por las que se realiza y los principales objetivos que buscamos alcanzar con él. Además se dan unas pequeñas pinceladas sobre la estructura y el contenido de los capítulos de la memoria.

- ***Capítulo 2: Estado del Arte***
  En el capítulo 2 se estudian las herramientas principales usadas en el proyecto, su repercusión en los sistemas implementados hasta ahora y todo lo necesario para poder entender después más en profundidad el diseño y la implementación de nuestra aplicación.

- ***Capítulo 3: Diseño e Implementación del sistema***
  En este capítulo se aborda el contenido fundamental del proyecto, cómo se ha desarrollado con todo detalle y explicado paso a paso para poder alcanzar los objetivos propuestos en un principio.

- ***Capítulo 4: Despliegue y resultados***
  El contenido de este capítulo es fundamentalmente analizar el modo en el que se ha probado la aplicación y las principales características y resultados analizados en la fase de pruebas del proyecto.

- ***Capítulo 5: Conclusiones y futuras líneas de trabajo***
  En el quinto capítulo y último del grueso de la memoria se analizan las principales conclusiones derivadas del proyecto y las posibles mejoras que se podrían realizar si se quisiera seguir desarrollando el proyecto en profundidad.

- ***Bibliografía***
  Apartado dedicado a la enumeración de las referencias bibliográficas utilizadas tanto en el desarrollo del proyecto como de la misma memoria.

- ***Glosario***
  En este apartado se definen gran parte de las palabras técnicas, conceptos y acrónimos usados a lo largo de la memoria para hacernos una mejor idea de lo que estamos analizando.

Capítulo 2

Estado del Arte

En este capítulo se analizarán las características principales de las diferentes tecnologías usadas en este proyecto, de modo que se pueda hacer una mejor idea de lo que se ha llegado a implementar más adelante.

2.1. Aplicaciones cliente-servidor

Para comenzar a explicar las herramientas que se van a usar en este Proyecto de Fin de Carrera deberíamos primero desglosar un poco en qué consiste la arquitectura de las aplicaciones cliente-servidor.

2.1.1. Definiciones

El modelo cliente-servidor se divide en dos partes fundamentales: por un lado tenemos el servidor encargado de soportar el grueso de la aplicación, ya que tiene en la mayoría de casos casi la totalidad de la información y los recursos disponibles y es capaz de gestionar la conexión de múltiples clientes; por el otro, está el cliente, que pide determinados recursos al servidor y éste se los proporciona.

Según la cantidad de clientes que pueda procesar al mismo tiempo se diferencian dos tipos de servidores:

- **Interactivos:** aquéllos en los que el servidor sólo puede atender a un cliente al mismo tiempo, si otro cliente desea hacer uso del servidor deberá esperar a que termine el primero. Este tipo de servidor es adecuado si sabemos que vamos a tener pocas peticiones y pocos clientes, pero en el momento en el que nos empiecen a coincidir se producirán tiempos de espera demasiado largos.

- **Concurrentes:** precisamente para solucionar esos tiempos de espera existen los servidores concurrentes que pueden atender a un número de clientes determinado simultáneamente. Esto se puede hacer creando procesos dedicados a cada cliente, si el servidor puede admitir multiprocesos, reduciendo de manera notable el tiempo de espera de los clientes.

La arquitectura cliente-servidor choca con lo que se usaba antes que era la arquitectura monolítica, en la que no hay distribución ni a nivel físico ni a nivel lógico. Lógicamente se ha derivado a este tipo de arquitectura porque supone varias ventajas:

- La mayoría de información la tiene el servidor, por tanto la seguridad de la aplicación está menos comprometida a ataques por parte de clientes malintencionados ya que el cliente sólo puede acceder a una pequeña porción de los datos totales y necesita la interacción del servidor para recibir más recursos, los cuáles el servidor tiene la opción de restringírselos.

- Al estar divididos en dos partes cada parte puede gestionarse de manera separada. Esto hace que se puedan añadir más funcionalidades a cada parte individualmente o cambiar la gestión de los datos del servidor sin que interfiera a los clientes e incluso sin que éstos tengan noción de los cambios internos que se hayan producido. Esto también ayuda al hecho de que si un servidor falla o comienza a dar un mal funcionamiento pueda sustituirse por otro de manera que el cliente sea totalmente ajeno a todos estos cambios.

- Como el modelo cliente-servidor ha sido una elección óptima para el desarrollo de aplicaciones web, cada vez son más los recursos e investigaciones que se generan para fomentar la correcta funcionalidad de este tipo de modelo, así como la evolución de las interfaces web cada vez más amigables e intuitivas para que los clientes hagan uso de las mismas.

Aunque tiene muchas, no todo en el modelo de cliente-servidor son ventajas, ya que precisamente derivadas del modelo surgen varias desventajas:

- Si un servidor recibe muchas peticiones de clientes puede saturarse, y provocar varios escenarios desastrosos: pérdida de peticiones, caídas del sistema o imposibilidad de acceder por parte de los clientes. Esto podría solucionarse si hubieran varios servidores trabajando unidos, de manera que se fueran repartiendo las peticiones y si uno llegara a caerse todavía tendríamos el respaldo de otro, o en el caso de que necesitemos mayor robustez, otros.

- No todos los hardware sirven para ser servidores. Debido a la cantidad de clientes que debe soportar y la ingente información de la que debe disponer, los servidores deben tener un hardware y un software que sea capaz de sobrellevar toda esta información y carga de trabajo sin que pueda repercutir en el uso de los clientes.
- El cliente debe, en la mayoría de casos, recurrir al servidor para recibir la información que necesite. Por tanto, si el cliente quiere hacer uso de gran cantidad de información de manera continuada deberá hacer múltiples peticiones al servidor, mientras que si lo tuviera todo integrado en el cliente no habría esa necesidad.

2.1.2. Gestión de servicios/recursos

A continuación, se procederá a describir cómo es el funcionamiento interno de un servidor y de un cliente y cómo se gestionan las peticiones desde cada punto de vista:

- **Servidor.** Primeramente el servidor inicia el proceso y se abre el canal de comunicaciones, comunicando a la red la dirección del mismo y poniéndose a la espera de peticiones. En el caso de que reciba una petición se crea un proceso hijo para atenderla y se dispone a esperar de nuevo peticiones (en el caso de un servidor interactivo lo que haga el proceso hijo es lo que haría el propio servidor). En el proceso hijo se atiende la petición del cliente dándole la información, recurso o servicio que requiera, e incluso puede quedarse esperando si se necesitan más interacciones desde ese cliente. Por último se finaliza el proceso hijo (en el caso del servidor interactivo se volvería a esperar más peticiones).
- **Cliente.** El cliente primero abre el canal de comunicación y después conecta con la dirección del servidor. En ese momento realiza la petición de servicio necesaria y se dispone a esperar una respuesta del servidor. Cuando le haya llegado la respuesta la procesa de la manera que se requiera y finaliza el proceso cliente.

2.2. Python

El lenguaje elegido para desarrollar este Proyecto de Fin de Carrera ha sido Python por diversas razones intrínsecas a las características de las que se hablarán a continuación.

Python es un lenguaje de programación interpretado con un nivel de abstracción alto y orientado a una sintaxis clara para disponer de un código legible. Además admite varios paradigmas de programación como programación orientada a objetos, programación imperativa y programación funcional.

Python fue creado a finales de los ochenta por Guido van Rossum en los Países Bajos. A lo largo de los años se ha ido perfeccionando añadiendo funcionalidad que tenían otros lenguajes de programación de por aquel entonces pero sin perder la esencia del lenguaje original.
Además Python es un ejemplo de FLOSS (Free/Libre and Open Source Software), de modo que pueden distribuirse copias, leer su código fuente e incluso hacerle cambios o usar partes de él en nuevos programas libres, un concepto con el que se desea compartir conocimiento sin ataduras y una de las razones por las que tanto ha proliferado.

Una de las características fundamentales de Python es la encapsulación que ofrece, el nivel de abstracción con el cual no tenemos que preocuparnos de cómo se distribuyen los datos en la memoria ya que Python se encarga de manejarla por sí mismo.
Otra es que es un lenguaje interpretado, esto quiere decir que no necesitamos un proceso de compilación separada de ejecución, se ejecuta directamente desde el código fuente. Internamente Python convierte ese código de forma que se pueda ejecutar independientemente del lenguaje nativo de la computadora al que lo traduce, y este hecho supone una gran portabilidad en lo que al lenguaje respecta, ya que podemos copiar nuestro código a otros ordenadores y directamente ejecutarlo.
Las librerías de Python son múltiples y permiten incluso la inclusión de bases de datos, XML u otras herramientas complejas que se puedan fusionar en la aplicación que estemos desarrollando.

Por tanto si pensamos en realizar un proyecto como el que nos compete es razonable pensar en Python como primera opción, por encima de otros lenguajes orientados a objetos como JAVA.

2.3. Django

Muy relacionado con Python está el framework de desarrollo web de código abierto Django. Esto es así porque Django está escrito en Python en todas las partes del framework, desde las configuraciones hasta los modelos de datos.

Se ha elegido Django debido a que necesitamos que nuestra aplicación soporte un gran número potencial de personas, al menos todos los alumnos y profesores de la Universidad Rey Juan Carlos. Por tanto necesitábamos una herramienta más potente que simplemente

usando el lenguaje de programación Python.

Django nació en principio con el objetivo de gestionar páginas orientadas a noticias basado en el paradigma MVC (Modelo Vista Controlador), pero después se liberó al público bajo una licencia BSD en 2005. Las características de Django dejan claro el uso que tuvo en sus inicios ya que está orientado al desarrollo rápido de páginas de contenidos, incluso con una interfaz de administración ya preestablecida que permite la creación de objetos de contenido y llevando un registro de los mismos. Además ya proporciona una interfaz de administración de usuarios y grupos de usuarios otorgándoles permisos.

Django también se centra en la reutilización de funciones y código, evitando la replicación siempre que sea posible bajo el lema DRY (Don't Repeat Yourself) poniendo énfasis en la conectividad y extensibilidad de componentes.

Algunas características de este framework de desarrollo web se indican a continuación:

- Asocia peticiones de URLs con el código que maneja esas peticiones de manera óptima.
- Facilita el manejo de formularios HTML así como su validación.
- Convierte los datos de los formularios a lenguaje Python para su correcta manipulación.
- Separa la parte más visual de un sitio web mediante plantillas sin afectar al contenido.
- Se integra con otras herramientas como bases de datos o HTTP de la mejor forma posible.
- Posee mecanismos de seguridad como XSS o protección CSRF.
- Tiene soporte de internacionalización integrado.

Todo esto más la necesidad de cumplir con los objetivos y motivaciones expuestos en el primer capítulo de la memoria, hacen que Django sea una herramienta que encaje perfectamente con nuestros propósitos.

2.4. HTML

HTML o HyperText Markup Language ("Lenguaje de marcado hipertextual") hace referencia al lenguaje utilizado para la creación de páginas web. Estandarizado por la W3C, HTML se basa en el concepto de referenciar elementos, desde imágenes hasta scripts. HTML se encarga de asociar mediante texto la ubicación de los mismos para que el navegador pueda unir todos los elementos y formar así la estructura de las páginas web que visualizamos normalmente. La primera versión de HTML disponible públicamente data de 1991, por Tim Berners-Lee, y fue un documento llamado HTML Tags formado por 22 elementos que describían la simpleza y el diseño inicial de HTML.

La forma de un documento HTML se destaca por los corchetes angulares (<,>) que hacen referencia a las etiquetas en las que se basa. A continuación se describen dos características fundamentales de este lenguaje:

- **Elementos.**
  Los elementos se componen de dos partes: los atributos y el contenido. Mientras que los atributos están incluidos dentro de las etiquetas, los contenidos se introducen entre etiquetas que delimitan su dimensión, aunque también existen algunas etiquetas que no necesitan contenido ya que tienen significado por sí mismas.
  Sin embargo las etiquetas normalmente están formadas por una de inicio y una de fin, y de este modo se pueden distinguir varios tipos de marcado según lo que hagan estas etiquetas:

  - *Marcado estructural:* éste describe el propósito del texto, hace referencia al nivel de texto al que nos estamos refiriendo, si son títulos o encabezados más importantes para que los navegadores puedan reconocerlos y, si pueden, aplicarles un formato determinado.
  - *Marcado presentacional:* éste está más relacionado con el estilo del texto, comúnmente ligado a las hojas de estilo ya que describe la apariencia del texto. Aquí se incluyen todas las etiquetas que cambian el texto a negrita, itálica o que le dan énfasis. Cada vez más en desuso ya que el lenguaje avanza y las hojas de estilo nos ayudan a diseñar la apariencia de las páginas web de manera más eficiente.
  - *Marcado hipertextual:* los enlaces con otras páginas o con partes del mismo se realizan mediante este marcado, caracterizado fundamentalmente por la etiqueta '<a>' junto con el atributo 'href' que describirá la dirección a la que

apunta el enlace.

- **Atributos.**
  Por otra parte tenemos los atributos, que ya se han empezado a nombrar más arriba. Éstos son intrínsecos a las etiquetas y tienen unos nombres determinados, no podemos asignar cualquier atributo. Están definidos mediante un par nombre-valor en el que el valor en muchos casos pertenece también a un diccionario finito de opciones dependiendo del atributo que estemos cambiando.

Si analizamos un poco la estructura básica de una página HTML podemos ver el siguiente esquema:

```
<html>
        <head>
                <title>

                </title>
        </head>

        <body>

        </body>
</html>
```

Podemos comprobar que todo el documento está distribuido entre las etiquetas *html* que se colocan al inicio y al final del documento para denotar que estamos desarrollando una página en este lenguaje.
Lo incluido entre las etiquetas *head* no afecta a la visualización de la página en sí, es donde van los links a hojas de estilo, la información oculta o el título de la página.
La etiqueta *title* es la encargada de mostrar un nombre de página en el navegador que estemos usando.
Dentro de las etiquetas *body* es donde va el grueso de nuestra página, toda la información que queramos que se muestre en el navegador y donde mayormente se trabaja cuando se realizan páginas en HTML.

Antes de usar HTML existían dos técnicas para vincular documentos electrónicos: los hipervínculos o enlaces y el lenguaje de etiquetas SGML. De éste último deriva el lenguaje HTML y viendo el éxito que se obtuvo debido a su sencillez y a la internacionalización del mismo, de modo que todos los navegadores comenzaron a usarlo, surgió la necesidad de evolucionar el mismo.

Primero se empezó a desarrollar HTML+ como una evolución gradual del lenguaje, pero no llegó a cuajar como estándar y se tuvo que esperar hasta que apareció HTML 3.0 en 1995. Con éste se introdujeron muchas mejoras como facilidades para la creación de tablas o la capacidad de mostrar elementos matemáticos complejos.
HTML 4.0 apareció en 1997 implementando características nuevas pero las tecnologías cada vez avanzaban más rápido, y la intrusión de los dispositivos móviles hicieron que fuera necesaria la evolución a nuevo tipo de HTML, lo que nos lleva a la aparición de HTML5.

2.4.1. HTML5

HTML5 es la quinta revisión del lenguaje de marcado HTML. Incluye numerosas mejoras que merecen tener un apartado propio en esta memoria, entre algunas de las cuales se distinguen:

- Simplifica el DOCTYPE que en anteriores versiones era mucho más largo para dejarlo en una simple línea:
  <!DOCTYPE html>

- Se pueden asociar imágenes a su correspondiente pie de imagen de una manera más sencilla gracias a la etiqueta *<figure>*.

- Nuevo tipo de input para emails para admitir sólo direcciones de email válidas.

- Nuevos atributos como *required* o *autofocus* que nos ayudan aún más a lidiar con los formularios de las páginas web.

- En consonancia con los formularios también se han añadido expresiones regulares con el atributo *pattern*.

- Numerosos avances derivados de las necesidades que estaban cubiertas con Javascript como el atributo *placeholder* que nos rellena cajas de formularios con un texto hasta que comenzamos a escribir en ellas.

- Un cambio organizativo en cómo se distribuye el cuerpo de la página mediante las etiquetas de *<header>*,*<nav>* y *<footer>*.

- Una de las características más destacables es la mejora en la que se da soporte para audio y vídeo. Para evitar el uso de plugins y librerías externas, en esta versión de HTML se han añadido los elementos de *<audio>* y *<video>*, que los nuevos navegadores ya están añadiendo para poder soportar estos elementos.

- Para finalizar cabe destacar que HTML5 se convierte en una plataforma de APIs para integrar funciones complejas como geolocalización o acciones de tipo deslizar y soltar (más orientadas a dispositivos móviles).

Así podemos ver que HTML5 supone grandes cambios estructurales pero sin perder la esencia del lenguaje HTML inicial, simplemente ajustándose a las nuevas tendencias e integrando todo de una manera más sencilla y funcional.

2.4.2. JQuery Mobile

JQuery Mobile es un framework de desarrollo de aplicaciones web para dispositivos móviles basado en JQuery y la interfaz de usuario JQuery-UI.

Primeramente la finalidad de un framework es la de desarrollar software de forma correcta para que se pueda ver en todos los navegadores desde donde se esté visitando, incluso estar preparado para nuevos navegadores sin tener que cambiar el código inicial. Aparte también sirve para escribir menos código fuente y hacer cosas más complicadas de una manera sencilla.
JQuery Mobile va más allá, de modo que está preparado para el mundo al que nos vamos dirigiendo, un mundo más móvil repleto de smartphones y tablets, y por tanto más complejo, ya que cada uno tiene su sistema operativo y su navegador propio, y ofrece una experiencia en la que no haya errores de compatibilidad. Además comprime mucho más el código necesario para hacer cosas que con el básico jQuery.
El framework divide los tipos de compatibilidad de modo que se establecen tres categorías:

- **Grado A**: Para dispositivos con compatibilidad máxima y transiciones mejoradas basadas en AJAX. Entre ellos destacan: iOS, Android, Blackberry 6-7, Opera Mobile y navegadores de escritorio como Chrome, Firefox, Internet Explorer y Opera.
- **Grado B**: Dispositivos compatibles con una experiencia mejorada pero sin características de AJAX. Entre ellos se incluyen: Blackberry 5.0, Opera Mini y Nokia Symbian.
- **Grado C**: Sin experiencia HTML mejorada como para Smartphones más antiguos o Windows Mobile.

Así la principal característica de este framework es que nos ofrece herramientas diseñadas para simplificar el proceso de creación de páginas web para móviles, desde la propia escritura del código HTML hasta la creación de efectos dinámicos con Javascript.

Una de las necesidades básicas del framework es el desarrollo en HTML5. Debido a que usa las herramientas más novedosas del lenguaje estamos obligados a trabajar en esta versión.
Además está repleto de automatismos que nos ayudarán a tener una experiencia más cómoda como las transiciones entre páginas o la propia personalización de las mismas.
Sin duda una de las características fundamentales de los dispositivos móviles actuales son las pantallas táctiles, y era de esperar que JQuery Mobile estuviera preparado para ello, haciendo hincapié en la gestión de eventos de este estilo. Aunque este hecho no hace que se olvide de otros métodos de ingreso de datos como el cursor de los navegadores antiguos o el propio ratón del ordenador.
Por último cabe destacar la personalización de temas que ofrece el framework, desde la elección de algunos que vienen prediseñados hasta la creación de temas propios que encajen mejor con el entorno de la página.

Capítulo 3

Diseño e implementación del sistema

Hasta ahora hemos estado exponiendo las motivaciones y objetivos de este Proyecto de Fin de Carrera, así como descrito las herramientas fundamentales que vamos a usar en el mismo. Este capítulo está dedicado enteramente al propio proyecto, comenzando con el diseño del mismo y acabando con la implementación, todo ello explicado de manera concisa y desvelando todos los detalles.

3.1. Introducción

Como buen método de desarrollo, al igual que dividir el objetivo fundamental en varios subobjetivos, también es bueno analizar el problema y buscar los elementos necesarios para poder desarrollarlo de manera óptima.

El primer elemento que nos compete es la información. Para manejar esta información, los eventos y actividades universitarias junto con los usuarios, necesitamos un soporte que almacene esos datos. De ahí surge la necesidad de tener una base de datos en la que guardar la información y que se pueda acceder a ella de manera consistente.
Enlazado con la base de datos debemos tener métodos de extracción de información y la posibilidad de mostrarla de manera clara. Por tanto necesitamos de una interfaz por la que se pueda acceder a los datos y que a la vez los muestre. Como en este caso buscamos hacer una aplicación web para que se acceda a la información desde cualquier lugar, el tipo de interfaz que necesitamos es una interfaz web.
Para dar soporte a estos dos elementos los unimos dentro de un servidor, que ofrece el acceso desde la web hacia los datos gracias a nuestra interfaz web.
Por otro lado existen los usuarios que quieran ver la información y acceder a los datos que precisen, algo que denominamos clientes. Éstos se conectan con el servidor mediante la interfaz web y así tienen la disponibilidad de los datos deseada.

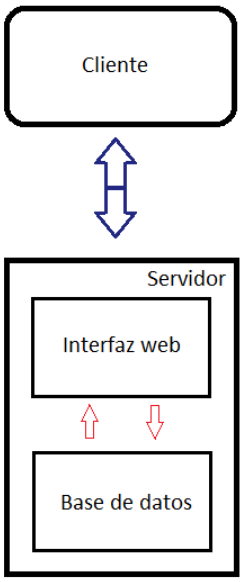

Figura 3.1: Esquema general de diseño del sistema

En la Figura 3.1 podemos ver todo el esquema que se ha ido especificando de una manera más gráfica. Aunque es una visión básica ya podemos ver los dos componentes principales y los flujos de información que hay entre los mismos.

3.2. Diseño del servidor

Para poder analizar el diseño del servidor es necesario primero separar el diseño en distintas partes para ayudar a su correcta comprensión.
Por una parte procederemos a analizar el diseño del modelo de datos del servidor y por otro su correspondiente interfaz para mostrar la información.

3.2.1. Diseño del modelo de datos

El modelo de datos es parte fundamental para el diseño de la base de datos, ya que la estructura de la misma depende de las consideraciones que tomemos a la hora de crearla y una decisión errónea en este punto podría derivar en la reestructuración de la base de datos por entero.
Así las especificaciones iniciales para su creación dependen de varios factores que se enumeran a continuación:

- **Distintos usuarios de la aplicación: alumnos y profesores.**
  Van a existir dos 'roles' fundamentales a la hora de crear usuarios en nuestro siste-

ma, a parte del administrador. Por un lado se encuentran los alumnos que tienen la necesidad de visualizar la información ordenada en listas y la posibilidad de apuntarse a ellas quedando así personalizada su experiencia.
Por otro lado existe el 'rol' de profesor, que aparte de contener las posibilidades de los alumnos suma a sus competencias la creación de listas y eventos, y también la gestión de los mismos.
Así podemos ver claramente que mientras que los usuarios de tipo alumno sólo van a poder disponer de una parte de la información, los de tipo profesor pueden acceder a toda la información de la base de datos excepto únicamente a la creación de usuarios nuevos o la gestión de los mismos. Para esto último aparece la figura del administrador, encargado de dar de alta en el sistema a los profesores que lo requieran y de gestionar la información de manera interna, así como controlar el buen funcionamiento de la base de datos.

- **Actividades o eventos universitarios.**
  La información principal que almacenará nuestra base de datos es la que concierne a las actividades o eventos universitarios. Ésta se dividirá en varios atributos relativos al lugar, la fecha o la duración de los mismos.

- **Listas de eventos relativas a la información que contienen.**
  La forma de organizar estos eventos será en forma de listas asociadas a la información que contengan. Esto supone que un evento puede incluirse en una o varias listas, dependiendo de cuántos temas conlleve su existencia. Además las listas se podrán crear posteriormente a la creación de eventos y no serán prediseñadas, con lo cual nos guardamos la posibilidad de añadir eventos antiguos a listas de reciente creación sin tener que renunciar a pertenecer a otras listas más antiguas.

- **Seguridad para la información de la base de datos.**
  Es importante separar la base de datos de cualquier intento de acceso por parte de usuarios extraviados o malintencionados. Por tanto es necesaria la inclusión de mecanismos de seguridad que aseguren la integridad de la información y que su acceso sea restringido en función del usuario que intente acceder.

Así, una vez expuestas las necesidades básicas de nuestro modelo de datos, podemos ver en la Figura 3.2 cómo podría resultar un diagrama relacional del diseño que se ha llevado a cabo en la base de datos.

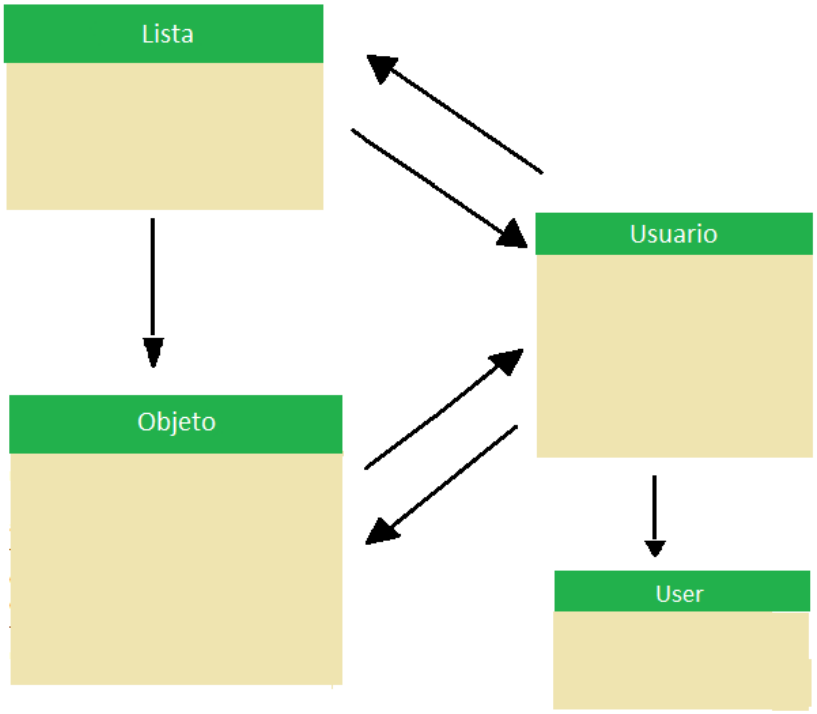

Figura 3.2: Diagrama relacional de la base de datos

Después de ver el diagrama relacional hemos de elegir un correcto gestor de bases de datos. Como el servidor que hemos elegido es Django, éste nos ofrece varias posibilidades de gestores de bases de datos integrados: PostgreSQL, MySQL, SQLite y Oracle.
Como el proyecto que queremos realizar deseamos que sea de tipo software libre, debemos descartar a Oracle como gestor ya que no es de código abierto y no cumple así con esta necesidad. Por otro lado los gestores PostgreSQL y MySQL son muy competentes para manejar gran volumen de información de la mejor manera posible, pero una de las razones fundamentales por la que se ha elegido SQLite3 en vez de otras opciones más potentes, es que el proyecto está encaminado a comprobar la funcionalidad mediante una fase de pruebas en las que no se espera gran concurrencia de usuarios ni gran cantidad de ellos. Por tanto si se desea ampliar el número de usuarios, o tener una base de datos más robusta, deberíamos valorar otras opciones como MySQL.

Por último comentar unas cuantas características muy básicas del gestor que hemos elegido: SQLite3.

- **Portabilidad:** ya que Python y Django nos ofrecían la portabilidad, era necesario que la base de datos la incluyera como característica.

- **Multiplataforma:** igual que en el punto anterior la multiplataforma era una característica del propio servidor con el que trabajamos.
- **Sin configuración:** la propia base de datos es el fichero, y no se necesita más que acceder al mismo para extraer la información.
- **Transacciones ACID:** Atomicidad, Consistencia, Aislamiento (Isolation) y Durabilidad, básicos en las bases de datos competentes.

3.2.2. Diseño de la aplicación web

Una vez desarrollado el diseño del modelo de datos debemos pasar al diseño de la aplicación web, pero para ello primero debemos explicar la arquitectura MVC (Modelo Vista Controlador). Una vez explicado genéricamente el concepto procederemos a ver la arquitectura asociada a nuestro proyecto.

La arquitectura MVC es un patrón de arquitectura de software basado en la división de los datos y la lógica de negocio proponiendo la construcción de tres conceptos:

- **Modelo.**
  Es el encargado de la representación de la información del sistema, teniendo en cuenta las restricciones de acceso que se hayan descrito en la lógica de negocio de la aplicación.
- **Vista.**
  Es el encargado de la visualización de la información de una manera adecuada para la interacción con el usuario, también llamada interfaz de usuario.
- **Controlador.**
  Parte fundamental que se encarga de gestionar los eventos solicitados por el usuario y de hacer peticiones al 'modelo' cuando se hacen solicitudes sobre la información. Es el intermediario entre la 'vista' y el 'modelo'.

En la Figura 3.3 podemos ver estos elementos de manera más visual, señalando que las líneas naranjas indican una asociación directa entre elementos y las azules una asociación indirecta.

Esta arquitectura introducida por Trygve Reenskaug a finales de los 70 ha dado lugar a numerosas variantes de la misma, cada una surgiendo de las nuevas necesidades que aparecían a lo largo del tiempo. Pero para nosotros se adapta perfectamente al proyecto con el que trabajamos ya que se basa en la reutilización de código y la separación de

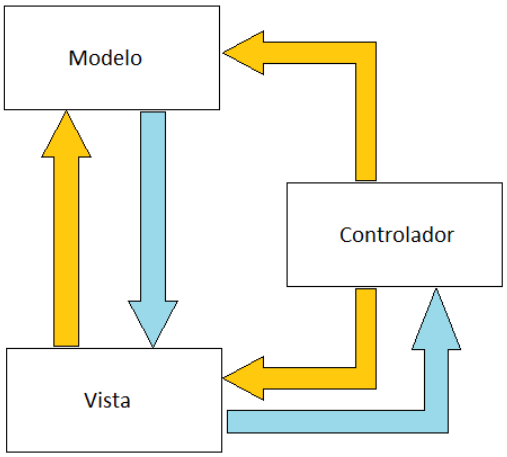

Figura 3.3: Diagrama de relaciones de la arquitectura MVC

conceptos. Así procederemos a explicar más en detalle las partes correspondientes de la arquitectura MVC aplicadas a este Proyecto de Fin de Carrera.

3.2.2.1. Modelo

Comenzaremos analizando el modelo de la aplicación correspondiente a nuestro proyecto. En esta parte se desarrollarán las clases y objetos necesarios para cumplir con los objetivos y los requisitos de diseño anteriormente expuestos. Podemos comprobar que se ajustan considerablemente a los mismos:

- **Objeto.**
  Clase fundamental que almacena la información más básica relativa a los eventos y actividades universitarias. Los campos que deberá contener son los siguientes: nombre del evento, lugar y aula asociados al mismo, fecha del evento, así como su duración, una descripción para aclarar el contenido del evento, el tiempo en que ha sido registrado y el usuario responsable de ese evento.

- **Lista.**
  La lista es el elemento que recoge todos los eventos o actividades relativos a un tema específico. Los campos que deberá contener son los siguientes: nombre de la lista, una pequeña descripción de la misma, el tiempo en el que ha sido registrada, un usuario responsable de la creación de la lista y todos los eventos que se asocien a esa lista.

- **Usuario.**
  Por último tenemos el tipo usuario que almacenará todo lo concerniente a los usuarios que hacen uso de la aplicación. En este caso tendremos que diferenciar entre dos tipos de usuarios: alumno y profesor, por tanto cada uno tendrá un rol distinto. Los campos que deberá contener son los siguientes: un tipo usuario asociado a la base de datos existente, un alias identificativo, el nombre completo del usuario, un correo electrónico para poder contactar con él, el rol de usuario (profesor o alumno), si es profesor la posibilidad de que sea un profesor activo para darle de alta en los permisos correspondientes, y las listas o eventos asociadas a los usuarios.

3.2.2.2. Vista

Una vez expuesto el modelo procederemos a analizar el siguiente elemento de la arquitectura MVC: la vista. La vista de nuestra aplicación está compuesta por todas las interfaces de usuario necesarias para la visualización de la información a los potenciales usuarios. En la siguiente lista se enumeran los requisitos necesarios que debe cumplir la interfaz:

- **Interfaz de administración.**
  Primeramente para poder administrar los usuarios de manera correcta y dar privilegios según el rol que tengan, necesitamos una interfaz para el administrador. Éste se encargará además de la correcta utilización de la aplicación y de añadir, editar o borrar eventos o listas si es necesario. Es conveniente añadir que a la interfaz de administración sólo puede acceder un administrador autorizado, los profesores tendrán su propia forma de añadir y gestionar eventos y listas.

- **Página de visualización.**
  El entrar en nuestra interfaz web lo primero que deberíamos ver es la información disponible de manera clara para todos los usuarios. Así se deberán mostrar los eventos en sus distintas listas para todo el público, ofreciendo la posibilidad de entrar al sistema para disfrutar de una experiencia más personalizada.

- **Página de acceso.**
  Si un usuario quiere entrar en el sistema necesita de un pequeño formulario para comprobar sus credenciales. Para ello es necesaria la creación de una interfaz que permita el acceso al mismo o la posibilidad de registrarse en el sistema para posteriores accesos.

- **Página de registro.**

  Si un usuario no está incluido en el sistema, debemos de ofrecer la posibilidad de registrarlo. Por eso, es necesaria la creación de una interfaz que permita la creación de nuevos usuarios con la correspondiente información necesaria, incluido el rol que quiere desempeñar en nuestro sistema: alumno o profesor.

- **Listas personalizadas.**

  Para personalizar más la experiencia de los usuarios se debe añadir una página que recoja las listas según la información que le interesa a cada usuario. También es necesaria la posibilidad de añadir o borrar listas de esa página de manera sencilla.

- **Administración de listas.**

  Cuando un profesor desee crear una nueva lista o ver las que ya existen desde el punto de vista administrativo, necesita una página en la que pueda realizar estas actividades. Por tanto una interfaz de creación de listas y de visualización de las mismas es necesaria, tanto como su correspondiente seguridad para que no accedan todo tipo de usuarios.

- **Edición de listas.**

  Si ya hemos creado una lista, la posibilidad de editarla o borrarla si ya no se necesita es algo que debemos contemplar. Así la página de edición de listas para profesores también es necesaria.

- **Inclusión o borrado de eventos en listas.**

  Para que tenga sentido la información con la que trabajamos, hemos decidido agrupar eventos en listas de manera que estén conectadas según la información que contengan. Debido a esta necesidad se ofrece la posibilidad de la inclusión o borrado de eventos de cada profesor en las listas ya existentes o en listas nuevas que cree el profesor asociado.

- **Administración de eventos.**

  Al igual que ocurre con las listas, la información básica de nuestro proyecto que son los eventos o actividades universitarias, necesita de un proceso de creación. Por tanto debemos tener una página en la que se puedan crear eventos y visualizar los pertenecientes al profesor que esté usando el sistema.

- **Edición de eventos.**

  Debido a que la información de los eventos es lo más importante que gestionar en nuestra aplicación, se necesita la posibilidad de volver a editar un evento ya creado

o incluso el borrado del mismo. Además los eventos sujetos a un horario, un lugar o un día determinado están más expuestos a cambios y se hace totalmente necesario que se puedan volver a editar.

3.2.2.3. Controlador

Por último, el elemento que queda por analizar de la arquitectura MVC es el controlador, que dará la lógica al servidor. A continuación se enumeran los controladores principales de los que debemos disponer con una pequeña descripción sobre lo que tienen que encargarse:

- **Controlador de acceso.**
  Aquel encargado de la gestión de usuarios, tanto concerniente al acceso de los mismos, como al registro de nuevos usuarios.

- **Controlador de personalización.**
  Debido a que cada usuario tiene un interés distinto en ver la información que cada uno desee, es necesario el control de la visualización de listas y eventos según el usuario que acceda a la aplicación.

- **Controlador de administración.**
  Al igual que se ha nombrado para la vista una interfaz de administración, debe existir un controlador que gestione lo concerniente a los administradores.

- **Controlador de eventos.**
  Debido a que los eventos o actividades tienen una información crucial, es necesario un controlador que gestione la correcta inclusión de información de los mismos así como la visualización de esta información según el usuario que la requiera.

- **Controlador de listas.**
  Al igual que ocurre con los eventos, las listas deben ser controladas en su proceso de creación y sobre todo en el proceso de inclusión o borrado de esos eventos en las listas correspondientes.

3.3. Implementación del servidor

Una vez desarrollado toda la parte del diseño de nuestro Proyecto de Fin de Carrera, vamos a continuar con la implementación del mismo.
Como el proceso de desarrollo de la aplicación es largo, y para establecer un orden, dividiremos la implementación en varias partes.

3.3.1. Estructura de directorios y ficheros

Para comenzar vamos a analizar la estructura de archivos de nuestro proyecto que se puede ver en la Figura 3.4.

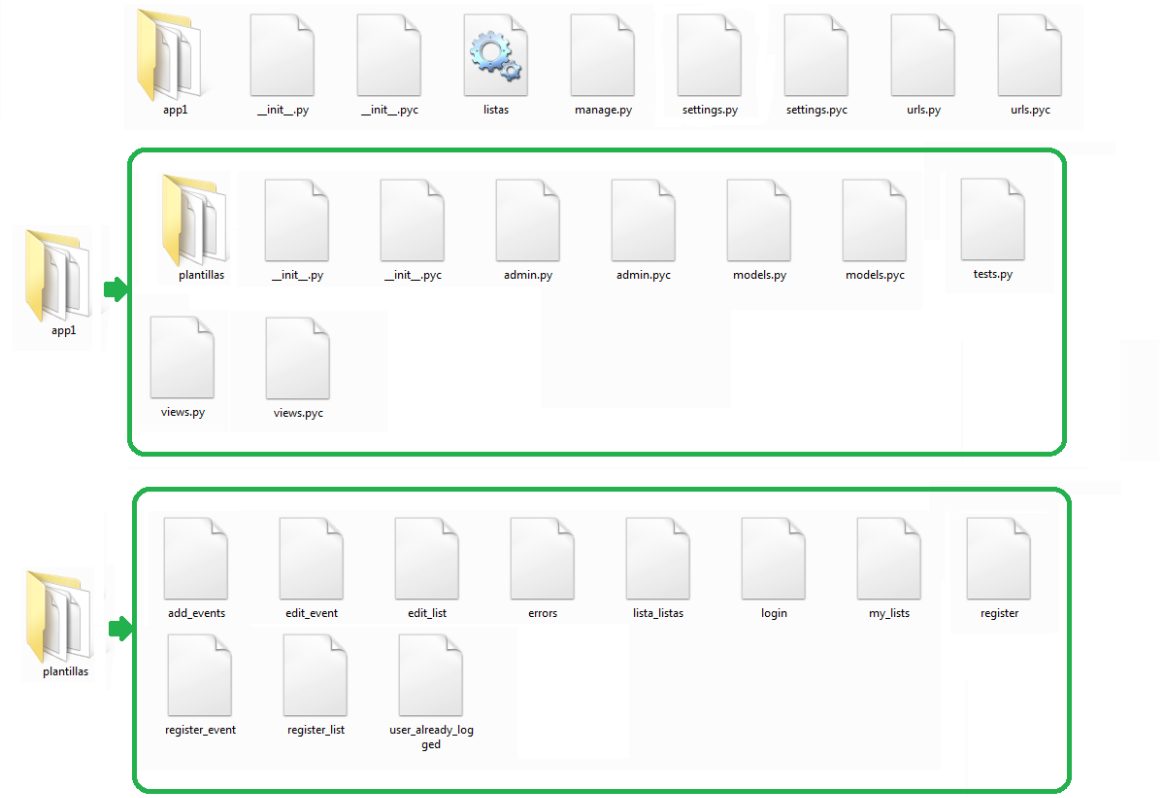

Figura 3.4: Estructura de directorios y ficheros del proyecto Django implementado

El servidor que se ha desarrollado se basa en el framework de desarrollo web Django, que tiene una estructura de directorios y ficheros asociada a lo forma de trabajar del mismo. Para poder entender mejor la implementación es necesario introducir brevemente los ficheros y directorios más importantes:

- **plantillas**

  En este directorio se almacenan todas las vistas correspondientes a la arquitectura MVC, asociadas a su vez con la interfaz gráfica que une cliente con servidor. Se puede observar que está compuesto de múltiples archivos, todos ellos con extensión '.html'.

- **app1**

  En esta carpeta se almacena toda la lógica de la aplicación, incluyendo los modelos de la arquitectura MVC y los controladores de la misma. Además está el fichero

correspondiente al administrador, que da permisos y almacena todo lo concerniente al panel de administración y el acceso a la base de datos.

- **listas.db**
  Este archivo es el correspondiente a la base de datos, donde vamos guardando toda la información relacionada con nuestra aplicación: desde los eventos hasta la personalización de los usuarios.

- **settings.py**
  Este fichero es el encargado de la configuración de nuestro servidor: se elige la base de datos a utilizar, las rutas de directorios de las plantillas, la zona horaria, el idioma, las aplicaciones instaladas, etc.

- **manage.py**
  Es el archivo de arranque de la aplicación, al cual se le pueden pasar distintos comandos para realizar distintos procedimientos como resetear la base de datos o iniciar la aplicación en un puerto determinado.

- **urls.py**
  Este fichero es el encargado de almacenar las URLs que dan acceso a nuestra aplicación web. A continuación se enumeran las URLs con una breve descripción sobre a donde nos dirigen (teniendo en cuenta que la dirección del servidor es la que se ha usado para la fase de pruebas):

  - *http://pfc-cdfuente.libresoft.es*: lleva a la página inicial de la aplicación independientemente del tipo de usuario que la requiera.
  - *http://pfc-cdfuente.libresoft.es/my_lists*: lleva a la listas personalizadas del usuario que esté autenticado.
  - *http://pfc-cdfuente.libresoft.es/register*: para poder registrar nuevos usuarios en el sistema.
  - *http://pfc-cdfuente.libresoft.es/register_event*: para poder crear nuevos eventos en el sistema.
  - *http://pfc-cdfuente.libresoft.es/register_list*: para poder crear nuevas listas en el sistema.
  - *http://pfc-cdfuente.libresoft.es/login*: para poder autenticarse en el sistema.
  - *http://pfc-cdfuente.libresoft.es/logout*: para salir del sistema.

- *http://pfc-cdfuente.libresoft.es/admin/*: para acceder a la interfaz de administración del sistema.

3.3.2. Implementación del Modelo Vista Controlador

Después de haber analizado la estructura de ficheros debemos de entrar más en detalle en los elementos de la arquitectura MVC asociados a nuestro proyecto. Primero se explicaran todos los modelos implementados, incluidos los que vienen ya en el framework de desarrollo web Django, a continuación se explicarán los controladores asociándolos con los descritos en el diseño de la aplicación y por último se analizarán todas las vistas según la interfaz web que hemos creado.

3.3.2.1. Modelos

Rememorando la estructura de ficheros de la Figura 3.4 veíamos que los modelos estaban incluidos en la carpeta app1, más concretamente en el archivo 'models.py'.
Debido a que Django es un framework de desarrollo web muy potente, nos ofrece grandes posibilidades en lo relativo a la generación de los modelos de datos, así como facilidades para su posterior acceso, borrado o modificación. Además las tablas de la base de datos son generadas automáticamente, de forma transparente para el desarrollador, con lo que aporta un nivel de encapsulación perfecto para centrarnos en lo que el desarrollador tenga en mente.
A continuación se describirán algunos de los modelos de datos que ya implementa Django por su cuenta para el correcto funcionamiento de los proyectos:

- **django.contrib.auth**
  Elemento dedicado a la autenticación de usuarios en el sistema manejando cuentas, grupos, permisos y sesiones basadas en cookies. Así con la máxima abstracción posible nos ofrece la creación de tablas que almacenan toda la información necesaria para poder tener usuarios fácilmente.

- **django.contrib.contenttypes**
  Para poder crear nuestros propios modelos, necesitamos un soporte de trabajo y de eso se encarga este elemento, ofreciendo una interfaz genérica que permite diferenciar el tipo de contenido que queramos crear en nuestros propios modelos.

- **django.contrib.sessions**
  Django nos ofrece sesiones de usuarios anónimos mediante el uso de cookies. Las cookies relativas a cada usuario único se almacenan en una tabla de forma totalmente

transparente para el usuario y para el desarrollador, asociando un identificador de sesión para el usuario.

- **django.contrib.sites**
  Heredado del objetivo con el que se creó Django, nos ofrece también herramientas asociadas a la creación de páginas web de contenidos. En este caso podemos asociar nombres de dominio y direcciones web de nuestro sitio web, tomando más importancia cuando se requieren varios sitios web.

- **django.contrib.messages**
  También basado en el uso de cookies Django nos ofrece la posibilidad de mostrar mensajes de notificación cuando se ha hecho alguna petición al servidor como por ejemplo el proceso de rellenar un formulario. Estos mensajes están divididos según su nivel de prioridad como de información, alarma o error entre otros.

- **django.contrib.admin**
  Una de las herramientas más poderosas con las que cuenta este framework de desarrollo web es la posibilidad de insertar directamente una interfaz de administración. Este módulo se encarga precisamente de eso: administra los metadatos de nuestros modelos y nos ofrece la posibilidad de incorporar contenido al servidor web desde la interfaz de administración.

Todos estos módulos se ven reflejados en un apartado del archivo 'settings.py' llamado 'INSTALLED_APPS', a los que se suma nuestro propio módulo creado en función de nuestro diseño, en este caso nuestra 'app1'.
Cabe la posibilidad de añadir más modelos de datos que se corresponderían con nuevos módulos pero en este Proyecto de Fin de Carrera sólo se ha visto necesaria la incorporación de uno.

A continuación se desarrollará más a fondo el contenido de nuestro modelo de datos, que está sujeto a las restricciones de diseño que planteamos anteriormente:

- **Objeto.**
  Modelo encargado de gestionar la información relativa a eventos o actividades universitarias. Los campos que contiene son:

  - *nombre: CharField.* Nombre del evento o actividad en cuestión. Contiene una restricción máxima de 50 caracteres.

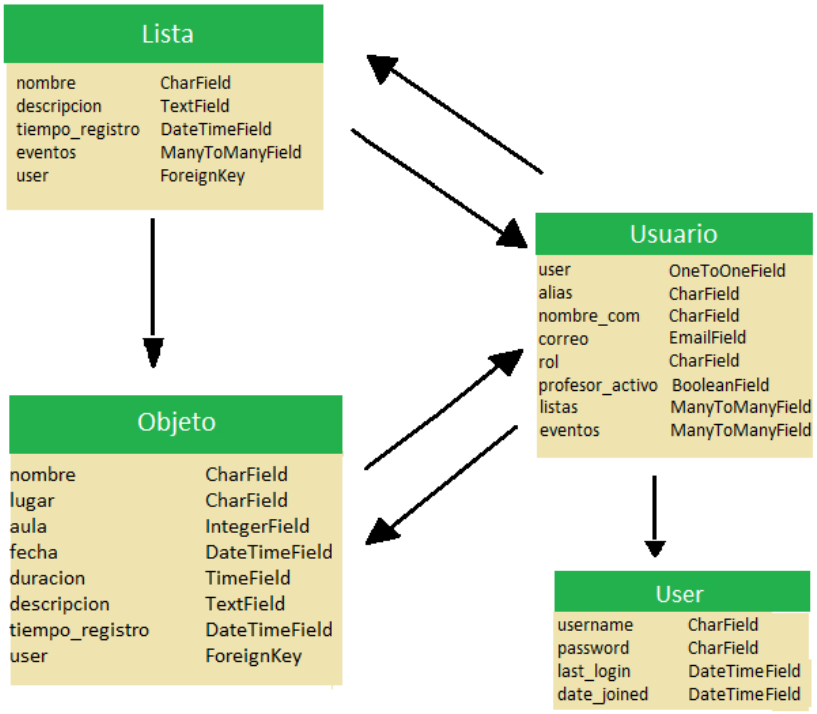

Figura 3.5: Diagrama relacional de los modelos desarrollados en Django

- *lugar: CharField.* Lugar destinado a la realización del evento. Contiene una restricción máxima de 50 caracteres.
- *aula: IntegerField.* Campo asociado al aula en la que se realizará el evento mediante un número identificativo.
- *fecha: DateTimeField.* Fecha de la realización del evento compuesta por día, mes, año y hora.
- *duracion: TimeField.* Cantidad de tiempo que se prevé que vaya a durar el evento en forma de horas y minutos.
- *descripcion: TextField.* Descripción de las actividades o eventos, lugar en el que introducir toda la información relativa a los mismos. Contiene una restricción máxima de 800 caracteres.
- *tiempo_registro: DateTimeField.* Momento exacto de la creación del evento que además se guarda automáticamente.
- *user: ForeignKey.* Usuario responsable de la creación del evento.

- **Lista.** Modelo que se encarga de asociar eventos según la información que contengan. Los campos que contiene son:

- *nombre: CharField.* Nombre de la lista. Contiene una restricción máxima de 50 caracteres.
- *descripcion: TextField.* Texto relativo a la información que se va almacenar en esta lista. Contiene una restricción máxima de 800 caracteres.
- *tiempo_registro: DateTimeField.* Momento exacto de la creación de la lista que además se guarda automáticamente.
- *eventos: ManyToManyField.* Lista de eventos relativos a la información requerida por la lista. Se comporta igual que un campo de tipo ForeignKey pero asociando varios elementos en vez de uno solo.
- *user: ForeignKey.* Usuario responsable de la creación de la lista.

- **Usuario.** Modelo creado para la gestión de usuarios en el sistema. Cabe destacar que se relaciona con el modelo ya creado por Django: 'User', pero con una funcionalidad extra que veremos a continuación. Además para la creación de este modelo hemos tenido que crear un tipo de datos nuevo relativo al rol que va a desempeñar el usuario en el sistema. Los campos que contiene son:

  - *user: OneToOneField.* Para poder extender el modelo de 'User' creado por Django se usa la relación 'OneToOneField'. Este campo contiene todos los que contiene el modelo 'User' entre ellos un alias y una contraseña para autenticarse en el sistema.
  - *alias: CharField.* Alias de autenticación en el sistema. Contiene una restricción máxima de 20 caracteres.
  - *nombre_com: CharField.* Campo que recoge el nombre completo del usuario. Contiene una restricción máxima de 50 caracteres.
  - *correo: EmailField.* Email de contacto del usuario.
  - *rol: CharField.* Rol que va a desempeñar el usuario en el sistema. Para ello hemos creado un 'CharField' específico en el que hay dos opciones: 'ROL_PROFESOR' y 'ROL_ALUMNO', asociadas a dos letras 'p' y 'a' respectivamente. Contiene una restricción máxima de 1 carácter y está predefinido como rol de alumno.
  - *profesor_activo: BooleanField.* Campo destinado a dar permiso a un profesor para que pueda hacer uso de la creación de eventos y listas. Tiene como valor predefinido 'False'.
  - *listas: ManyToManyField.* Listas asociadas al usuario para mostrar a las que está apuntado.

- *eventos: ManyToManyField.* Eventos asociados al usuario para mostrar a los que está apuntado.

3.3.2.2. Controladores

Una vez vistos los modelos implementados necesitamos los controladores que manejen la lógica de la aplicación. Como vimos en la estructura de ficheros la parte de los controladores estaba contenida dentro de la carpeta 'app1', más concretamente en el fichero 'views.py'. En este fichero se han incluido todos los controladores necesarios que se habían expuesto en la fase de diseño.
Como sabemos los controladores se encargan de todas las funciones lógicas que son necesarias para el correcto desarrollo de la aplicación, ya sea concerniente a la generación de contenidos como el acceso a la base de datos. Además deben servir para conectar el modelo de datos con las vistas de la arquitectura MVC, por tanto unir la base de datos de alguna manera a la interfaz web desarrollada para la visualización por parte de los usuarios.
A continuación se analizará en detalle el fichero 'views.py' que contiene todos estos controladores separados en distintas funciones:

- **lista_listas(request)**
  Cuando un usuario hace uso por primera vez de la aplicación es dirigido hacia esta función. Ésta se encarga de hacer varias comprobaciones: primero analiza si el usuario ha sido validado en el sistema, y le pasa la información a la vista correspondiente; a parte recoge todas las listas y eventos y los muestra en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es*.
  Además si el usuario está dado de alta en el sistema le pasa las listas que está siguiendo de modo que pueda apuntarse a más listas o borrarse de la que quiera.
  Por último si el usuario es incorrecto o ha habido un fallo de seguridad le redirige a la página de errores correspondiente.

- **my_lists(request)**
  Debido a que queremos personalización de la experiencia es bueno tener la posibilidad de ver sólo las listas que le interesen al usuario. Este controlador se encarga precisamente de eso, de forma parecida al anterior, saca las listas del usuario y se las pasa a la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/my_lists*.
  Además como en el caso anterior se encarga de controlar qué usuario está en el sistema y derivarlo a una página de error si se detecta actividad sospechosa.

- **register(request)**

  Controlador encargado del registro de nuevos usuarios. Primero comprueba que no haya ya un usuario autenticado en el sistema y después se encarga del registro de los campos que nos hayan introducido así como de la comprobación de los mismos, de modo que no exista un usuario con anterioridad con las mismas credenciales y que todos los campos sean acordes al modelo de datos especificado. En el caso de que haya algún campo que esté vacío, sea erróneo o que el usuario ya exista, se devolverá un mensaje de error para que la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/register* pueda mostrarlo correctamente.

- **login(request)**

  En esta función se controla la correcta validación de un usuario en el sistema. Primero comprueba que no haya ningún usuario ya validado y después comprueba las credenciales a partir de la variable 'request'. Dependiendo del rol que tenga directamente los da de alta en el sistema y los redirige a la vista correspondiente. Si hay algún error, o en el caso de que el usuario sea de tipo profesor y todavía no esté autorizado a entrar, se informa a través de un mensaje de error en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/login*.

- **logout(request)**

  Controlador muy sencillo que simplemente se encarga de dar de baja a un usuario que esté validado en el sistema.

- **register_event(request)**

  Este controlador es uno de los más importantes y más complejos de la implementación ya que se encarga de todo lo concerniente a los eventos. Debido a que es tan complejo dividiremos sus funciones en varios puntos:

  - Por un lado primeramente comprueba el usuario que está validado en el sistema, ya que sólo los profesores autorizados son los que pueden acceder a la posibilidad de creación, edición o borrado de eventos.
  - Además recoge la información del profesor en cuestión y elabora una lista con todos los eventos pertenecientes al mismo para su correcta visualización en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/register_ event*.
  - Por otro lado se encarga de la creación de nuevos eventos, comprobando que los campos de información introducidos a través de la variable 'request' sean correctos y se ajusten debidamente al modelo de datos descrito en el apartado

anterior. En el caso de que no se correspondan con lo esperado avisará mediante un mensaje de error.

- Aparte también es el encargado de controlar la edición de eventos que ya haya creado en algún otro momento el usuario, comprobando, como se ha expuesto anteriormente, que todos los campos sean correctos. Estos campos se mostrarán en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/edit_event.*
- De la misma forma, este controlador se encarga del borrado de eventos del sistema, teniendo en cuenta que el profesor sólo puede borrar eventos que le pertenezcan, es decir que sólo puede borrar aquellos que él mismo haya creado en algún momento.

- **register_list(request)**

  Al igual que en el caso anterior, este controlador es de vital importancia para el correcto funcionamiento de la aplicación. En este caso es el encargado de las listas de eventos y como es igual de complejo lo dividiremos en varias funciones:

  - Por un lado primeramente comprueba el usuario que está validado en el sistema, ya que sólo los profesores autorizados son los que pueden acceder a la posibilidad de creación, edición o borrado de listas, así como añadir o borrar eventos de las mismas.
  - Además recoge la información del profesor en cuestión y elabora una lista con todas las listas pertenecientes al mismo para su correcta visualización en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/register_list.*
  - Por otro lado se encarga de la creación de nuevas listas, comprobando que los campos de información introducidos a través de la variable 'request' sean correctos y se ajusten debidamente al modelo de datos descrito en el apartado anterior. En el caso de que no se correspondan con lo esperado avisará mediante un mensaje de error.
  - Aparte también es el encargado de controlar la edición de listas que ya haya creado en algún otro momento el usuario, comprobando como se ha expuesto anteriormente que todos los campos sean correctos. Estos campos se mostrarán en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/edit_list.*
  - De la misma forma, este controlador se encarga del borrado de listas del sistema, teniendo en cuenta que el profesor sólo puede borrar listas que le pertenezcan, es decir que sólo puede borrar aquéllas que él mismo haya creado en algún momento.

- Por último este controlador es el encargado de añadir o borrar eventos de cualquier lista existente, ofreciendo así la posibilidad de añadir eventos propios en listas ajenas que coincidan con la información de las mismas. Esta posibilidad se mostrará en la vista asociada a la URL: *http://pfc-cdfuente.libresoft.es/add_events*, mostrando mensajes de error si los eventos ya estaban añadidos a la lista cuando se intentaban añadir de nuevo o borrar eventos que no estén incluidos en la lista correspondiente.

- **errores(request)**
  El último controlador implementado es aquél encargado de los errores producidos por el acceso a páginas inexistentes. Este controlador está asociado a la vista relacionada con la URL: *http://pfc-cdfuente.libresoft.es/errors*.

3.3.2.3. Vistas

Una vez analizados los modelos y controladores debemos finalizar con el último elemento de la arquitectura MVC: las vistas. Las vistas se centran en el desarrollo de la interfaz gráfica que ofrece el servidor para que los usuarios desde sus navegadores tengan una experiencia cómoda y se acceda al contenido de manera sencilla.
Si recordamos la estructura de ficheros de la que hemos hablado, vemos que las vistas se guardan en el directorio 'plantillas', el cual contiene archivos con extensión *html*. Cada uno de estos archivos corresponde a una vista distinta y a continuación analizaremos todos estos archivos de manera detallada:

- **lista_listas.html**

  Siguiendo con el orden que hemos ido viendo a lo largo de esta memoria, comenzaremos con lo que verá un usuario nuevo al llegar a la aplicación. Esta vista, que se puede observar en la Figura 3.6, está dividida en varias partes ayudado por la tecnología de JQuery Mobile.
  Por un lado tenemos en la parte superior el encabezado compuesto por el título de la página que hemos decidido llamar 'Inicio Listas' y una serie de botones de navegación. Estos botones estarán determinados por el tipo de usuario que esté validado o no en el sistema, de modo que un usuario anónimo tendrá los botones expuestos en la Figura 3.6. Si está validado como profesor se asociarán más botones ya que puede acceder a más funciones como se puede observar en la Figura 3.7. Si el rol del usuario fuera alumno, la única diferencia sería que los botones que puede ver serán 'Inicio', 'Mis Listas' y 'Salir'.

Figura 3.6: Captura de imagen de la vista lista_listas.html

Por otro lado en la parte inferior vemos que hay una especie de pie de página en la que se encuentra el botón por el cual se puede acceder a la valoración de la aplicación y una dirección de contacto por si se quiere contactar con el administrador.

Figura 3.7: Captura de imagen de la vista lista_listas.html con usuario validado

Por último la parte central de la vista se compone del grueso de la información. Se puede observar primeramente una barra de búsqueda para poder encontrar eventos de manera más cómoda, sobre todo si las listas se hacen muy voluminosas. Con

poner una palabra clave en este buscador, sin tener que apretar ningún botón, se acortará la lista ajustándose a esa palabra clave. Debajo podemos ver todas las listas con sus correspondientes eventos en el interior. Cabe destacar los distintos botones que disponemos: en los encabezados de las listas, si estamos autenticados en el sistema, se podrán observar dos tipos de botones: 'Seguir' y 'Abandonar'. Éstos son los encargados de personalizar la información de modo que al presionar sobre un botón de 'Seguir' inmediatamente pasa a ser parte de las listas seguidas por el usuario, que se podrán ver por separado en su vista personalizada correspondiente. Del mismo modo al presionar 'Abandonar' dejaremos de seguir esa lista y desaparecerá de la vista personalizada ubicada en 'Mis Listas'.
Aparte de esos dos botones, si centramos nuestra atención en los eventos, podemos ver que en la parte derecha hay un pequeño botón con el símbolo de un '+'. Si presionamos este botón abriremos un menú desplegable en el que aparece toda la información del evento seleccionado. Así en la Figura 3.8 se observa un ejemplo de un evento desplegado.

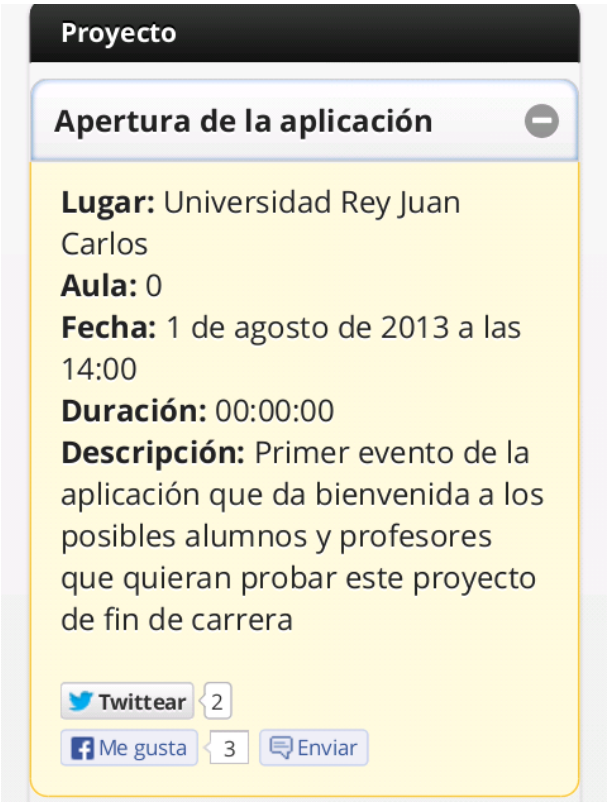

Figura 3.8: Captura en detalle de un evento expandido

En esta figura podemos ver toda la información de manera ordenada y aparte dos pequeños apartados en la parte inferior. Como queríamos cumplir con uno de los

objetivos de difusión más novedosos mediante las redes sociales, podemos ver este hecho reflejado en estos apartados. Primero se ha incluido un botón para twittear el evento en la red social Twitter, de modo que al presionar se nos abrirá una interfaz en la que tendremos que validar nuestras credenciales de Twitter y twittear el evento. Cabe destacar que al hacerlo se propone mencionar a la cuenta de Twitter de la propia aplicación, @ActividadesUrjc, creada explícitamente para el propósito del proyecto, junto con el nombre del evento, pero es totalmente personalizable.
Debajo de este botón tenemos la conexión con otra red social del momento: Facebook. El botón que aquí se ha colocado permite dar un 'Me gusta' a la página de la aplicación, de modo que pueda difundirse la dirección de la propia página en esta red social.

- **my_list.html**
  Muy relacionada con la vista anterior está la vista correspondiente a las listas personalizadas del usuario autenticado que podemos apreciar en la Figura 3.9. Accediendo a través de la URL: *http://pfc-cdfuente.libresoft.es/my_list* o presionando el botón correspondiente a 'Mis Listas' en los botones de navegación de la parte superior, en esta vista se ven las listas que el usuario ha decidido seguir. Podemos observar que sólo hay botones del estilo 'Abandonar', ya que aquí no están las que no sigue el usuario, y si presionáramos alguno de estos botones, sería de inmediato borrada la lista de sus preferencias, y por tanto de la vista asociada.

Figura 3.9: Captura de imagen de la vista my_lists.html

- **login.html**
  Hemos visto listas personalizadas por usuario pero no la opción de autenticarlo en el

sistema. Para esto existe el acceso mediante la URL: *http://pfc-cdfuente.libresoft.es/login* o el botón 'Entrar' de la página inicial.

En esta vista recogida en la Figura 3.10, se puede observar los campos 'Usuario' y 'Contraseña' además del botón de validación. Una vez introducidas las credenciales correctas y pulsado el botón 'Entrar', si son válidas, se le redirigirá la página inicial.

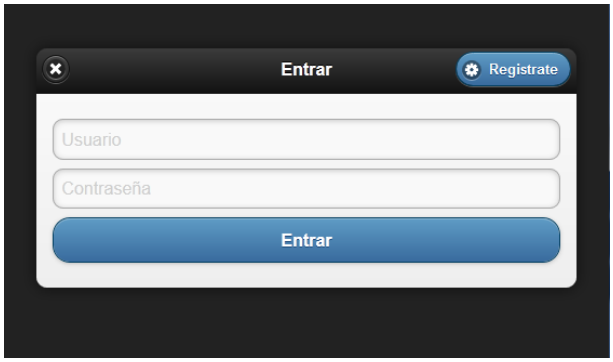

Figura 3.10: Captura de imagen de la vista login.html

Aparte podemos ver que el diseño de esta vista es ligeramente distinto a los anteriores. Además, aunque no se aprecia en la imagen, la transición a la misma también es distinta, de modo que al presionar el botón de entrar se abre como una especie de diálogo con su correspondiente botón de cerrar en la esquina superior izquierda. Si presionamos el botón de cerrar volveremos a la vista en la que nos encontrábamos antes.

Del mismo modo en la esquina superior derecha hay un botón de registro que nos redirige a la vista correspondiente al registro de nuevos usuarios, por si todavía no tenemos credenciales que validar en nuestro sistema.

Por último hay que aclarar que si las credenciales son erróneas al intentar validarlas, se redirigirá a una página de autenticación muy similar, pero ya sin en formato de diálogo, y en la que nos mostrará el error correspondiente.

- **register.html**

  Si un usuario quiere validarse en el sistema pero todavía no dispone de credenciales debe tener la posibilidad de registrarse para poder llegar a autenticarse. Esa es la función de esta vista a la cual se accede mediante la URL *http://pfc-cdfuente.libresoft.es/re gister* o presionando el botón de 'Registrar' de la vista anteriormente descrita.

  En la Figura 3.11 se pueden ver los elementos que componen esta vista.

Figura 3.11: Captura de imagen de la vista register.html

En la parte superior se observan dos botones: uno destinado a la opción de volver a la página inicial y otro para ir a la vista encargada de autenticarse en el sistema.
Después podemos ver que la vista se compone de cuatro campos en los que introducir la información necesaria para un correcto registro en la aplicación. Además existe un botón para elegir el rol que va a tomar el usuario en la aplicación: profesor o alumno, estando por defecto seleccionada la opción de alumno.
Una vez rellenados los campos debemos presionar el botón de 'Registrar' de la parte inferior, y si todo es correcto aparecerá un mensaje que confirme que el registro ha sido realizado. En el caso de que el usuario sea de tipo profesor se le mostrará un mensaje con la dirección de correo del administrador para que se ponga en contacto con el mismo y que pueda acceder a toda la funcionalidad de la aplicación.
Si hubiera ocurrido algún error en el registro también aparecerá un mensaje informando sobre el mismo.

- **register_event.html**
  Si el usuario validado en el sistema es de tipo profesor, tiene la posibilidad de acceder a esta vista mediante la URL *http://pfc-cdfuente.libresoft.es/register_event* o pulsando el botón 'Eventos' del navegador.
  Esta vista tiene varias partes que se detallarán por separado:

  - Primero nos ofrece la posibilidad de crear nuevos eventos como podemos apreciar en la Figura 3.12. Con siete campos a rellenar se pueden crear eventos que pasarían al conjunto de eventos que posee el usuario validado en el sistema. Una vez rellenados, se presiona el botón 'Registrar evento' para comprobar que los datos introducidos coinciden con lo esperado. Si por algún caso surge algún error se mostrará mediante un mensaje en esta misma interfaz.
  - Por otro lado en esta misma vista, en la parte inferior, tendremos una lista de

Figura 3.12: Captura de imagen de la vista register_event.html

los eventos que pertenecen al usuario. Así como en la vista inicial los eventos pueden desplegarse para ver su información, y además en este caso darnos más opciones como se puede apreciar en la Figura 3.13.

Figura 3.13: Captura en detalle de los eventos de la vista register_event.html

De modo que en el evento desplegado se observa que está toda la información del mismo a parte de dos botones: 'Editar' y 'Borrar'. Al pulsar el botón de borrar desaparecerá de forma inmediata el evento de la lista de eventos del

usuario y por tanto de todas las listas en las que estuviera incluido. Al pulsar el botón de editar nos redirigirá a la vista de edición de eventos.

- **edit_event.html**
  Una vez creado un evento tenemos la posibilidad de volver a editarlo. Como hemos visto en la vista anterior al presionar el botón de 'Editar' nos redirigirá a esta vista en la que se muestran todos los campos del evento en cuestión.

Figura 3.14: Captura de imagen de la vista edit_event.html

Como podemos apreciar en la Figura 3.14, todos los campos ya están rellenos con la información correspondiente al evento que estemos editando. Lo único que tenemos que hacer es cambiar el campo que deseemos y pulsar el botón de 'Editar evento'. Si todos los campos son correctos aparecerán en sus debidos espacios ya actualizados, en cambio si hay algún tipo de error se mostrará mediante un mensaje en esta misma vista.

Una vez terminada la edición podemos pulsar el botón 'Volver' situado en la esquina superior izquierda para volver a la vista 'register_event.html'.

- **register_list.html**
  La siguiente vista a analizar es la que permite la creación de listas. Si el usuario validado en el sistema es de tipo profesor podrá acceder a esta vista a través de la URL *http://pfc-cdfuente.libresoft.es/register_list* o presionando el botón de 'Listas' del navegador.

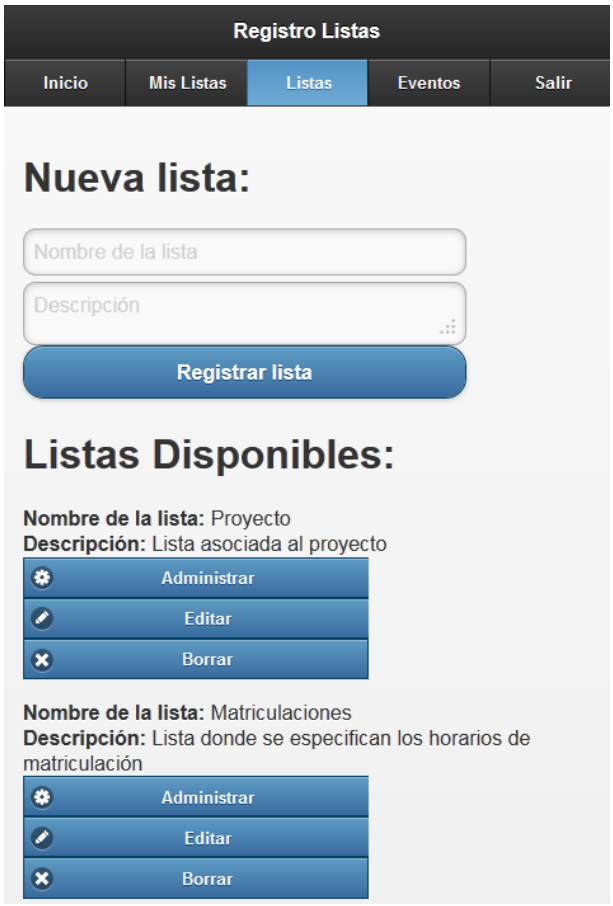

Figura 3.15: Captura de imagen de la vista register_list.html

Como se puede apreciar en la Figura 3.15, esta vista también se divide en dos partes que podemos describir por separado:

- Por un lado tenemos en la parte superior una sección dedicada a la creación de nuevas listas. Ésta se divide en dos campos, 'Nombre' y 'Descripción', los cuales se deben rellenar correctamente y pulsar después el botón de 'Registrar lista'. Si todo ha ido bien la lista se sumará a las que ya tenga el usuario creadas y si no, mostrará un mensaje de error informando al usuario.

- Por otro lado, en la parte inferior de la vista, se dispone de la visualización de todas las listas existentes en la base de datos. Tanto si pertenecen al usuario como si no, se pueden ver todas las listas, puesto que debemos disponer de algún modo de añadir un evento propio a una lista que no nos pertenezca si la información que contiene es relativa a nuestro evento. Esto se hace mediante el botón 'Administrar' situado en cada una de las listas, que nos llevará a la vista correspondiente. Aparte de este botón podemos ver que existen otros dos botones, que sólo aparecerán si somos los propietarios de la lista: 'Editar' y 'Borrar'. Si presionamos el botón de borrado, se eliminará inmediatamente esa lista de la base de datos. Si pulsamos el botón de editado nos redirigirá a la vista de edición de listas.

- **edit_list.html**

  Muy en consonancia con el apartado anterior, la vista que nos compete ahora es la de edición de listas. Accediendo a través del botón de 'Editar' que se ha expuesto en la vista anterior, aparecerá la vista que podemos apreciar en la Figura 3.16.

Figura 3.16: Captura de imagen de la vista edit_list.html

Como se puede observar, ésta es una vista muy simple en la que aparecen los campos de la lista en cuestión ya con su correspondiente información y nos ofrece la posibilidad de cambiarlos. Una vez hechos los cambios pulsamos el botón de 'Editar lista' y si todo es correcto se actualizará la información, si no, aparecerá un mensaje de error en la parte superior. Una vez terminada la edición se puede pulsar el botón 'Volver' situado en la esquina superior izquierda para poder volver a la vista anterior.

- **add_events.html**

  La última vista relacionada con las listas y los eventos es la que permite añadir distintos eventos a las listas existentes. A esta vista se accede mediante el botón 'Administrar' de la vista 'register_list.html'. Como se puede observar en la Figura 3.17, esta vista está compuesta de varias partes.

Figura 3.17: Captura de imagen de la vista add_events.html

Primero nos muestra el nombre de la lista por si nos hemos equivocado de lista y no queremos añadir eventos a la elegida. Inmediatamente después tenemos una lista con los eventos incluidos en la lista que estamos administrando, con la opción de desplegar la información para ver si coincide con lo que queramos añadir o borrar.
En la parte inferior tenemos dos opciones: añadir o borrar eventos. Ambos apartados están compuestos de un menú desplegable en el que elegiremos un evento propio y un botón debajo. Una vez elegido el evento que queramos pulsamos el botón con lo que queramos hacer: 'Añadir Evento' para incluir ese evento en la lista o 'Borrar Evento' para que desaparezca de ella si ya estaba incluido. Tanto si añadimos un evento que ya estuviese incluido como si borramos uno que no lo estuviese, se nos mostrará un mensaje informándonos del error.
Una vez terminadas las acciones que queramos en esta vista, tenemos la opción de volver pulsando el botón 'Volver' situado en la esquina superior izquierda.

- **admin.html**

  Una vez analizadas las vistas más importantes de nuestro proyecto hace falta men-

cionar la vista de la interfaz de administración. Accediendo a través de la URL: *http://pfc-cdfuente.libresoft.es/admin/*, se puede llegar a lo que vemos en la Figura 3.18, vista destinada al administrador del sistema.

Figura 3.18: Captura de imagen de la vista de validación del administrador

En la Figura se observa que, de forma similar a nuestra vista de 'login.html', aquí se piden unas credenciales que sólo podrán validarse por un administrador. Si intentáramos acceder con el nombre y contraseña de un usuario corriente, ya sea de tipo alumno o profesor, no conseguiríamos acceder al sitio de administración.
Una vez validadas las credenciales de administrador pasaremos a ver lo que se muestra en la Figura 3.19.

Figura 3.19: Captura de imagen de la vista del sitio de administración

Como podemos observar el estilo de estas vistas es predeterminado por Django y difiere del resto, debido a que no se ha visto necesario el cambio de las mismas ya que aquí sólo debe acceder el administrador, es decir, no es una interfaz pública.
Dentro del sitio se puede observar el nombre de nuestra aplicación Django 'app1' y una serie de campos correspondientes al modelo de datos del sistema. El administrador puede añadir, editar o borrar cualquiera de estos elementos ya que tiene

todos los permisos necesarios. Cabe destacar que tenemos dos apartados con el mismo nombre pero en distintos módulos: 'Usuarios' en 'App1' y 'Usuarios' en 'Auth'. Esto se debe a que tenemos una correspondencia en el modelo de datos en la que extendemos el tipo usuario ya incluido en 'Auth' y debemos tener cuidado con el manejo de ambos. Si eliminamos usuarios debemos eliminarlos de ambos apartados ya que si no podría producir alguna inconsistencia en el sistema.

- **errores.html**

  Por último existe una vista destinada a mostrar errores como cuando un usuario intenta acceder a una página que no existe en nuestro proyecto. En esta vista simplemente se muestra el mensaje de error correspondiente con un botón para poder volver a la página inicial si lo deseamos.

Capítulo 4

Despliegue y resultados

Este capítulo está destinado a abordar el proceso de pruebas que se ha realizado para probar el correcto funcionamiento de la aplicación. Aquí se expondrán todos los detalles de este proceso de prueba, los objetivos que se buscaban al inicio y los resultados del mismo, así como unas conclusiones iniciales sobre lo que supone este experimento.

4.1. Introducción

Es importante una vez realizado un proyecto de cierta envergadura y potencialmente complejo, poder probarlo con personal ajeno al equipo de desarrollo del mismo. Con el fin de encontrar fallos que se hayan podido pasar por alto y de comprobar el funcionamiento en distintos dispositivos, la fase de pruebas es fundamental en el desarrollo de proyectos como el que estamos enfrentando. Además es necesario un proceso de retroalimentación con los usuarios, de modo que así podemos hacernos una idea de lo que les parece la aplicación a parte de encontrar posibles errores.
Con este propósito se ha realizado el proceso de pruebas en un servidor proporcionado por la Universidad Rey Juan Carlos en el que se ha dejado operativa la aplicación y libre para todo aquel que quisiera probarla. A continuación se explica el experimento de manera más detallada:

- **Objetivos del experimento:**
    - Comprobar el correcto funcionamiento de la aplicación más allá de los dispositivos en los que ya se ha probado.
    - Comprobar la usabilidad de la interfaz web por usuarios ajenos al desarrollo de la misma.
    - Comprobar el origen de uso de la aplicación, dispositivos móviles o PCs.
    - Ver la difusión de los eventos mediantes las redes sociales incluidas.

- **Fecha de inicio:** 20 de Agosto de 2013.
- **Fecha de finalización:** 1 de Octubre de 2013
- **Servidor de pruebas donde se ha alojado la aplicación:** *http://pfc-cdfuente.libresoft.es*
- **Método de retroalimentación:** encuesta realizada por el desarrollador ubicada en la dirección *http://www.encuestafacil.com/RespWeb/Cuestionarios.aspx?EID=1568667&MSJ=NO#Inicio*

4.2. Resultados

El éxito o fracaso de los resultados de la fase de pruebas ha sido determinado por la experiencia de los usuarios que han probado la aplicación y las encuestas que han realizado algunos de ellos.
Debido a que la difusión del proyecto ha sido escasa, sobre todo por las fechas de exposición del mismo que coincidían con vacaciones y con el inicio del curso, el número total de usuarios registrados en el sistema ha sido de diez, todos ellos de tipo alumno.
Los resultados técnicos han sido positivos, ya que no se han detectado fallos por parte de los usuarios y la interfaz web ha sido bien acogido por todos ellos.
Los resultados en torno a la difusión en las redes sociales ha sido más bien un fracaso ya que apenas se han usado los botones de 'Twittear' y 'Me gusta', y en la cuenta de Twitter asociada al proyecto sólo se han conseguido dos seguidores.
A continuación se relatarán las experiencias en torno a la encuesta realizada a los usuarios, en el que el número total de personas que la han realizado ha sido de siete:

1. *¿Cuántas veces ha visitado usted la página en los últimos 30 días?*

Figura 4.1: Respuestas de la primera pregunta de la encuesta

De esta pregunta se saca de resultado que la mayoría de usuarios que han probado la aplicación, se han metido precisamente con esa intención y que sólo alguno ha vuelto a meterse en otra ocasión.

2. *¿Cree que volverá a visitarnos en los próximos 30 días?*

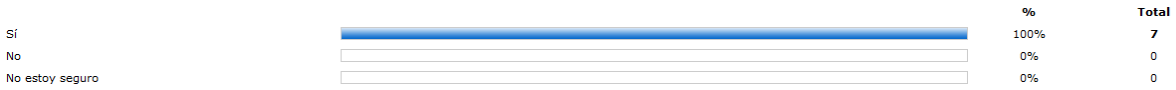

Figura 4.2: Respuestas de la segunda pregunta de la encuesta

Con esta pregunta descubrimos el potencial de la aplicación ya que el 100 % de los usuarios piensa que volverá a usar la interfaz web en algún momento.

3. *¿Por qué motivo/s ha visitado usted la página?*

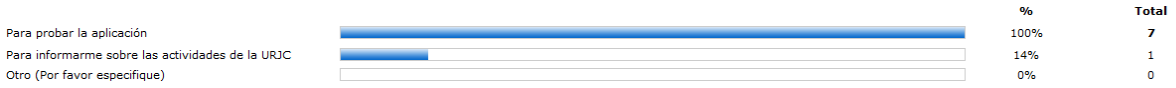

Figura 4.3: Respuestas de la tercera pregunta de la encuesta

De nuevo el resultado que se saca con esta pregunta es que los usuarios han decidido probar la aplicación en primer lugar y sólo a un usuario le ha servido la información que contiene (ya que en ese momento tampoco se disponía de mucha información).

4. *¿Cuál es su nivel de satisfacción general con la página?*

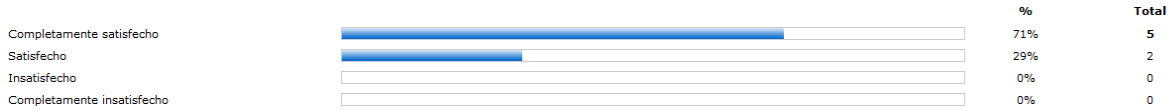

Figura 4.4: Respuestas de la cuarta pregunta de la encuesta

Con esta pregunta queríamos medir la satisfacción del usuario al usuar la aplicación y vemos que la gran mayoría están completamente satisfechos y no hay ninguno que esté insatisfecho con la experiencia.

5. *¿Qué errores ha encontrado al usar la aplicación?*

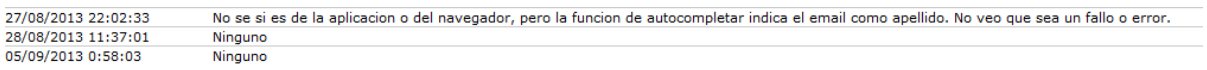

Figura 4.5: Respuestas de la quinta pregunta de la encuesta

También se puede comprobar que no se ha encontrado ningún tipo de error al usar la aplicación, al menos por parte de los usuarios que han contestado a esta pregunta.

6. *¿Qué características cree que se pueden añadir a la aplicación?*

23/08/2013 0:45:45	Añadir en la parte superior bienvenido fulanito cuando fulanito registrado entre en la aplicación, junto con un contador que indique el tiempo que lleva online
28/08/2013 11:37:01	Un listado con las direcciones de correo de los profesores para tenerlo integrado en la aplicación y no tener que visitar el directorio de la URJC
28/08/2013 15:24:33	un icono para seguir la aplicacion via twitter facebook rss
05/09/2013 0:58:03	Crear Notificaciones para avisar de cambios, novedades, etc.

Figura 4.6: Respuestas de la sexta pregunta de la encuesta

Esta pregunta es muy interesante ya que permite saber al desarrollador lo que han echado en falta los usuarios al usar la aplicación, y propone evolucionar el sistema con las críticas de los propios usuarios.

7. *Basándose en su experiencia, por favor, puntúe los siguientes aspectos:*

	Muy buena	Buena	Regular	Mala	Muy mala	No aplicable	Total
Facilidad de uso	86%(**6**)	14%(1)	0%(0)	0%(0)	0%(0)	0%(0)	**7**
Rapidez de descarga de las páginas	57%(**4**)	29%(2)	0%(0)	0%(0)	0%(0)	14%(1)	**7**
Diseño atractivo	57%(**4**)	43%(3)	0%(0)	0%(0)	0%(0)	0%(0)	**7**
Utilidad	71%(**5**)	29%(2)	0%(0)	0%(0)	0%(0)	0%(0)	**7**

Figura 4.7: Respuestas de la séptima pregunta de la encuesta

Con esta pregunta queríamos ver qué era lo más valorado de la aplicación y sorprende encontrar que es la facilidad de uso y su usabilidad, teniendo en cuenta que todos los apartados tienen buena valoración.

8. *¿Qué es lo que más le gusta de la página?*

20/08/2013 19:15:03	El diseño y la facilidad de uso. Además de su finalidad.
23/08/2013 0:59:27	Que es muy intuitiva y puede ser muy útil tanto para alumnos como profesores.
28/08/2013 11:38:36	Su interfaz
28/08/2013 15:26:30	Es bastante intuitiva para manejarse con ella
02/09/2013 13:16:49	Interfaz muy sencilla
05/09/2013 0:59:19	La posibilidad de verlo desde el móvil con rapidez.

Figura 4.8: Respuestas de la octava pregunta de la encuesta

En todo proyecto con estas características viene bien saber el grado de satisfacción de los usuarios y qué es lo que más aprecian, y en este caso ha quedado claro que la interfaz es de fácil uso y que la aplicación tiene potencial usabilidad.

9. *¿Qué es lo que menos le gusta de la página?*

23/08/2013 0:59:27	Que tiene pocos colores, podría añadirse una paleta de colores para que cada uno lo personalice como quiera.
28/08/2013 11:38:36	Nada
05/09/2013 0:59:19	La web para navegador es muy grande

Figura 4.9: Respuestas de la novena pregunta de la encuesta

Así como está bien saber qué es lo que les gusta a los usuarios, aún más importante es saber que no les ha gustado para poder mejorarlo. En este caso lo único que

apreciamos son mejoras en cuanto al diseño de la web más puramente comercial, se podría mejorar este aspecto adaptándolo a los navegadores en los PCs y ofreciendo personalización de la interfaz.

10. *¿Tiene usted algún comentario adicional o alguna sugerencia para ayudarnos a mejorar?*

Figura 4.10: Respuestas de la décima pregunta de la encuesta

Como abarcábamos gran información con las demás preguntas, los usuarios no han visto necesario contestar a esta en concreto.

11. *¿Es usted profesor o alumno?*

Figura 4.11: Respuestas de la undécima pregunta de la encuesta

Como la aplicación divide su uso en dos roles, era necesario ver que tipo de usuarios estaban validando el sistema, para ver si se abarcaban todas las vistas del mismo.

12. *¿Podría decirnos el navegador que usa para visitarnos?*

20/08/2013 19:15:29	Mozilla Firefox
23/08/2013 1:00:06	Firefox
28/08/2013 11:38:52	Firefox
28/08/2013 15:26:59	firefox
02/09/2013 13:17:10	Mozilla
05/09/2013 0:59:35	google chrome

Figura 4.12: Respuestas de la duodécima pregunta de la encuesta

Con el fin de analizar el uso de la aplicación, es bueno saber qué navegadores han sido usados para si hubiera algún error asociado a un navegador, poder detectarlo.

13. *¿Nos visita desde un Smartphone?*

Figura 4.13: Respuestas de la decimotercera pregunta de la encuesta

Esta última pregunta se ha realizado con el fin de saber si la aplicación se ha usado desde un dispositivo móvil, y el resultado ha sido más bien decepcionante, ya que sólo un tercio de los usuarios ha usado la aplicación desde un Smartphone.

4.3. Conclusiones iniciales

Se pueden extraer varias conclusiones iniciales tras analizar los resultados de esta fase de pruebas:

- **El sistema final funciona.** Se ha podido comprobar que el sistema desarrollado funciona de la forma esperada y, al menos técnicamente, de manera correcta.
- **La interfaz ha sido un éxito.** Se ha valorado por encima de todo el carácter intuitivo y de fácil manejo de la aplicación, por tanto el uso del framework JQuery Mobile ha sido un acierto.
- **El acceso sigue siendo predominantemente desde dispositivos fijos.** Se ha podido comprobar tristemente que la aplicación ha sido usada en menor medida desde dispositivos móviles, que era una parte fundamental del proyecto, pero también depende de las preferencias de uso por parte de los usuarios.
- **La difusión social ha fracasado.** Una de las bazas más potentes del proyecto que era su difusión en las redes sociales ha sido un fracaso, ya que apenas se han usado los botones de 'Twittear' y 'Me gusta'. Tampoco se ha seguido en Twitter a la cuenta asociada al proyecto. De nuevo esto depende de las preferencias de cada usuario y a lo mejor era necesaria la inclusión de otras redes sociales más acordes con los gustos de los mismos.

Capítulo 5

Conclusiones y Futuras Líneas de Trabajo

5.1. Conclusiones

Después de cuatro capítulos en los que se han expuesto desde las motivaciones y objetivos iniciales por los que se propuso hacer este Proyecto de Fin de Carrera, hasta el diseño e implementación del mismo, es necesario analizar el trabajo realizado. El objetivo principal de este proyecto era la implementación de una aplicación de ayuda a la gestión de actividades universitarias orientada a smartphones. Como era un objetivo complejo por sí mismo establecimos una serie de objetivos secundarios que sirvieran para cumplir el objetivo principal.
A lo largo de un año de trabajo en el que se ha desarrollado el sistema, podemos concluir primeramente que hemos cumplido con el objetivo principal, creando un sistema que además pueda ser usado por usuarios ajenos al desarrollo del proyecto y que se desenvuelven perfectamente gracias a la interfaz intuitiva y útil que se ha desarrollado. A continuación se expondrán varias conclusiones que se pueden sacar de este Proyecto de Fin de Carrera:

- El sistema implementado es único hasta el momento, ya que las interfaces de gestión de eventos que hay ahora mismo en las universidades se han quedado bastante por detrás tecnológicamente hablando. El framework JQuery Mobile ha resultado ser muy útil para el desarrollo de la aplicación y nos ha aportado una interfaz moderna y muy intuitiva. Además hemos conseguido orientar nuestro sistema hacia el mundo de los dispositivos móviles como se requería en un principio.

- Aunque la teoría dice que cada vez son más los que utilizan los dispositivos móviles como smartphones y tablets, hemos comprobado que aún le queda mucho a esta transición ya que según los resultados obtenidos en este proyecto vemos que sólo un tercio de los usuarios ha accedido a la aplicación mediante un smartphone, por

tanto la mayoría todavía prefiere conectarse desde un dispositivo fijo.

- Independientemente del impacto del sistema en los usuarios ajenos al desarrollo, este proyecto ha sido una gran experiencia para el desarrollador del mismo. Gracias a esta aplicación el creador ha podido ganar conocimientos en muchos ámbitos como ha sido Django, Python y, en mayor medida, JQuery Mobile, ya que nunca había desarrollado con nada parecido a lo largo de la carrera. Además el desarrollo de una memoria junto con todo el proceso de creación de un proyecto desde cero, supone un acercamiento con el terreno laboral muy valioso, ya que simula de manera acertada lo que sería trabajar realizando proyectos como se hace actualmente en el mundo empresarial.

- En consonancia con el punto anterior cabe destacar la fase de pruebas en las que el desarrollador se enfrenta a las críticas por parte de usuarios que son ajenos al desarrollo del proyecto. Siempre es importante el hecho de tener una realimentación en un proyecto de cierta envergadura, y reparar los errores si fuera necesario, además de tener en cuenta las opiniones de los demás.

- Cabe destacar el grado de satisfacción del autor de este Proyecto de Fin de Carrera, después de horas y horas de trabajo, es gratificante comprobar el resultado. Así es importante llegar a la conclusión de que después de mucho trabajo y sacrificio, se puede llegar a desarrollar un proyecto como este y sentirse satisfecho con la labor realizada, creando ganas de abordar otros proyectos de manera similar.

- Por último, a modo de finalización de las conclusiones, se puede decir que este proyecto no es más que el comienzo de algo que podría llegar a usarse de verdad en el ámbito universitario. Así en el apartado siguiente se expondrán posibles futuras líneas de trabajo para que se pueda seguir trabajando en este proyecto y llegar a estar al alcance de cualquier alumno o profesor de la universidad.

5.2. Futuras líneas de trabajo

Así como hemos puntualizado en las conclusiones anteriores, este Proyecto de Fin de Carrera puede tener mucho futuro si se pone interés y se trabaja un poco más en el mismo. A continuación se exponen posibles futuras líneas de trabajo en las que se podría orientar el sistema desarrollado:

- El primer avance que podríamos implementar sería el uso de una base de datos más potente como MySQL. De este modo estaría preparado para el uso de varios miles

de usuarios y la información quedaría aún más robusta frente a fallos de seguridad, además de ofrecer gran velocidad a la hora de realizar las operaciones.

- Por una parte nos ofrece una visión de una interfaz web similar a lo que resultaría una pequeña red social universitaria. Partiendo de ahí podríamos unirlo a más redes sociales existentes: desde Facebook, pasando por Google+, hasta una infinidad de posibilidades en las que compartir la información. Así cada alumno, con sus gustos propios, podría difundir los eventos en las redes sociales en las que más cómodo se encuentre, personalizando aún más la aplicación al usuario.

- Aparte está el hecho de que muchos usuarios ya usan algún gestor de eventos o calendarios propio. Un gran avance sería poder integrar estos calendarios con el de la aplicación, de modo que de forma automática pudieran agregarse los eventos de las listas que el alumno 'siga' en sus propios calendarios, como por ejemplo Google Calendar. Esto también podría servir para los profesores que usen alguna herramienta del estilo y poder tener toda la información aunada o incluso exportar eventos de sus propios calendarios a listas de nuestra aplicación.

- En consonancia con la difusión de la información también podría ser exportada mediante RSS. Debido a que los lectores de RSS se usan desde hace mucho tiempo y aunque ahora no estén muy de moda, la posibilidad de exportar la información mediante RSS resulta ser otra buena opción para la redifusión de información. Así cuando se añadieran nuevos eventos mediante agregadores se puedan ver éstos mismos de otra manera a la propia de la aplicación.

- Otra buena opción sería el poder avisar a los usuarios de nuevos eventos en las listas a las que estén apuntados. Esto nos ofrece dos posibilidades:

  - Usar una bandeja de correo personalizada asociada a cada usuario en la propia aplicación, o incluso la posibilidad de mandar un correo al usuario en cuestión.
  - Llevar el proyecto a una aplicación Android de modo que cuando hubiera nuevos eventos pudiera aparecer un aviso en nuestro Smartphone o Tablet.

  Mientras que la primera idea está basada en cómo se desarrollaba en el pasado, la segunda nos ofrece posibilidades nuevas más orientadas al futuro próximo.

- También se podría unir aún más con el ámbito universitario, si pudiera formar parte de la propia web oficial o, derivando de ella, pudiéramos acceder a la propia aplicación. A parte, las cuentas de alumno y profesor que hay preestablecidas podrían

ser ya las propias de autenticación en nuestra web y así no ser necesaria la creación de cuentas nuevas. Además podría integrarse con la herramienta actual de Moodle de alguna manera para que la difusión fuera máxima y a la vez formara parte del propio tejido web de la universidad.

- El servidor de Django es bastante limitado, una buena mejora sería poder cambiar el mismo a un servidor más potente como Apache o IIS, de manera que nos pudiera ofrecer información adicional sobre el uso de la aplicación y poder elaborar estadísticas de uso más complejas que con una simple encuesta. De este modo podrían surgir más ideas y recoger más fallos según los navegadores y dispositivos desde los que se accede.

- Sería interesante la unión con otros Proyectos de Fin de Carrera como por ejemplo el 'Sistema de Control de Presencia Implementado en Smartphones Vía Geolocalización' del autor Félix Redondo Sierra. En este caso el proyecto se centra en eventos universitarios, tiene una interfaz web y una base de datos de alumnos y profesores, y la posibilidad de hacer 'check-in' cuando se realicen las actividades tanto por parte de los alumnos como de los profesores. Por todo esto la unión de estos dos proyectos podría ser un paso más en la red universitaria: alumnos que puedan seguir listas y poder apuntarse a eventos y en el momento en el que ocurran llevar un control de asistencia de los mismos por parte de los profesores.

- Debido a que cada vez queremos que las acciones sean los mas automatizadas posibles, sería interesante automatizar aún más la creación de nuevos usuarios. Más concretamente la posibilidad de que al registrarse un profesor en el sistema, se mandase un correo al administrador de manera automática, e incluso la posibilidad de validarlo si pertenece a un conjunto de posibles direcciones de profesores que sean confiables.

Apéndice A

Planificación del proyecto

Este apéndice supone la finalización de la memoria con un desglose de las fases principales por las que ha pasado el proyecto.

A.1. Diagrama de Gantt

En la Figura A.1 se puede observar el diagrama de Gantt correspondiente a nuestro proyecto con las distintas fases de las que se compone. Además se ven las fechas iniciales y finales de cada fase así como los ligeros solapamientos que ha habido entre ellas. Un diagrama de Gantt muestra el origen y el final de las distintas tareas asociadas a un proyecto de manera gráfica y es una herramienta muy útil en cuanto a la elaboración de proyectos.

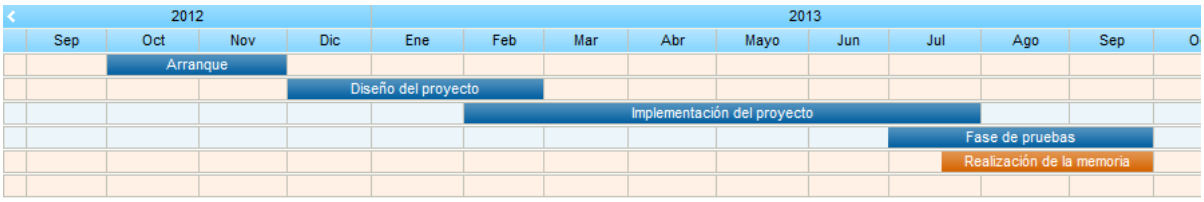

Figura A.1: Diagrama de Gantt

Cabe destacar la duración de las dos primeras fases del proyecto, ya que teníamos que planificar muy bien el tiempo para poder desarrollar la aplicación en un tiempo total menor a un año. También vemos que la fase de implementación comenzó incluso antes de terminar el diseño total, esto es debido a que se comenzó a implementar el servidor de forma simple en html para luego, cuando estuviera diseñada la implementación con JQuery Mobile, fuera más sencilla la transición. Además vemos que la implementación siguió durante un tiempo en el que se empezó la fase de pruebas, por si había algún error

grave poder mitigarlo al instante y que la fase de pruebas estuviera totalmente libre de errores durante al menos dos meses. Por último destacar que durante la fase de pruebas se realizó la elaboración de esta memoria, para así aprovechar más el tiempo e incluir los últimos resultados al final de esta fase.

Bibliografía

[AH07] Jacob Kaplan-Moss Adrian Holovaty. *The Definitive Guide to Django: Web Development Done Right.* Apress, 2007.

[Ben09] J. Bennett. *Practical Django Projects.* Apresspod Series. Apress, 2009.

[Cla10] G. Clapperton. *This is Social Media: Tweet, blog, link and post your way to business success.* Wiley, 2010.

[dlI10] J.L.M. de la Iglesia. *Web 2.0. Una Descripcin Muy Sencilla de Los Cambios Que Estamos Viviendo.* POCKET INNOVA. Netbiblo, 2010.

[FC12] M.R. Firtman and S.L.G. Cruz. *jQuery Mobile: Aplicaciones HTML5 para móviles / Up and Running.* O'Reilly (Anaya Multimedia). Anaya Multimedia-Anaya Interactiva, 2012.

[Fir10] M. Firtman. *Programming the Mobile Web.* O'Reilly Media, 2010.

[Gli13] S. Gliser. *Creating Mobile Apps with jQuery Mobile.* Packt Publishing, Limited, 2013.

[Haw10] M.D. Hawker. *Developer's Guide to Social Programming: Building Social Context Using Facebook, Google Friend Connect, and the Twitter API, The.* Developer's Library. Pearson Education, 2010.

[LAS10] P. Lubbers, B. Albers, and F. Salim. *Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development.* Expert's voice in Web development. Apress, 2010.

[LR01] Avraham Leff and James T. Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, EDOC '01. IEEE Computer Society, 2001.

[MBW07] Dana Moore, Raymond Budd, and William Wright. *Professional Python Frameworks: Web 2.0 Programming with Django and Turbogears*. Wrox Press Ltd., 2007.

[NL06] J. Nielsen and H. Loranger. *Usabilidad*. Diseno Y Creatividad / Design and Creativity. Anaya Multimedia-Anaya Interactiva, 2006.

[Ree79] T. Reenskaug. THING-MODEL-VIEW-EDITOR: an example from a planning system. *Xerox PARC Technical Note*, May 1979.

[vdVAB+07] E. van der Vlist, D. Ayers, E. Bruchez, J. Fawcett, and A. Vernet. *Professional Web 2.0 Programming*. Wrox professional guides. Wiley, 2007.