Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Construcción de una herramienta para la gestión de las prácticas externas en la ETSI Informáticos de la UPM

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro Otero Ortiz de Cosca

Tutora: Mª Luisa Córdoba Cabeza

25 de junio de 2014

Índice general

Capítulo 1

Introducción

La ETSI. Informáticos de la UPM ofrece un sistema de prácticas externas como parte de los créditos de optatividad de algunas de las titulaciones que se imparten. El COLFI (Centro de Orientación Laboral de la Facultad de Informática), se encarga de gestionar este sistema de prácticas externas, almacenando y manejando datos de las mismas a través de hojas de cálculo.

El centro debe establecer un convenio con la empresa que acogerá al alumno, además, a cada alumno le son asignados dos tutores, un docente de la Escuela como tutor interno y un trabajador de la empresa como tutor externo. Los datos de los tutores, las empresas y los alumnos se almacenan para la posterior consulta, elaboración de informes que recogen distintas estadísticas y generación de certificados a los tutores que los soliciten. Todos estos datos son almacenados mediante hojas de cálculo y la generación de certificados e informes son realizadas actualmente a mano. Por ello, en este proyecto, se propone dotar al Centro de Orientación Laboral de una herramienta que permita realizar todas estas tareas de forma rápida, cómoda e incluso automática, facilitando el acceso y el almacenamiento de los datos.

Capítulo 2

Objetivos

El objetivo de este proyecto es proporcionar una herramienta de gestión que permita agilizar, sistematizar y registrar la actividad de los principales procesos que se llevan a cabo en las prácticas externas. Además la herramienta pretenderá ser lo más modular posible para permitir la inclusión de nuevas funcionalidades para futuras necesidades.

Capítulo 3

Las prácticas externas en la Escuela

Como ya se ha dicho anteriormente, la Escuela ofrece un sistema de prácticas en empresas como parte de la optatividad de algunas titulaciones.

En la asignación de un proyecto de prácticas externas intervienen 4 entidades, un alumno solicitante, una empresa de destino, un tutor docente de la Escuela (tutor interno) y un tutor dentro de la empresa (tutor externo).

En la Figura 3.1. se puede apreciar un esquema del proceso de gestión de estas relaciones a través del COLFI en el cual, las entidades aparecen representadas como rectángulos, los trámites como flechas, los informes generados como rectángulos recortados y los datos intercambiados en los trámites como banderas.

Para que un alumno pueda realizar sus prácticas externas en una empresa se sigue el siguiente proceso:

1. El COLFI debe realizar un convenio con la empresa, de esta relación se almacenan todos los datos de la misma.

2. El alumno solicita al COLFI realizar las prácticas en la empresa, para ello debe indicar una persona dentro de la empresa que hará el papel de tutor externo. En esta interacción se almacenan todos los datos del alumno y del tutor externo.

3. El docente de la Escuela que juega el papel de tutor interno del alumno, puede asignarse en el momento de la solicitud del propio alumno o posteriormente. En el momento en el que el tutor interno va a ser asignado se almacenan todos los datos del mismo.

Con los datos almacenados de todas las relaciones entre estas entidades, se generan informes de actividad de los distintos semestres. Además, un tutor

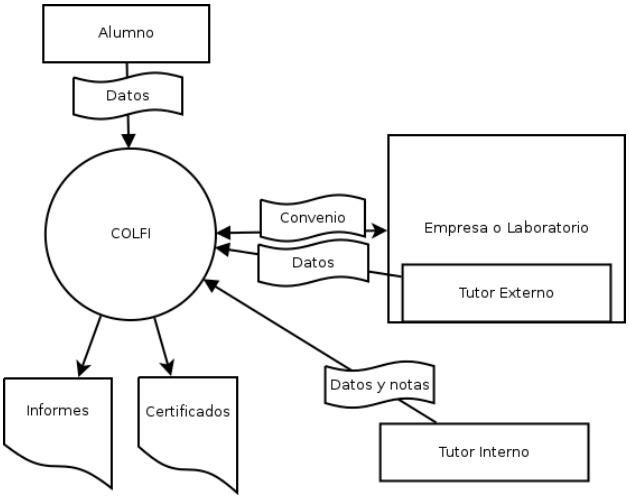

Figura 3.1: Diagrama de alto nivel de la gestión de las prácticas externas.

(interno o externo), puede solicitar un certificado que acredite su trabajo dentro del sistema de prácticas externas de la Escuela, es este certificado deben aparecer todos los alumnos que ha tutoreado y en qué fechas.

Actualmente el almacenamiento de todos los datos de los alumnos se realiza con hojas de cálculo y en caso de ser necesario, los informes de actividad y los certificados de participación, se generan a mano.

El almacenamiento de datos en hojas de cálculo es poco eficiente, está sujeto a la introducción de errores y dificulta el tratamiento de los mismos, haciendo necesarios conocimientos de *softwares* de esto tipo por parte de los miembros del COLFI. Además, de esta forma los datos no están centralizados y es menos sencillo compartirlos entre miemebros y mantener un histórico de toda la activdad de las prácticas externas en la empresa.

Se propone crear una base de datos, alojada en un servidor, con una *interfaz* de acceso cómoda, rápida y sencilla, que automatize la generación de informes y certificados.

Capítulo 4

Elecciones software

4.1. Definición del problema

Como se ha visto en el capítulo anterior, toda la gestión de las prácticas externas en la Escuela pasa por el almacenamiento de los datos de todas las entidades participantes para posteriormente ser tratados; lo que implica la necesidad de realizar una base de datos. A su vez, puesto que las gestiones las pueden realizar distintas personas simultáneamente, el acceso a la base de datos debe poder ser realizado desde varios equipos y, por supuesto, a través de una interfaz cómoda que abstraiga al usuario de la programación de las *queries*.

Se necesita pues, una base de datos accesible desde varios clientes simultáneos con una interfaz que permita la introducción, tratamiento y procesamiento de los datos.

4.2. Aplicaciones web, aplicaciones móviles y aplicaciones de escritorio

Una vez definido el problema a resolver, es muy importante elegir correctamente el tipo de herramienta que se va a desarrollar. Puesto que es necesario crear una base de datos con acceso múltiple desde varios equipos, será preciso la instalación de un servidor de bases de datos, sin embargo, la creación de la parte cliente debe ser analizada minuciosamente, el software apropiada puede mejorar ampliamente el rendimiento en la gestión de las prácticas, la facilidad de adaptación a la herramienta y la introducción de la misma dentro del entorno laboral de los miembros del COLFI. Por tanto, se proponen los siguientes tres modelos:

- Una **aplicación web** con una base de datos centralizada en un servidor
- Una **aplicación móvil** con una base de datos centralizada en un servidor
- Una **aplicación de escritorio** con una base de datos centralizada en un servidor

Dada la naturaleza de la necesidad, una aplicación móvil no tendría sentido puesto que los miembros del COLFI no necesitan el nivel de accesibilidad que proporciona una aplicación en un dispositivo de este tipo. Por otra parte, una aplicación de escritorio implicaría la necesidad de instalarla en todos los puestos que fueran a ser utilizados, lo que requiere un esfuerzo extra que se puede evitar fácilmente con una aplicación web.

Por tanto, una aplicación web proporcionaría:

- Mayor transparencia de instalación frente a una aplicación de escritorio. Una vez que la aplicación está funcionando en el servidor, el usuario simplemente debe conectarse al mismo y no necesita instalar ningún software en su puesto. Esto facilita el acceso a la herramienta, evitando que la ampliación del número de ordenadores o su renovación implique tiempo extra de instalación.
- Independencia del sistema operativo. Al ser una página web, lo único que el usuario necesita es un navegador web, que se proporciona con cualquier sistema operativo inclusive con los sistemas operativos móviles. Esto evita la necesidad de programar versiones distintas de la aplicación en función del sistema operativo.
- Facilidad en el control de acceso. Una de los requisitos de la aplicación es que el acceso a la misma, no sólo se controle a través de un usuario con contraseña, si no que además, se permita el acceso únicamente a los ordenadores pertenecientes a la red local del COLFI. Proporcionando una aplicación web, se puede limitar el acceso al servidor a los equipos del área local en la que se encuentra, evitando accesos externos.

Por todo esto, una aplicación web parece la mejor opción ante las necesidades del proyecto. Se plantea entonces, un modelo cliente servidor a través de la red de área local del COLFI, el servicio de la aplicación alojado en un equipo y los clientes con acceso a través de sus navegadores (Figura 4.1).

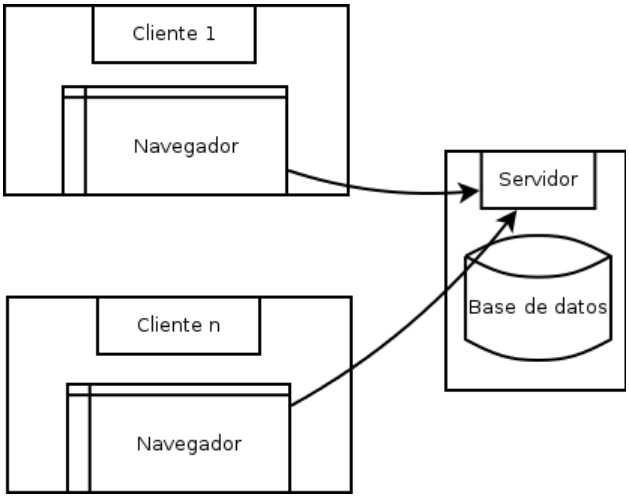

Figura 4.1: Diagrama de alto nivel del modelo de aplicación web.

4.3. Estado del arte de las aplicaciones web

Con los avances tecnológicos cada vez se demandan aplicaciones más rápidas, ligeras y robustas que permitan ser usadas sin importar el lugar u horario. Aprovechando las ventajas que brinda la red se comenzaron a desarrollar aplicaciones de servicio y de solución de problemas entre las que se encuentran las aplicaciones Web.

Por la gran expansión que ha tenido Internet así como el uso de las Intranet corporativas, ha habido una evolución en la necesidad de información de las organizaciones, que ha provocado la proliferación del cambio de las aplicaciones de escritorio por las aplicaciones web.

En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una Intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Todas las aplicaciones web tienen dos bloques fundamentales, el *Front-end* o parte visual con la que lidia el usuario final y el *Back-end* o lógica interna de la aplicación. Cada una de estas partes puede ser desarrollada en distintos lenguajes de programación.

4.3.1. Programación del lado cliente (*Front-end*)

Los lenguajes de programación utilizados para el lado del cliente son:

HTML5

Es la quinta revisión importante del lenguaje básico de la *World Wide Web*, HTML. Todavía se encuentra en modo experimental, lo cual indica la misma W3C, aunque ya es usado por múltiples desarrolladores web por sus avances, mejoras y ventajas.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada, como los elementos <audio> y <video>. Las novedades que incorpora esta nueva versión son:

- Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con codecs para mostrar los contenidos multimedia.
- Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime,...) y facilidades para validar el contenido sin Javascript.
- Visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales). En general se deja abierto a poder interpretar otros lenguajes XML.
- Drag&Drop. Nueva funcionalidad para arrastrar objetos como imágenes.
- Añade etiquetas para manejar la Web Semántica (Web 3.0): header, footer, article, nav, time (fecha del contenido),... Estas etiquetas permiten describir cuál es el significado del contenido. Por ejemplo su importancia, su finalidad y las relaciones que existen. No tienen especial impacto en la visualización, se orientan a buscadores. Los buscadores podrán indexar e interpretar esta meta información para no buscar simplemente apariciones de palabras en el texto de la página.
- Permite incorporar a las páginas ficheros RDF / OWL (con meta información) para describir relaciones entre los términos utilizados.
- Además, ofrece versatilidad en el manejo y animación de objetos simples, imágenes etc.

- API para hacer Drag&Drop. Mediante eventos.
- API para trabajar Off-Line. Permite descargar todos los contenidos necesarios y trabajar en local.
- API de Geolocalización para dispositivos que lo soporten.
- API Storage. Facilidad de almacenamiento persistente en local, con bases de datos (basadas en SQLite) o con almacenamiento de objetos por aplicación o por dominio Web (Local Storage y Global Storage). Se dispone de una Base de datos con la posibilidad de hacer consultas SQL.
- WebSockets. API de comunicación bidireccional entre páginas. Similar a los Sockets de C.
- WebWorkers. Hilos de ejecución en paralelo.
- Estándar futuro. System Information API. Acceso al hardware a bajo nivel: red, ficheros, CPU, memoria, puertos USB, cámaras, micrófonos... Muy interesante, pero con numerosas salvedades de seguridad.

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

CSS3

Mientras que HTML nos permite definir la estructura una página web, las hojas de estilo en cascada (Cascading Style Sheets o CSS) son las que nos ofrecen la posibilidad de definir las reglas y estilos de representación en diferentes dispositivos, ya sean pantallas de equipos de escritorio, portátiles, móviles, impresoras u otros dispositivos capaces de mostrar contenidos web.

Las hojas de estilo nos permiten definir de manera eficiente la representación de nuestras páginas y es uno de los conocimientos fundamentales que todo diseñador web debe manejar a la perfección para realizar su trabajo.

CSS3 es la última versión de este lenguaje que incluye las siguientes mejoras:

- Inclusión de bordes redondeados.
- Textos con sombras.
- Asignación de multiples fondos.
- Mejora en el manejo de las tablas con la opción del estilo cebra.

- Multicolumnas.
- Vinculación mejorada de fuentes mediante el uso de la regla @font-face.
- Nuevo soporte para formatos de fuentes web:
  - WOFF.
  - Fuentes instalables sin formato TrueType y OpenType (sin conjunto de permisos incrustados).
- Funciones tipográficas avanzadas que tienen propiedades de fuentes nuevas y actualizadas:
  - La nueva propiedad font-stretch.
  - Propiedades font-weight y font-size actualizadas que hacen que su comportamiento de representación sea más coherente con otros exploradores.

JavaScript

JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Actualmente, en lugar de programar JavaScript directamente, los desarrolladores optan por utilizar bibliotecas JavaScript que proporcionan funcionalidad muy buenas pre-programadas. Una de estas bibliotecas es **jQuery**.

Con jQuery se simplifica la forma de seleccionar y manipular los elementos del DOM, de la hoja de estilos CSS y además incluye funciones para realizar peticiones AJAX.

4.3.2. Programación del lado servidor (*Back-end*)

Por otro lado, la programación del Back-End no tiene un lenguaje que se desmarque del resto como elección mayoritaria.

El servidor de bases de datos

Un servidor de bases de datos se utiliza para almacenar, recuperar y administrar los datos de una base de datos. El servidor gestiona las actualizaciones de datos, permite el acceso simultáneo de muchos servidores o usuarios web y garantiza la seguridad y la integridad de los datos. Y cuando hablamos de datos, podemos estar hablando sobre millones de elementos a los que acceden al mismo tiempo miles de usuarios.

Así como sus funciones básicas, el software de servidores de bases de datos ofrece herramientas para facilitar y acelerar la administración de bases de datos. Algunas funciones son la exportación de datos, la configuración del acceso de los usuarios y el respaldo de datos.

Los servidores de bases de datos más conocidos son Oracle, SQL Server, DB2, Sybase y MySQL.

El servidor web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI.

Lenguajes de programación del lado servidor

Existe una multitud de lenguajes concebidos o no para Internet. Cada uno de ellos explota más a fondo ciertas características que lo hacen más o

menos útiles para desarrollar distintas aplicaciones.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente.

Los lenguajes de lado servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el ASP, JSP, PERL y PHP.

4.3.3. *Frameworks* de programación web

Al comienzo, los desarrolladores web escribían cada una de las páginas a mano. Actualizar un sitio web significaba editar HTML; un rediseño implicaba rehacer cada una de las páginas, una por vez.

Como los sitios web crecieron y se hicieron más ambiciosos, rápidamente se hizo obvio que esta situación era tediosa, consumía tiempo y al final era insostenible. Un grupo de emprendedores del NCSA (Centro Nacional de Aplicaciones para Supercomputadoras, donde Mosaic, el primer navegador web gráfico, fue desarrollado) solucionó este problema permitiendo que el servidor web invocara programas externos capaces de generar HTML dinámicamente. Ellos llamaron a este protocolo Puerta de Enlace Común, o CGI, y esto cambió la web para siempre.

Ahora es duro imaginar la revelación que CGI debe haber sido: en vez de tratar con páginas HTML como simples archivos del disco, CGI te permite pensar en páginas como recursos generados dinámicamente bajo demanda. El desarrollo de CGI hace pensar en la primera generación de página web dinámicas.

Muchos de los que se dedican al desarrollo de *software* utilizan, conocen o, como mínimo, se han encontrado con el concepto de *framework* (cuya traducción aproximada sería "marco de trabajo"). Sin embargo, el concepto de *framework* no es sencillo de definir, a pesar de que un programador con experiencia captará su sentido de manera casi intuitiva, y es muy posible que esté utilizando su propio *framework* (aunque no lo llame así).

Framework es por tanto, un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. Su utilización fomenta la reutilización de código, promueve buenas prácticas de desarrollo y proporciona una reducción de tiempo en los procesos de desarrollo.

Los *frameworks* más extendidos son:

- Ruby on Rail. *Framework* MVC basado en Ruby, orientado al desarrollo de aplicaciones web.

- Django. *Framework* Python que promueve el desarrollo rápido y el diseño limpio.
- Symfony. *Framework full-stack* PHP.
- NodeJS. *Framework* que utiliza JavaScript para la programación del lado servidor de la aplicación.

4.4. El *framework* de desarrollo web Django

Django es un *framework* web de código abierto escrito en Python, que respeta el paradigma conocido como Modelo-Vista-Controlador, permitiendo construir aplicaciones web más rápido y con menos código.

Django permite construir en profundidad, de forma dinámica, sitios interesantes en un tiempo extremadamente corto. Django está diseñado para hacer foco en las partes interesantes del trabajo, al mismo tiempo que alivia el dolor de las partes repetitivas. Al hacerlo, proporciona abstracciones de alto nivel de patrones comunes del desarrollo web, atajos para tareas frecuentes de programación y claras convenciones sobre cómo resolver problemas. Al mismo tiempo, Django intenta no poner restricciones al programador, permitiendo que se salga del ámbito del *framework* cuando así lo necesite.

Django fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de World Online, más tarde se liberó bajo licencia BSD. Django se centra en automatizar todo lo posible y se adhiere al principio DRY (Don't Repeat Yourself)[2][3].

Las principales características de Django son[3]:

- Un mapeador modelo E/R a objetos python.
- Aplicaciones *Plug&Play* que pueden instalarse en cualquier página gestionada con Django.
- Una API de acceso a la base de datos robusta.
- Un sistema incorporado de "vistas genéricas"que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.

- Un sistema *middleware* para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

Todas estas características sumadas a mi conocimiento previo del *framework*, hacen que sea la elección más apropiada para el desarrollo de la herramienta planteada en el proyecto.

4.4.1. Base de datos a partir de clases python

A pesar de que se puede utilizar Django sin base de datos, este proporciona un mapeo de objetos a modelo relacional, con el cuál puedes describir la base de datos con código python. Además Django permite seleccionar el lenguaje de programación en el cual se creará la base de datos tras el mapeo de los objetos creados en python.

4.4.2. API automática de acceso a los datos

Por otra parte al crear la base de datos, Django crea, sin necesidad de codificar, una API de acceso a tus datos.

Mediante la importación python de las clases declaradas para crear la base de datos, Django permite extraer e insertar datos en las entidades correspondientes de forma fácil y sin necesidad de formular *queries*.

4.4.3. Portal de administración auto generado

Una vez el modelo esta definido, Django puede automáticamente crear una interfaz de administración profesional y lista para la producción. Este portal de administración permite que un usuario autenticado añada, modifique o borre objetos.

La filosofía es evitar que el desarrollador implemente una *interfaz backend* sólo para manejar el contenido del sitio web. La interacción típica es que el desarrollador cree los modelos de la base de datos y los dé de alta en el sitio de administración, tan rápido como sea posible para que el personal de

administración(o los clientes) puedan comenzar a subir contenido. Entonces, el desarrollador decide como el contenido se muestra al público.

El diseño del portal de administración es completamente customizable para adaptarse a las necesidades del proyecto.

4.4.4. Las URL's

Tener un buen esquema de URL's dentro de un sitio web es importante para una buena calidad del portal web. Django proporciona un diseño limpio de URL's que elimina extensiones de fichero y demás caracteres innecesarios.

Django ofrece un mapeo entre las URL's y las funciones *callback* python de la aplicación, mediante la creación de tablas. El desarrollador define con expresiones regulares qué URL llama a qué función python.

4.4.5. Las vistas

Las funciones que se ejecutan en el lado servidor para atender a las peticiones web del cliente (funciones *callback*) se almacenan en un archivo y son llamadas por el manejados de URL's explicado en el apartado anterior. Estas funciones programadas en python permiten renderizar el HTML con los datos extraídos a través de la API de la base de datos.

4.4.6. Los templates

Los *templates* son las plantillas HTML que modifican su contenido en función de las peticiones del cliente. Este modelo de plantillas permite ahorrar líneas de código y duplicidad del mismo.

4.4.7. Resumen de funcionamiento

En la Figura 4.2 se aprecia el funcionamiento a vista de pájaro de este framework de programación web.

Las peticiones HTTP son atendidas por un archivo "urls.py"que identifica, mediante expresiones regulares, que servicio se está solicitando de los ofrecidos. Posteriormente se ejecutan las acciones requeridas dentro del archivo "views.py". Este archivo realiza peticiones a la base de datos, de ser necesarias, y devuelve al cliente la respuesta HTTP que incluirá un fichero HTML renderizado con los parámetros oportunos. La renderización consiste en reducir el número de líneas de código HTML utilizando código propio de django que se sustituye por el código HTML apropiado para cada servicio solicitado.

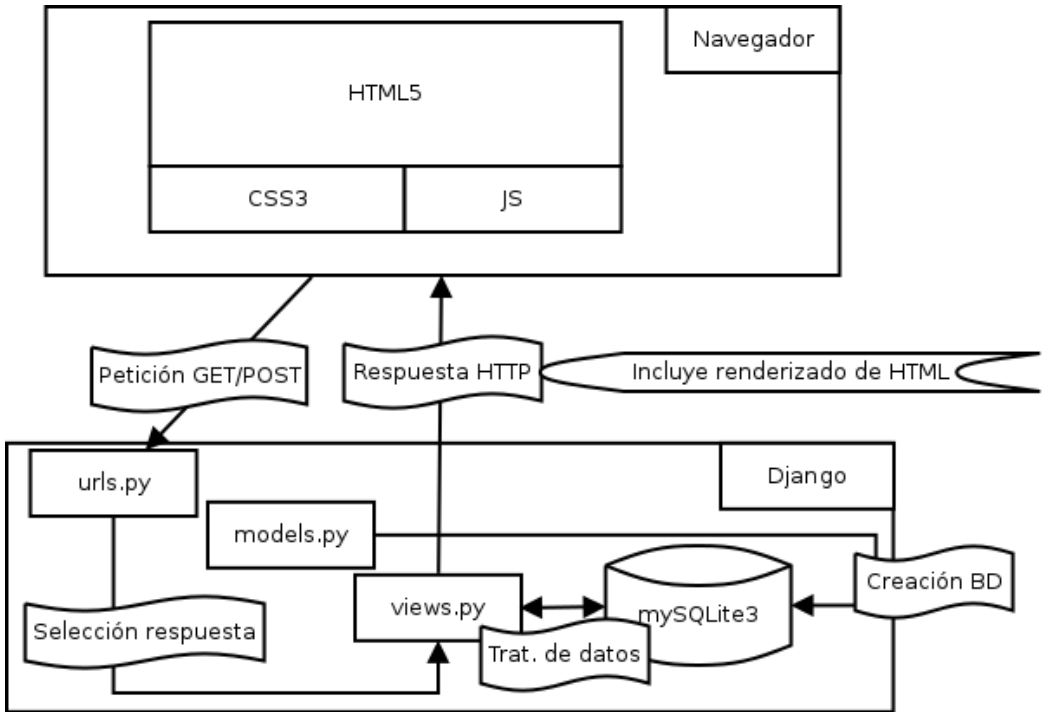

Figura 4.2: Diagrama de alto nivel del funcionamiento de Django.

Capítulo 5

Fases del desarrollo

Para la realización de este proyecto se ha elegido un **ciclo de vida clásico en cascada**, que consiste en ordenar rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida[9].

Típicamente, este cilo de vida tiene cinco fases: definición de requisitos, diseño del sistema, implementación, fase de pruebas y post-mortem. Para este proyecto, se propone un ciclo de vida en cascada con las siguientes fases:

- Fase de planificación
- Fase de definición de requisitos
- Fase de diseño
- Fase de implementación
- Fase de pruebas
- Postmortem

De cada una de estas fases se obtiene un "producto"que sirve de entrada para las siguientes fases. De la fase de planificación se obtiene la planificación de todo el proyecto, de la fase de definición de requisitos se obtiene el documento de requisitos que define los requisitos del sistema, de la fase de diseño se obtiene el diseño a vista de pájaro de la aplicación (o diseño de alto nivel) y el diseño de bajo nivel de la aplicación, de la fase de implementación se

obtiene el software, en la fase de pruebas se verifican las funcionalidad de la aplicación obteniendo un documento de comprobación de los requisitos implementados y, por último, del post-moretem se obtienen unas conclusiones de todo el proceso de desarrollo.

5.1. Fase de planificación y estandarización.

En esta fase se determinan los tiempos dedicados a cada una de las partes del proyecto. El producto generado en esta fase es el "Plan de trabajo" en el que se muestra el diagrama de Gantt del proyecto con los tiempos asignados a cada parte, hitos y productos generados. En la Figura 5.1 se puede observar claramente el ciclo de vida en cascada y la evolución temporal de las distintas fases del desarrollo.

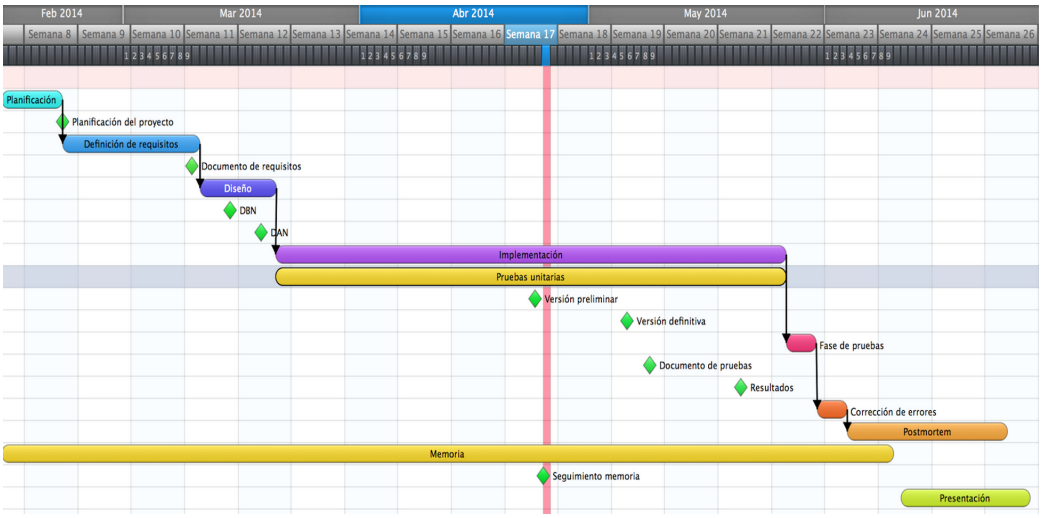

Figura 5.1: Diagrama de Gantt del proyecto.

5.2. Fase de análisis y especificación de requisitos

Durante esta fase se establecen los requisitos de la aplicación. Dichos requisitos se negocian con el usuario final en función de sus necesidades. El análisis global de los requisitos de una aplicación es un proceso de conceptualización y formulación de los conceptos que involucra la aplicación. Es

una parte fundamental del proceso de desarrollo, la mayor parte de los defectos encontrados en el software entregado se originan en la fase de análisis de requisitos, y además son los mas caros de reparar. Durante esta fase se establecen los requisitos que debe cumplir la aplicación. Dichos requisitos se negocian con el usuario final en función de sus necesidades.

Nombre del requisito	**Definición del requisito**
Req. 1	La aplicación debe mostrar un formulario para la autenticación del usuario. El formulario constará de dos campos, uno para el nombre de usuario y otro para la contraseña. Todas las URL's de la aplicación deben requerir que el usuario haya iniciado sesión.
Req. 2	Una vez autenticado el usuario, se mostrará la página principal de la aplicación.
Req. 3	La página principal de la aplicación mostrará el logo del COLFI, una página principal con un pequeño manual de uso y enlaces al perfil del usuario, a la pestaña de gestión de las prácticas externas y a cerrar la sesión.
Req. 4	El enlace al perfil de usuario mostrado en la página principal se indicará con el propio nombre del usuario y estará situado a la derecha de la pantalla.
Req. 5	El enlace al cierre de sesión estará indicado con la palabra "Salirz borrará todos los datos de sesión del usuario, redirigiendo de nuevo al formulario de autenticación especificado en el Req. 1.
Req 6.	El enlace a la página de gestión de las prácticas externas se mostrará como una pestaña con el nombre "Prácticas", al igual que la página principal que se monstrará con el nombre "Inicio".
Req 7	A la izquierda del logo del COLFI aparecerá el logo de la Universidad Politécnica de Madrid.

Sigue en la página siguiente.

Nombre del requisito	Definición del requisito
Req. 8	Cuando un usuario acceda a la pestaña de gestión de las prácticas externas se cargará en el cuerpo de la página un menú en el lateral izquierdo que mostrará las siguientes opciones: ▪ Alumnos ▪ Tutores Externos ▪ Tutores Internos ▪ Empresas ▪ Informes ▪ Certificados
Req. 9	A la derecha del menú de navegación de la pestaña de prácticas, al acceder pinchando en el enlace a la misma, se mostrará la opción de Alumnos por defecto.
Req. 10	La opción de "Alumnos"del menú lateral de la pestaña de prácticas externas mostrará la lista de los alumnos en la base de datos,
Req. 11	En la opción "Alumnos"se ofrecerá la opción de filtrar los alumnos por titulación, semestre y curso.
Req. 12	Para cada uno de los alumnos listados en la opción de "Alumnos"de las prácticas externas, se permitirá visualizar y modificar sus datos personales.
Req. 13	En la esquina inferior derecha de la opción "Alumnos"de las prácticas externas existirá un enlace al formulario de creación de un alumno nuevo.
Req. 14	En la lista de alumnos nombrada en el Req. 10 se mostrará el nombre de cada alumno, sus apellidos y su correo electrónico.
Req. 15	La opción de "Tutores externos"del menú lateral de la pestaña de prácticas externas mostrará la lista de los tutores externos en la base de datos,
Req. 16	En la opción "Tutores externos"se ofrecerá la opción de filtrar los tutores por empresa, semestre y curso.

Sigue en la página siguiente.

Nombre del requisito	Definición del requisito
Req. 17	Para cada uno de los tutores listados en la opción de "Tutores externos"de las prácticas externas, se permitirá visualizar y modificar sus datos personales.
Req. 18	En la esquina inferior derecha de la opción "Tutores externos"de las prácticas externas existirá un enlace al formulario de creación de un tutor externo nuevo.
Req. 19	En la lista de tutores externos nombrada en el Req. 15 se mostrará el nombre de cada tutor, sus apellidos y su correo electrónico.
Req. 20	La opción de "Tutores internos"del menú lateral de la pestaña de prácticas externas mostrará la lista de los tutores internos en la base de datos,
Req. 21	En la opción "Tutores internos"se ofrecerá la opción de filtrar los tutores por departamento al que está asignado, semestre y curso.
Req. 22	Para cada uno de los tutores listados en la opción de "Tutores internos"de las prácticas externas, se permitirá visualizar y modificar sus datos personales.
Req. 23	En la esquina inferior derecha de la opción "Tutores internos"de las prácticas externas existirá un enlace al formulario de creación de un tutor interno nuevo.
Req. 24	En la lista de tutores externos nombrada en el Req. 20 se mostrará el nombre de cada tutor, sus apellidos y su correo electrónico.
Req. 25	La opción de "Empresas"del menú lateral de la pestaña de prácticas externas mostrará la lista de las empresas en la base de datos,
Req. 26	En la opción "Empresas"se ofrecerá la opción de filtrar las empresas semestre y curso en los cuales han tenido alumnos realizando prácticas.
Req. 27	Para cada una de las empresas listadas en la opción de "Empresas"de las prácticas externas, se permitirá visualizar y modificar sus datos.
Req. 28	En la esquina inferior derecha de la opción "Empresas"de las prácticas externas existirá un enlace al formulario de creación de una empresa nuevo.

Sigue en la página siguiente.

Nombre del requisito	Definición del requisito
Req. 29	En la lista de empresas nombrada en el Req. 25 se mostrará el nombre de cada empresa y su correo electrónico de contacto.
Req. 30	La opción de "Informes"del menú lateral de la pestaña de prácticas externas mostrará un informe de actividad en el que se mostrará,: ▪ Número de alumnos matriculados. ▪ Nota media de los alumnos. ▪ Número de tutores externos participantes. ▪ Número de tutores internos participantes. ▪ Número de empresas participantes.
Req. 31	La opción "Certificados"del menú lateral de la pestaña de prácticas externas mostrará la configuración para generar los certificados de participación para los tutores
Req. 32	En esta opción se mostrarán dos cajas seleccionables para elegir el tutor (externo o interno) para el cuál se desea generar el informe.
Req. 33	Debe existir un botón "Generar certificado"que al ser pulsado abra en el navegador el certificado para el tutor previamente seleccionado en formato pdf descargable.

Sigue en la página siguiente.

Nombre del requisito	Definición del requisito
Req. 34	De los alumnos se debe almacenar: ▪ Nombre. ▪ Apellidos. ▪ Correo electrónico. ▪ Teléfono de contacto. ▪ DNI. ▪ Número de matrícula. ▪ Curso y semestre de matriculación. ▪ Nota final obtenida. ▪ Titulación. ▪ Nombre de la asignatura de prácticas externas. ▪ Nombre y apellidos de su tutor externo. ▪ Nombre y apellidos de su tutor interno. ▪ Nombre de la empresa en la que ha trabajado.
Req. 35	De los tutores externos se debe almacenar: ▪ Nombre. ▪ Apellidos. ▪ Correo electrónico. ▪ Teléfono de contacto. ▪ Empresa a la que pertenece.

Sigue en la página siguiente.

Nombre del requisito	Definición del requisito
Req. 36	De los tutores internos se debe almacenar: ▪ Nombre. ▪ Apellidos. ▪ Correo electrónico. ▪ Teléfono de contacto. ▪ Departamento al que pertenece.
Req. 37	De las empresas se debe almacenar: ▪ Nombre. ▪ CIF. ▪ Fecha en la que se realizó el convenio. ▪ Nombre de la persona de contacto. ▪ Correo electrónico de contacto. ▪ Teléfono de contacto.

Tabla 5.1: Listado de requisitos.

5.3. Fase de diseño

De esta fase se obtiene el diseño de alto nivel del sistema y el diseño de bajo nivel. Esta fase es fundamental para el proceso de desarrollo puesto que una buena definición del diseño del sistema facilita mucho la programación del mismo.

5.3.1. Diseño de alto nivel

En este diseño se puede apreciar el funcionamiento general de la aplicación. Por otro lado también se incluye el modelo entidad-relacción de la base de datos. La Figura 5.2 muestra el DAN del sistema y la Figura 5.3 el DAN de la base de datos.

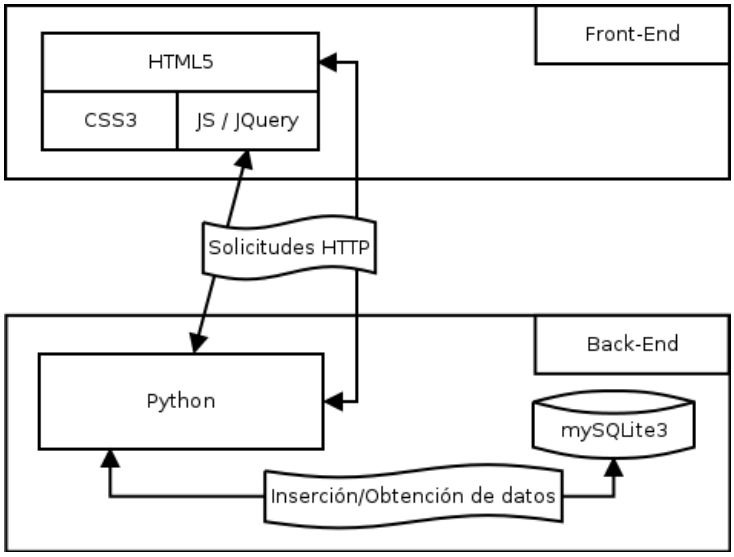

Figura 5.2: Diagrama de alto nivel del software a desarrollar.

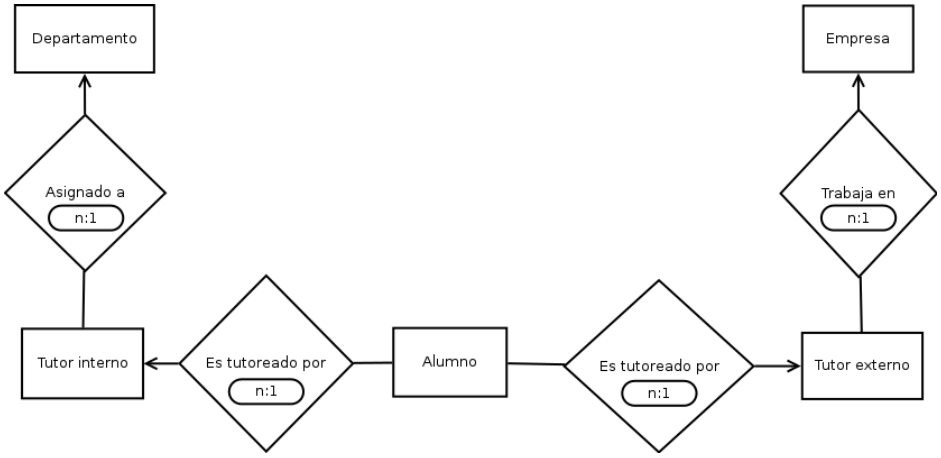

Figura 5.3: Diagrama de alto nivel del software a desarrollar.

5.3.2. Diseño de bajo nivel

En este diseño se pueden apreciar los pormenores del funcionamiento de la aplicación. Por otro lado también se incluye el diseño de la base de datos obtenido a través de un análisis minucioso de las necesidades y de los requisitos. A continuación se muestra el DAN de la base de datos:

```
 BEGIN;
CREATE TABLE "practicas_externas_empresa" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(200) NOT NULL,
    "cif" varchar(9) NOT NULL,
    "fecha_convenio" datetime NOT NULL,
    "nombre_contacto" varchar(300) NOT NULL,
    "correo_contacto" varchar(75) NOT NULL,
    "telefono_contacto" decimal NOT NULL
)
;
CREATE TABLE "practicas_externas_departamento" (
    "id" integer NOT NULL PRIMARY KEY
)
;
CREATE TABLE "practicas_externas_tutorinterno" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(50) NOT NULL,
    "apellidos" varchar(100) NOT NULL,
    "correo" varchar(75) NOT NULL,
    "telefono" decimal NOT NULL
)
;
CREATE TABLE "practicas_externas_tutorexterno" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(50) NOT NULL,
    "apellidos" varchar(100) NOT NULL,
    "correo" varchar(75) NOT NULL,
    "telefono" decimal NOT NULL,
    "empresa_id" integer NOT NULL REFERENCES "practicas_externas_empresa" ("id")
)
;
CREATE TABLE "practicas_externas_titulacion" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(200) NOT NULL
)
;
CREATE TABLE "practicas_externas_asignatura" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(200) NOT NULL,
    "creditos" decimal NOT NULL
)
;
CREATE TABLE "practicas_externas_alumno" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(50) NOT NULL,
    "apellidos" varchar(100) NOT NULL,
    "correo" varchar(75) NOT NULL,
    "telefono" decimal NOT NULL,
    "dni" varchar(9) NOT NULL,
    "numero_matricula" varchar(7) NOT NULL,
    "semestre_matricula" varchar(200) NOT NULL,
    "curso_matricula" varchar(200) NOT NULL,
    "nota" decimal NOT NULL,
    "titulacion_id" integer NOT NULL REFERENCES "practicas_externas_titulacion" ("id"),
    "asignatura_id" integer NOT NULL REFERENCES "practicas_externas_asignatura" ("id"),
    "tutorInterno_id" integer NOT NULL REFERENCES "practicas_externas_tutorinterno" ("id"),
    "tutorExterno_id" integer NOT NULL REFERENCES "practicas_externas_tutorexterno" ("id"),
    "empresa_id" integer NOT NULL REFERENCES "practicas_externas_empresa" ("id")
)
;
COMMIT;
```

5.4. Fase de implementación

Esta es la fase de desarrollo del software. Como ya se ha dicho en el capítulo anterior, se ha elegido el *framework* de programación *Django*. Con este *framework* se ha creado una base de datos a partir del diseño obtenido en la fase de diseño y se han programado las funcionalidades definidas en el documento de requisitos de la fase de requisitos.

Para el desarrollo de las funcionalidades se ha seguido la Tabla 5.1, procurando aprovechar la similitud entre tareas para ahorrar tiempo y líneas de código.

Autenticación de usuario

En la Figura 5.4 se puede ver el resultado de la implementación del requisito X. El formulario consta de dos campos de introducción de texto, uno para el nombre del usuario y otro para la contraseña (que oculta los caracteres introducidos), y de un botón para enviar los datos. Una vez enviados los datos, se comprueba la validez usuario-contraseña y se redirige al usuario a la pantalla principal de la aplicación.

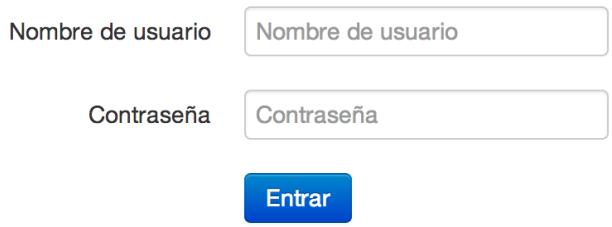

Figura 5.4: Pantalla para la autenticación del usuario.

Pantalla de inicio

En la pantalla principal de la aplicación (Figura 5.5) se muestra un pequeño manual de usuario para la aplicación, tal y como se especifica en los requisitos.

Como se aprecia en la Figura 5.5 sólo es necesario hacer click en la pestaña "Prácticas" para acceder al apartado de gestión de las prácticas externas. La disposición de los elementos en esta pestaña sigue las necesidades especificadas en la tabla de requisitos.

Con el menú de la derecha de la Figura 5.6 se puede acceder a la gestión de los datos de las distintas entidades (alumnos, tutores externos, tutores internos y empresas) y a la generación de certificados y de informes de actividad.

Figura 5.5: Pantalla de inicio de la aplicación.

Gestión de alumnos

En la pantalla de los alumnos podemos ver el listado total del los mismos, filtrarlos por titulación, semestre y curso, añadir un alumno nuevo y ver y editar los datos de otro.

Figura 5.6: Pantalla de gestión de los datos de los alumnos.

Gestión de tutores externos

En la pantalla de los tutores externos podemos ver el listado total del los mismos, filtrarlos por empresa, semestre y curso, añadir un tutor nuevo y ver y editar los datos de otro (Figura 5.7).

Figura 5.7: Pantalla de gestión de los datos de los tutores externos.

Gestión de tutores internos

En la pantalla de los tutores interno podemos ver el listado total del los mismos, filtrarlos por departamento, semestre y curso, añadir un tutor nuevo y ver y editar los datos de otro (Figura 5.8).

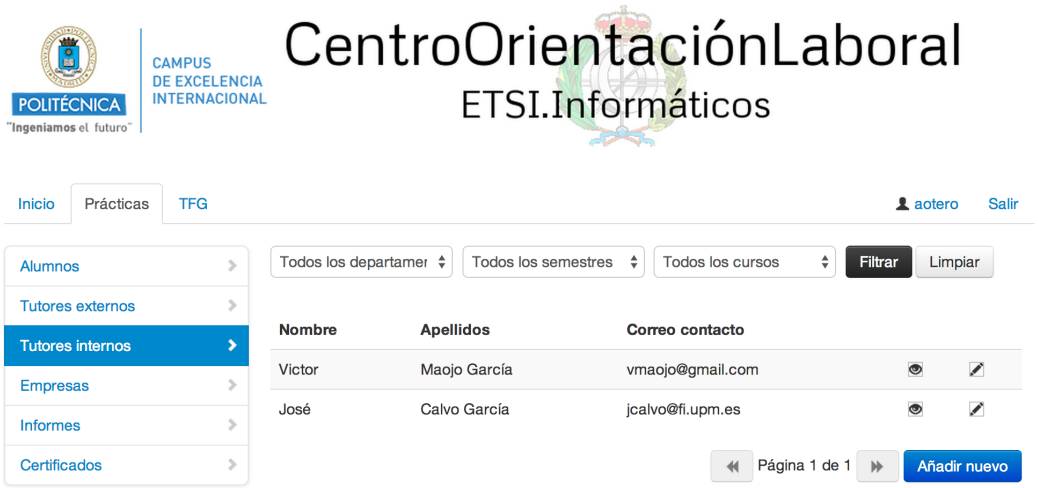

Figura 5.8: Pantalla de gestión de los datos de los tutores internos.

Gestión de empresas

En la pantalla de las empresas podemos ver el listado total del las mismas, filtrarlas por semestre y curso, añadir una empresa nueva y ver y editar los datos de otra (Figura 5.9).

Figura 5.9: Pantalla de gestión de los datos de las empresas.

Generación de informes de actividad

En la pantalla de generación de informes de actividad se distribuyen los distintos objetos y menús siguiendo los requisitos establecidos como se puede ver en la Figura 5.10.

Figura 5.10: Pantalla de generación de los informes de actividad.

Generación de certificados

En la pantalla de generación de informes de actividad se distribuyen los distintos objetos y menús siguiendo los requisitos establecidos como se puede ver en la Figura 5.11.

Figura 5.11: Pantalla de generación de los certificados a los tutores.

Modales de visualización, edición e introducción

La edición y visualización de los datos de todas las entidades de la base de datos y la introducción de una nueva se realiza a través de modales desplegados con JavaScript que muestran formularios auto-generados por *Django*. Este tipo de formularios permite automáticamente con una mínima programación realizar todas las tareas de comprobación y visualización de errores necesarias a partir de clases python. En las figuras 5.12, 5.13 y 5.14, podemos ver los modales de visualización, edición y creación de alumnos respectivamente.

Los modales son pantallas emergentes generadas con CSS y cuyo comportamiento está controlado con JavaScript.

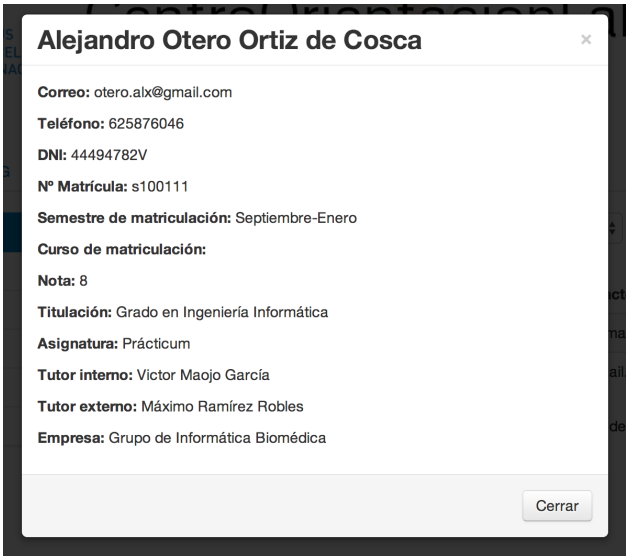

Figura 5.12: Modal para ver los datos de los alumnos.

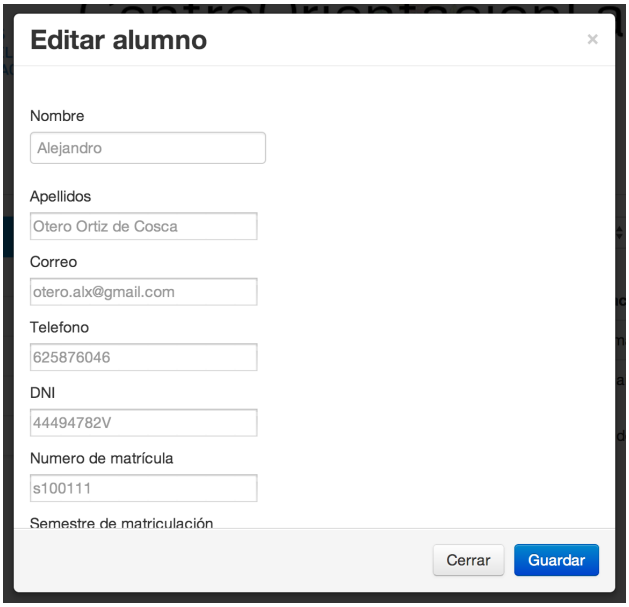

Figura 5.13: Modal para editar los datos de los alumnos.

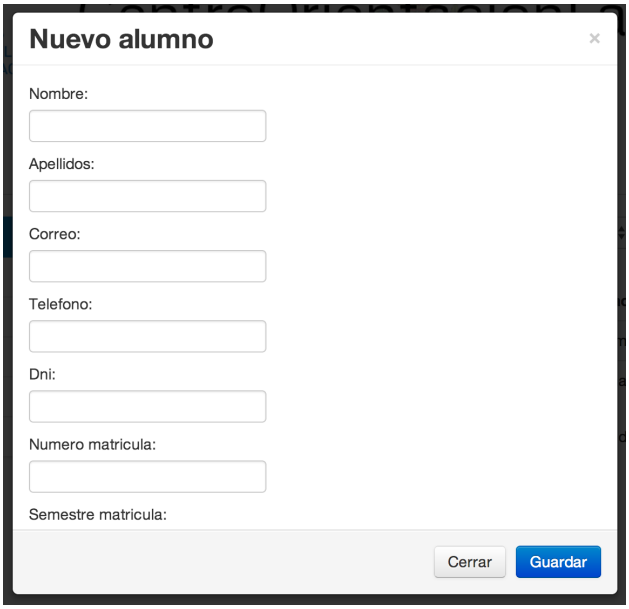

Figura 5.14: Modal para incluir un nuevo alumno.

Estos modales son desplegados tanto para los alumnos como para los tutores y las empresas de igual forma.

El funcionamiento es sencillo, cuando el usuario pincha en el enlace de visualización de datos de un alumno, tutor o empresa se despliega un modal como el de la Figura 5.12, este modal se puede cerrar pinchando fuera del mismo o con el botón cerrar.

Por otro lado, cuando un usuario pincha en el enlace a la modificación de los datos de una entidad, se despliega un modal como el de la Figura 5.13, en el cual se muestran los datos actuales del usuario y se permite modificarlos enviando el formulario.

Por último si el usuario pincha en el botón de creación de una nueva entidad, se muestra un modal como el de la Figura 5.14, en el cual se ve el formulario de creación de la nueva entidad y los botones de cerrar el modal y de enviar el formulario.

Perfil de usuario de la aplicación

A la derecha de la Figura 5.6 está enlace de desconexión y la pestaña de acceso a los datos del usuario autenticado, haciendo click en esta pestaña veremos la pantalla de la Figura 5.15. En esta pantalla podemos modificar (como también se especifica en los requisitos) la contraseña del usuario y su email, además podemos visualizar la información actual del usuario. Para

modificar cualquier otro dato será necesario acceder al portal de administración y solamente un usuario con privilegios suficientes podrá acceder. Los usuarios con privilegio de administración podrán ver un botón verde en la pantalla del usuario que los redirigirá al portal de administración.

Figura 5.15: Pantalla de gestión de los datos del usuario.

5.5. Fase de pruebas

La fase de pruebas tiene como base el plan de pruebas diseñado durante la fase de planificación. A partir de este plan se obtiene el documento de revisión del plan de pruebas y se corrigen los errores detectados.

Para realizar todas las pruebas se introducen datos de prueba en la base de datos.

5.5.1. Plan de pruebas

Se ha planteado el desarrollo de las pruebas como pruebas unitarias durante la fase de implementación, de tal forma que según se va desarrollando el

sistema se van realizando pruebas para comprobar el correcto funcionamiento de las funcionalidad. Con este modelo de *gesteo*, se evitan errores muy grandes con una complejidad de resolución igualmente grande y se ahorra tiempo al compatibilizar las pruebas de partes similares del sistema.

Por otra parte, al final del desarrollo se realizaran los siguientes casos de prueba para comprobar la funcionalidad general:

- **Casos de éxito**:
  - Caso 1: Caso de éxito de autenticación.
  - Caso 2: Caso de éxito de filtrado de los alumnos por la titulación.
  - Caso 3: Caso de éxito de filtrado de los alumnos por semestre.
  - Caso 4: Caso de éxito de filtrado de los alumnos por curso.
  - Caso 5: Caso de éxito de introducción de un nuevo alumno.
  - Caso 6: Caso de éxito de filtrado de los tutores externos por la empresa.
  - Caso 7: Caso de éxito de filtrado de los tutores externos por semestre.
  - Caso 8: Caso de éxito de filtrado de los tutores externos por curso.
  - Caso 9: Caso de éxito de introducción de un nuevo tutor externo.
  - Caso 10: Caso de éxito de filtrado de los tutores internos por la departamento.
  - Caso 11: Caso de éxito de filtrado de los tutores internos por semestre.
  - Caso 12: Caso de éxito de filtrado de los tutores internos por curso.
  - Caso 13: Caso de éxito de introducción de un nuevo tutor interno.
  - Caso 14: Caso de éxito de filtrado de las empresas por semestre.
  - Caso 15: Caso de éxito de filtrado de las empresas por curso.
  - Caso 16: Caso de éxito de introducción de una nueva empresa.
  - Caso 17: Caso de éxito de generación de informe de actividad combinando los tres distintos filtros.
  - Caso 18: Caso de éxito de generación de certificado para tutor externo.
  - Caso 19: Caso de éxito de generación de certificado para tutor interno.

- Caso 20: Caso de éxito de cambio de contraseña del usuario.
- Caso 21: Caso de éxito de cambio del correo electrónico del usuario.

- **Casos de error**:
  - Caso 1: Caso de error de autenticación.
  - Caso 2: Caso de error introduciendo campos erróneos en el formulario de creación de alumno.
  - Caso 3: Caso de error dejando campos obligatorios en blanco en el formulario de creación de alumno.
  - Caso 4: Caso de error introduciendo campos erróneos en el formulario de creación de tutor externo.
  - Caso 5: Caso de error dejando campos obligatorios en blanco en el formulario de creación de tutor externo.
  - Caso 6: Caso de error introduciendo campos erróneos en el formulario de creación de tutor interno.
  - Caso 7: Caso de error dejando campos obligatorios en blanco en el formulario de creación de tutor interno.
  - Caso 8: Caso de error introduciendo campos erróneos en el formulario de creación de empresa.
  - Caso 9: Caso de error dejando campos obligatorios en blanco en el formulario de creación de empresa.
  - Caso 10: Caso de error de cambio de contraseña del usuario.
  - Caso 11: Caso de error de cambio del correo electrónico del usuario.

5.5.2. Ejecución de las pruebas y resultados

Para la ejecución de las pruebas se sigue el siguiente proceso: se realiza cada caso de prueba tres veces, en caso de error en cualquiera de las ejecuciones se entiende que existe un error, se corrige y se vuelve a ejecutar el mismo caso otras tres veces. En caso de tres ejecuciones correctas se entiende que la funcionalidad está correcta y se pasa a la ejecución del siguiente caso de prueba.

Gracias a la ejecución de pruebas unitarias, el resultado de la ejecución de los casos de prueba ha tenido una tasa de éxito de un 100 %, debido a que la mayoría de los casos de prueba habían sido comprobados en pruebas unitarias durante la implementación.

5.5.3. Corrección de errores

El tiempo invertido en corrección de errores finalizada la implementación ha sido nulo. Todos los errores se han corregido durante la fase de implementación al ser detectados durante las pruebas unitarias.

5.6. Post-mortem

En la fase de Post-mortem se obtienen las conclusiones de desarrollo del proyecto, las dificultades encontradas, los puntos fuertes y débiles y las lecciones aprendidas. En el Capítulo 7 se exponen estas conclusiones.

Capítulo 6

Líneas futuras

Las posibles líneas futuras del proyecto son:

- Mejorar los informes de actividad, incluyendo nuevos datos de las entidades, nuevas relacciones y mejores manipulaciones de los mismos, de tal forma que se generen informes más interesantes con gráficas y otras cosas que puedan interesar a los miembros del COLFI.
- Implementar la creación de nuevos usuario para el portal a través de la propia interfaz de la aplicación y no a través del portal de administración generado por Django.
- Mejorar la interfaz del portal de administración para hacerla más intuitiva, eliminando opciones innecesarias e incluyendo otras más necesarias.
- Incluir un módulo para el control de los alumnos que realizan su Trabajo de Fin de Grado en una empresa externa.
- Incluir un módulo para el control de los alumnos que trabajan en empresas a la salida de la carrera, permitiendo comprobar si realizaron sus prácticas en esa empresa o su Trabajo de Fin de Grado.

Gracias a la utilización de Django, la inclusión de estas nuevas funcionalidades no implicaría la modificación del trabajo realizado. Además las modificaciones en el portal de administración están ampliamente documentadas por los desarrolladores de Django.

Capítulo 7

Conclusiones

Con este proyecto se ha realizado una herramienta web de gestión de las prácticas externas en la Escuela Técnica Superior de Ingenieros Informáticos. Esta herramienta mejora el proceso de asignación de prácticas, generación de informes y generación de certificados de acreditación a tutores, facilitando a los miembros del Centro de Orientación Laboral la introducción de datos de las entidades participantes en el proceso. Por otro lado, al haberse desarrollado una herramienta web, los datos están centralizados en un servidor evitando duplicidad de los mismos y copias desactualizadas. Además, se permite el acceso y la manipulación de todos estos datos de forma simultánea por varios usuarios.

El desarrollo de una aplicación web otorga a la herramienta final de grandes ventajas que la hacen ser la mejor opción frente a otras como una aplicación móvil o una aplicación de escritorio.

El uso del *framework* de programación web Django para el desarrollo, facilita las tareas del desarrollador y que introduce modularidad en el proyecto haciendo fácil su posterior ampliación funcional. Con el uso de *frameworks*, el desarrollo ahorra mucho tiempo de implementación, sistematizando tareas comunes como la generación de formularios a través de entidades de una base de datos.

Para la interfaz de usuario se ha procurado utilizar la tecnología más actual para dotar a la aplicación de una apariencia agradable y actual. A su vez, utilizando JavaScript, se libera al servidor de cierta carga trasladándola al lado del cliente; evitando así retardos por tiempo de espera en respuestas HTTP.

Apéndices

Apéndice A

Requisitos de instalación

A continuación se listan los requisitos de instalación de la aplicación en un equipo:

1. Sistema operativo linux.
2. Servidor nginx.
3. Django.
4. PISA para el tratamiento de pdf's a través de Django (y sus dependencias).

Apéndice B

Inclusión de nuevos módulos

En este apéndice se muestra el proceso a seguir para poder incluir nuevos módulos a la aplicación web y así poder ampliar las funcionalidades de la misma.

1. Abre una consola en el servidor.
2. Acede al directorio de instalación del proyecto "COLFI".
3. En el directorio raiz comienza el nuevo módulo con:

   ```
   django-admin.py startapp ‘‘nombre del nuevo módulo"
   ```

   Esto creará un nuevo directorio con el nombre del nuevo módulo. Es en esta carpeta donde se debe incluir toda la nueva lógica deseada para la aplicación.

Adicionalmente se instalará en el servidor que contenga la aplicación el software de control de versiones *Git* que permitirá descargarse el código de la aplicación para modificarlo sin provocar modificaciones en el código del servidor, *testear* e incluir las modificaciones requeridas incluyendo la creación de los nuevos módulos y sus funcionalidades.

Para aprender el funcionamiento del software de control de versiones *Git* acceda a la página web del proyecto `http://git-scm.com/`

Bibliografía

[1] Sitio web de Django: https://www.djangoproject.com

[2] Sitio web de Django en español: https://www.django.es

[3] Software de desarrollo web Django: http://es.wikipedia.org/wiki/Django_(framework)

[4] Maestros del web: Curso Django para perfeccionistas con deadlines.

[5] Estado del arte de metodologías, herramientas y lenguajes para el desarrollo de aplicaciones Web: http://www.monografias.com/trabajos98/estado-del-arte-metodologias-herramientas-y-lenguajes-desarrollo-aplicacione estado-del-arte-metodologias-herramientas-y-lenguajes-desarrollo-aplicacione shtml

[6] El lenguaje de programación HTML5: http://es.wikipedia.org/wiki/HTML5

[7] Los 11 mejores frameworks gratuitos para aplicaciones web: http://elbauldelprogramador.com/los-10-mejores-frameworks-gratis-de-aplicaciones-web

[8] Artículo sobre Django: http://www.linuxlinks.com/article/20120525000545879/Django.html

[9] Ciclo de vida en cascada: http://es.wikipedia.org/wiki/Desarrollo_en_cascada

[10] Una introducción a los servidores de bases de datos: http://blog.iweb.com/es/2014/04/servidores-de-bases-de-datos/2487.html

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Wed Jun 25 21:51:25 CEST 2014
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)