UNIVERSIDAD
REY JUAN CARLOS

INGENIERIA DE TELECOMUNICACIONES

Curso Académico 2016/2017

Proyecto Fin de Carrera

WEBAPP SOPA: SISTEMA DE OPINION SOBRE PRACTICAS DE ALUMNOS

Autor : Gregory Joel Sparton Alcobendas

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

WEBAPP SOPA: SISTEMA DE OPINION DE PRACTICAS DE ALUMNOS

Autor : Gregory Joel Sparton Alcobendas

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2017, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

Dedicado a

mi familia

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se realiza una breve introducción sobre el contenido de esta memoria, las webs de nuestros días y su evolución en el tiempo

1.1. Motivación

Este proyecto busca continuar el camino que inician los foros web tradicionales fusionado con las funcionalidades de las redes sociales en un entorno Web 2.0, consiguiendo la colaboración entre alumnos para formar una biblioteca de opiniones y encuestas sobre la calidad y contenido de sus prácticas en empresa. La idea principal es la de conseguir resolver las principales dudas que puedan surgir a la hora de realizar unas prácticas en una determinada empresa. Existen foros y webs donde los usuarios reflejan opiniones sobre su experiencia laboral pero suelen ser sitios donde la gente vierte sus críticas sin objetividad.

1.1.1. SOPA: Sistema de Opinión sobre Prácticas de Alumnos

Esta web busca servir de ayuda a la hora de elegir o afrontar unas prácticas en empresa. La idea es que los alumnos con experiencia laboral reflejen sus impresiones y condiciones laborales para que sirvan de ayuda a sus compañeros. Estas cuestiones pueden ser de diversa índole, desde los conocimientos que son necesarios a la hora de trabajar a si se trata de prácticas remuneradas pasando por las relaciones con el tutor en la empresa o los compañeros. Todo con la idea de elegir las prácticas que mejor se adapten a las necesidades e inquietudes del alumno.

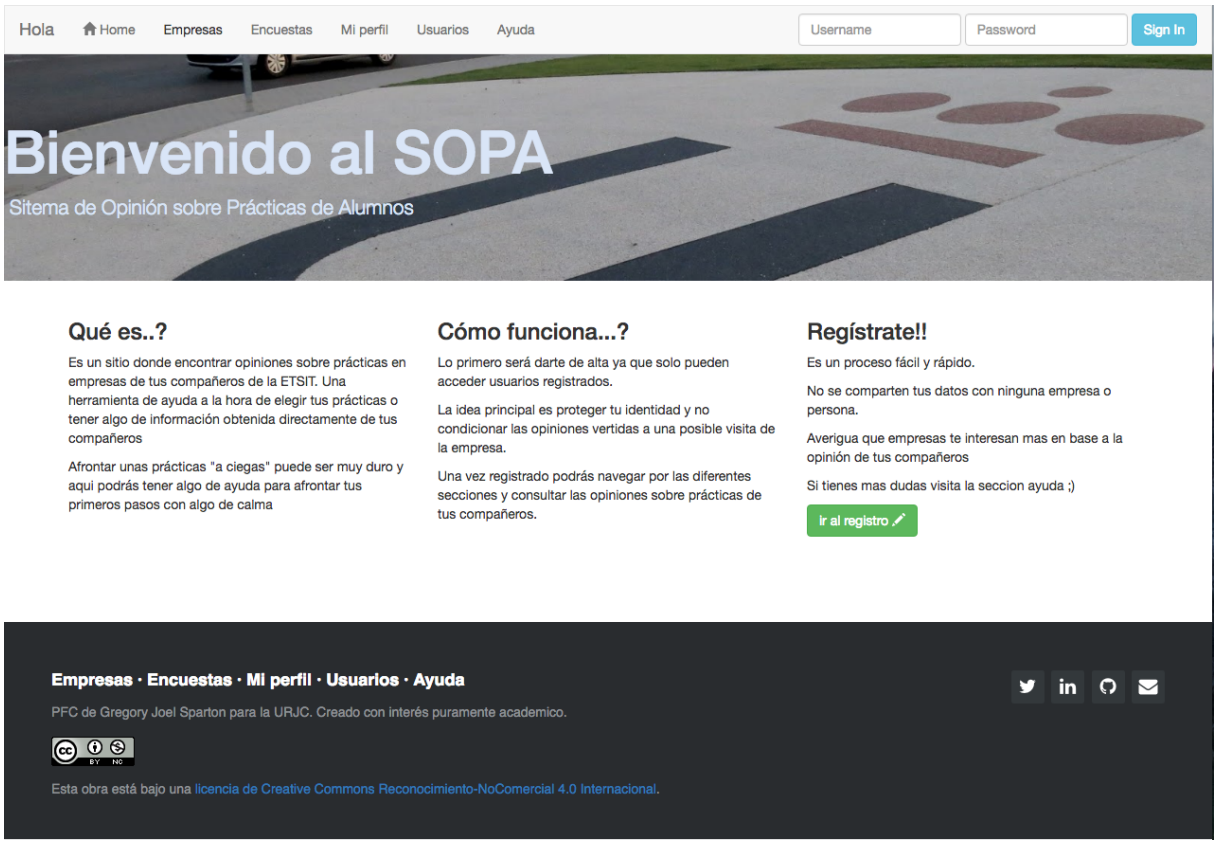

Figura 1.1: Home del sitio SOPA

1.2. Prácticas en empresa

1.2.1. ¿Qué son?

Las prácticas en empresa son un periodo corto de tiempo en el que el alumno se integra en la empresa con unas condiciones diferentes a las de un empleado. El objetivo es la iniciación en el ejercicio de actividades profesionales relacionadas principalmente con su formación permitiendo que el alumno conozca de primera mano algunos aspectos de una futura vida laboral y obtenga alguna experiencia que incorporar a su curriculum.. A día de hoy las prácticas son un componente obligatorio de la formación del alumno y deben realizarse para poder conseguir el título Universitario. También es posible realizar prácticas en empresa de manera extracurricular para completar el expediente del alumno.

1.2.2. Condiciones laborales

Tal y como se ha mencionado anteriormente el objetivo de las prácticas es la integración del alumno, durante un periodo de tiempo determinado, dentro de la empresa. Las condiciones no son las mismas que la de una relación laboral estándar. El alumno irá asumiendo responsabilidades a medida que adquiera conocimientos, sus tareas deben ser supervisadas por su tutor en la empresa y en ningún caso el alumno debe servir de remplazo de un trabajador. Todas las condiciones se indican en reglamento de esta universidad, adaptado a la reciente normativa aprobada por el Gobierno de la Nación, mediante Real Decreto 592/2014, de 11 de julio, por el que se regulan las prácticas académicas externas de los estudiantes universitarios. Las prácticas no son obligatoriamente remuneradas. Opcionalmente la empresa puede otorgar al alumno algún tipo de remuneración (económica, de transporte, bono comida, etc.) siendo esto algo habitual en las empresas del sector de las telecomunicaciones.

1.3. Internet hoy

1.3.1. Proceso evolutivo

Desde la aparición de las primeras webs hasta el día de hoy la evolución de las mismas ha sido sustancialmente notable. De simples páginas llenas de texto se ha dado paso a sitios que se comportan de diferente manera en función del dispositivo que se utiliza para acceder a dicho sitio o de quien es la persona que lo visita. Es la evolución de la **Web 1.0** a la **Web 2.0** donde el usuario tiene un papel relevante, más allá de ser un simple observador el usuario interactúa con el sitio para aportar su información o para obtener la información que necesita. Cuando se habla de **Web 2.0** se hace referencia a la segunda generación de las páginas web que son el resultado de la conversión del las antiguas webs, de texto plano y escasa o nula interacción donde el usuario es un simple lector, en webs de contenido dinámico y personalizado en función del usuario, buscando en muchas ocasiones la interacción entre usuarios y con cada vez más frecuencia y eficacia buscando de facilitar todo tipo de tareas al usuario. El incremento del ancho de banda em las conexiones de internet y de la capacidad de cómputo de los dispositivos ha propiciado un aumento exponencial del tráfico generado en internet. Se ha pasado de webs de texto plano a web repletas de contenido multimedia.

1.3.2. WebApps

Paralelamente a la evolución de la Web, la aparición de los *smartphones* y posteriormente de las *tablets* supuso la aparición de las tiendas de aplicaciones móviles. En ellas se venden aplicaciones de todo tipo, de juegos a *suites* de ofimática o diseño. El principal problema de estas aplicaciones es que deben ser programadas para diferentes sistemas operativos y dispositivos, lo que resulta bastante costoso a la hora de desarrollarlas. En este punto es donde convergen la **Web 2.0** y las aplicaciones, dando como resultado lo que conocemos como **WebApps**. En su forma más sencilla las **WebApps** son páginas adaptadas al dispositivo desde el que se visitan limitando o cambiando su comportamiento y forma de interacción con el usuario. Actualmente una tendencia creciente es desarrollar WebApps complejas que evitan el problema de las aplicaciones nativas, funcionan en cualquier dispositivo sin necesidad de instalarse en ningún momento, y solo exigen acceder desde un navegador web medianamente actualizado. Los ejemplos más

representativos con los clientes de correo. Hoy en día son pocos los usuarios o empresas que no utilizan un cliente web (WebApp), al menos de manera paralela con una aplicación nativa, para gestionar su correo, organizar citas, etc. Uno de los más notables en el caso de correo es *Gmail*, servicio de correo de **Google**. Hay todo tipo de software como las *suites* de ofimática o redes sociales que tienen una versión WebApp paralela a la aplicación nativa o directamente sólo son accesibles en su forma **WebApp**. Actualmente se puede asegurar que todo sitio relevante de internet ha sido transformado en **WebApp**. Sitios como **Wikipedia**, **Amazon**, **YouTube**, **Facebook**, **Google** son su *Google Docs*, o **Apple** y su *iCloud* son solo unos poco ejemplos. En todo caso estas aplicaciones suponen como mucho la instalación de algún complemento para el navegador o simplemente la utilización de un navegador determinado como es el caso de **Apple**.

Figura 1.2: WebApps

1.4. Estructura de la memoria

Esta memoria está compuesta por seis bloques correspondientes a seis capítulos:

- En el primer capítulo se hace una introducción al proyecto. Se indica el contexto y la
- En el capítulo 2 se muestran los objetivos del proyecto.
- A continuación se presenta el estado del arte.
- En el capítulo 4 se explican los procesos de diseño e implementación.
- Los resultados se muestran en el capítulo 5.
- Después, en el capítulo 6 se presentan las conclusiones.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo general de este proyecto es la elaboración y puesta en marcha de una aplicación web (**WebApp**) con dos funciones: elaborar una base de datos de encuestas que contengan las condiciones y opiniones de los alumnos sobre sus prácticas en empresa y de forma paralela que esas opiniones sean accesibles a otros alumnos y les sean útiles a la hora de elegir una empresa donde realizarlas, o sirvan de respuesta ante las dudas que puedan surgir una vez que sean seleccionados.

2.2. Objetivos específicos

Los objetivos específicos que se persiguen en este proyecto son, por orden de importancia:

- **Hacer uso de tecnologías actuales como HTML5, CSS y JavaScript:** Éstas, junto a **PHP y MySQL** son los pilares principales de cualquier web actual. Mediante la interacción entre estas tecnologías se construyen las principales WebApps y sitios de internet. Actualmente el dominio de estas tecnologías es una condición con alta demanda en ciertos sectores del mundo laboral lo que es un gran aliciente para mejorar en su manejo.

- **Facilitar el manejo de información en la aplicación:** Conseguir que generar y consultar información en la aplicación sea intuitivo y no provoque dudas a la hora de hacerlo. Por otra parte se busca que consultar información sea rápido y generar opiniones mediante las

encuestas sea un proceso breve y eficiente.

- **Componer una interfaz atractiva:** Mejorar en el manejo de **Bootstrap** para trabajar con **CSS3** y crear una interfaz que resulte atractiva y agradable a la vista e intuitiva a la hora de utilizarla.
- **Que la interfaz sea responsiva:** Que todos los elementos se muestren de manera correcta y agradable para el usuario en función del dispositivo.
- **Integrar Google Maps:** Aprovechar la versatilidad de uso de la API de Google,una de las más reconocidas, para generar mapas que muestren la ubicación de las empresas.

Capítulo 3

Estado del arte

A continuación se describen las tecnologías utilizadas para realizar este proyecto profundizando de manera breve en el estatus actual de su evolución.

3.1. Python y Django

Figura 3.1: Logo de Django y Python

Python[1]. es un lenguaje de programación de alto nivel y con un enfoque simple pero efectivo a la programación orientada a objetos. Sus principales características son:

- Se trata de un lenguaje interpretado que no necesita compilación. Al ejecutar el código fuente el interprete lo transforma en un lenguaje comprensible para la máquina. Esto convierte a Python en un lenguaje portable al poder ser ejecutado con independencia de la máquina donde se haga. Además facilita la depuración de código.

[1] `http://www.python.org`

- Su sintaxis es muy simple y fácil de manejar, lo que le hace fácil de aprender. Al ser de alto nivel de abstracción el programador no se debe preocupar de la gestión de memoria o registros. Se trata principalmente de escribir las instrucciones que deseas que se ejecuten en el orden correcto.

- Soporta múltiples estilos de programación, ya sea orientada a objetos, funcional o imperativa.

- Tanto el intérprete como la extensa biblioteca estándar son de código abierto y gratuito.

Django[2]. es un *framework* para crear aplicaciones web gratuito y de código abierto escrito en Python. se trata de un *WEB framework*, un conjunto de herramientas para crear aplicaciones web de manera rápida y fácil. Fue publicado bajo licencia BSD en junio de 2005 siendo actualizado con frecuencia por la Django Software Foundation. Sus características principales son:

- Respeta el patrón de diseño modelo-vista-controlador (MVC). Este modelo separa los datos de la lógica de la aplicación y fomenta la reutilización de código y el principio DRY (Don?t repeat yourself) que promueve evitar la duplicidad de código, aportando claridad y limpieza facilitando las tareas de desarrollo. El cliente genera una petición que recibe el **controlador**, este procesa la petición y solicita los datos que necesita al **modelo**, una vez que obtiene respuesta el controlador genera una **vista** como respuesta que se envía al cliente. Los desarrolladores de Django no sienten la necesidad de ceñirse a ningún paradigma por ello lo que en Django llamamos ”vista.[es] el çontrolador”del MVC y lo que para Django es una ”plantilla”para el MVC es una ”vista”. Por ello se dice que Django respeta el paradigma MTV o *Model Template View*

[2] `https://www.djangoproject.com/`

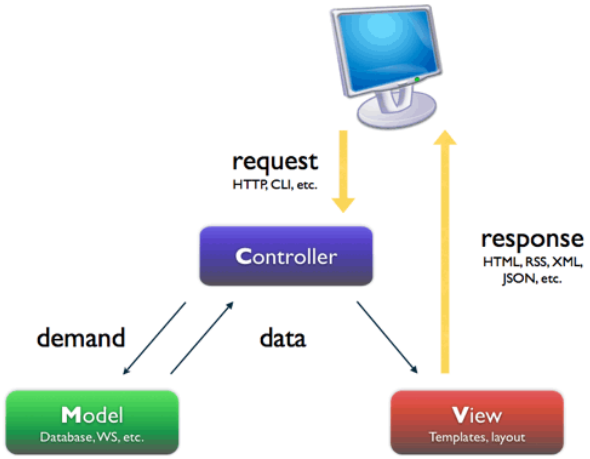

Figura 3.2: Diagrama del patrón MVC.

- Posee varios módulos previamente instalados que facilitan la gestión de nuestra aplicación. El más popular es el módulo de administración que permite administrar los usuarios, contraseñas y bases de datos con unos sencillos pasos.

- Su sistema de plantillas extensibles permite heredar código, lo que facilita la presentación de los datos ya sea en HTML o en otros formatos.

- Una robusta API de bases de datos.

- La gestión de URLs se basa en expresiones regulares que hace corresponder una URL con el código que maneja la petición.

- Utiliza Python en prácticamente todo lo que tenga que ver con el código de la parte servidor, la configuración, las vistas o los modelos de datos.

3.2. HTML5

Figura 3.3: Logo de HTML5

HTML5 es la quinta versión de *HyperText Markup Language*, lenguaje principal de internet y núcleo de la *World Wide Web*. HTML es el estándar para la elaboración de páginas web, define una estructura básica y un código, basado en etiquetas para denominar o *marcar* contenido como texto o imágenes o video. En su quinta versión se incorporan nuevos elementos y atributos acordes a los sitios web modernos, siempre en forma de etiqueta pero son significado semánticos como *audio* o ”video”para elementos multimedia, *header* para el encabezado, o *nav* para una barra de navegación. La lista de novedades es larga, algunas de las más interesantes son:

- Etiquetes nuevas como *canvas* 2d y 3d, que junto con las anteriormente mencionadas *audio* y *video* y los códecs incorporados sirven para mostrar contenido multimedia.

- Etiquetas para el manejo de la web semántica como la mencionada *header*, *footer* para el pié de página o *time* para la fecha del contenido.

- Mejoras para los buscadores, haciendo que busquen por tipos de contenido en lugar de por coincidencias de texto.

- APIs para trabajar *Off-line*, descargando el contenido necesario y trabajar en local. De geolocalización o de *Drag & Drop* para manejar archivos arrastrándolos.

- Mejoras en los formularios con nuevos tipos de datos y facilidades para validarlos.

3.3. CSS3

Figura 3.4: Logo de CSS3

CSS3, tercer nivel estandarizado del lenguaje de las hojas de estilo en cascada, *Cascading Style Sheets*, que amplia el nivel 2 que tardó 9 años en conseguir ser un estándar. Es un lenguaje de diseño utilizado para definir y modelar la presentación de un documento estructurado marcado por etiquetas, como es el caso de HTML5. Busca aislar el contenido de la presentación del mismo y define todo lo que tenga que ver con el aspecto visual, coles, estructura, estilo de las imágenes, etc. Permite mantener, con un único documento, el mismo estilo en todas las páginas de un sitio.

3.4. Bootstrap

Figura 3.5: Logo de Bootstrap

Bootstrap[3] es un *framework front-end* de código abierto que se utiliza para generar documentos CSS con el objetivo de diseñar páginas o aplicaciones web. Fue creado por y para

[3] `http://getbootstrap.com/`

Twitter aunque actualmente se ha convertido en un estándar en el diseño de todo tipo de páginas y WebApp. Dispone de una gran biblioteca de plantillas HTML, hojas de estilo CSS y *scripts* JavaScript para mejorar el diseño de los formularios, botones, barras de navegación y cualquier elemento que se desee de nuestra página o WebApp. Las principales ventajas son:

- Ahorro de tiempo a la hora de crear el diseño de una página web pues se parte de elementos prediseñados por lo que no se parte de cero.
- Con unas pocas lineas de código podemos construir una interfaz realmente agradable, con un poco más de esfuerzo se consiguen grandes resultados.
- Implementa un diseño responsivo, que hace que el diseño se adapte al dispositivo que se utiliza
- Funciona en la mayoría de los navegadores actuales.

3.5. JavaScript

Figura 3.6: Logo de JavaScript

JavaScript es un lenguaje de programación interpretado, derivado del estándar ECMAScript. Orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Desarrollado por Netscape Communication, conocido como JS, es uno de los lenguajes de *scripting* más conocidos. Aunque puede ser usado en lado del servidor, lo mas normal es que sea utilizado en le lado cliente, mejorando la interfaz de la aplicación haciendo que responda a eventos de todo tipo, como clicks del ratón o formularios, lo que resulta en páginas web dinámicas. Al ser interpretado en el lado cliente por el navegador se alivia la carga de trabajo del servidor y se reducen los tiempos de espera del usuario ante ciertas peticiones.

3.6. Google Maps

Figura 3.7: Logo de Google Maps

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc. Ofrece imágenes de mapas desplazables, fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle con Google Street View. Anunciado por primera vez en 2005 hoy en día es una de las aplicaciones mas utilizadas para tareas relacionadas con la localización. Su API ofrece multitud de posibilidades que permiten crear mapas de manera sencilla y rápida. Con sólo unas líneas de código y la ayuda de su API podemos realizar una consulta sobre un lugar a la extensa base de datos de Google y recibir una rápida respuesta que será interpretada en un mapa. Con un poco mas de trabajo podemos personalizar a nuestro antojo los mapas para mostrar la información de la manera que estimemos oportuna.

3.7. Git y GitHub

Figura 3.8: Logos de Git yGitHub

Git es un software de control de versiones pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Busca la rapidez en la gestión de ramas y mezclado de diferentes versiones.

Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework *Ruby on Rails*. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. Ofrece:

- Wiki para cada proyecto
- Página web para cada proyecto1
- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto
- Funcionalidades como si se tratase de una red social, como por ejemplo: seguidores;
- Herramienta para trabajo colaborativo entre programadores.

3.8. Pythonanywhere

Figura 3.9: Logo de Pythonanywhere

textbf Pythonanywhere es un servicio de *hosting* web con un entorno de desarrollo integrado basado en Python. Tiene como principales características:

- Múltiples versiones de Python con las que trabajar, con una multitud de librerías instaladas.
- Permite alojar y depurar aplicaciones creadas con Django y otros frameworks como Flask o web2py.

- Consolas interactivas en el navegador que permiten trabajar con el código alojado en los servidores de manera compartida con otros usuarios.
- Editores de código con marcado de sintaxis.
- Permite establecer tarea tipo *Cron* como mantenimientos o reportes.

Utilizaremos la modalidad gratuita con limitaciones en los accesos y usos de CPU pero con un margen mas que amplio para nosotros. Estas limitaciones crecen o desaparecen en las opciones de pago.

Capítulo 4

Diseño e implementación

Este capitulo describe las fases de desarrollo del proyecto, describe los módulos por lo que está formado el proyecto y se describe su funcionamiento.

4.1. Arquitectura general

La base de este proyecto es el modelo cliente-servidor. En este modelo existen dos actores con funciones diferenciadas: el cliente elabora las peticiones y las envía al servidor, pasando a esperar una respuesta; el servidor, que tiene como misión atender dichas peticiones, generar la respuesta correspondiente a dicha petición y devolverla al cliente.

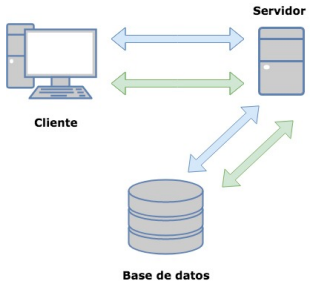

Figura 4.1: Modelo cliente servidor

Podemos considerar un tercer actor que a efectos prácticos es una abstracción del servidor, es el servidor de datos o la base de datos, donde se almacenan y se sirven los datos que el servidor procesa en la elaboración de respuestas. En este caso el servidor pasa a comportarse como cliente de la base de datos y la base de datos toma el rol de servidor. La separación entre

servidor y base de datos no está condicionada a una separación física y pueden estar alojados en la misma máquina.

4.1.1. El cliente

El **cliente** es el navegador web mediante el que el usuario de la aplicación accede a ella. Este se encarga de procesar la solicitudes del usuario que realiza escribiendo urls o pulsando en enlaces o botones. Estas acciones son transformadas por el cliente en peticiones *HTTP* que envía al servidor pasando a esperar la respuesta del mismo. Una vez que recibe la respuesta la procesa y muestra al usuario.

4.1.2. El servidor

El **servidor** es el lugar donde podemos encontrar toda la lógica de la aplicación. Para la fase de desarrollo **Django** incorpora un servidor de desarrollo, ligero y escrito en **Python**. Este Servidor no se debe utilizar en entornos de producción ya que no es para lo que está pensado. El servidor de Django está pensado para escuchar conexiones locales, puede configurarse para que se comporte de manera diferente pero no es recomendable. Este servidor recibe una petición, mediante el archivo urls.py determina que información se está solicitando, para componer la respuesta consulta el archivo views.py que contiene la lógica de la aplicación, solicita información la base de datos si es preciso y devuelve el código **HTML** que se debe enviar al cliente. Para ejecutar el servidor de desarrollo de Django basta con ejecutar desde la shell la orden *python manage.py runserver 8000* en el directorio de nuestro proyecto Django.

4.1.3. La base de datos

la **base de datos** en nuestro caso se encuetra en el mismo equipo que el servidor, tanto en la fase de desarrollo como en la de despliegue y puesta en producción. **Django** proporciona compatibilidad con un gran número de tecnologías de bases de datos. En nuestro caso como la aplicación se basa en entrada y salida de texto elegimos una opción de fácil sintaxis y totalmente integrada dentro de Django como es *SQLite3*.

4.2. Desarrollo de la aplicación

A continuación se describen las diferentes fases que se han ido completando a medida que se avanzaba en el desarrollo de la aplicación.

4.2.1. Documentación

Una vez decidido el objetivo con el que se construirá la aplicación el primer paso, como en todo proyecto, es el de documentarse tanto sobre las aplicaciones similares disponibles, si existen, como de las tecnologías necesarias para realizar la aplicación. La cantidad de contenido que se puede encontrar en internet sobre todas las tecnologías utilizadas facilita conseguir información, pero lo hace laborioso a la hora de seleccionar lo que nos interesa. Posteriormente se instala Django y se prepara el entorno de programación adecuado para elaborar el proyecto.

4.2.2. Diseño de la base de datos

Un objetivo básico de toda aplicación es el manejo de datos, en mayor o menor medida, de manera rápida y eficiente. Tener una base de datos estructurada de manera correcta es imprescindible para poder hacerlo. Buscamos que los recursos importantes solo sean accesibles para los usuarios registrados, por esto creamos un modelo que hereda de la clase *Users* predefinida de Django y le añadimos el campo *id_grado*. Esto nos facilita la tarea de trabajar con los datos y formularios de creación de cuentas y autenticación en el sistema. La tabla *grados* contiene los diferentes grados que se imparten en la URJC y se utilizan para definir el Grado que estudia o ha estudiado el usuario.

En lo referente a las **empresas** sobre las que se realizarán encuestas tenemos una tabla con el nombre de la misma, el departamento, nombre del tutor y la dirección para obtener su ubicación.

Para las **encuestas** que reflejan las opiniones de los usuarios sobre sus experiencias en prácticas tenemos la taba *encuestas*. Esta contiene todos los campos que se recopilan mediante una formulario en varios pasos.

4.2.3. Registro y Login de los usuarios

Capítulo 5

Resultados

Capítulo 6

Conclusiones

En nuestro caso el *framework* **Django** cumple todos los requisitos para crear nuestra aplicación ya que se trata de una herramienta que ya hemos utilizado previamente. Esto no evita que debamos pasar unas cuantas horas actualizando e incrementado nuestros conocimientos de la herramienta. Con **HTML5** tampoco ha habido grandes problemas una vez que se han. El esfuerzo en esta fase viene provocado por el escaso conocimiento del *framework* **Bootstrap** que en los primeros pasos puede ser algo mas complejo de lo que termina siendo una vez que se manejan bien un par de conceptos. Otro desafío ha sido actualizar los conocimientos

6.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

6.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

6.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a
2. b

6.4. Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

6.5. Valoración personal

Finalmente (y de manera opcional), hay gente que se anima a dar su punto de vista sobre el proyecto, lo que ha aprendido, lo que le gustaría haber aprendido, las tecnologías utilizadas y demás.

Apéndice A

Manual de usuario