UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado de Ingeniería en Sistemas de Comunicación

Trabajo de fin de Grado

Diseño y desarrollo en django de una aplicación web para la edición, creación y gestión de cuestionarios tipo test on-line

Autor: Eric Wikstrom Pujante
Tutor: Iria Estévez Ayres

Junio 2014

"Life is not about how to survive the storm but how to dance in the rain."

Índice general

Índice de figuras

Acrónimos

CGI	**C**ommon **G**ateway **I**nterface
CSS	**C**ascading **S**tyle **S**heets
HTML	**H**yper **T**ext **M**arkup **L**anguage
JS	**J**ava **S**cript
LOPD	**L**ey **O**rgánica de **P**rotección de **D**atos
MVC	**M**odelo **V**ista **C**ontrolador
ORM	**O**bject **R**elational **M**apping
RGBD	**R**egistro de **P**rotección de **D**atos
SGBD	**S**istema de **G**estión de**B**ase de **D**atos
SGML	**S**tandar **G**eneralized**M**arkup **L**anguaje
SQL	**S**tructured **Q**uery **L**anguage
XML	**E**xtensible **M**arkup **L**anguage

Todos los caminos conducen indudablemente a un fin. Por fortuna o por desgracia mi aventura como estudiante está aproximándose a su final. Han sido unos años llenos de momentos muy buenos y otros no tanto pero al final del camino, lo más importante es recordar aquellos buenos momentos con una sonrisa en la cara y esperar que esta nueva etapa en la que embarco me recompense con momentos aún mejores. Solo tengo palabras de agradecimiento a todas las personas que me han motivado a seguir adelante y sobre todo a mis padres por animarme en todo momento a sacar lo mejor de mi mismo.

Capítulo 1

Introducción

1.1. Motivación del proyecto

Como bien se explicará más adelante, a principios de la década de los 90 empezaron a aparecer las primeras páginas web. En un primer momento, las oportunidades de desarrollo en estas páginas eran más bien escasas y únicamente existía la posibilidad de desarrollar una página web con texto informativo.

Con el paso del tiempo el número de usuarios y programadores de páginas web fue incrementando y aparecieron nuevas tecnologías que permitían llevar el desarrollo a un nivel superior. En la última década este desarrollo ha sufrido cambios muy significativos que han permitido que los programadores puedan implementar funciones que hace 20 años parecían imposibles.

El desarrollo que han sufrido las tecnologías en la red en la última década han hecho que éstas sean imprescindibles en cualquier negocio. Para cualquier empresa o institución, la página web sirve de primera toma de contacto entre el usuario o cliente y la empresa o institución en cuestión. Es por ello fundamental que las personas conozcan las muchas posibilidades que pueden llegar a ofrecer las tecnologías en la red.

Dentro de este contexto entran en juego las páginas web destinadas a la educación. Muchas instituciones se han centrado en desarrollar aplicaciones que buscan facilitar el aprendizaje para el alumno y hoy en día el denominado e-learning está presente en muchos sistemas educativos. El trabajo de fin de grado tiene como objetivo empezar a desarrollar la primera fase de una herramienta de evaluación que consiga que los alumnos se motiven en aprender.

La elaboración de una aplicación web didáctica e intuitiva ha supuesto la perfecta elección para poder asentar buenas bases sobre como se llevan a cabo proyectos relacionados con aplicaciones en la red. Personalmente ha sido muy gratificante elegir este trabajo de fin de

grado ya que ha permitido que añada conocimientos nuevos sobre una serie de tecnlogías que esán muy presentes en la actualidad en cualquier sector.

1.2. Sistema de competición

El trabajo de fin de grado en cuestión es el inicio de una propuesta compleja de aplicación, y como tal, no se va a acometer en un solo trabajo de fin de grado. La aplicación se basará en un sistema de competición entre los alumnos de una determinada asignatura.

Los alumnos se inscribirán o registrarán en una página y una vez autorizados por la aplicación podrán acceder a realizar una competición con los demás integrantes de la clase.

La competición consistirá en retar a todos los compañeros (de forma individual) a realizar duelos de preguntas. Las preguntas irán relacionadas con la asignatura y al final de la competición, los alumnos que tengan más puntos serán recompensados.

La idea principal de realizar esta aplicación es motivar al alumno en aprender conocimientos sobre una determinada asignatura y fomentar el trabajo diario por parte del alumno en dicha asignatura(para conseguir un aprendizaje progresivo). Los alumnos en cuestión competirán con todos sus compañeros y lucharán por ver quién se sitúa en lo más alto de la clasificación. El objetivo de fomentar la competición es producir una motivación extra en el alumno que lo lleve a esforzarse al máximo en realizar bien los duelos. Realizar las pruebas de una forma más interactiva e divertida ayudará al aprendizaje del alumno y podrá incluso cambiar la mentalidad de la persona hacia la asignatura en cuestión.

En la figura 1.1 se ilustra un esquema global sobre los diferentes módulos que compondrán la aplicación.

En el servidor se almacenará la información que posteriormente el cliente solicitará mediante una petición. En el módulo de preguntas se situarán las preguntas que se vayan a emplear en la aplicación, dichas preguntas las insertará el profesor pertinente a través de la aplicación a la cual tendrá permiso de acceso. En el módulo de competición se encontrará la lógica aplicada a la competicón (algoritmos de puntuación, envío de preguntas, etc) y finalmente, en el módulo de gestión de usuarios se encontrará la base de datos con los usuarios habilitados para acceder a la competición.

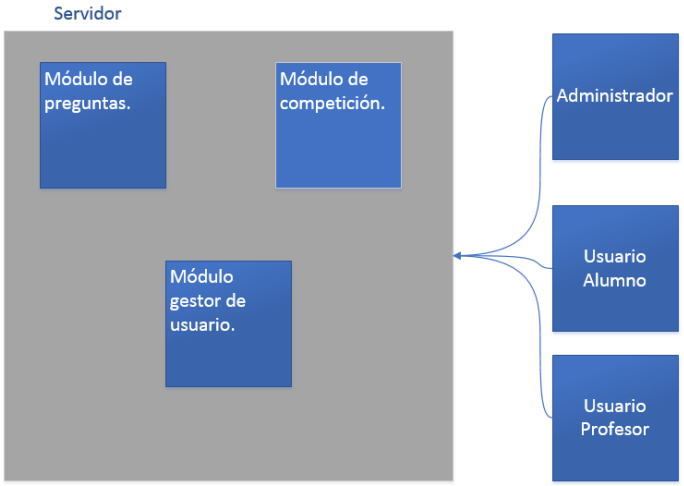

FIGURA 1.1: Esquema del sistema de competición

1.3. Objetivos

El objetivo de este trabajo fin de grado es realizar la primera fase del sistema de competición planteado en la sección anterior. Por ello es necesario crear una aplicación web interactiva, que permita al profesor o administrador subir preguntas de tipo test a la aplicación. En esta primera fase el administrador accederá a una página de administración donde podrá editar las preguntas y de la cual la aplicación solicitará la información necesaria.

El objetivo final es que todos los alumnos respondan a dichas preguntas y se guarde un historial con cada una de sus puntuaciones.

El alumno en cuestión accederá a una página principal donde podrá elegir realizar pruebas de distintos temas, y por ello será necesario crear un módulo de gestión de usuarios. En este módulo se situará la base de datos con los nombres de los usuarios registrados y habilitados para acceder a la aplicación.

Con respecto a las preguntas, cada tema consistirá en preguntas relacionadas con él y las cuales variarán en dificultad. Una vez el alumno esté realizando la prueba, a medida que vaya contestando a las cuestiones le aparecerá su puntuación actual. El alumno podrá decidir en cualquier momento abandonar la prueba ya que como se trata de contestar a preguntas de tipo test, una respuesta incorrecta restará puntuación a la puntuación total.

Cada usuario podrá además consultar la clasificación global de la clase y según el puesto final que ocupe el alumno, le corresponderá más o menos nota a sumar en la nota final de la asignatura.

1.4. Marco Regulador

El trabajo de fin de grado en cuestión maneja datos de un determinado número de usuarios. Es por ello que para el marco regulador es necesario hablar sobre la LOPD[8] (Ley Orgánica de Protección de Datos).

A nivel nacional, la organización encargada del cumplimiento de la protección de datos es la AEPD (Agencia Española de Protección de Datos). La LOPD establece las obligaciones que los responsables de los ficheros o tratamientos y los encargados de los tratamientos, tanto de organismos públicos como privados, han de cumplir para garantizar el derecho a la protección de los datos de carácter personal.

El responsable de un fichero o tratamiento es la entidad, persona u órgano administrativo que decide sobre la finalidad, el contenido y el uso del tratamiento de los datos personales.

En primer lugar es importante saber que la LOPD[9] trabaja con 3 tipos de protección:

- Nivel básico: el nivel básico corresponde a datos relacionados con nombres, apellidos, códigos postales, domicilio, ciudad o teléfono.
- Nivel medio: el nivel medio corresponde a datos relacionados con el curriculum, datos bancarios, etc.
- Nivel alto: el nivel alto corresponde sobre todo a información relacionada con la salud o información de tipo política.

Una vez establecidos los 3 niveles de protección de LOPD es necsario identificar que nivel corresponde a la información que se va a emplear. Teniendo en cuenta el trabajo de fin de grado en cuestión, es fácil determinar que le nivel de protección que corresponderá sera el nivel básico (ya que utilizaremos únicamente información relacionada con el nombre, apellido o teléfono del usuario).

El siguiente paso a realizar es inscribir o dar de alta el fichero en la LOPD. Dentro de la LOPD existen multitud de diferentes ficheros pero en el caso de este trabajo correspondera con:

Nombre del fichero	**Identificador RGPD**
Usuario de la web	345235548 (Fictício)

Tabla 1.1: Ficheros a registrar en la LOPD

El identificador RGPD (Registro de Protección de Datos) es el número de alta cuando se inscirbe un cierto fichero en la LOPD. Es un identificador numérico al que estará relacionada la información que se desea proteger.

Una vez inscrito el fichero que el propietario (o autónomo) o empresa desea proteger se lleva a cabo el documento de seguridad en el cual encontraremos el nivel de datos a proteger y el tipo de datos exactos que se desea inscribir en el fichero. En la tabla de abajo observamos como se realizá un documento de seguridad para delimitar de forma precisa el contenido que se desea registrar para proteger.

NOMBRE DEL FICHERO	ORIGEN
Usuario de la Web	El Propio Interesado ☑ Registros Públicos ☐ Otras Personas Físicas ☐ Entidad Privada ☐ Fuentes Accesibles al Público ☐ Administraciones Públicas ☐
Sistema de Tratamiento Automatizado ☐ Manual ☐ Mixto ☑	**Nivel de Seguridad del Fichero** Básico ☑ Medio ☐ Alto ☐

Finalidades Previstas		Colectivos Interesados	
Gestión de Clientes Contable, Fiscal y Administrativa	☐	Empleados	☑
Recursos Humanos	☐	Clientes y Usuarios	☑
Gestión de Nóminas	☐	Proveedores	☐
Prevención de Riesgos Laborales	☐	Asociados/Miembros	☐
Prestación de Servicios de Solvencia Patrimonial y Crédito	☐	Propietarios/Arrendatarios	☐
Cumplimiento/Incumplimiento deObligaciones Dinerarias	☐	Pacientes	☐
Servicios Económicos Financieros y Seguros	☐	Estudiantes	☐
Análisis de Perfiles	☐	Personas de Contacto	☐
Publicidad y Prospección Comercial	☐	Padres/Tutores	☐
Prestación de Servicios de Comunicación Electrónica	☐	Representante Legal	☐
Guías/Repertorios de Servicios de Comunicaciones Electrónicas	☐	Solicitantes	☐
Comercio Electrónico	☐	Beneficiarios	☐
Prestación de Servicios de Certificación Electrónica	☐	Cargos Públicos	☐
Gestión de Asociados/Partidos Políticos/Iglesias/Sindicatos (sin lucro)	☐	Otros Colectivos	☐
Actividades Asociativas, Culturales, Recreativas, Deportivas y Sociales	☐		
Gestión de Asistencia Social	☐		
Educación	☐		
Investigación Epidemiológica y Actividades Analógicas	☐		
Gestión y Control Sanitario	☐		
Historial Clínico	☐		
Seguridad Privada	☐		
Seguridad y Control de Acceso a Edificios	☐		
Video Vigilancia	☐		
Fines Estadísticos, Históricos o Científicos	☐		
Otros tipos de finalidad	☐		

FIGURA 1.2: Documento de seguridad

1.5. Estructura de la memoria

Para comprender de una forma clara la composición e implementación del trabajo de fin de grado se dotará a la memoria de los siguientes bloques:

1. **Introducción**: en la introducción ya comentada se ha procedido a explicar de una forma sencilla los objetivos a conseguir en el trabajo de fin de grado y además, hablar del marco regulador que rodea a la aplicación.

2. **Estado del Arte**: en este bloque analizaremos los aspectos que han influido a la hora de elegir el entorno de trabajo para la realización del proyecto. Se tratarán las diferentes tecnologías empleadas y se explicará de forma breve las características y usos que tiene cada tecnología.

3. **Diseño e Implementación**: explicaremos en este bloque la composición principal del trabajo. Especificando aspectos técnicos de la misma y explicando también la implementación de los diferentes módulos que constituyen el trabajo de fin de grado.

4. **Validación**: en este capítulo se llevará a cabo una demostración, con ayuda de capturas de imágenes de la aplicación, sobre el cumplimiento de los requisitos mencionados en el capítulo de Diseño e Implementación.

5. **Conclusiones y Trabajo futuro**: capítulo destacando las conclusiones sobre el desarrollo del trabajo de fin de grado y ampliaciones futuras posibles sobre el trabajo.

6. **Planificación y Presupuesto**: capítulo dedicado a la planificación seguida en el trabajo de fin de grado y también al presupuesto necesario para llevar a cabo la aplicación.

Capítulo 2

Estado del Arte

2.1. Introducción

Una gran parte de las aplicaciones web, incluyendo este trabajo de fin de grado, se basan en la arquitectura de red de Modelo Cliente-Servidor. Este modelo basa su funcionamiento en un cliente que realizará una serie de peticiones, y un servidor que a partir de estas peticiones aplicará una lógica programada determinada y ofrecerá una respuesta.

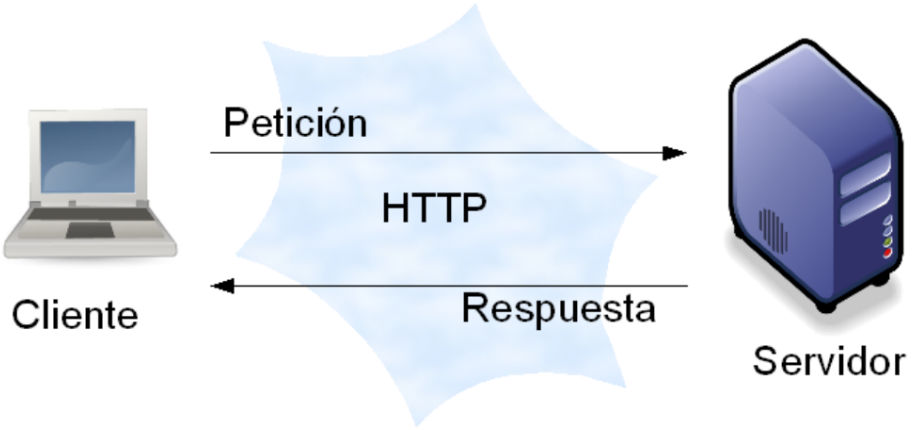

Figura 2.1: Modelo Cliente-Servidor

En la figura 2.1 podemos observar el esquema general del modelo Cliente-Servidor por el cual el cliente o usuario envía peticiones y el servidor otorga respuestas a esas peticiones.

En el lado servidor será necesario contratar a un data center el alojamiento del servidor. Este servidor está basado en un ISP (Internet Solution Provider) que nos podrá ofrecer 2 tipos de servicios:

- **Housing**: el servicio de housing es el más completo ya que la máquina en el ISP es exclusiva para la empresaoel particular que contrata el servicio.
- **Hosting**: el servicio de hosting significa que la máquina ISP es compartida por un número de particulares que contratan el servicio.

En este capítulo se describirán principalmente todas las tecnologías que han permitido la realización del trabajo de fin de grado. Y en algún caso una breve explicación de por que se ha elegido dicha tecnología frente a otras.

2.2. Desarrollo Web

El desarrollo web ha evolucionado exponencialmente en la última década. En la actualidad se ha convertido en un servicio muy solicitado y que puede ofrecer una inmensa cantidad de funciones muy distintas.

Cualquier aplicación web en la actualidad es la suma de diferentes lenguajes de programación y lenguajes de etiquetado. Esto se debe principalmente a que la web ya no es únicamente un conjunto de páginas HTML estáticas. Se han desarrollado distintas tecnologías que han permitido páginas dinámicas y aplicaciones complejas. Esta complejidad hace que sea preciso un diseño cuidadoso y una lógica de aplicación en servidor o cliente que permita al usuario realizar las tareas pertinentes.

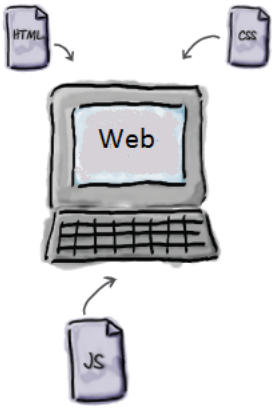

Figura 2.2: Conjunto de tecnologías web

Este trabajo fin de grado tiene como requisito la utilización de Django como motor para la realización de la aplicación web. El hecho de utilizar Django implica emplear Python en la lógica de servidor, esto se debe a que Django está escrito en Python y se explicará con más

detalle en este capítulo. Además, ha sido necesario emplear otros lenguajes como Javascript, HTML, CSS y también MySQL como motor de la base de datos.

A continuación se explicará detalladamente las características que tienen dichos lenguajes, y salvo en el caso de Django, la razón principal por la que se han elegido para desarrollar la aplicación.

2.3. Tecnologías Web

2.3.1. Python

Creado por Guido van Rossum[1], su nombre proviene de la afición del autor por el grupo Monty Python. Python es un lenguaje de programación de alto nivel [1]. La filosofía de su diseño se centra en poder desarrollar un código fácil de leer para el programador (como si de pseudo-código se tratara). Esta característica es probablemente la más importante y su gran ventaja es que permite al programador, en caso de estar un determinado tiempo sin ver el código, adaptarse y volver a entenderlo con suma facilidad.

2.3.1.1. Características Principales

1. Python es un lenguaje de programación dínamico y orientado a objetos[1].
2. Facilidad en la lectura y en el diseño.
3. Pyhton es un lenguaje de programación multiparadigma y permite tanto la orientación a objetos como la programación imperativa.
4. Gran soporte e integración con otros lenguajes y herramientas.
5. Con Python es posible realizar cualquier programa desde aplicaciones Windows a servidores de red o incluso páginas web.
6. Python es un lenguaje interpretado, lo que significa que no es necesario compilar el código para poder ejecutarlo.

2.3.1.2. Ventajas y Desventajas

- Ventajas:
  - Rápido de desarrollar.
  - Bibliotecas muy potentes que favorecen muchas tareas al programador.

- Variedad en el uso de bases de datos.
- Programa interpretado.
- Gran soporte con otros lenguajes y herramientas.

- Desventajas:

  - Lentitud: los programas interpretados son más lentos que los compilados.
  - Documentación más escasa que en otras tecnologías (independientemente del idioma) debido a que su comunidad de programadores es más pequeña.
  - Sintaxis basada en el acomodo de espacios e indentación, lo cual puede llevar a producir errores facilmente.

Posibles desarrollos[1]

- Web (Django, TurboGears).
- GUI (Tk, wxWidgets, GTK+, QT).
- Educación, juegos, redes.
- Computación científica y numérica.
- Desarrollo de software.
- Ejemplos de páginas con soporte Python: Youtube y Google.

El empleo de Python en el trabajo de fin de grado se debe únicamente al requisito del trabajo en sí, que es el de realizar la aplicación en Django. El *framework* Django está escrito en Python y por ello el aprendizaje de Python es necesario para conseguir la realización de la aplicación.

2.3.2. Django

Django es un *framework* de desarrollo web de código abierto, escrito en Python[6]. Es un conjunto de bibliotecas y herramientas que permitirán al programador la creación de sitios web. Django permite al usuario construir sitios web dinámicos e interesantes en un periodo de tiempo relativamente corto.

¿Que es un *Framework* Web?

A finales de la década de los 90 el diseño de una aplicación web se programaba mayoritariamente mediante un estándar muy popular llamado Common Gateway Interfaces (CGI). Cuando se programaba una aplicación en CGI[6], el programador tenía que estar al control de todos los aspectos. El programador tenía que encargarse de la correcta comunicación entre

sus ficheros HTML y las bases de datos y una vez terminada esta comunicación establecer el fin de la misma. Desde un punto de vista práctico, este método podía ser manejable si el programador estaba trabajando con poco flujo de información. Pero a medida que una aplicación web crece más allá de lo trivial, el enfoque que se le daba en CGI se desmorona y el programador se enfrenta a una serie de problemas:

- Múltiples páginas conectándose a la misma base de datos. ¿Es necesario que el programador se encargue de la conexión a la base de datos de cada uno de los scripts CGI?
- ¿Es necesario que el programador tenga que ocuparse cada vez de cerrar la conexión con la base de datos?
- Re-utilización en múltiples entornos cada uno con bases de datos diferentes.
- ¿Qué ocurre cuando un diseñador web con poca experiencia desea rediseñar la página? Lo ideal sería que las diferentes tareas a realizar estuviesen separadas y fueran independientes.

En este contexto entran en juego los *Framework*. La misión principal de un *Framework* Web es la de proveer una infraestructura de programación centrada en las aplicaciones, donde el programador pueda centrarse en escribir un código limpio y de fácil mantenimiento[6]. Django pertenece a una nueva generación de *framework's* Web. El término Framework hace referencia a una serie de motores para crear sitios web que permiten al programador centrarse en los diferentes módulos que componen una aplicación web, sin tener que destinar tiempo excesivo en la comunicación entre estos módulos.

2.3.2.1. Características de Django

La meta de Django es facilitar la creación de sitios web complejos. Django pone énfasis en la reutilización , la conectividad, el desarrollo rápido y el principio DRY (del inglés 'Dont repeat yourself' o 'No te repitas').

Las principales ventajas de Django son[6]:

1. **Modelo Vista Controlador (MVC)**: en Django se emplea la arquitectura MVC de tal forma que se separa la aplicación en tres partes:
    - **Modelos**: los modelos van a corresponder con la parte de la aplicación que define la estructura de la base de datos y se encarga de la comunicación con ella. Se define qué estructura seguirá toda la información que se maneje en la aplicación.

- **Vistas**: las vistas van a corresponder a la parte interfaz del usuario, con el código que elige qué datos pedirle o mostrarle al usuario en cada momento.
- **Controladores**: corresponde a la parte de la aplicación que elige qué vistas ejecutar en respuesta a las acciones o peticiones del usuario.

Actualmente en Django la arquitectura MVC cambia ligeramente frente a la tradicional, en el sentido de que las vistas contendrán las funciones que van a devolver el contenido que debe ser entregado al usuario. Desde las vistas ejecutaremos la lógica de la aplicación y entregaremos a los "Templates", que en este caso corresponden a los archivos HTML, toda la información que se desea manejar y plasmar.

2. **Mapeador Objeto-Relacional (ORM)**: el ORM en Django va a ser el que nos permita interactuar constantemente con la base de datos. Django se va a encargar de traducir nuestras operaciones sobre los objetos, en sentencias SQL (en el caso de este trabajo) que se ejecutarán sobre las tablas de la base de datos. Esto permitirá al programador más flexibilidad ya que no tendrá que centrarse en estas acciones, sino que Django detectará que el programador está modificando un dato y se lo hará saber a la base de datos.

3. **Panel de administración de fácil manejo**: en Django una de las principales ventajas es la posibilidad de utilizar su panel de administrador para la página web en desarrollo. Mediante el panel de administración, el administrador de la página tendrá fácil acceso a toda la información circulando por su página o páginas vinculadas a este panel. Desde este punto podrá además controlar de una forma muy sencilla toda la información que contienen las bases de datos de la página, permitiendo la posibilidad de actualizar, borrar o crear cualquier objeto de contenido en la página. Proporciona también una forma sencilla de controlar a los usuarios de la página web, pudiendo modificar desde aquí los permisos de acceso de cada uno de ellos.

FIGURA 2.3: Panel de Administración

4. **Bibliotecas incorporadas con funciones de gran utilidad**: Django está dotado de una gran librería (escrita en Python) con funcionalidades dirigidas para ayudar al programador a ahorrar tiempo en ejecutar ciertas tareas. Entre la gran variedad de librerías que existen podemos destacar funciones como el envío de correos, leer datos de páginas web o también trabajar con archivos comprimidos.

5. **Alta portabilidad**: en Django vamos a tener Proyectos y Aplicaciones:

   - **Proyectos**: el proyecto contendrá la configuración general de nuestro sitio. El proyecto en Django es la parte que suministra un archivo de configuración, el cual define la información hacia la conexión a la base de datos, la lista de las aplicaciones instaladas y por ejemplo la ruta hacia las plantillas.
   - **Apliciaciones**: la aplicación es la parte con la funcionalidad en sí de la página web, aquí reside la lógica de la aplicación en cuestión. Una aplicación es un conjunto portable de una funcionalidad de Django.

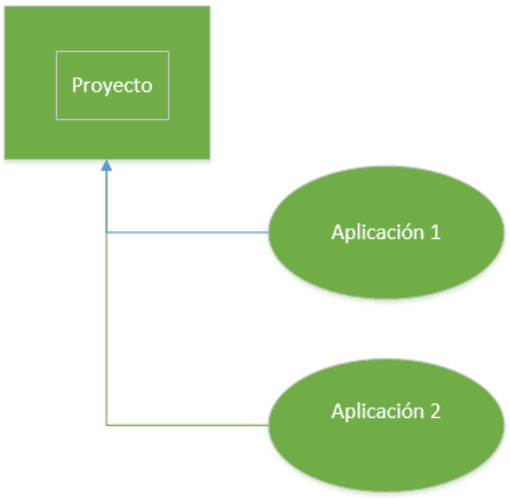

FIGURA 2.4: Diagrama de proyecto con posibles aplicaciones

6. **Compatibilidad con múltiples tipos de bases de datos**: Django es compatible con 4 motores de bases de datos:

   - **PostgreSQL.**
   - **SQLite3.**
   - **MySQL.**
   - **Oracle.**

Presencia de Django en la actualidad

Django está presente en páginas web muy utilizadas en la actualidad. Las principales son:

- **Instagram**: http://instagram.com/
- **Mozilla**: http://www.mozilla.org/es-ES/
- **Disqus**: http://www.disqus.com/
- **Pinterest**: https://www.pinterest.com/

2.3.3. HTML (Hyper Text Markup Language)

HTML es un lenguaje de marcado que se emplea para definir la estructura que seguirán los datos de una página web. Fue creado por Tim Berners-Lee a principios de los años 90 y desde entonces ha supuesto el pilar en el que se basan todas las tecnologías web en la actualidad[2][12].

HTML es un formato estandar basado en lenguaje de marcas para usar en internet bajo el protocolo HTTP (HyperText Transfer Protocol). El protocolo HTTP es el método más común de intercambio de información en la world wide web[16].

Además, HTML es perfectamente compatible con Django y gracias al modelo MVC que nos proporciona Django nos permitirá trabajar de forma independiente con las plantillas. Las llamadas a los archivos HTML se producen desde las funciones lógicas situadas en las vistas de Django[6].

¿Por qué HTML?

Como ya mencionado anteriormente, HTML es el pilar fundamental de cualquier tecnología web. Es por esto que a la hora de desarrollar una aplicación web es muy importante utilizar HTML. Una consecuencia directa de esto se refleja en la cantidad de información, tutoriales y libros que se pueden encontrar en la red, permitiendo así, a un usuario principiante en HTML asentar buenas bases en poco tiempo sobre el diseño de páginas web en HTML.

En Django, como mencionamos arriba, los documentos o plantillas HTML son llamados desde las vistas[6]. Las vistas en Django serán las responsables de informar a la aplicacón que plantilla se debe ejecutar. Esta acción en Django es sencilla de implementar y es por ello fácil llevar a cabo estas funciones.

Alternativas

Para el sistema de plantillas se podría hacer uso también de XML (Extensible Markup Languaje). XML es, al igual que HTML, una ampliación de SGML (Standard Generalized Markup Language) pero con mejoras y además sigue siendo un lenguaje basado en las marcas[10]. XML es un formato basado en texto, específicamente diseñado para almacenar y

transmitir datos. Además, XML admite un conjunto ilimitado de etiquetas, no para indicar el aspecto que debe tener algo (como en HTML), sino lo que significa[10]. Se trata también de una tecnología libre (su uso no implica pagos) y manejable en el sentido de que incluye métodos para declarar y reforzar las estructuras documentales, como las bases de datos.

No obstante, la principal característica de XML[11] (ya mencionada arriba) es que permite el almacenamiento de datos y por ello, no se empleará en la aplicación puesto que el almacenamiento de datos se conseguirá trabajando con MySQL, Python y Django.

2.3.4. CSS (Cascading Style Sheets)

CSS surge gracias a la W3C (World Wide Web Consortium), organismo que impulso su estandarización a mitades de la década de los 90. CSS es un lenguaje que sirve para describir las propiedades de diseño de un contenido en particular en una estructura. Generalmente esta estructura consiste en un archivo HTML, pero CSS también es perfectamente compatible con otras tecnologías como XML o SVG[2].

Características Principales

CSS permite separar los contenidos de la página y la información sobre su aspecto. Dentro de la propia página HTML se crea una zona en la que se incluye toda la información relacionada con los estilos de la página. Esto permite al programador generar un código muy limpio y de fácil acceso. El programador tendrá la opción de centrarse en el código por módulos, es decir, para modificar el diseño de los página (Colores, Margenes, Fondos, etc) únicamente tendrá que modificar la parte del código donde se encuentra escrito el CSS.

Como mencionado en el párrafo anterior, utilizando CSS, se pueden establecer los mismos estilos con menos esfuerzo y sin ensuciar el código HTML. Delimitando el archivo con los marcadores "style"se crea una zona especial donde se incluyen todas las reglas CSS que se aplican en la página. Con CSS estamos escribiendo una serie de reglas, y cada regla se asocia a un elemento o parte de la página que deberá respetar estas reglas de diseño establecidas.

2.3.5. Javascript

Con HTML y CSS podemos llegar a crear páginas web visiblemente muy detalladas y precisas. Pero en ambos lenguajes únicamente se declara la composición o estructura de la página web en cuanto al diseño (Formatos, párrafos, colores, márgenes, etc), es decir, mediante estos lenguajes de marcado no podremos ejecutar cierto tipo de lógica y esta es la razón por las que surgieron los lenguajes de Scripting como Javascript. Desde su aparición a mediados de la

década de los 90, Javascript tenía como objetivo dotar a la página web de un comportamiento adicional[3].

Características Principales

Este "comportamiento" mencionado anteriormente, está centrado en darle al programador un control total sobre la página web. Con Javascript se pretende que el usuario pueda interaccionar dinámicamente con la página web en cuestión. De tal forma que Javascript y HTML pueden interactuar, permitiendo a los programadores web utilizar contenido dinámico[3]. Por ejemplo, permite responder rápidamente a acontecimientos iniciados por el usuario (como podría ser introducir datos en un formulario, registro o blog).

La programación en Javascript se lleva a cabo dentro del mismo archivo HTML, es decir, las funciones que se programarán en Javascript se escribirán dentro del mismo archivo donde este escrito el código HTML.

Teniendo en cuenta que el objetivo del trabajo de fin de grado es permitir a los alumnos competir entre sí realizando test sobre una asignatura determinada, esta última característica sobre Javascript encaja perfectamente en las necesidades de la aplicación.

Al igual que HTML, Javascript es un lenguaje que ha tenido mucho éxito a nivel mundial, y por ello, es de suma facilidad encontrar tutoriales, libros o información de apoyo. Esto es fundamental para gente en estado de iniciación en desarrollo de páginas web, y supone una gran ayuda a la hora de aprender.

Estas características sobre Javascript han supuesto las razones principales por las que se decidió emplear esta tecnología en el trabajo.

2.4. Base de datos

Además de los lenguajes de programación web, que hemos visto anteriormente, existe otro módulo imprescindible sin el cual no se podría desarrollar el proyecto, este módulo es la implementación de la base de datos(también conocidas como SGBD o Sistemas de Gestión De Bases de Datos). La base de datos ,como se mencionará en la siguiente sección, es un pilar fundamental de cualquier aplicación web interactiva, ya que nos permitirá almacenar toda la información de interés para el programador de forma ordenada, y de esta forma emplearla a posteriori para un fin determinado.

El propósito general de las bases de datos es el de manejar de forma clara y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para el usuario.

Como hemos mencionado anteriormente Django es compatible con numerosos tipos de bases de datos. Finalmente se ha elegido emplear MySQL como motor de la base de datos, y en la siguiente sección describiremos brevemente los diferentes tipos de bases de datos que son compatibles con Django y porque se decidió emplear MySQL en el trabajo.

2.4.1. Tipos de base de datos

- **PostgreSQL:** PostgreSQL es un sistema de gestión de bases de datos objeto-relacional basado[15] en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Entre las características principales estan:
  - Soporta distintos tipos de datos
  - Permite la declaración de funciones propias, así como la definición de disparadores.
  - Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
  - Permite la gestión de diferentes usuarios.
- **SQLite3:** SQLite es un sistema de gestión de bases de datos relacional[13] compatible con ACID(Atomicidad, Consistencia, Aislamiento y Durabilidad).Su diseñador fue D. Richard Hipp quien lo lanzó en el año 2000. Entre las características principales están:
  - Es mono-usuario, es decir, no permite concurrencia de conexiones. Si un usuario está modificando datos, otro no podrá hacerlo hasta que el anterior no termine.
  - Funciona perfectamente para sitios de tráfico bajo-medio (99 por ciento de la web).
  - No tiene tipos de datos. Puedes meter una cadena de texto en un campo que estás utilizando como numérico.
- **MySQL:** MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario, rápida, segura, fácil de usar, y con un gran número de usuarios que la utilizan. Los derechos del código están en posesión de MySQL AB, quién desarrolla MySQL como software libre, ofreciendo MySQL bajo licencia GNU GPL[2].
  - MySQL usa el lenguaje SQL estandarizado para el almacenamiento, actualización y acceso a información.
  - MySQL es muy rápido y capaz de almacenar grandes cantidades de datos.
  - Soporta diversos lenguajes de programación diferentes, C, C++, Eiffel, Java, Perl, PHP, Python y TCL [2].

- **Oracle:** oracle es una herramienta cliente/servidor para la gestión de bases de datos[14]. Es un producto vendido a nivel mundial, aunque esta orientado a empresas multinacionales debido a su potencia y su alto coste. Oracle soporta el manejo de un volumen de información muy elevado. Tiene un coste muy alto por lo cual no está al alcance de todos los usuarios, lo que produce que muchos acaben empleando bases de datos como MySQL. Esta característica fue un factor determinante para descartar Oracle como motor de la base de datos del trabajo de fin de grado.

¿Por qué MySQL?

La elección de la base de datos se basó principalmente en la comunidad de usuarios que tiene cada una de ellas. Teniendo en cuenta esta característica MySQL es la base de datos más usada entre la comunidad de desarrolladores web, y es muy abundante la cantidad de tutoriales y libros disponibles para usuarios principiantes. Además, las características de MySQL, como su capacidad multiusuario y su capacidad de alamcenar grandes cantidades de información, encajan perfectamente con los requisitos de la aplicación y es por esto que ha supuesto la elección perfecta.

2.5. Resumen

En conclusión es importante mencionar que las aplicaciones web son fruto de un desarrollo conjunto de diversas tecnlogías. En el desarrollo de aplicaciones web existe una gran variedad de tecnologías que se pueden implementar, y es por ello que el programador debe realizar un estudio a priori sobre los requisitos que tendra su aplicación con el fin de elegir las herramientas de desarrollo que más se adaptarán a su trabajo.

En esta aplicación se ha procedido a emplear una serie de tecnlogías que se pueden ver en la siguiente tabla:

Tecnología	**Razón**
Django	Es un framework sencillo de emplear para principiantes y muy novedoso, emplea el modelo MVC y era un requisito emplear Django para la realización del trabajo de fin de grado.
Python	El requisito de realizar la aplicación en Django obliga a emplear Python como lenguaje de programación puesto que Django esta escrito en Python.
JavaScript	Lenguaje de scripting con una comunidad de usuarios muy extensa y fácil de usar. Se empleará en el lado cliente para realizar tareas lógicas que con HTML no son posibles.
HTML	Lenguaje de marcado que determinara la estructura que sigue la información en la página web.
CSS	Se empleará como motor del diseño visual del contenindo HTML en la página web que verá el cliente o usuario.
MySQL	Es un de los motores de bases de datos más usados. Junto con Oracle son las bases de datos más potentes pero además no tiene coste lo cual es una ventaja frente a Oracle.

Tabla 2.1: Tabla con tecnologías empleadas.

Capítulo 3

Diseño e Implementación

A lo largo de este capítulo explicaremos los pasos más importantes que han intervenido para desarrollar el software de la aplicación. Se hara hincapié en explicar la estructura de los datos así como la lógica seguida para la ejecución de la aplicación.

3.1. Requisitos

Los requisitos de la aplicación se centrarán en el usuario alumno de la aplicación, los requisitos pedidos en la aplicación son:

- **Registrar:** el usuario debe tener la posibilidad de registrarse en la base de datos con la información que inserte en la vista correspondiente al registro. El registro se debe producir únicamente una vez y si el usuario registrandose ya se encuentra en la base de datos recibirá una notificación indicandole esta información.
- **Autenticar:** una vez cumplido el requisito número 1, el usuario debe tener la posibilidad de ingresar su información en la vista correspondiente al acceso a la aplicación con el fin de acceder a la página principal de la página web.
- **Realizar test:** una vez cumplido los requisitos anteriores, el usuario debe poder elegir en la página principal de la aplicación la opción de realizar test, donde deberá poder elegir, entre una serie de temas o niveles, el test a realizar. El usuario además sabra en cada pregunta la puntuación que ha obtenido hasta el momento como también el número de preguntas restantes.
- **Ver clasificación:** una vez finalizado el test, el alumno o usuario deberá poder acceder a la opción 'ver clasificación' dentro del menú principal de la aplicación, y una vez seleccionada esta opción, ver la clasificación general de la clase. La opción de clasificación se elegirá en función del tema que se desea comprobar y habrá una clasificación por cada tema existente en la aplicación.

- **Abandonar test:** en medio de un test, si el usuario lo estima oportuno, podrá finalizar el test sin la necesidad de terminarlo y su puntuación se guardará de la misma forma que si hubiera realizado el test entero. Este requisito se debe a que una respuesta incorrecta restará puntuación y se le otorgará al almuno la opción de no realizar más preguntas si considera que ha obtenido una buena puntuación hasta el momento.
- **Realizar test únicamente una vez:** el usuario deberá quedar limitado por la aplicación a realizar los test de los diversos temas únicamente una sola vez. Si por razones ajenas se perdiese la conexión, el alumno no perdería su opción a realizar la prueba y podría seleccionarla de nuevo al inicar la sesión en la aplicación.
- **Modificar datos:** el usuario, una vez habilitado, podrá elegir del menú principal de la página la opción de modificar datos. En esta primera fase la información que se almacena del usuario es básica pero será una función más importante a la hora en la que se incorporen nuevos campos de datos. Para modifcar los datos será necesario que el alumno introduzca su información antigua para poder llevar a cabo una validación correcta, si la información antigua no se verifica la aplicación notificará al usuario con un mensaje de error.

3.2. Funcionalidades

En la figura 3.1 se puede observar un diagrama general en el que se representan todas las funcionalidades del trabajo de fin de grado.

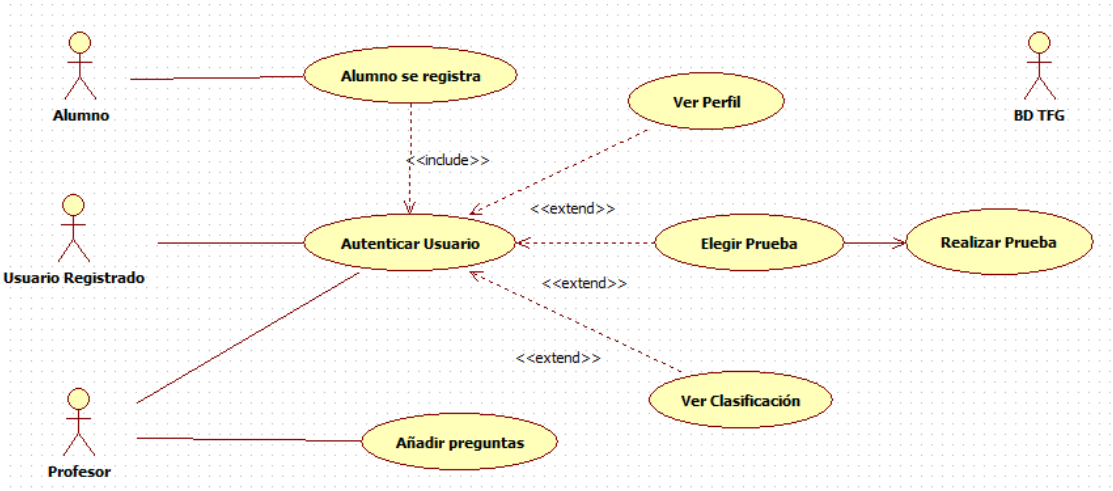

FIGURA 3.1: Diagrama funcional del TFG

Las funcionalidades de la aplicación (resumidas en la figura de arriba) se centran en 2 tipos de usuarios:

- **Usuario Administrador**: el administrador (Profesor) de la aplicación fundamentalmente tendrá los permisos adecuados para acceder al panel de administración (Activado desde Django) y, a través de él, añadir y modificar las preguntas de los diferentes temas o niveles. Además podrá acceder a la tabla de clasificación autenitcándose desde la página principal de la aplicación.

- **Usuario Almuno**: como se puede observar en el diagrama de los casos de uso (figura 3.1) el alumno o usuario podrá registrarse a través de un formulario de registro situado en la página principal de la aplicación. Una vez rellenado los datos del usuario se procederá a guardar la información en las tablas de la la base de datos de la aplicación. Después de realizar este registro el alumno ya estará habilitado para acceder a la aplicación introduciendo los datos anteriores y una vez autenticado, el alumno podrá realizar las pruebas de los diferentes niveles o temas (únicamente 1 vez) y consultar la clasificación global de la clase.

3.3. Modelo de Datos

En esta sección se tratará un tema fundamental en cualquier aplicación que son los modelos de datos. En este módulo se determinará la estructura lógica que seguirán los diferentes datos o mejor dicho, la forma en la que se organizará la base de datos.

El propósito de estructurar bien el modelo de datos es fundamentalmente de mejorar la comunicación y la precisión en aplicaciones que requieren un intercambio de datos continuo.

A continuación se explicará la estructura que seguirá el modelo de datos de este trabajo fin de grado.

3.4. Modelos

En el trabajo se han utilizado 4 clases diferentes para llevar a cabo la aplicación:

1. **Class Usuario:** en primer lugar, es necesario establecer todos los campos de información que llevara el usuario de la aplicación. Como se puede observar, en la figura 3.2 cada usuario se guardará en el sistema con su nombre (nickname), la contraseña que empleará para acceder al portal web (password) y un espacio reservado para el almacenamiento de una foto.

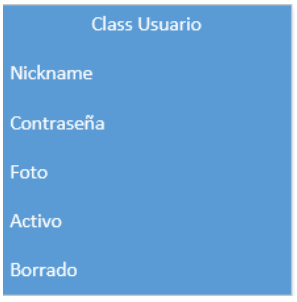

FIGURA 3.2: La clase Usuario empleada

2. **Class Tema:** la clase Tema unicamente se utilizará para poder diferenciar las preguntas por tema. Se le asociará un número de identificación que permitirá al administrador de la aplicación separar las preguntas por temas desde el interfaz de administrador ('admin site') de django. También tendra asociado un espacio para establecer el nombre del tema o el nivel de la prueba.
3. **Class Pregunta:** la clase Pregunta será el modelo que se encargue de establecer el formato que tendrán las preguntas en la base de datos. En primer lugar, es necesario reservar espacio para la propia pregunta y, como es lógico, también para las respuestas. Otro campo característico que tendrá la clase será un valor numérico que indicará la respuesta correcta de la ristra de respuestas.

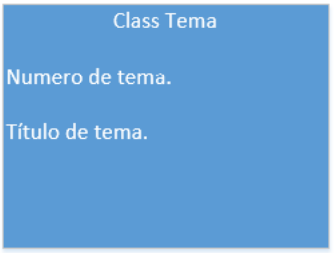

FIGURA 3.3: La clase Tema empleada

Se ha procedido a emplear un modelo de pregunta típico de una prueba de tipo test constituida por la corresponidente pregunta y 4 posibles respuestas.

Cada pregunta pertenece a su vez a un tema. Por ello, es necesario indicar también a que tema pertenece, esto se consigue creando un campo tema y asociando este a su vez a la clase tema mediante el uso del ForeignKey.

Esto permitirá al administrador de la aplicación establecer de una forma muy intuitiva a que tema pertenece la pregunta que se incorporará a los test.

La composición de la clase pregunta se puede observar en la figura 3.4.

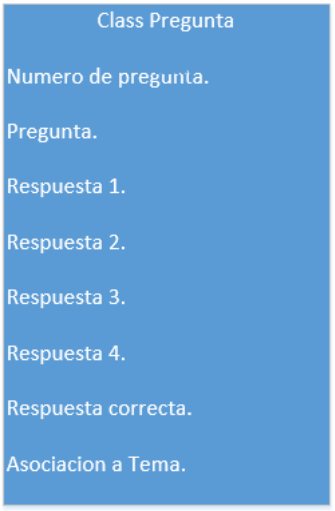

FIGURA 3.4: La clase Pregunta empleada

4. **Class Partida:** por último, tendremos la clase Partida. Esta clase contendrá la información que relaciona al usuario con el test realizado y con su respectiva puntuación.

   Cada vez que un usuario termine una ronda de preguntas, la aplicación procede a guardar su partida en la base de datos. En esta acción se incluirá en la base de datos la información del tema, puntuación y nombre del usuario.

   Con el fin de evitar que un mismo usuario pueda volver a realizar la prueba una vez finalizada, cada vez que un usuario acceda a una prueba, la aplicación se encargara

de buscar en las tablas correspondientes si existe algún historial entre el usuario y el tema elegido. En el caso de existir, se notificará al usuario la información correspondiente.

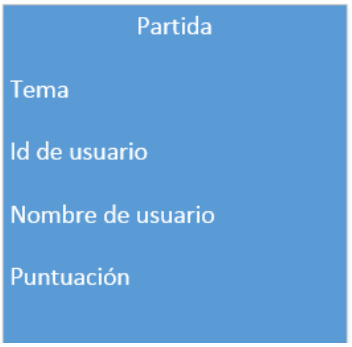

FIGURA 3.5: La clase Partida empleada

3.4.1. Diagrama de datos relacional

En la siguiente figura se puede observar un diagrama de modelo relacional de los datos empleados en el trabajo fin de grado. La función de estos diagramas es mostrar de forma visual como está constituida la estructura de datos de una aplicacón. Los diagramas de datos relacionales nos permitirán entender que como están conectados los diferentes datos que se emplean en una aplicación.

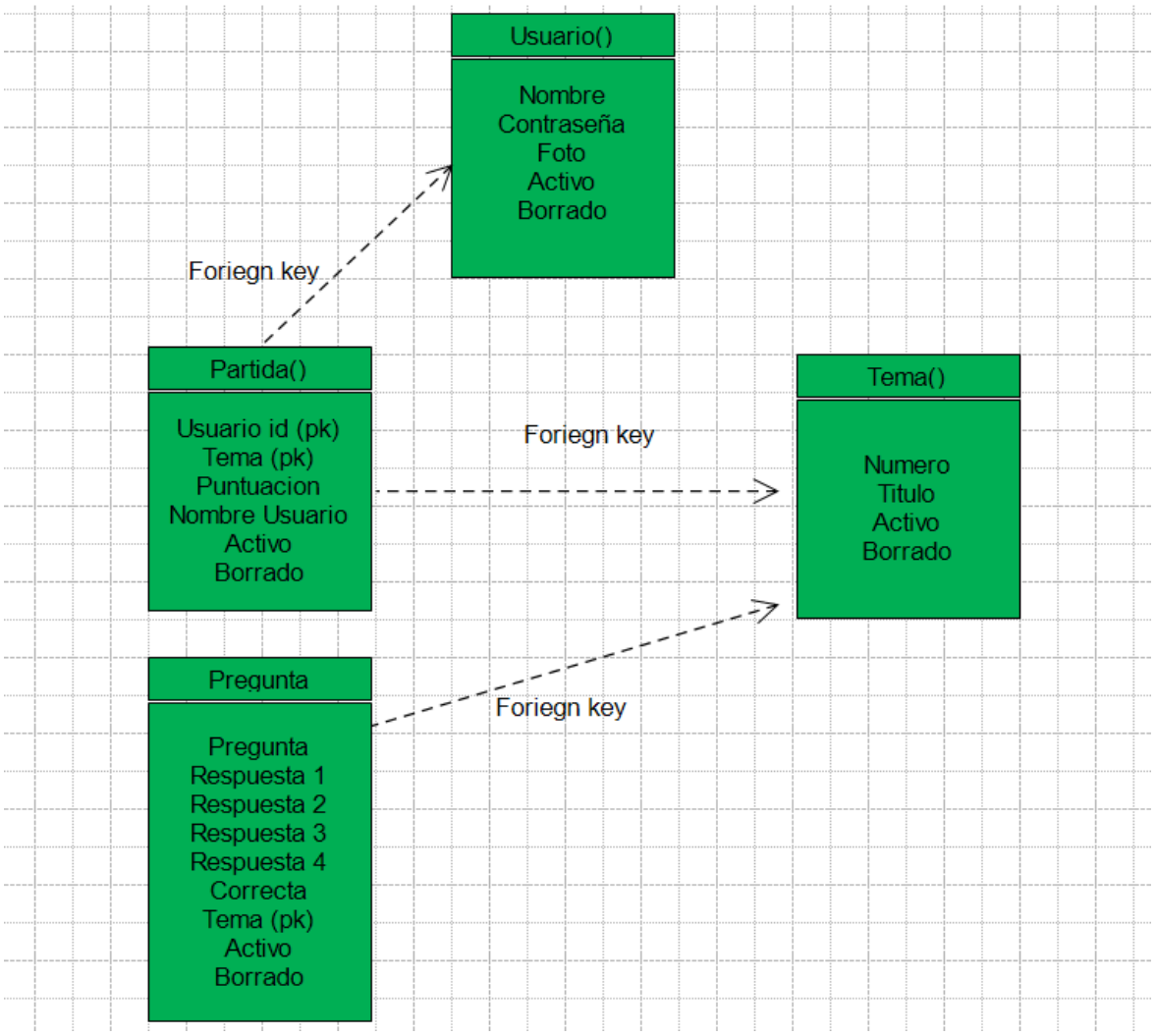

FIGURA 3.6: Diagrama de modelo relacional

El modelo de datos principal es Partida. Como se puede obsdervar en la figura de arriba Partida() tiene relación directa con el Usuario y con el Tema. Y Tema a su vez tendrá relación directa con Pregunta.

De esta forma, en la base de datos existirán diferentes tablas que contendrán información distinta. Las tablas de la base de datos correspondientes a la aplicación se pueden encontrar en la siguiente tabla:

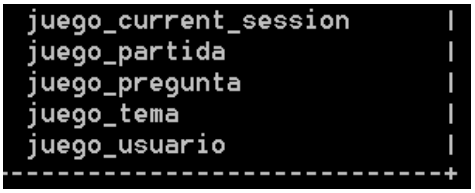

FIGURA 3.7: Tablas de la DB

Un ejemplo sería la tabla de tipo partida que tendrá la información de las puntuaciones de los diferentes temas que cada usuario tiene. Y será imprescindible a la hora de recuperar información para plasmar en la tabla de las las clasificaciones.

3.5. Vistas

En Django, una parte fundamental son las denominadas vistas. Todas las vistas se deben localizar en el archivo views.py(se explicará más adelante en este capítulo), generado automáticamente al iniciar una nueva aplicación en Django.

En views.py encontraremos todas los métodos necesarios para llevar a cabo las tareas lógicas de la aplicación.

En el caso de este trabajo fin de grado, se tratará de las funciones encargadas principalmente de la gestión de todos los usuarios de la aplicación, imprimir en la pantalla de forma adecuada el contenido de las pruebas y permitir al usuario modificar su información personal.

En esta sección se describirá, empleando diagramas de flujos, las principales funciones de la aplicación que son:

1. **Registrar**

2. **Validar**

3. **Modificar**

4. **Realizar Test**

3.5.1. Registrar

La función registrar recibe los parámetros nickname y password a través de una función Javascript (que envía esta información a través de la cabecera url) dentro de un archivo HTML. La primera acción que ha de realizarse es comprobar si los datos ya se encuentran en la base de datos. Si los datos ya están en las tablas correspondientes de los usuarios, enviará un mensaje diciendo que el registro no ha sido posible.

En caso contrario la función crea un nuevo usuario sin información y procede a guardar todos los campos correspondientes del nuevo usuario.

En la figura 3.8 podemos ver el diagrama de flujo correspondiente a la función Registrar() de la aplicación.

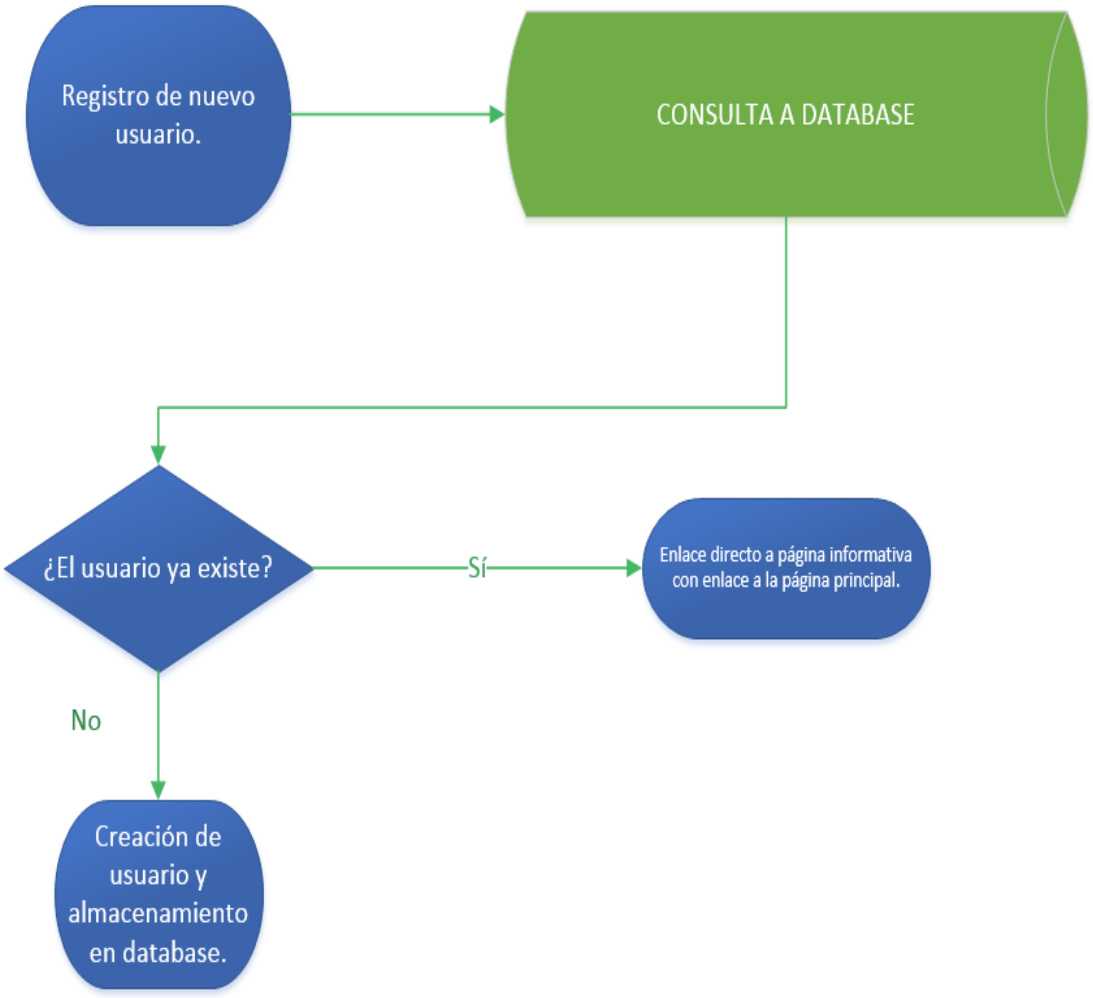

FIGURA 3.8: Diagrama de flujo Registrar()

3.5.2. Validar

El funcionamiento de la función validar tiene la misma estructura que registrar pero cambian las consultas y las operaciones.

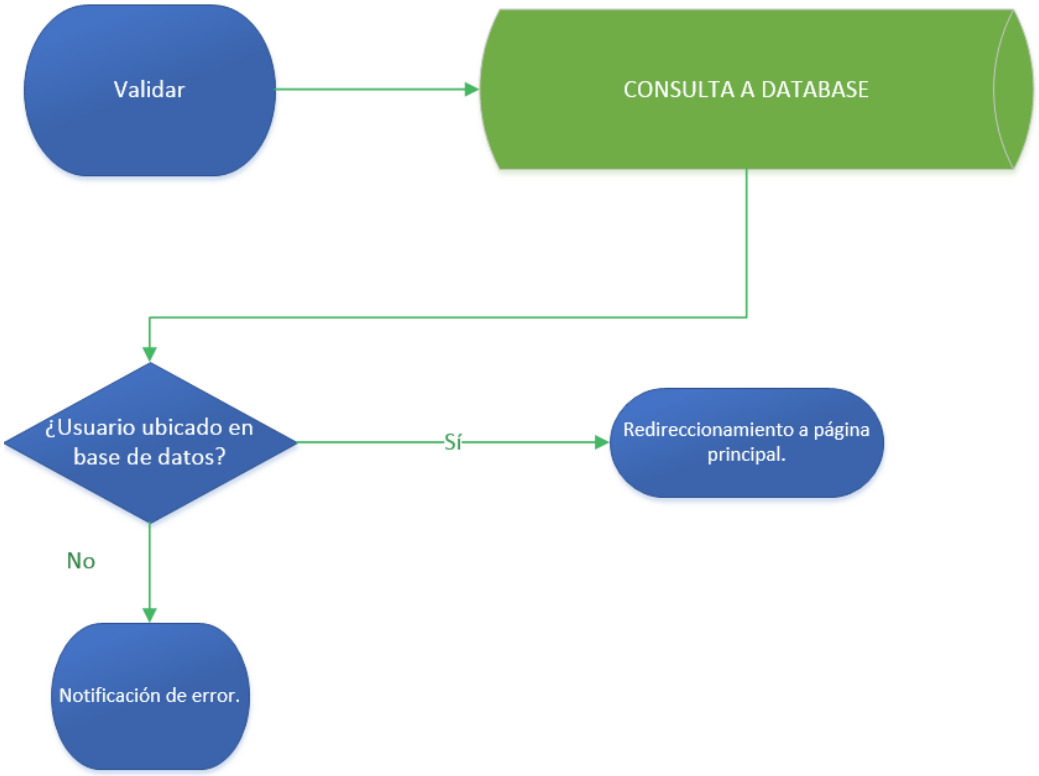

FIGURA 3.9: Diagrama de flujo Validar()

Al introducir los datos (Usuario y contraseña) en la página principal, el archivo HTML le pasa la información correspondiente a la función validar en las vistas de Django. La función validar realiza una consulta a la base de datos preguntando si el usuario esta inscrito en sus tablas. En caso afirmativo, Django redireccionará al usuario a la página principal de la aplicación, y en el caso contrario, informará al usuario que los datos introducidos son erróneos.

El paso de la información de control a Django desde el lado cliente de la aplicación se consigue mediante una función Javascript que se encuentra en el archivo HTML en cuestión. Al seleccionar el botón de login, se ejecuta la función Javascript que transmitirá el nombre del usuario y su correspondiente contraseña a la función validar (localizada en el archivo views.py) a través de la cabecera (de la misma forma que la función de registro).

3.5.3. Modificar

Cada usuario registrado de la aplicación tendrá la opción de modificar sus datos personales (Nombre de usuario, contraseña, foto, etc).

La primera acción que realiza la función es solicitar de nuevo al usuario su nombre y contraseña, con esto evitaremos que otra persona ajena pueda modificar los datos si la sesión del usuario no se ha cerrado.

Posteriormente, la función buscará en las tabla de usuarios al usuario que solicita cambiar sus datos. Una vez encontrado, al aceptar la petición se actualizaran los datos correspondientes.

En la figura 3.10 podemos ver el diagrama de flujo correspondiente a la función Modificar() de la aplicación.

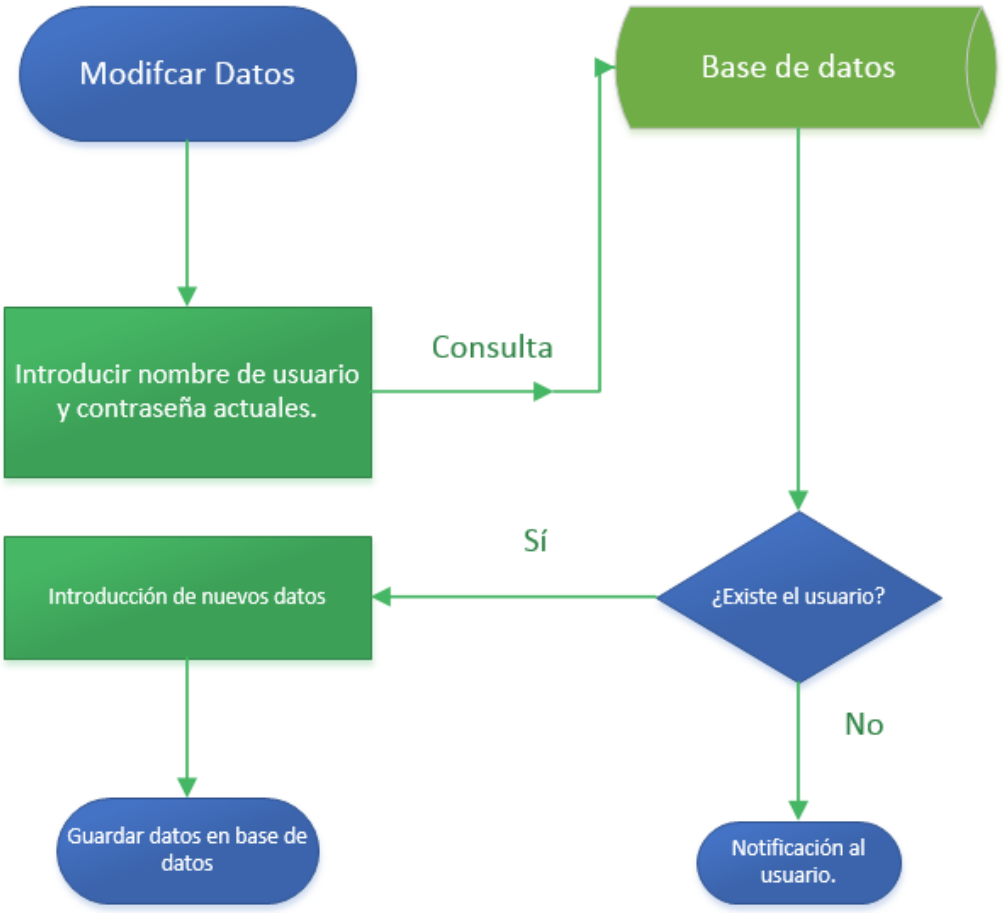

FIGURA 3.10: Diagrama de flujo Modificar()

3.5.4. Test

La función Test corresponderá a la función que se encarga de mandar las correspondientes preguntas al HTML y una vez que finalice el test, guardara un registro con la partida realizada por el usuario y su puntuación correspondiente en la base de datos.

Es importante destacar que las preguntas correspondientes a cada test son diferentes en cada usuario. Se hace uso de una variable aleatoria que permitirá que las preguntas que conteste un usuario sean distintas a las que realice otro usuario en otra sesión.

A continuación, en la figura 3.11 podremos observar un diagrama de flujo representantivo de la función encargada de sacar por pantalla las preguntas que realizarán los alumnos o usuarios.

La puntuación se tendrá que modificar cada vez que se elija una opción en la pregunta y se pulse al botón siguiente (una vez pulsado el botón aparecerá otra pregunta diferente). La puntuación se guardara únicamente cuando el usuario haya terminado el test o haya elegido abandonarlo. La razón primordial de esto se debe a que es poco eficiente consultar la base de datos y modificar el campo de puntuación cada vez que se conteste a una pregunta.

Para la realización de esta operación se ha procedido a utilizar las variables de sesión, que evitarán que la aplicación este constantemente consultando a la base de datos y mejorará la eficiencia de ésta.

3.5.4.1. ¿Qué son las variables de sesión?

Las variables de sesión son variables que guardaremos en el caché del navegador web. Serán individuales a cada usuario puesto que dependerán únicamente de la sesión del usuario.

Las variables de sesión nos permitirán guardar y recuperar información arbitraria que sea de interés para el programador. En el caso expuesto anteriormente, guardaremos la puntuación que vaya obteniendo el alumno en una variable de sesión llamada puntuación que se ejecutará mediante el comando request.session['puntuacion']. Cada pregunta que acierte o falle el alumno modificara esta variable y cuando el alumno finalice el test se procederá a guardar en la tabla juego_partida (figura 3.7) la puntuación que aparezca en esta variable de sesión 'puntuación'.

Además de esta función, las variables de sesión ayudan a mejorar la eficiencia de cualquier aplicación, ahorrando al sistema a realizar múltiples consultas y modificaciones a la base de datos. En el caso de una aplicación con poco tráfico de usuarios este aspecto puede parecer poco significativo, pero cuando se trata de aplicaciones muy concurridas puede suponer un incremento sustancial en la eficiencia de la página.

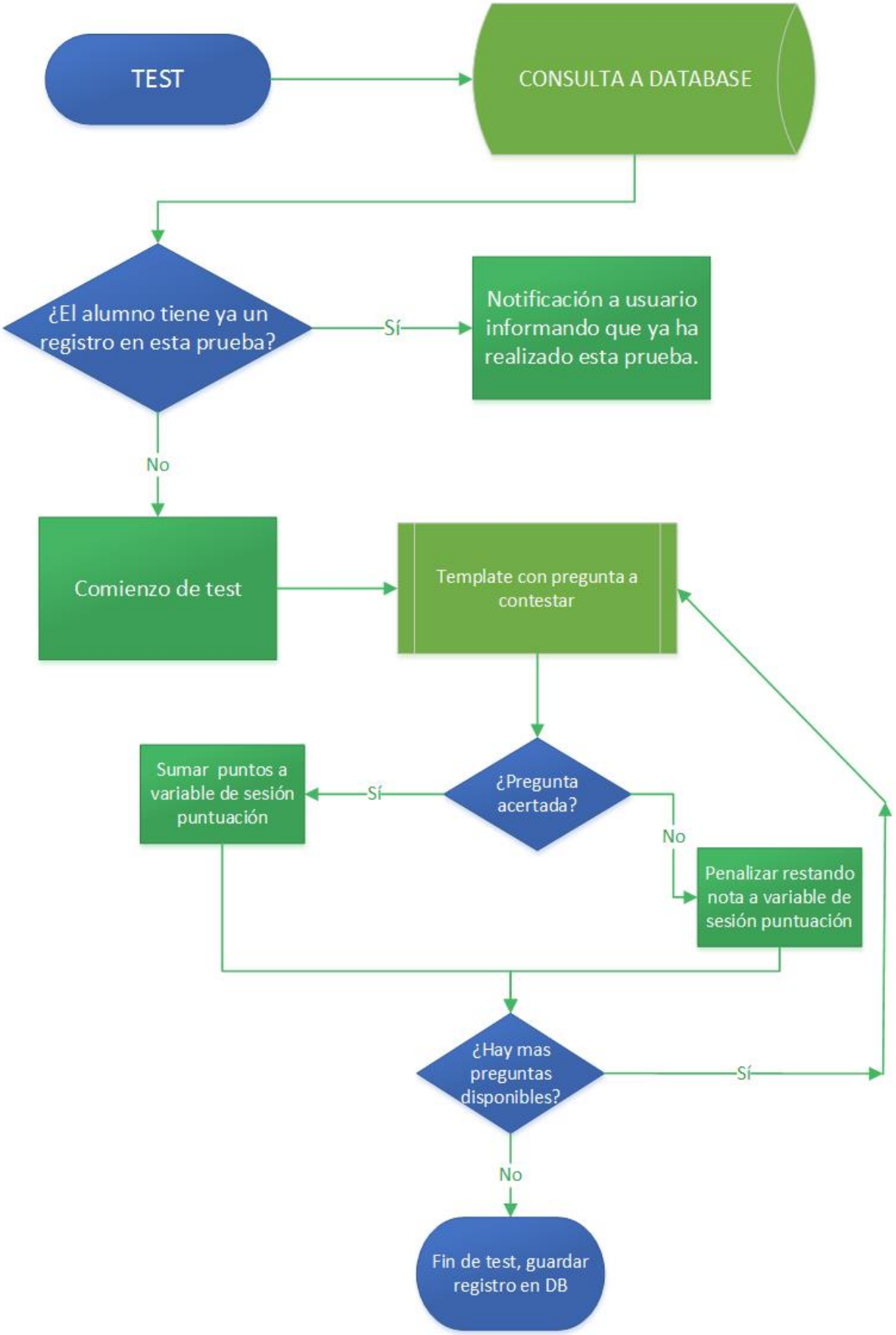

FIGURA 3.11: Diagrama de flujo función Test

3.6. Estructura del Código

En el capítulo 2 mencionamos las características primordiales de Django, entre ellas se mencionaba que Django empleaba la arquitectura MVC (Modelo, Vista y Controlador). El código en Django se va a dividir de la siguiente forma:

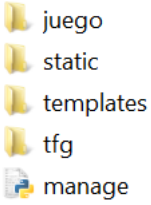

FIGURA 3.12: Estructura de carpetas en Django

Dentro de esta estructura podemos diferenciar las carpetas principales con las que se trabaja en Django.

3.6.1. Templates

En la carpeta de templates colocaremos todas las vistas (HTML) que se emplearán en la aplicación. En la figura 3.13 podemos observar la composición parcial de esta carpeta.

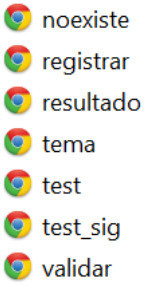

FIGURA 3.13: Contenido de templates

3.6.2. Juego

En la carpeta juego se localizarán todas las funcionalidades propias de la aplicación en cuestión. También incluirá los modelos de la aplicación y la activación del panel de administración ya explicado anteriormente.

En el archivo denominado views.py (figura 3.14) se encontrarán todas las funciones lógicas que realiza la aplicación.

3.6.3. Carpeta TFG

Por último, tendremos la carpeta tfg (figura 3.15). En esta carpeta encontraremos principalmente 2 archivos importantes. En primer primer lugar, situaremos en esta carpeta el archivo urls.py cuya función sera la de indicar todas las cabeceras de la página web y relacionarlas con sus respectivas funciones (de la carpeta view.py). En segundo lugar encontraremos el archivo settings.py donde principalmente estableceremos la información acerca de qué tipo de base de datos emplearemos y todas las rutas de los archivos que se utilizarán en la aplicación.

Esta estructura otorgará al programador gran flexibilidad a la hora de trabajar permitiendo al programador centrarse en los diferentes módulos de forma individual.

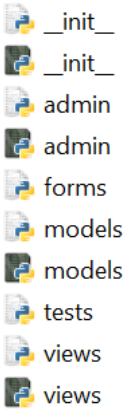

FIGURA 3.14: Carpeta juego

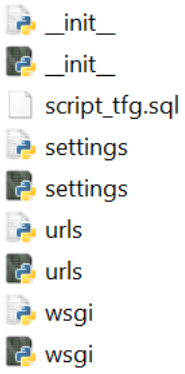

FIGURA 3.15: Carpeta tfg

3.7. Diseño de vistas

Para el diseño de las vistas se hace uso de una plantilla común llamada *base1.html*. El objetivo de esto es permitir que todas las plantillas de la aplicación sigan la misma estructura y que únicamente varie la información plasmada en la vista (controlado por las vistas de la aplicación). Todas las plantillas van a heredar la estructura de base1.html y hará que la apariencia de la página sea muy ordenada. Por ello, haciendo uso de CSS, diseñamos las diferentes partes de la vista (se puede ver el diseño en el capítulo de validación) y elegimos un estilo diferente para cada parte. En este aspecto trabajar con CSS es muy cómodo puesto que deja al programador centrarse exclusivamente en el diseño cuando el resto de los módulos ya estan implementados.

3.8. Facilidades para el administrador

La interfaz de administración en Django es una característica, como ya mencionada en el capítulo 2, que va a proporcionar facilidades al administrador de la página. El acceso a está página en principio esta destinado a un administrador, pero Django nos permite modificar los permisos de los usuarios pudiendo el administrador autorizar a más usuarios a acceder a este contenido y modificarlo.

Dentro de las acciones que deberá poder realizar el administrador se encuentran:

- **Autenticación en aplicación:** el administrador de la página debe poder acceder a la aplicación de la misma forma que los usuarios o alumnos.

  Esta funcionalidad viene de serie en la aplicación y no tiene mayor dificultad. El administrador como cualquier otro usuario esta registrado en la base de datos de la aplicación, y por tanto el registro se produce de la misma forma que un usuario normal.

- **Autenticación en panel de administrador:** el administrador en cuestión deberá poder acceder al panel de administración, y desde este punto, controlar todos los datos que circulan por la página (agregar, modificar y eliminar preguntas, verificar a los usuarios, etc).

  En el caso de éste trabajo de fin de grado, el panel tendrá un aspecto como el siguiente:

FIGURA 3.16: Página principal del panel de administración

Desde este punto, el administrador podrá acceder a toda la información contenida en la base de datos de una forma muy sencilla e intuitiva. Por ejemplo, si el administrador o profesor quisiera añadir una pregunta a la aplicación, únicamente tendría que seleccionar el enlace 'Preguntas' y seleccionar la opción 'añadir' donde encontraría la siguiente pantalla:

FIGURA 3.17: Añadir Pregunta

Nota: *es importante recordar que los campos de la opción 'Añadir Pregunta' provienen directamente del modelo de Pregunta situado en la base de datos MySQL.*

El administrador además, podrá proporcionar permisos a otros usuarios a través de la página de administración, permitiendo que otro usuario pueda también añadir preguntas. Esta última característica es de gran utilidad si se requiere que existan varios profesores (superusuarios) subiendo preguntas a la vez.

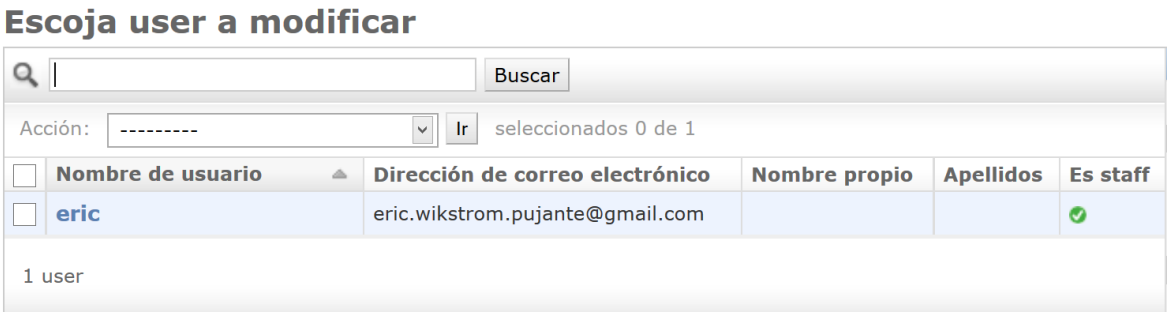

FIGURA 3.18: Modificación de permisos

- **Ver clasificación:** el administrador al acceder a la página principal deberá poder seleccionar la opción de mostrar la clasificación.

  Una vez autorizado por la aplicación, en la página principal tendrá que elegir la opción de 'Ver Clasificación' para poder acceder al contenido relacionado con los resultados obtenidos por los alumnos.

FIGURA 3.19: Ejemplo de tabla de clasificación

Capítulo 4

Validación de requisitos

El objetivo de este capítulo es verificar el correcto funcionamiento de los requisitos mencionados en el capítulo de Diseño e Implementación (Capítulo 3).

Por ello, haremos uso de una serie de capturas de pantalla obtenidas directamente de la aplicación en ejecución. Con esto se conseguirá obtener una visión de como funciona la aplicación para el usuario, alejandonos de una descripción más técnica de la misma.

La primera toma de contacto que tiene el usuario con la aplicación se produce en la vista índice. Como se puede obsevar en la siguiente figura se aprecia una división de la plantilla, por un lado para registrarse y por otro para realizar el login:

Figura 4.1: Pantalla de inicio

4.1. Requisitos Usuario

- **Requisito 1:** registrar usuario en la base de datos.

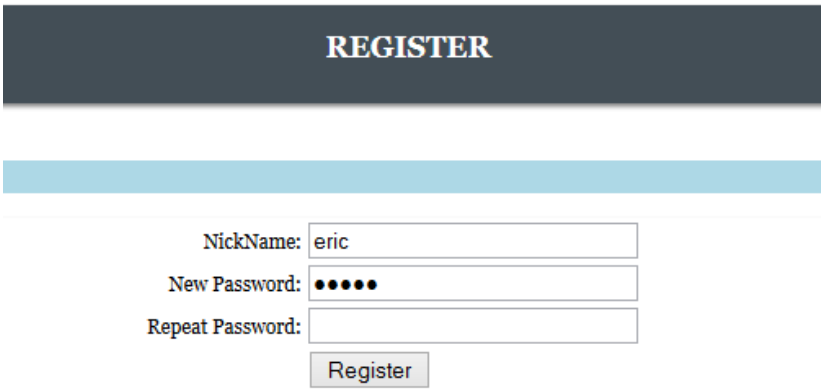

FIGURA 4.2: Registro de usuario

Después del procedimiento de registro, el usuario estará inscrito en la tabla de usuarios de la base de datos MySQL, tal y como se puede observar en la tabla usuario (ejemplo).

```
mysql> select *from juego_usuario;
+----+------------------+----------+------+--------+---------+
| id | nickname         | password | foto | activo | borrado |
+----+------------------+----------+------+--------+---------+
|  1 | Eric Wiktrom     | admin    |      | 1      | 0       |
|  2 | alfonso          | admin    |      | 1      | 0       |
|  3 | paco             | admin    |      | 1      | 0       |
|  4 | Victor Jiménez   | hola     |      | 1      | 0       |
|  5 | Cristina Sanchez | hola     |      | 1      | 0       |
|  6 | alejandro        | admin    |      | 1      | 0       |
|  7 | fernando         | admin    |      | 1      | 0       |
|  8 | alicia           | admin    |      | 1      | 0       |
|  9 | eric             | admin    |      | 1      | 0       |
| 10 | laura            | pleite   |      | 1      | 0       |
| 11 | ernesto          | villa    |      | 1      | 0       |
| 12 | arancha          | muelas   |      | 1      | 0       |
+----+------------------+----------+------+--------+---------+
```

FIGURA 4.3: Tabla de usuarios en DB

En la figura 4.3 podemos observer también como los datos correspondientes al modelo usuario (Capítulo 2) están correctamente establecidos en la base de datos (Nombre usuario, contraseña, activo, borrado, etc)

- **Requisito 2:** Después del procedimiento de autenticación, el usuario será redirigido a la página principal de la aplicación donde podra elegir las funciones correspondientes a realizar. Las opciones que el usuario está habilitado a realizar se encuentran en la parte superior y se ha diseñado de tal forma que ocupe todo el ancho de la página.

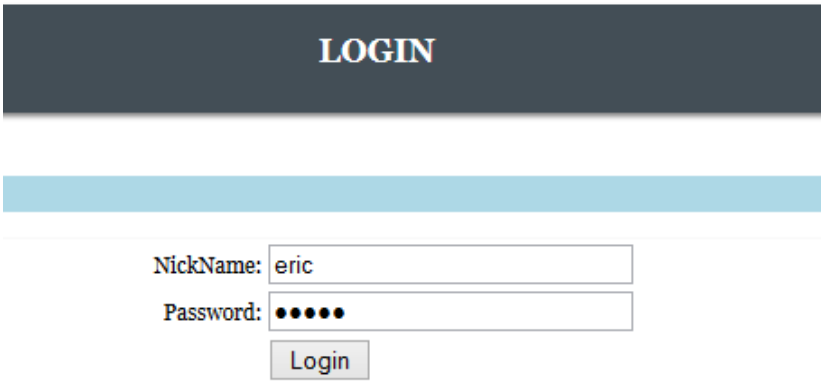

FIGURA 4.4: Login de usuario

FIGURA 4.5: página principal de la aplicación

- **Requisito 3:** realizar test.

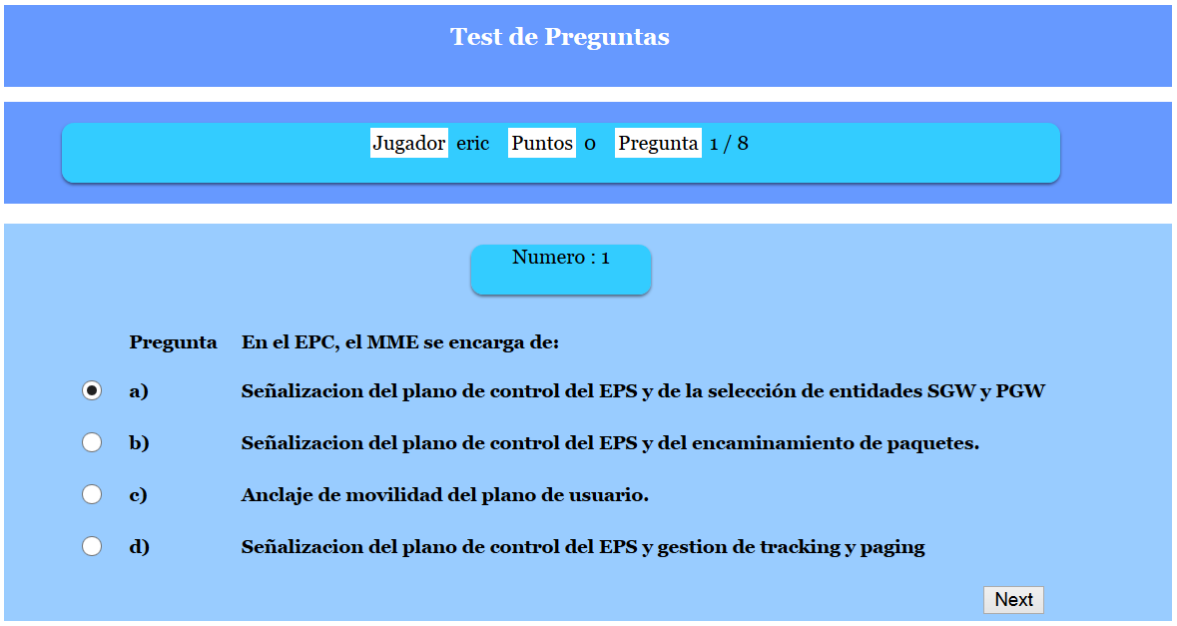

FIGURA 4.6: Ejemplo de pregunta

En esta vista, el alumno empezará a realizar el test. La aplicación le imprimirá el número de preguntas a realizar y la puntuación actual. Esto permitirá al alumno tener la opción de abandonar el test en caso de conformarse con la nota obtenida hasta el momento determinado. Las preguntas, como ya mencionamos en el capítulo anterior, saldran también de forma aleatoria.

Una vez finalizado el test, la aplicación imprimirá la puntuación conseguida y el alumno podrá acceder a la clasificación para ver el puesto en el que se ha situado.

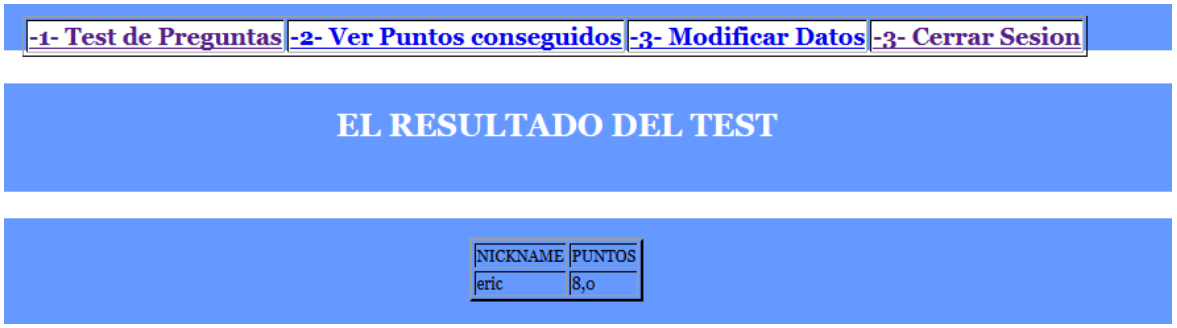

FIGURA 4.7: Notificación de resultado

- **Requisito 4:** ver clasificación global de la clase.

FIGURA 4.8: Clasificación global de la clase en un tema

El usuario podrá elegir ver la clasificación de los determinados temas eligiendo la opción 'Ver puntos conseguidos'.

- **Requisito 5:** abandonar test.

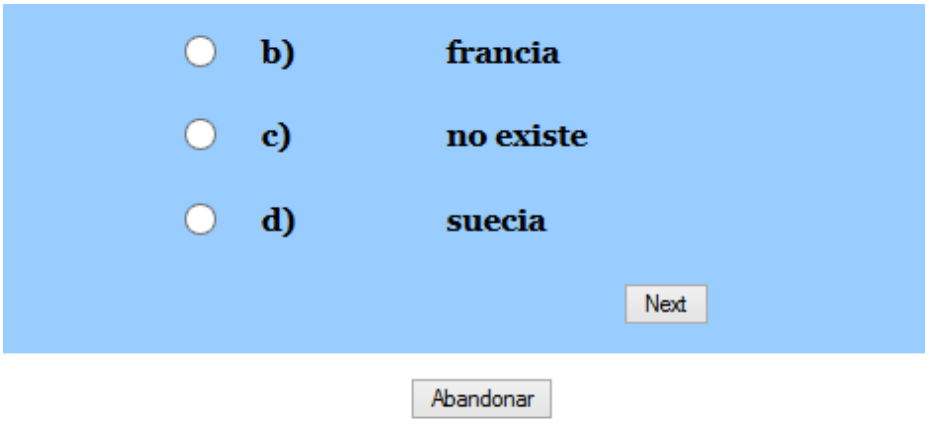

FIGURA 4.9: Opción de abandonar la prueba

Como ya se explicó en el capítulo anterior, la idea de permitir que el alumno abandone la prueba cuando lo crea conveniente se debe principalmente a que las preguntas restan puntuación. Como el objetivo final del alumno es quedar lo más alto posible en la

clasificación, podrán abandonar la prueba cuando consideren que han obtenido una buena puntuación.

- **Requisito 6:** control de accesos a los test.

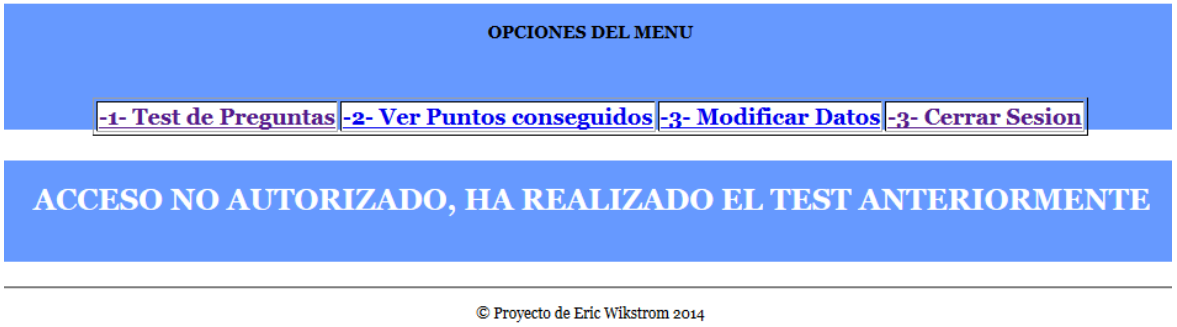

FIGURA 4.10: Control de accesos por parte de los usuarios

El control de los accesos por parte de los usuarios se controlará de una forma muy sencilla. Antes de ejecutar la función que imprimirá por pantalla la primera pregunta de la prueba, la aplicación hará una consulta a la base de datos.

Esta consulta corresponderá en preguntar a la base de datos si en la tabla partida (donde esta la información de las puntuaciones de todos los alumnos) existe algún registro entre el usuario realizando la petición de test y la puntuacón obtenida.

Si la búsqueda tiene un resultado igual a 0 significará que el usuario jamás ha realizado la prueba elegida (en la base de datos no existirá ningun registro que relacione al usuario con esa prueba) y por tanto tendrá acceso a la realizacón de la prueba.

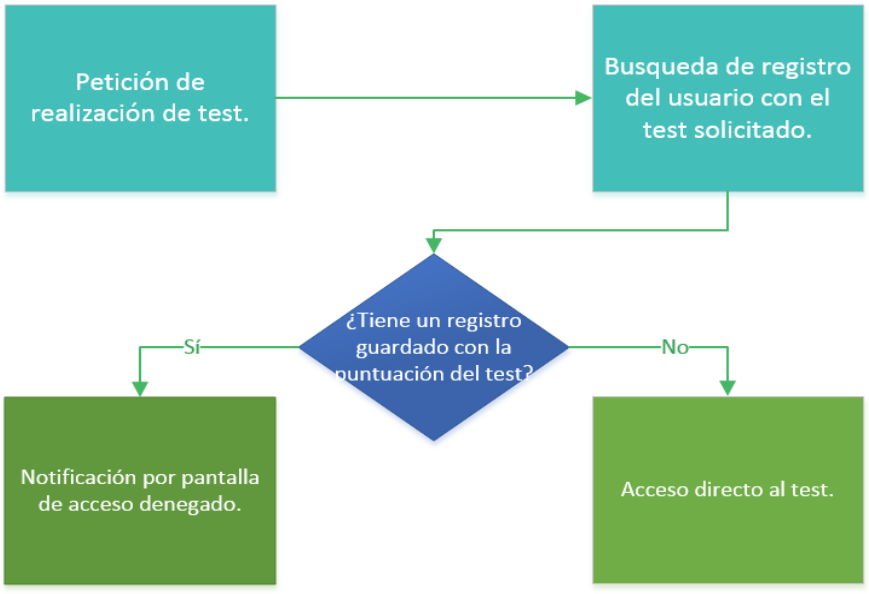

FIGURA 4.11: Diagrama de flujo de control de acceso a test

- **Requisito 7:** modificar datos de usuario.

FIGURA 4.12: Modificar datos de usuario

El usuario podrá introducir sus datos personales e modificarlos. Como ya mencionado en el capítulo de 'Diseño e Implementación' la aplicación comprobará si los datos del usuario son válidos y procederá a la modificación de los datos en las tablas de la base de datos. Si los datos introducidos por el usuario no son correctos, la aplicación notificará al usuario de esta información mediante un mensaje.

Capítulo 5

Conclusiones y Trabajo futuro

5.1. Conclusión

En este trabajo de fin de grado se ha implementado la primera fase de un sistema interactivo de competición en la web empleando el *framework* Django.

En el se ha creado una aplicación web dinámica cuyo objetivo principal es permitir al profesor de cualquier asignatura crear, editar y gestionar una serie de pruebas con preguntas tipo test orientado hacia los alumnos. El profesor en cuestión no deberá tener ningún conocimiento acerca de bases de datos puesto que se le facilitará un panel de administración intuitivo en el cual podrá introducir toda la información relacionada con la aplicación.

Esta aplicación ha servido como iniciación para aprender una base sobre le funcionamiento principal que rodea una aplicación web. Este conocimiento pasa por aprender como se comunican entre sí todos los lenguajes empleados en una aplicación web (Django, HTML, CSS, JavaScript, MySQL).

Finalmente, cabe destacar que la elección de realizar esta aplicación ha supuesto, personalmente, un gran acierto. Ha posibilitado la absorción de conocimientos no vistos en la carrera y que serán sin duda de gran utilidad y una herramienta muy importante para mi carrera profesional en un futuro.

5.2. Trabajo futuro

Una ventaja que tienen muchas aplicaciones web es la posibilidad de siempre incorporar mejoras o cambios en ella. Centrandonos a este trabajo de fin de grado se podrían incorporar mejoras del tipo:

- Incorporar la edición de los test en la página principal (únicamente para el administrador) para facilitar tareas al administrador.
- Poblar la base de datos con una amplia categoría de preguntas.
- Mejorar el aspecto visual de las plantillas.
- Incorporar un foro para la resolución de conflictos.
- Incorporar la clasificación de las pruebas en la página de administración.

En el primer capítulo mencionabamos que este trabajo de fin de grado corresponde a la fase 1 de un proyecto más complejo sobre un sistema de competición. En la siguiente figura podemos observar los diferentes módulos (ya comentados en el primer capítulo) del sistema de competición, y junto a ellos se resalta el porcentaje que este trabajo o fase 1 corresponde de la totalidad del sistema de competición.

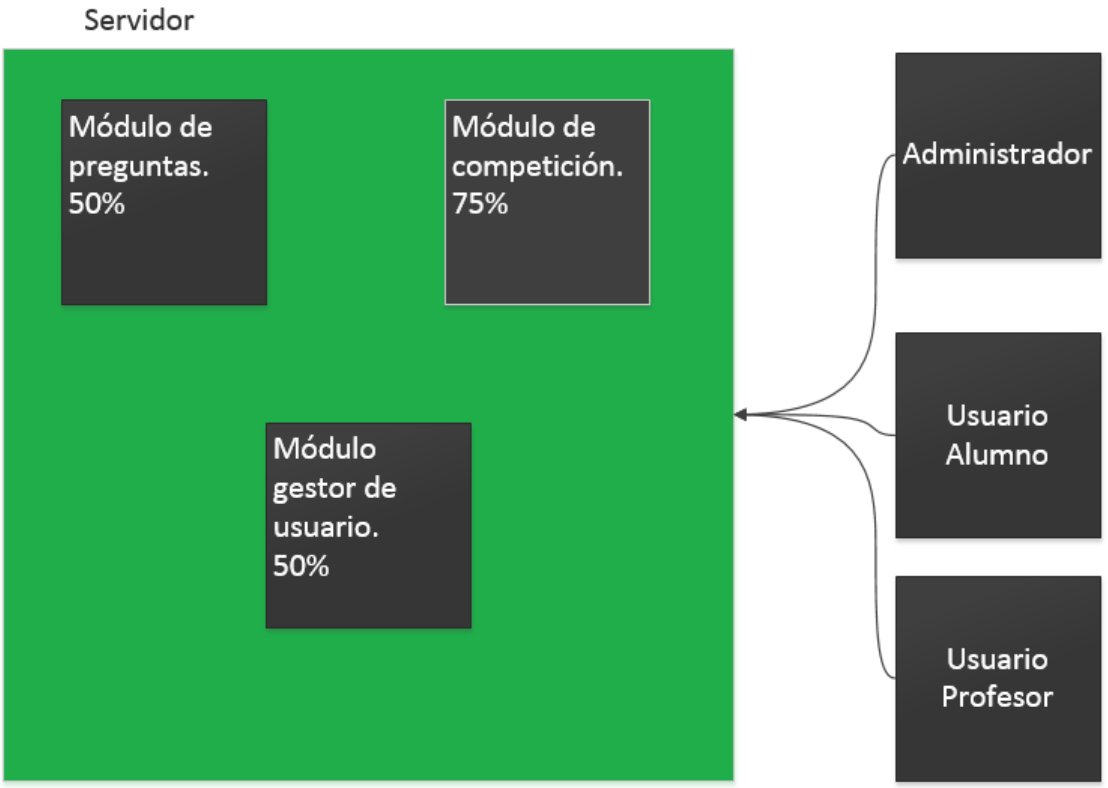

FIGURA 5.1: Módulos del sistema de competición

Para el futuro desarrollo de los diferentes módulos sería necesario incorporar nuevas funcionalidades:

- Para el módulo de preguntas sería conveniente permitir que pudiesen acceder a la aplicación una gran variedad de profesores de diferentes asignaturas y permitir que

agregasen y modificasen desde aquí las preguntas (sin necesidad de conectarse a la página de administración). Con esto se facilitaría la tarea de la edición de los test para los profesores que les resulte más lioso acceder a la página de administración.

- Para el módulo de competición sería necesario guardar el registro de las puntuaciones de los duelos entre los diferentes alumnos para posteriormente tratar esta información de la forma adecuada. Sería necesario modificar y añadir nuevas tablas a la base de datos que permitiese guardar un registro de información entre la partida de 2 alumnos en concreto.

- Para el módulo de gestión de usuarios sería necesario separar a los diferentes usuarios por clase y habilitar el acceso de éstos únicamente a la página principal de sus correspondientes asignaturas. De esta forma los alumnos podrán acceder únicamente a la página que lleva el sistema de competición de la asignatura en la que están matriculados.

Como mencionamos en el primer capítulo, en la actualidad son abundantes las aplicaciones que mezclan diversión y conocimientos. Nos encontramos en un época donde el denominado e-learning está sufriendo una amplia evolución y donde se realizan estudios demostrando que cuanto más entretenido este el alumno mientras este aprendiendo algo, más información se recolecta en su cerebro y más fácil es para el individuo aprender. Esto está provocando que cada dia, miles de profesionales de la docencia se interesen cada vez más en que sus técnicas de aprendizaje se centren en procesos más divertidos para los alumnos.

El objetivo del sistema de competición es aumentar la motivación individual del alumno y añadir al estudio un componente lúdico que favorezca que trabaje día a día y permitir así una absorción de conocimientos progresiva y eficiente.

Capítulo 6

Planificación y Presupuesto

6.1. Planificación

El proyecto tiene una duración aproximada de 4 meses. Para comenzar su realización primero fue necesario un estudio acerca de las tecnologías de programación para la web existentes (estudiando cual se adaptaría mejor a estas características).

En la siguiente tabla podemos observar las distintas tareas que han permitido el desarrollo de la aplicación y el tiempo empleado en cada una de ellas.

	Tarea	**Fecha de inicio**	**Fecha de fin**
1	Estudio sobre las diferentes tecnologías de desarrollo web	15/1/2014	31/1/2014
2	Aprendizaje en Django y Python	3/2/2014	1/4/2014
3	Instalación y aprendizaje MySQL	2/4/2014	4/4/2014
4	Creación de Modelos	7/4/2014	14/4/2014
5	Activacion página de administración	15/4/2014	15/4/2014
6	Creación de página principal de registro y acceso	16/4/2014	18/4/2014
7	Creación del resto plantillas web para la aplicación	21/4/2014	16/5/2014
8	Lógica de la aplicación	21/4/2014	16/5/2014
9	Pruebas y actualizaciones	19/5/2014	30/5/2014

Como se puede observar en tabla de arriba, la duración del proyecto es de aproximadamente 4 meses. En la tabla podemos apreciar que se han dividido las etapas de desarrollo en 9 tareas correspondientes.

A continuación explicaremos brevemente en que consisten las diferentes tareas pertenecientes al desarrollo del trabajo de fin de grado:

- **Tarea 1**: la primera tarea corresponde a la iniciación del proyecto. Es una estado donde el tiempo empleado se dedica exclusivamente a estudiar como funcionan las aplicaciones web y que conocimientos van a requerirse para realizar el trabajo.
- **Tarea 2**: después de la primera fase, es necesario empezar a estudiar tecnologías que serán necesarias para desarrollar la aplicación como el caso de Django y Python. Esta fase es la más larga puesto que es necesario asentar un base sólida sobre las materias para después realizar el trabajo de la forma más eficiente. En esta fase se hace gran uso de tutoriales[4][5] en internet para facilitar el aprendizaje.
- **Tarea 3**: el aprendizaje de Django se llevo a cabo empleando la base de datos SQLite3 , pero por motivos de eficiencia se empleó MySQL como motor de base de datos de la aplicación y por ello esta fase se centra en la instalación y aprendizaje de MySQL.
- **Tarea 4**: para empezar a trabajar en la aplicación es necesario establecer los modelos que seguirán los datos de la aplicación. Esta fase se centra en determinar la estructura de los determinados modelos necesarios en la aplicación.
- **Tarea 5**: el primer paso en cualquier aplicación web en Django es activar el panel de administración para poder controlar los datos. Esta fase se corresponde con la activación de la página de administración.
- **Tarea 6**: como la aplicación va a requerir un registro para los usuarios, el paso siguiente es crear una página de acceso a la aplicación y controlar el acceso únicamente a los usuarios registrados en la base de datos.
- **Tarea 7**: despues de crear esta página de acceso, se procedera a diseñar el resto de plantillas (estas plantillas luego podrán sufrir ciertos cambios) que se emplearán en la aplicación.
- **Tarea 8**: esta fase es fundamental puesto que se va a trabajar con la lógica principal de la aplicación, relacionada principalmente con la realización de los test.

  Nota: *Las tareas 7 y 8 son tareas diferentes pero el tiempo en las que se desarrollan es el mismo puesto que la lógica de la aplicación incluye lo que el usuario esta observando en la pantalla y por ello, se trabaja a la par tanto con las plantillas como con las funciones lógicas de la aplicación.*
- **Tarea 9**: la última etapa se dedica a la realización de pruebas y modificaciones en el proyecto para que todos los requisitos se cumplan de la fomra esperada.

A continuación se muestra un diagrama de gantt sobre la planificación del trabajo. En el diagrama de gantt las tareas están distribuidas de forma mensual y es una forma de elaborar una idea visual de la evolución del trabajo.

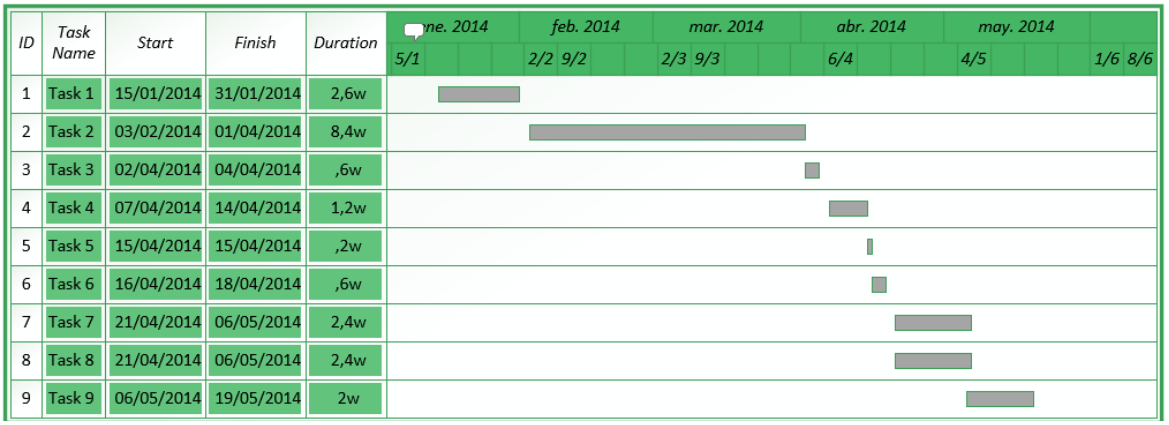

FIGURA 6.1: Diagrama de gantt

6.2. Presupuesto

En esta sección presentaremos el presupuesto total para la realización de la aplicacion web que fundamentalmente se calcula teniendo en cuenta 2 datos:

- **Horas de dedicación**: en la tabla de acontinuación podremos observar la distribución de las horas dedicadas para la realización del trabajo de fin de grado teniendo en centa la duración de las tareas en la planificación.

Tarea 1	**52 Horas**
Tarea 2	**168 Horas**
Tarea 3	**12 Horas**
Tarea 4	**24 Horas**
Tarea 5	**4 Horas**
Tarea 6	**12 Horas**
Tarea 7	**32 Horas**
Tarea 8	**48 Horas**
Tarea 9	**20 Horas**

Para el cálculo de las horas empleadas finales se hace una estimación del tiempo empleado para cada tarea (consultando la tabla de planificación). Y simplifcando las tareas podemos agruparlas finalmente en 3 fases:

Fase 1	**Documentación y Aprendizaje**	**220 Horas**
Fase 2	**Desarrollo de Software**	**132 Horas**
Fase 3	**Desarrollo de la memoria**	**60 Horas**

De forma individual cada fase se compone por:

- **Fase 1**: en la fase 1 se incluye tanto el tiempo relacionado con el estudio de las tecnologíaas como el tiempo empleado en aprender a emplear dichas tecnlogías.
- **Fase 2**: en la fase 2 vamos a incluir el desarrollo de todos los modulos que componen el proyecto (Plantillas, base de datos, modelos, etc)
- **Fase 3**: en la fase 3, para la realización de la memoria también se incluye el tiempo empleado en aprender a manejar el editor Latex[7] para la realización de la memoria.

Cabe destacar que el cálculo de las horas hace referencia únicamente a días laborables. Cada día laborable corresponde con una serie de horas de dedicación, y cabe resaltar que no todas las horas corresponden exclusivamente a la realización del trabajo de fin de grado. Estas consideraciones son necesarias para finalemente calcular una estimación del presupuesto total.

- **Material empleado**: en la tabla que se muestra acontinuación se observarán los costes relacionados con el material empleado para la realización de este trabajo de fin de grado.

Objeto	**Uso**	**Coste**	Precio de amortización
Ordenador	Desarrollo de aplicación	900 euros	75 euros
Libros	Documentación	60 euros	-

El coste final del ordenador se calcula considerando que la vida útil del objeto es de 5 años, y aplicando la devaluación correspondiente al tiempo que se ha empleado en desarrollar la aplicación.

Para calcular el presupuesto final se incluirá el precio amortizado del ordenador empleado y además el precio de las horas dedicadas en realizar la aplicación en cuestión (Se establece que le precio por hora de dedicación es de 25 euros sin el I.V.A incluido).

Gastos	**Precio Final sin I.V.A**
75 + 60 + 132 Horas * 25 (euros)	3435 euros

Bibliografía

[1] **Autor: Raúl González Duque** Titulo: *Pyhton para todos.*

[2] **Autor: Robin Nixon** Titulo: *Learning PHP, MySQL, Javascript, CSS, HTML5. A step by step guide to creating dynamic web sites.*

[3] **Autor: Eric Freeman, Elisabeth Robson** Titulo: *Head First JavaScript Programming.*

[4] *Tutorial youtube de Django: http://www.youtube.com/user/MickeySoFine1972* **Autor: Mike Hibbert**

[5] *Tutorial youtube de Django: http://www.youtube.com/watch?v=-ahaS1g58x4* **Autor: Gonzálo Caminos**

[6] **Autor: Adrian Holovaty, Jacob Kaplan-Moss** Titulo: *The Definitive Guide to Django.*

[7] *Tutorial Latex, http://ece.uprm.edu/f_perez/Latex/2005/manual.pdf* **Autor: Freddy Pérez**

[8] *Página web, http://www.agpd.es* **Consultado: 5-6 Junio 2014**

[9] *Archivo LOPD:* http://www.agpd.es/portalwebAGPD/canal-documentacion/publicaciones/common/pdfs/guia_seguridad_datos_2008.pdf **Consultado: 5-6 Junio 2014**

[10] *Página web, XML* http://www.monografias.com/trabajos7/xml/xml .shtml#_Toc518143720 **Consultado: 22-23 Enero 2014**

[11] *Página web, XML* http://www.w3schools.com/xml/xml_whatis.asp **Consultado: 22-23 Enero 2014**

[12] *Página web, HTML* http://www.w3schools.com/html/default.asp **Consultado: 4 Febrero 2014**

[13] *Página web, SQLite3* http://www.php.net//manual/es/book.sqlite3.php **Consultado: 18 Febrero 2014**

[14] *Página web, Oracle* http://www.slideshare.net/moRado2/oracle14874943 **Consultado: 18 Febrero 2014**

[15] *Página web, PostgreSQL* http://postgresql-dbms.blogspot.com.es/p/limitaciones-puntos-de-recuperacion.html **Consultado: 18 Febrero 2014**

[16] *Página web, HTTP* http://www.mastermagazine.info/termino/5288.php **Consultado: 23 Febrero 2014**