



GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

Creación de un portal web corporativo

Autor:

Albert ARAUJO FABREGAT

Supervisor:

Antonio JOSÉ BALAGUER VALLS

Tutor académico:

José Manuel BADIA CONTELLES

Fecha de lectura: 16 de noviembre de 2015

Curso académico 2014/2015

Resumen

En este documento explicamos el proyecto desarrollado durante la estancia en prácticas realizada en el marco de asignatura EI1054. La idea principal del proyecto es la de dar a conocer a la empresa “Materiales de Construcción Fadrell” en el mundo de internet. En esta memoria describiremos fundamentalmente la creación de un portal web realizado con Django-cms.

Para lograrlo además de crear un portal web, hemos implementado e integrado en él proyecto en Django para mostrar información de los productos y hemos creado diversos perfiles en redes sociales para dar difusión a la empresa también en este ámbito.

El portal web se ha diseñado e implementado usando el framework Django-cms que es un gestor de contenidos desarrollado basándose en el framework Django.

Palabras clave

Django; Django-cms; Portal web; Bases de datos;

Keywords

Django; Django-cms; Portal web; Databases;

Índice general

1. Introducción	5
1.1. Contexto y motivación del proyecto	5
1.2. Objetivos del proyecto	6
2. Descripción del proyecto	9
3. Planificación del proyecto	11
3.1. Metodología y definición de tareas	11
3.2. Planificación temporal de las tareas	12
3.3. Estimación de recursos del proyecto	13
3.4. Presupuesto	14
3.5. Desarrollo del proyecto	16
4. Entorno de desarrollo	19
4.1. Django	23
4.2. Django-cms	24
5. Análisis y diseño del sistema	27
5.1. Visión global del sistema	27
5.2. Diseño del sistema	28
5.3. Diseño de la interfaz	29
5.4. Estilo del portal web	31
5.5. Diseño de la base de datos	32
6. Implementación, pruebas y documentación	35
6.1. Promoción creación de perfiles	35
6.2. Integración de la venta online en el portal	37
6.3. Desarrollo del portal	42
6.4. Implementación del catálogo	50
6.5. Integración del catálogo en el portal	60
6.6. Desplegado del portal	60
6.7. Documentación y formación	63
7. Conclusiones	65

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto descrito en esta memoria se ha desarrollado en la empresa “Materials de Construcció Fadrell”, situada en la ciudad de Castellón de la Plana. Esta joven empresa local constituida en 2012 se dedica a la compra-venta de todo tipo de materiales de construcción, maquinaria industrial y afines. Esta empresa forma parte del grupo Ibricks, que es un centro de compras y una plataforma de gestión de servicios. Sus servicios se dirigen a almacenes distribuidores de materiales de construcción. Esto les aporta un gran valor añadido, ya que es el segundo grupo más grande a nivel nacional por número de asociados y esto les hace formar parte de un grupo de más de 200 almacenes a nivel nacional.

En cuanto a personal, la empresa está formada por cuatro miembros, todos socios a partes iguales. Sus roles están bastante definidos. En primer lugar tenemos a un empleado que se encarga de realizar las ventas y aconsejar al cliente. Por otro lado tenemos al encargado del almacén, que es quien decide cuándo hay que realizar otro pedido de algún material que se esté agotando. Además se encarga de organizar el almacén para que todo esté en su sitio. Otro de los miembros de la empresa es quien se encarga de repartir los materiales, ya que se trata de materiales que se venden por toneladas y hay que usar un transporte específico. Este miembro de la empresa es el único que tiene el carnet necesario para poder realizar las entregas, así que se encarga de la totalidad de los repartos. Para ello cada pedido está preparado y estudiado, para que tras servir uno, el siguiente sea lo más rápido posible. Por último tenemos al encargado de la gestión contable y financiera y que además se encarga de hablar con distribuidores.

En el apartado técnico en la actualidad la empresa dispone de 2 ordenadores personales con el sistema operativo Windows 7. Se tratan de ordenadores muy básicos ya que solo se usan para manejar el software de contabilidad GESTWIN, utilizar programas ofimáticos y acceder a internet. Además ambos ordenadores utilizan una base de datos local integrada en GESTWIN compartida por una red local.

Esta red local es usada por el software GESTWIN [1] que es su software principal. Este se encarga de la gestión contable y financiera, realización de albaranes de entrada de material y facturación. En la base de datos tenemos los productos y su precio, pero no llega ser un catálogo, ya que en la empresa utilizan los catálogos de los proveedores en papel. Al ser estos catálogos tan extensos y tener muchos proveedores, sería muy complejo tener todos los productos registrados. Por eso la empresa ha seleccionado un subconjunto de materiales y los ha integrado en la base de datos de GESTWIN.

En la empresa ninguno de los miembros es informático, sino que tienen contratado un informático externo, que en caso de error acude a la empresa y lo arregla presencialmente o por internet si la urgencia es extrema y el error lo permite.

La intervención de este informático externo es esporádica, pero fue este mismo informático quien les aconsejó qué ordenadores debían comprar para una empresa de esas características. Además instaló todo el software que hay en la actualidad y se encarga de la totalidad del mantenimiento, ya sea del software como del hardware.

Este informático se eligió, por ser conocido por un miembro de la empresa, que ya había trabajado con él en el pasado. Se dedica a este tipo de gestiones en diversas empresas a nivel individual.

Debido a lo anterior, se acordó que el supervisor de la estancia en la empresa se encargaría de toda la parte de las prácticas que no tuvieran que ver con cuestiones técnicas, y sería el tutor de la UJI el encargado de hacer de director del proyecto.

En cuanto a la motivación del proyecto, esta radica en que mucha de la competencia directa de la empresa sí que tiene una web corporativa. Además su presencia en internet es nula, lo cual les hace estar en una gran desventaja. Con la creación del portal web la empresa quiere cambiar esta desventaja para transformarla en una ventaja, ofreciendo mejoras que la competencia no tiene. Con estas mejoras se pretende dar a conocer la empresa en un ámbito global pero sobre todo local, para poder llegar al número máximo de clientes posibles y al mismo tiempo ofrecerles un mejor servicio a los clientes actuales. Por otro lado, otra forma de darse a conocer será con la creación de diferentes perfiles en las redes sociales más populares. Como se ha demostrado estas redes son de gran utilidad, para dar visibilidad a la empresa y mantener un contacto con los clientes.

Con todo esto la empresa pretende conseguir:

- Crear y promocionar su imagen en internet.
- Acceder a nuevos clientes.
- Tener otro canal de comunicación con los clientes.
- Mantener información actualizada sobre eventos.
- Promocionar artículos y ofertas.

1.2. Objetivos del proyecto

El objetivo principal de este proyecto es dar a conocer en internet a la empresa. Para ello el proyecto a desarrollar girará entorno a la idea de un portal web, ya que actualmente no poseen ningún tipo de presencia en internet y un portal Web les parece necesario. Este portal Web debe estar orientado sobre todo al cliente potencial, sin olvidarnos de los clientes actuales, ofreciéndole un buen diseño, que lo haga atractivo y sencillo.

Con todo esto pretendemos hacer que sea fácil de usar, que nos proporcione información importante como pueden ser horarios actualizados, ofertas, servicios que ofrece la empresa, etc. El portal contendrá diversas páginas que mostrarán información de contacto, información de la empresa, horarios, servicios, un apartado específico para la localización, información de los últimos eventos de la empresa y un pequeño catálogo.

Aunque se muestre en el catálogo un subconjunto de todos los productos, no se podrán comprar todos por Internet. Esto se debe a las características de los productos que se venden, ya que muchos de ellos no podrían ser transportados por paquetería convencional. Además

la empresa quiere poder hacer cambios en el futuro tanto en el portal como en el catálogo, por lo que habrá que escribir un pequeño manual con los conceptos básicos para el manejo y actualización de su contenido.

Podemos subdividir todos estos objetivos en 2 tipos.

Objetivos empresariales:

- Dar a conocer la empresa en el ámbito de Internet mediante:

- La creación de un portal web.
- La creación de perfiles en distintas redes sociales.
- La optimización de su posicionamiento en los buscadores principales.

Objetivos técnicos:

- El principal objetivo de este proyecto es el diseño, realización e implantación de un portal web para la empresa Materials de Construcció Fadrell.
- Diseño e implantación de una base de datos de los diferentes materiales que puede hacerse accesible desde el portal web en forma de catálogo.
- Integración del proceso de venta de artículos en el portal web.
- Formación del personal en el uso y manejo tanto del portal web como de los diversos perfiles en redes sociales.
- Realización de un manual de administrador del portal web.

Capítulo 2

Descripción del proyecto

Este proyecto se ha realizado en la empresa “Materials de Construcció Fadrell” que se dedica a la compra-venta de materiales de construcción. El código CNAE de la empresa es 4752 que corresponde a “Comercio al por menor de ferretería, pintura y vidrio en establecimientos especializados”.

El proyecto se divide en 3 bloques, el más sencillo de los cuales es la creación de diversos perfiles en las redes sociales más importantes, con el objetivo de darse a conocer. En otro bloque estará la creación de un portal Web, para el que utilizaremos el Framework Django-cms [2]. Por último abordaremos la creación de un proyecto en Django [3] que integraremos con el proyecto realizado en Django-cms y que funcionará como catálogo de la empresa.

Para el portal web se diseñarán diversos apartados para satisfacer las ideas generales del cliente. Cada función básica del portal dará lugar una página diferente:

- Página principal: con información general de la empresa, desde donde poder tener una idea global de a qué se dedica.
- Catálogo: mostrará un subconjunto de todos los artículos que posee la empresa en venta. Esto es debido a que el stock total es muy extenso y la empresa pretende mostrar solo algunos artículos, ya sean por que son nuevos o para destacarlos.
- Servicios: apartado para mostrar todos los servicios que ofrecen tales como el asesoramiento personalizado, entrega de los productos directamente en la obra, etc.
- Localización: información detallada de la localización de la empresa.
- Ofertas: se pretende realizar ciertos productos, ya sean por innovación o por estar rebajado su precio.
- Noticias: apartado para publicar noticias y mostrar imágenes de las reuniones organizadas.

Todos los apartados del portal están abiertos a todos los usuarios. Aunque en las primeras reuniones con el supervisor se habló de la posibilidad de hacer un registro de usuarios sólo para los clientes que fuesen a comprar on-line, se acordó que este no iba ser necesario, ya que no aportaba casi ninguna ventaja y la empresa lo veía casi más como un problema por el tipo de público al que está orientada.

El portal web permite dos niveles de acceso. Por un lado, los usuarios podrán obtener información tanto de la empresa como de los servicios que ofrece. Por otro lado el administrador podrá modificar la información de la empresa el aspecto visual de la web.

Otra de las tareas abarcadas en este proyecto ha sido la creación de perfiles en redes sociales para lo que, se escogieron las dos más famosas, como son Facebook y Twitter. Su función

principal es la de empezar a crear una imagen en Internet para la empresa. Además esto supone hacerle entender la empresa la importancia que tiene este tipo de herramientas en la actualidad.

La idea principal de las redes sociales, es la de dar a conocer la empresa, Facebook está más orientado a publicar fotos y publicidad de la empresa, dónde está, las reuniones que se hacen, etc. La idea de Twitter es más la de promocionar artículos de oferta de manera puntual.

Ambas herramientas se usarán para permitir un contacto permanente y actualizado con los clientes. También serán utilizadas para tener un lugar donde los clientes pueden dejar reseñas y opiniones o para responder a sus dudas.

Capítulo 3

Planificación del proyecto

3.1. Metodología y definición de tareas

Como se ha explicado anteriormente, al no tener un departamento de informática en la empresa, la metodología que se empleará para desarrollar el proyecto será la tradicional [7]. Esto es debido a que es una metodología contrastada y verificada que puede aplicarse de modo individual. No necesita de un supervisor constante, se estudia la idea inicial, se acuerda lo que se pretende obtener y se desarrolla. Esta metodología se adapta a nuestras circunstancias. No se seguirá la metodología tradicional al pie de la letra, sino que se harán algunas variaciones, ya que el contacto con el cliente en la empresa es continuo. Esto nos proporcionará mucho feedback, el cual utilizaremos para corregir pequeños errores o desacuerdos que aparezcan durante el desarrollo del proyecto.

En lo referente a la web, no se tiene ninguna guía o ayuda de la empresa. En las reuniones con el supervisor se acordaron objetivos, contenido y funcionalidad, pero no formatos, estilos, distribución. El portal deberá ser diseñado y configurado al mismo tiempo que se avanza en el proyecto, haciendo las adaptaciones que se obtienen del feedback. En realidad siguiendo la metodología tradicional, se debería realizar un diseño inicial, proponérselo al cliente y luego adaptarlo a su opinión, este cambio debería realizarse una sola vez en todo el proyecto. En nuestro caso al tener un contacto continuo se fue adaptando a las necesidades de la empresa durante la creación del portal.

Con el catálogo sucede lo mismo. Se acordó mostrar información muy general del producto. Al no poder acceder a la base de datos local que usa el software GESTWIN, se tuvo que crear una nueva, para el subconjunto de los productos que la empresa quiere mostrar. Se acordó la funcionalidad que debería tener el catálogo pero no el formato.

El proyecto no se ha desarrollado en equipo, sino que ha sido realizado completamente por mí durante mi estancia en la empresa.

La metodología tradicional tiene pautas muy marcadas: planificación del proyecto, determinar objetivos, análisis del riesgo y desarrollo del proyecto y pruebas. Antes de empezar se necesitarán unas cuantas reuniones con el cliente para ver qué se quiere obtener. Luego habrá que seguir los pasos de la metodología tradicional: Inicio, planificación, ejecución, monitorización y control y cierre del proyecto.

Como podemos ver en los diagramas de Gantt, de las figuras 3.2 y 3.3, sí que se ha seguido las etapas que marca la metodología utilizada. Empezamos definiendo el contexto, alcance y objetivos del proyecto y planificándolo todo previamente. Esto dio lugar a una propuesta técnica inicial.

Dentro de la metodología tradicional, las tareas se intentarán llevar acabo según se indicó en el diagrama de Gantt que se realizó previamente en la propuesta técnica como muestra la figura 3.2. Estas se dividen en:

- **Inicio del proyecto:** En este periodo se definen los objetivos generales del proyecto, el método de trabajo, el entorno de desarrollo y documentación necesaria, así como el formato y estándares del proyecto. Para ello se necesitará hacer diversas reuniones con el cliente.
- **Documentación y análisis:** En este apartado lo principal es definir bien el contexto y a partir de este, buscar la información necesaria, identificar bien el alcance y los objetivos detallados.
- **Planificación:** Consiste en definir las tareas y estimar el tiempo. Este trabajo se verá reflejado en un diagrama de Gantt.
- **Desarrollo técnico:** Esta tarea se divide en 4 subtareas, lo que hace de esta la parte más extensa del proyecto y la de mayor importancia.
 - **Definición de requisitos técnicos:** Definición de casos de uso.
 - **Diseño:** Se definirá el aspecto del portal Web y se realizarán bocetos de sus distintas páginas. Por otro lado se deberá definir y diseñar la base de datos asociada al catálogo.
 - **Desarrollo del producto:** En este apartado se acoplará todo lo realizado anteriormente para la implementación del portal Web, haciendo que encajen todos los pasos previos. El objetivo es obtener un producto acabado y funcional.
 - **Puesta en marcha:** Aquí observamos todos los pasos que hay que dar, para una vez terminado el proyecto desplegar el portal en un hospedaje.

Como se ha explicado anteriormente, aunque en la metodología tradicional durante las primeras etapas ya se decide y planifica todo el proyecto, en nuestro caso durante el desarrollo del mismo se han ido cambiando algunas de las ideas previstas.

Estos cambios han sido debidos a problemas técnicos o a que durante el desarrollo se ha encontrado una solución mejor, o lo acordado resultaba no encajar completamente con la idea que tenía la empresa.

3.2. Planificación temporal de las tareas

Como se muestra en el diagrama de Gantt 3.2 nombrado en el apartado anterior, la suma de las horas totales que se le dedican al proyecto son 300. Otras 135 horas de dedicación son tanto para la redacción de los informes quincenales, como de este documento y para la preparación de la presentación del resultado ante un tribunal. Las 15 horas restantes para completar las 450 horas de la asignatura EI1054 son para reuniones con el tutor durante el transcurso del proyecto.

Las 300h fueron divididas en 5h diarias de 8:00 a 13:00, de lunes a viernes. En las reuniones previas se acordó que durante las fiestas de la Magdalena no se trabajaría.

3.3. Estimación de recursos del proyecto

Los recursos utilizados para el desarrollo del proyecto los podemos dividir en 3 clases: hardware, software y recursos humanos.

Software:

Tras hablar con la empresa en diferentes reuniones, se acordó que todo el software que se utilizase para el desarrollo debería ser gratuito. Con esta premisa seleccionamos las diferentes herramientas y entornos de desarrollo:

- **Django:** Se usará para la realización de un catálogo. Es un Framework de aplicaciones web de código abierto, desarrollado en Python [5], que respeta el patrón Modelo-Vista-Controlador.
- **Django-cms:** Este software será la piedra angular del portal Web. La principal diferencia con Django es que Django-cms es un Framework de aplicaciones web desarrollado con Django. Está creado para facilitar la creación de portales web. Django por su parte es más genérico, se podría decir que Django-cms es elegir la rama de la creación de portales web, que es una de las muchas cosas que se puede hacer con Django y adaptarlo para que su uso sea mucho más intuitivo y fácil para personas que nunca han usado Django.
- **Latex:** Lo usamos para la escritura de los documentos creados durante el desarrollo del proyecto, por su sofisticación y su gran adaptabilidad, ofreciendo un acabado muy profesional.
- **SublimeText:** Es un editor de texto muy moldeable y adaptable. En nuestro caso lo elegimos para poder trabajar con comodidad tanto con Django-cms como para la creación del catálogo en Django. Su principal singularidad es que nos permite variar el lenguaje con el que trabajamos en cada fichero.

A parte de este software, que sería el principal, se han utilizado otros como pueden ser diversas aplicaciones web, también gratuitas, para la realización tanto de diagramas de clases, mockups, redimensionar imágenes, etc. Por ejemplo para la realización de los diagramas de Gantt se ha utilizado el Microsoft Project Management, software para la planificación de proyectos y colaboración.

Hardware:

En cuanto al hardware utilizado, al principio de la estancia se me asignó un portátil con Windows 7 que ya poseía la empresa. Durante la estancia, y tras realizar algunas pruebas y leer diversos manuales tanto de Django como Django-cms, se decidió usar un sistema operativo basado en OS X por los siguientes motivos.

- En los manuales siempre se referencia al sistema operativo Linux.
- Al no estar acostumbrado al terminal de Windows no me resultaba natural su uso.
- Leer una instrucción pensada para un sistema operativo como es Linux y tener que traducirla, resultaba ser muy complejo y ralentizaba mucho el desarrollo del proyecto. Ya que muchas veces no era capaz de hacer la “traducción” literal de la orden.
- Daba la casualidad de que yo poseía un portátil con estas características. Tras probarlo por mi cuenta en mi tiempo libre y ver que me resultaba mas naturales e intuitivo, y tras consultar con la empresa, no tuvieron objeciones en que cambiara.

Características técnicas:

- Sistema Operativo: OS X Yosemite
- Procesador: Intel Core i5 2,7GHz
- Memoria Ram: 8GB
- Disco Duro: 256 GB
- Gráficos: Intel Iris Graphics 6100 1536 MB

Recursos humanos:

Como se ha explicado anteriormente la totalidad del proyecto ha sido realizado por una sola persona. Aunque se ha realizado el trabajo de un Ingeniero Informático y de un Diseñador web.

3.4. Presupuesto

En este apartado vamos a calcular cuál sería el coste que la empresa debería haber pagado por nuestro trabajo. Esto se puede calcular de diferentes formas.

Hemos realizado el cálculo para dos situaciones diferentes: como si fuésemos trabajadores de la empresa y como si la empresa contratara a otra empresa de informática para el desarrollo del proyecto.

Para la obtención de los datos nos hemos basado en el Boletín Oficial del Estado (BOE) del miércoles 20 de mayo del año 2015 [6] , con la excepción del coste por hora en empresa externa, que se ha elegido teniendo en cuenta el coste real de diversas empresas de informática. Este dato se ha obtenido de preguntar a diferentes empresas locales cuánto pagan por servicios de informática.

Para los cálculos de la figura 3.1 del coste para la empresa de la Seguridad social del informático, hemos visitado diferentes apartados de la web [9].

Como podemos ver en la imagen 3.1, se calcula por un lado el precio teórico por hora de un titulado superior según el BOE y se multiplica por las horas de la estancia en prácticas.

Y por otro lado observamos el cálculo teniendo como precio por hora, el obtenido de consultar con diversas empresas de la comunidad y simulando que somos una empresa externa que nos contratan para el desarrollo del proyecto, teniendo en cuenta el IVA que debería pagar la empresa.

COSTO PROYECTO TRABAJO FINAL GRADO

EFFECTUADO POR DEPARTAMENTO INFORMATICO EMPRESA

TITULADO SUPERIOR S/B.O.E.	SALARIO ANUAL	19.643,14 €
	S.S. A CARGO EMPRESA	23,60% 4.635,78 €
	TOTAL COSTO EMPRESA	24.278,92 €

HORAS LABORABLES/CONVENIO 1776, COSTO HORA : 13,67

HORAS PROYECTO 300'-- X 13,67: 4.101,00 €

CONTRATADO EMPRESA EXTERNA

COSTO HORA (incluyen costo personal, gastos empresa y beneficios)	26,00
Total horas	300
Importe facturado	7.800,00 €
IVA 21%	1.638,00 €
<u>TOTAL FACTURADO</u>	<u>9.438,00 €</u>

Figura 3.1: Resumen de los costes.

3.5. Desarrollo del proyecto

Para poder observar con detalle, el desarrollo del proyecto, y poder ver su evolución lo más útil es revisar el diagrama de Gantt de la figura 3.3 que refleja la planificación final. Sin embargo deberemos explicar qué ocurrió durante el desarrollo del proyecto para entender las modificaciones con respecto a la planificación inicial.

En la siguiente figura 3.2, podemos ver la idea previa que se tenía. Estas eran las etapas que íbamos a seguir para el desarrollo del proyecto.

Imprevistos/Incidentes. Para entender por qué no se ha cumplido al pie de la letra lo preestablecido es necesario resumir todos los problemas encontrados. Los describiremos con más detalle en capítulos posteriores, cuando hablemos del diseño, implementación y despliegado del portal web.

Una de las primeras ideas de la empresa era que se pudieran realizar algún tipo de compra en la web. Realmente no querían nada muy sofisticado pero sí que tuviera esa funcionalidad.

Esto desembocó en un problema, como se explica en el capítulo 5 mucho más a fondo. Ya que trabajamos muchas horas sobre 2 frameworks que al final no nos permitieron añadir esa funcionalidad que buscábamos: poder realizar ventas on-line.

Otro de los imprevistos fue la curva de aprendizaje tan grande que resultaron tener Django-cms y Django. Sumado al continuo cambio de estos frameworks en sus diversas versiones, que hicieron que el proyecto tuviera que ir adaptándose y evolucionando casi al mismo tiempo.

Como consecuencia de todo lo anterior podemos ver que no se han podido realizar las últimas tareas del primer diagrama de Gantt. La configuración y desarrollo del proyecto se estimó con una duración muy por debajo del que después resultó ser. En lugar de las 170 horas inicialmente previstas, hemos dedicado más de 200 a la instalación del entorno y la implementación del portal y el catálogo. Las ausencias más destacadas en las tareas finalmente desarrolladas son la del “alojamiento al hospedaje seleccionado”, en consecuencia tampoco se pudo “comprobar y adaptar la web al servidor”. Por último la “realización de un pequeño manual de administración”. Esto es debido a la limitación del tiempo que tenemos de 300h asociadas a la estancia. Esto nos obligó a cambiar bastante el plan establecido inicialmente, haciendo que el diagrama de Gantt definitivo quedaría como podemos ver en la figura 3.3.

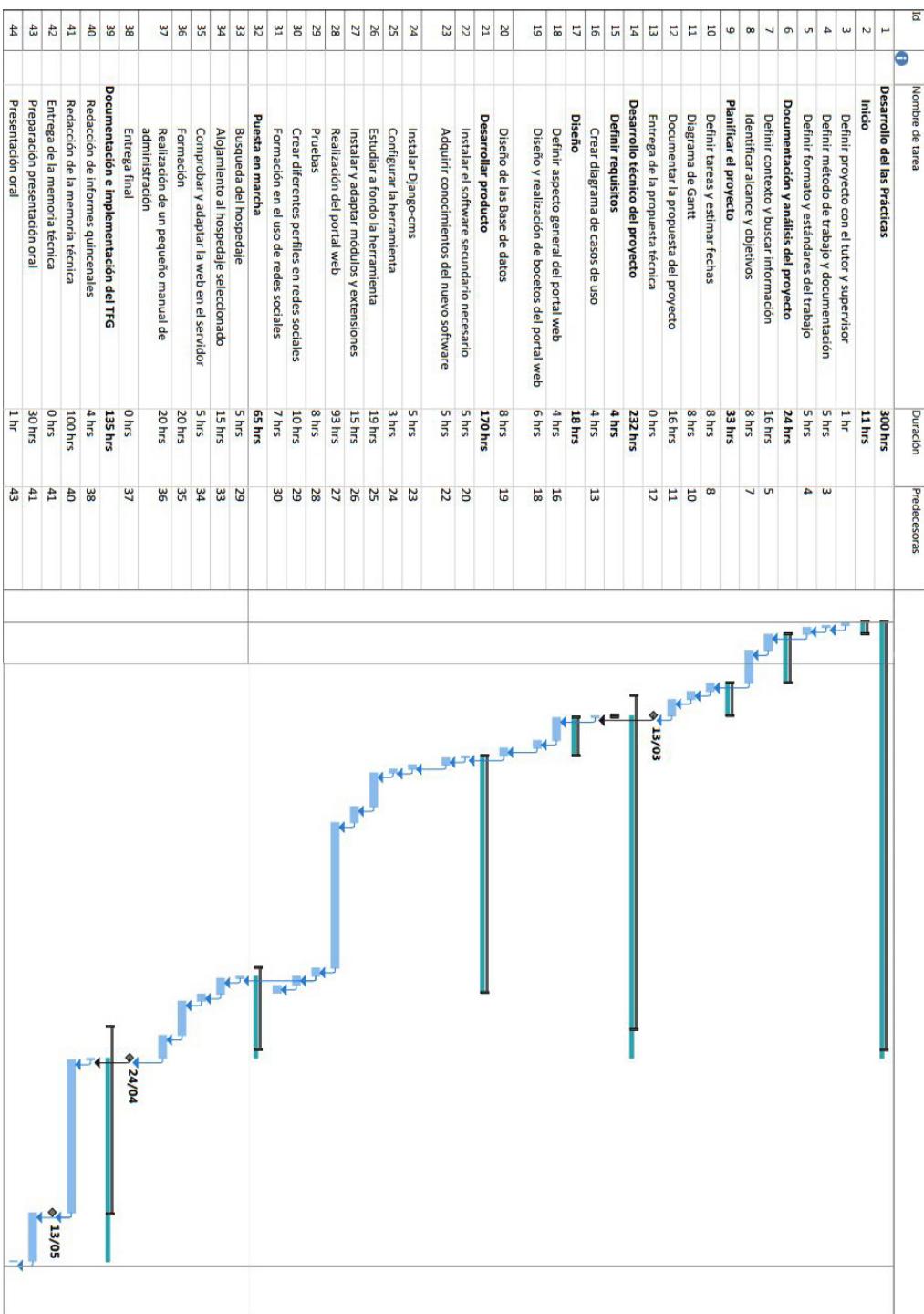


Figura 3.2: Diagrama de Gantt a priori.

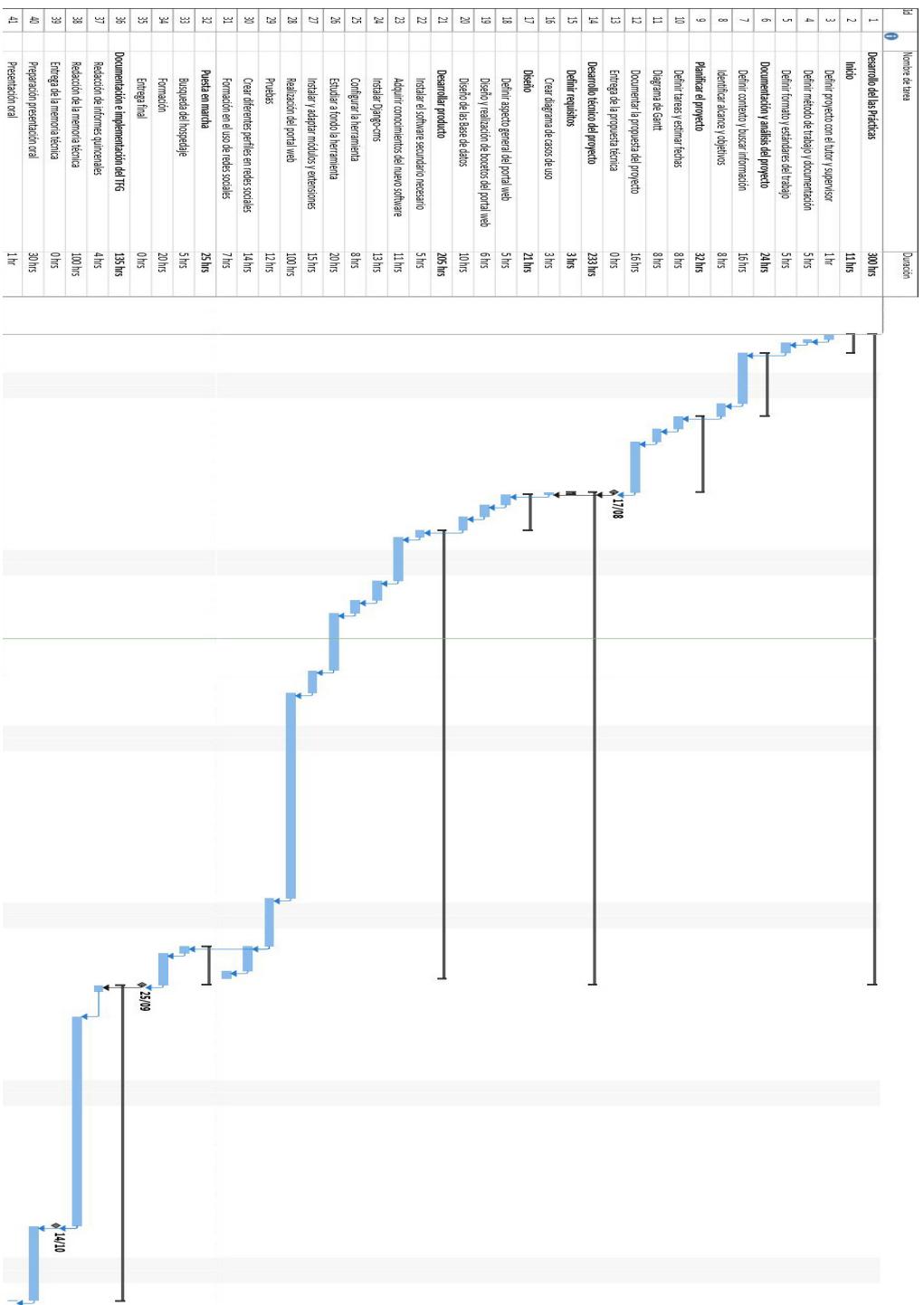


Figura 3.3: Diagrama de Gantt a posteriori.

Capítulo 4

Entorno de desarrollo

En este capítulo explicaremos por qué hemos elegido cada una de las herramientas utilizadas para desarrollar el proyecto. Además explicaremos cómo hemos instalado cada una de estas herramientas.

Como podemos ver, la figura 4.1 nos muestra las relaciones entre las herramientas y las capas que estas forman.

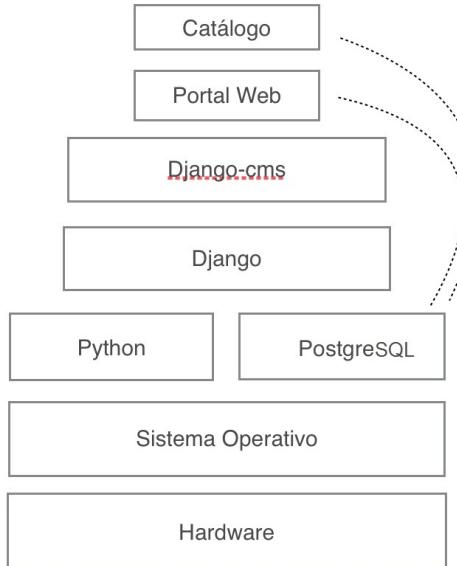


Figura 4.1: Esquema global del proyecto.

Si dividimos la imagen en escalones, podemos ver que en el inferior, tenemos el hardware sobre el que se ejecutará la aplicación.

Sobre el hardware tenemos el Sistema operativo. En nuestro caso el proyecto se ha desarrollado sobre un sistema Mac OS X, pero todo el entorno y la aplicación pueden desplegarse también sobre Linux. En lo referente al sistema operativo decir que la elección de usar Mac OS resultó ser la mejor opción tras realizar varias pruebas en Windows. Como hemos explicado anteriormente, el sistema operativo cambió en las primeras semanas del proyecto sin ningún tipo de problema.

En el tercer escalón encontramos el lenguaje de programación y el sistema de gestión de base datos elegido para desarrollar el portal: Python y PostgreSQL.

El mayor inconveniente que encontramos fue que la totalidad de los manuales de las principales herramientas que necesitamos para el desarrollo del proyecto están pensados para ser implementados en un sistema operativo basado en Unix. Esto nos supuso un gran inconveniente cuando intentamos trabajar con Windows y fue una de las principales razones por las que se decidió elegir usar un ordenador con el sistema Mac OS X..

La ventaja fundamental que nos ofrece el Mac OS X es la sencillez de instalar cualquier software. La mayoría se han instalado por línea de órdenes desde el terminal.

- **Python:** Es un lenguaje de programación que permite trabajar con gran rapidez e integrar sus sistemas con efectividad.
- **PostgreSQL:** Es un sistema de gestión de base de datos relacional, orientado a objetos. Se utilizó tanto para Django como para Django-cms. Se eligió por ser compatible con Django y Django-cms y estar muy extendido su uso. Y además es utilizado por estos frameworks para guardar en tablas todo el contenido y distribución de nuestro proyecto.

En cuanto a la base de datos, tanto django como django-cms, nos ofrece una propia por defecto, que es SQLite. Por motivos de seguridad decidimos cambiarla por una base de datos en PostgreSQL. Esto es debido a que nos ofrece una mayor seguridad, su gran popularidad, el hecho de ser multiplataforma y su facilidad de administración.

La elección de **Python** fue muy sencilla, Django se basa en Python así que necesitamos instalarlo. Además al ser un lenguaje de programación que ya dominamos nos encontramos más seguros al abordar el proyecto. Por otro lado, para los proyectos en Django se aconseja usar entornos virtuales y Python nos ofrece esto de forma muy sencilla. Con los entornos virtuales, lo que buscamos es tener nuestro proyecto aislado del resto de software del sistema operativo y al mismo tiempo tenemos todo el software del proyecto bien identificado con las versiones que queremos.

En cuanto a la instalación, para Python es tan sencillo como entrar en su web y bajar el archivo. Arrastramos a la carpeta de Aplicaciones y ya está instalado.

Para el entorno virtual, necesitamos haber instalado previamente Python.

Para la instalación por línea de órdenes usaremos la instrucción “easy_install virtualenv”. Con esto instalado podremos a crear entornos virtuales. Para ello haremos uso del comando “virtualenv env”. En este caso el nombre elegido es “env” pero podría ser cualquier otro, esto nos creará un directorio con el nombre que le hemos dado. Dentro nos encontraremos 3 carpetas que son las que vemos en la figura 4.2.

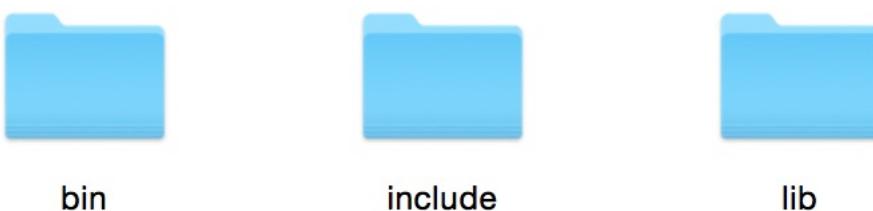


Figura 4.2: Esquema creado por virtualenv.

- **bin**: Contiene los ejecutables necesarios para interactuar con el virtualenv.
- **include**: Contiene algunos archivos necesarios para compilar algunas librerías de Python.
- **lib**: Contiene una copia del Python que hemos instalado previamente y dentro de este directorio encontramos otro nombrado “site-packages”, en el cual vemos el código fuente de los paquetes de Python instalados en el entorno virtual.

Para activar el entorno virtual necesitamos introducir esta orden “source env/bin/activate”

Para desactivar el entorno virtual en el cual nos encontramos trabajando es tan sencillo como poner el comando “deactivate” y automáticamente saldremos del mismo.

En el caso de **PostgreSQL** la instalación es bastante sencilla. Accedemos a su página web y descargamos el archivo necesario para la instalación. Luego hay que configurarlo según nos interese, para que arranque automáticamente al arrancar el ordenador, o activarlo manualmente cuando se tenga que utilizar.

En la siguiente capa de la figura 4.1 se encuentra Django. Antes de empezar con el proyecto, se estudiaron diversas opciones para la creación de un portal web, las 2 principales fueron, “Drupal” [4] y el propio “Django” [3] más “Django-cms” [2].

Antes de empezar con un proyecto de estas dimensiones, y sin nunca haber realizado nada parecido, al comparar estas 2 herramientas, ambas nos ofrecen grandes ventajas, y cada una tiene sus inconvenientes.

“Drupal” nos pareció que es un CMS con el que sólo se podía hacer uso de un sistema de gestión de contenidos. En este sentido se nos quedaba un poco corto, o por lo menos nos daba la sensación a priori que no nos permitiría definir, programar los detalles y el comportamiento del portal tanto como “Django-cms”.

Llegados a este punto tuvimos que decantar la balanza y decidimos usar “Django-cms” por ser un framework programable. Además este se apoya en Python que es un lenguaje de programación que ya conocíamos anteriormente y en el cual hemos programado bastante.

Su elección fue bastante fácil vistas sus ventajas. Para entender bien “Django-cms” primero tenemos que explicar las ventajas de “Django”.

- Código abierto.
- Se basa en Python, el cual utiliza en todos los apartados del framework: configuración, archivos, modelado de datos....
- Respeta el patrón Modelo-Vista-Controlador.
- Facilita la creación de sitios web complejos.
- Se centra en el re-uso, conectividad y extensibilidad de componentes.
- Pone mucho énfasis en el principio de “no te repitas” .
- La seguridad es un apartado muy importante.
- Incluye un mapeador de objeto-relacional.
- Incluye una API de base de datos robusta.
- Incorpora “vistas genéricas” para ahorrar tiempo en tareas repetitivas.

- Sistema extensible de plantillas basado en etiquetas.
- Herencia de plantillas.
- Sistema “middleware” encargado de desarrollar características adicionales, como pueden ser las sesiones, protección Cross Site Request Forgery, normalización de URLs.
- Soporte internacional, traducciones incluidas.
- Gran documentación.

La instalación y configuración de este framework y del siguiente, “Django” y “Django-cms” se explicarán en el siguiente apartado.

En la capa siguiente de la figura 4.1 encontramos “Django-cms”.

Aunque este framework se basa en el anteriormente explicado, y por lo tanto muchas de sus cualidades son las mismas, es importante apreciar las características que hacen de este framework una herramienta pensada para la creación de portales web:

- Software libre y código abierto.
- Comunidad muy activa.
- Muy moldeable.
- Pensado tanto para grandes como pequeños portales web.
- Robusta internacionalización. Perfecto para páginas multilenguaje.
- Guarda un historial con los cambios realizados, lo que permite volver a versiones anteriores.
- Rápido acceso a la gestión de contenidos.
- Fácil edición del *front-end*.
- Soporta gran variedad de editores de texto, con avanzadas funciones de edición de texto.
- Sistema de plugins muy flexible, ofreciendo poderosas herramientas, sin abrumar con una interfaz compleja.
- Documentación muy completa.
- Por su afán por el buen código, incluye una gran cantidad de pruebas automáticas.

Los 2 últimos escalones de la figura 4.1, constituyen el software desarrollado, por lo que se explicarán con detalle en capítulos posteriores.

Tanto Django como Django-cms desarrollan el código en forma de proyectos y aplicaciones, por lo que es importante diferenciar ambos conceptos.

- **Un proyecto:** es una instancia de un cierto conjunto de aplicaciones de Django, más las configuraciones de esas aplicaciones. Técnicamente, el único requerimiento de un proyecto es que este suministre un archivo de configuración, el cual define la información sobre la conexión a la base de datos, la lista de las aplicaciones instaladas, la ubicación de los componentes, y así sucesivamente. En un proyecto podemos tener varias aplicaciones.

- **Una aplicación:** es un conjunto portable de una funcionalidad de Django, típicamente incluye modelos y vistas, que conviven en un solo paquete de Python. Una misma aplicación, como puede ser un sistema de base de datos de registros, puede ser utilizado en diversos proyectos.

Por ejemplo, Django incluye un número de aplicaciones, tales como un sistema de comentarios y una interfaz de administración automática. Una cosa a hacer notar sobre estas aplicaciones es que son portables y reusables en múltiples proyectos. Existe un requisito respecto a la convención de la aplicación: si estás usando la capa de base de datos de Django (modelos), debes crear una aplicación. Los modelos deben vivir dentro de aplicaciones.

4.1. Django

En este apartado explicaremos cuales son los pasos que hay que seguir para la instalación y configuración de un proyecto genérico en “Django”.

Antes de empezar tenemos que tener instalado Python, como hemos explicado anteriormente.

En este punto, como también se ha dicho antes, es aconsejable crear un entorno virtual. No es obligatorio pero en el futuro, resultará de gran ayuda. Otro de los puntos importantes es decidir la base de datos que vamos a usar. Esto es necesario puesto que Django tiene su propia forma de guardar el contenido de nuestro portal web en forma de modelos sustentados en tablas de bases de datos. Por defecto el usa SQLite, y lo tiene preconfigurado, para que en caso de no elegir un sistema de gestión de bases de datos, él haga toda la configuración necesaria.

“Un modelo de Django es una descripción de los datos en la base de datos, representada como código de Python. Esta es tu capa de datos - lo equivalente a sentencias SQL CREATE TABLE - excepto que están en Python en vez de SQL, e incluye más que sólo definición de columnas de la base de datos. Django usa un modelo para ejecutar código SQL y retornar estructuras de datos convenientes en Python representando las filas de tus tablas de la base de datos. Django también usa modelos para representar conceptos de alto nivel que no necesariamente pueden ser manejados por SQL.” La explicación es bastante más extensa y compleja, y la podemos encontrar en [10].

Para instalar Django basta con descargarlo de una página web. En la web nos ofrecen distintos formatos de descarga y diferentes versiones del framework. Lo podemos conseguir mediante órdenes Linux o en un archivo comprimido. En caso de elegir la opción mediante órdenes Linux deberemos teclear “pip install Django==1.7.8”. En este punto, para verificar que tenemos la versión bien instalada, abrimos un interprete de Python y tecleamos “import django”.

En este momento ya estamos preparados para crear nuestro primer proyecto. Es preferible crear un directorio exclusivo para cada proyecto. Una vez en ese directorio tecleamos “django-admin.py startproject prueba”. Esto nos creará un directorio nuevo llamado “myapp”. Esta será la distribución que crea dicho comando, figura 4.3.

Esto nos generará los siguientes archivos:

- El fichero `manage.py`: te permite tener una utilidad de línea de órdenes para así interactuar con el proyecto.
- El fichero `myproject/__init__.py`: archivo vacío que le indica a Python que este paquete debería ser considerado un paquete del propio Python.
- El fichero `myproject/settings.py`: configuración para este proyecto Django.
- El fichero `myproject/urls.py`: contiene las URLs para este proyecto.

- El fichero `myproject/wsgi.py`: fichero para la configuración del desplegado del proyecto en un servidor.

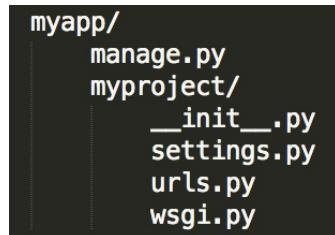


Figura 4.3: Esquema de un proyecto en Django genérico.

Ahora ya podríamos arrancar el proyecto. Para ello nos situamos dentro de la carpeta que nos ha creado, “myapp” y ejecutamos la siguiente orden “python manage.py runserver”. En este punto arrancará el proyecto y nos indicará que todo ha ido correctamente.

Todos estos serían los pasos para conseguir crear un proyecto en “Django” completamente vacío y sin ninguna funcionalidad. En capítulos posteriores describiremos cómo hemos creado los proyectos asociado a nuestro portal.

4.2. Django-cms

En el caso de Django-cms al tratarse de un framework basado en “Django” observamos que hay múltiples semejanzas en la creación de un proyecto.

En este caso en concreto, sí que nos piden desde el propio manual de Django-cms que creamos y activemos un entorno virtual. Recordamos que en “Django” no es necesario, sino que es aconsejable.

Como hemos explicado anteriormente creamos y activamos el nuevo entorno virtual. Una vez estamos en él, debemos instalarlo. Para ello usaremos la orden “pip install djangocms-installer”

Otra de las diferencias es que Django-cms nos ofrece una instalación asistida, que es la que hemos usado durante el desarrollo del proyecto. Para la instalación del framework necesitamos crear una carpeta vacía e introducirnos dentro. Ahora ya podemos crear nuestro proyecto de portal desorrollado con este framework, para ello simplemente pondremos “djangocms -p . mysite”. En este momento nos aparecerá un cuestionario con una serie de preguntas que definirán la configuración del framework. El propio Django-cms nos da unas respuestas genéricas para la realización de un proyecto estándar, podemos verlo en la figura 4.4.

Al crearlo, automáticamente nos proporciona la estructura que podemos ver en la figura 4.5, formado por una carpeta para archivos multimedia, otra para archivos estáticos, una carpeta con el nombre del proyecto que contiene las especificaciones concretas del mismo y por último el archivo encargado del arranque del proyecto.

Si desplegamos la carpeta con el nombre del proyecto encontramos unas carpetas y archivos que nos recuerdan a los vistos para un proyecto “Django”, podemos verlo en la figura 4.6.

- la carpeta `__pycache__`: que se usa para agilizar las compilaciones en python.
- El fichero `__init__.py`: se utiliza en Python para formar una jerarquía interna de la estructura del proyecto y que aunque existan nombres repetidos siempre sepa que módulos hay en cada sitio.

- Database configuration (in URL format): sqlite://localhost/project.db
- django CMS version: stable
- Django version: stable
- Activate Django l18N / L10N setting: yes
- Install and configure reversion support: yes
- Languages to enable. Option can be provided multiple times, or as a comma separated list: en, de
- Optional default time zone: America/Chicago:
- Activate Django timezone support: yes
- Activate CMS permission management: yes
- Use Twitter Bootstrap Theme: yes
- Use custom template set: no
- Load a starting page with examples after installation: yes

Figura 4.4: Respuestas aconsejadas para la creación de un portal genérico.

```
pruebaCMS/
  env/
  tutorial-project/
    media/
    mysite/
    static/
    manage.py
```

Figura 4.5: Esquema de un proyecto en Django-cms genérico.

```
pruebaCMS/
  env/
  tutorial-project/
    media/
    mysite/
      __pycache__/
      static/
      templates/
      __init__.py
      settings.py
      urls.py
      wsgi.py
    static/
    manage.py
```

Figura 4.6: Esquema de un proyecto en Django-cms genérico.

- El fichero `settings.py`: es el fichero principal de configuración, donde se nos muestra toda la información del proyecto, desde los módulos instalados, apps, base de datos, formatos, etc.

- El fichero `urls.py`: contiene todas las urls del proyecto y las relaciones entre ellas.
- El fichero `wsgi.py`: Es un fichero utilizado para configurar el despliegue del proyecto web sobre un servidor web como Apache.

En este momento solo quedaría arrancar el proyecto creado con la orden “`python manage.py runserver`”. Al arrancar podemos ver la web genérica en la figura 4.7.

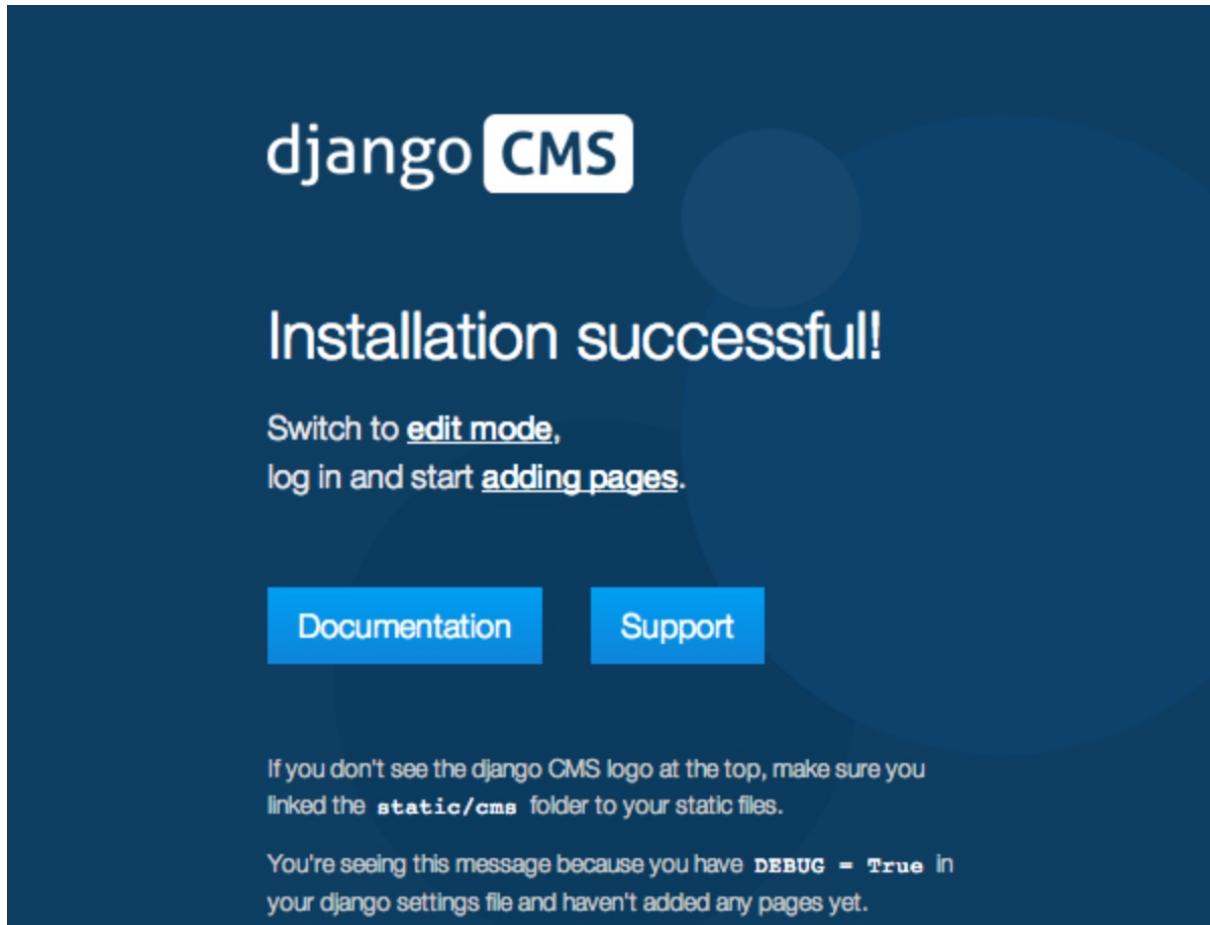


Figura 4.7: Web genérica creada con Django-cms.

Capítulo 5

Análisis y diseño del sistema

En este capítulo abordamos lo que es una visión global del sistema, su diseño, un análisis del proyecto, además de concretar aspectos relacionados con el estilo del portal y su distribución. Todo lo referente a la implementación del proyecto se explicará en el próximo capítulo.

5.1. Visión global del sistema

La manera más fácil de expresar una visión global del sistema es mediante un diagrama de casos de uso.

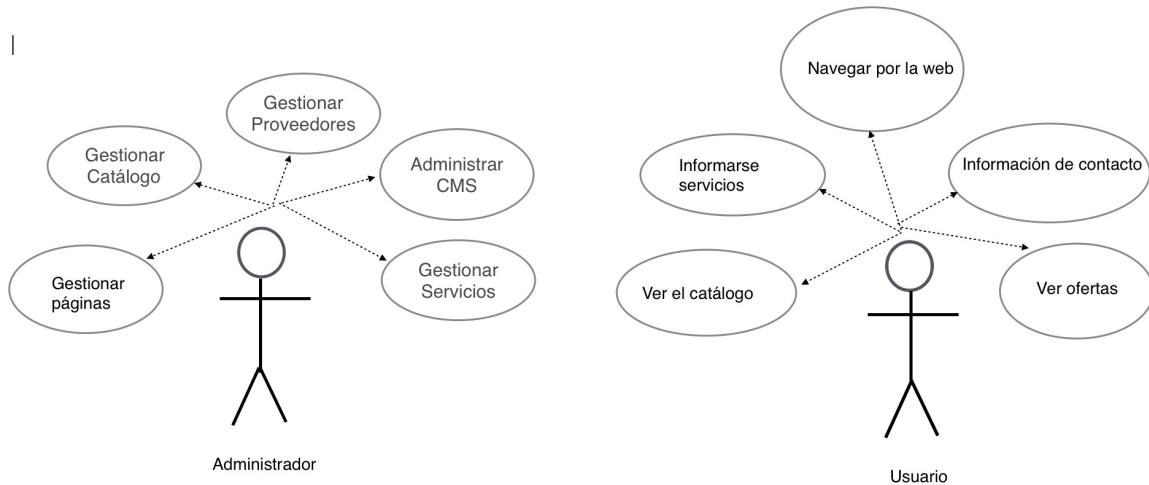


Figura 5.1: Diagrama de casos de uso.

Como podemos observar en la 5.1 se muestran los dos roles que tenemos, usuario y administrador del portal.

Dentro de cada rol, podemos observar las diferentes acciones que pueden realizar cada uno.

Administrador

1. Gestionar páginas: ser capaz mover, crear, modificar y borrar las páginas de nuestro portal, según nuestros gustos o deseos.
2. Gestionar catálogo: poder añadir, borrar o modificar, alguno de los productos del catálogo.
3. Gestionar proveedores: poder añadir, borrar o modificar, alguno de los proveedores del catálogo, haciendo referencia a las páginas web corporativas.
4. Administrar CMS: ser capaz de modificar el diseño de nuestro portal web.
5. Gestionar servicios: administrar la parte privada de nuestro portal web, añadiendo o quitando los servicios en cada página web como pueden ser la disposición de los placeholder, los permisos de cada página, etc.

Usuario

1. Ver el catálogo: poder visitar el apartado de catálogo, donde ver los productos
2. Informarse de servicios: observar los distintos servicios que ofrece la empresa.
3. Navegar por la web: poder viajar entre páginas web dentro del portal con facilidad.
4. Información de contacto: obtener toda la información necesaria para ponerse en contacto con la empresa.
5. Ver ofertas: poder ver qué productos se encuentran en oferta en un momento determinado.

5.2. Diseño del sistema

Para un buen diseño del sistema es importante explicar que aunque Django-CMS se base en Django, su gran diferencia principal es la distribución que hace de los archivos al crear un proyecto y su implicación en todo el funcionamiento.

Como se ha explicado en el capítulo anterior podemos ver cómo ha quedado nuestro proyecto una vez acabado en la figura 5.2.

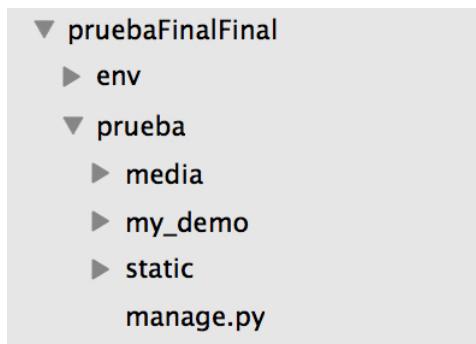


Figura 5.2: Esquema estructura proyecto Django-cms.

Si abrimos el directorio “my_demo” observamos en la figura 5.3, que hay una carpeta nueva llamada “apps”. Esto se debe a que en nuestro proyecto hemos insertado una aplicación.

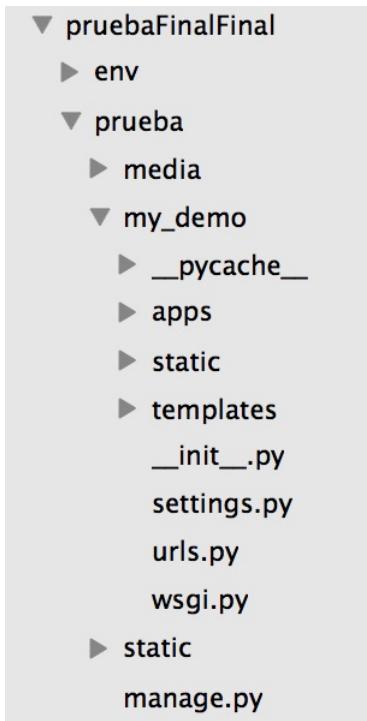


Figura 5.3: Esquema estructura proyecto my_demo django.

Para tener una idea de cómo sería la estructura de nuestra aplicación desarrollada en Django podemos verla en la figura 5.4. Que está formado principalmente por 2 directorios, la primera es “catalogo” donde está todo lo referente a la aplicación y por otro lado tenemos “myproject”, que sería el directorio que nos crea cualquier proyecto realizado con Django. Más a delante explicaremos cómo hemos llegado a esta estructura de ficheros y directorios.

5.3. Diseño de la interfaz

En este apartado la empresa no tenía ninguna idea previa, por lo que tuvimos que realizar un trabajo de investigación sobre diseños modernos en páginas web. La investigación se centró en buscar diferentes webs actuales y estudiar su distribución, aspecto y contenido. Sobre todo nos basamos en aquellas que estaban desarrolladas con “django-cms”. Por otro lado intentamos averiguar cuales eran los puntos fuertes que favorecían el éxito de un portal web. Dentro de los puntos clave que favorecen el éxito de una página web tuvimos que centrarnos en aquellos que sí que podíamos llegar a realizar, debido a la limitación del tiempo disponible y a las restricciones del propio framework “django-cms”.

Uno de los aspectos más importantes que queríamos destacar era el diseño Flat [12], que consiste en hacer que el aspecto de la web sea simple y minimalista. Con esto conseguimos que el cliente encuentre lo que está buscando con mayor facilidad. Otra idea fundamental es la del uso de imágenes de gran tamaño, sin llegar a abusar de estas. Por último la idea de que cada página se centre en una funcionalidad distinta del portal, hace que el uso de la web para el visitante sea más intuitiva.

Con estos tres ideas empezamos a crear diversos Mockups, como el de la figura 5.5, que utilizaríamos para obtener feedback de la empresa y así obtener un primer prototipo.

La estructura de la página principal del portal está formada por 4 bloques:

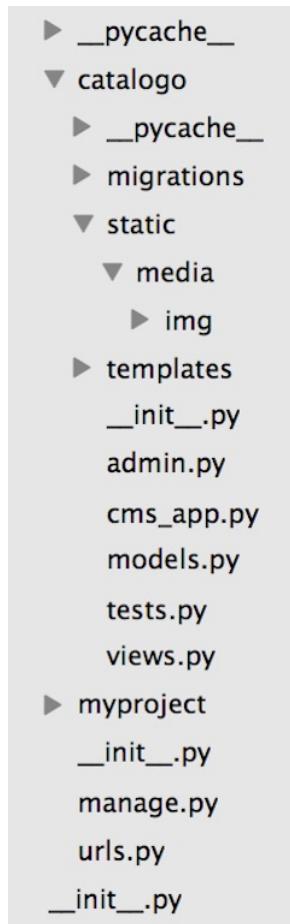


Figura 5.4: Estructura directorio apps.

- Cabecera: Formada por el nombre y el logo de la empresa.
- Barra de navegación: Tendrá las distintas páginas que forman parte del portal web.
- Cuerpo: En este caso estará formado por una imagen grande, que ocupe la totalidad de su espacio.
- Pie de página: Se mostrará un texto con información de la empresa.

Se acabó acordando con el cliente que la web dispondría de siete secciones/páginas:

- Home: corresponde a la página principal, contiene el eslogan de la empresa y una gran foto de la empresa.
- Empresa: nos muestra información referente a la empresa, como son horarios y servicios.
- Productos: pequeño subconjunto de productos: catálogo.
- Oferta del mes: destacamos las cualidades del producto que está en oferta.
- Localización: apartado donde encontraremos información de la localización exacta de la empresa.

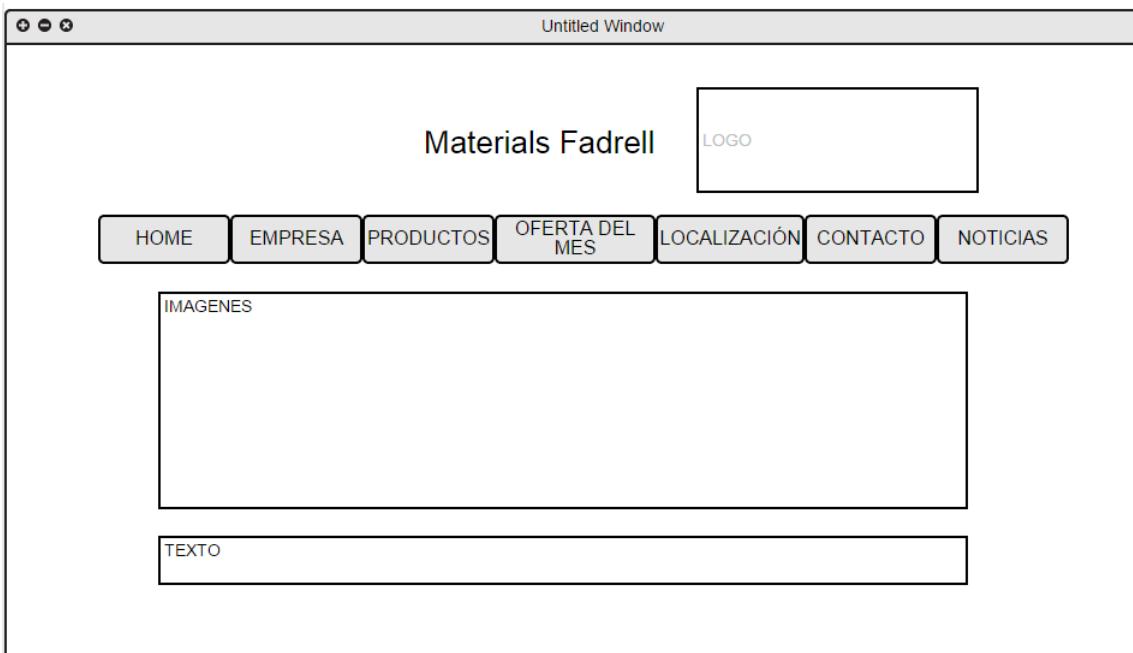


Figura 5.5: Primer prototipo de la página principal del portal.

- Contacto: mostrará los distintos métodos para poder ponerse en contacto con la empresa.
- Noticias: está enfocado a dar a conocer los cursos realizados en la empresa y las conferencias que se realizan para presentaciones de productos.

5.4. Estilo del portal web

También se definió un estilo corporativo para la web, dando un aspecto homogéneo a todas las páginas como vemos en la figura 5.6. Se aprovechó la idea de que ellos ya poseían un logo para la elección de los colores.



Figura 5.6: Logo de la empresa.

La idea es utilizar el color blanco como fondo, ya que se trata de un color que no cansa a la vista y permite destacar el resto de elementos incluidos. Además el color azul claro del logo, se utiliza para enfatizar aquello que queremos que destaque, como por ejemplo el eslogan en la página principal. Se ha repetido este color, para así homogeneizar más aun la web. Con las letras en blanco para el eslogan, se buscaba que se pudiese leer bien, y en este caso en concreto, el negro no acababa de encajar con lo que estábamos buscando. Para poder destacar la barra

con las secciones, hemos utilizado el color negro de fondo y las letras en blanco, es un juego con el que conseguimos diferenciar muy bien la parte de la cabecera con el resto de la web.

En la cabecera se ha decidido colocar tanto el logo de la empresa como el del grupo a la que esta pertenece, ver figura 5.7.



Figura 5.7: Estilo del portal web.

Hemos decidido que el pie de página también sea común a todas las páginas mostrando la información de contacto de la empresa. Le hemos dado un color de fondo un poco más oscuro, para realzar que no forma parte de cada página, sino que es información global del portal como vemos en la figura 5.8.



Figura 5.8: Estilo del pie de página.

Al hacer fijos tanto la cabecera como el pie de página, logramos mantener una continuidad en el aspecto para todas sus páginas y mantenemos la idea de que seguimos conectados al mismo portal web de la empresa.

5.5. Diseño de la base de datos

En esta sección nos centraremos en explicar la base de datos que hemos creado para el desarrollo del proyecto en Django, en concreto para poder realizar el catálogo.

La idea principal de este proyecto era poder mostrar un subconjunto de los materiales, así que no teníamos ningún tipo de restricción al respecto.

Para la organización y diseño de la base de datos se realizó una reunión con la empresa, para decidir que datos necesitaban guardar, el principal es tener una lista de productos de los cuales necesitaremos guardar el nombre, el tipo de producto, una pequeña descripción del mismo y por último una imagen. Además al tener productos tan parecidos como pueden ser ladrillos,

que hay de diferentes tamaños y colores, se pensó en tener los productos por categorías, de forma que cada producto forme parte de una sola categoría. Por lo cual deberemos guardar el nombre de la misma, una pequeña descripción y una imagen representativa. Otro de los puntos importantes es saber que cada producto pertenece a un proveedor, pero diferentes proveedores pueden suministrarnos el mismo producto. Por eso necesitaremos guardar el nombre de cada proveedor, su correo electrónico y su dirección.

Durante la fase de Diseño conceptual, la primera idea que tuvimos era que necesitábamos 3 entidades distintas: Productos, Categorías y Proveedores.

Como podemos ver en la figura 5.9 hicimos un diagrama entidad-relación, para definir los atributos que necesitaba cada entidad y también para ver el tipo de relaciones que necesitábamos y su cardinalidad. Este diagrama es la versión final del mismo. Durante el proyecto ha tenido algunas modificaciones, adaptándose a lo que se nos pedía o simplemente para mejorarlo.

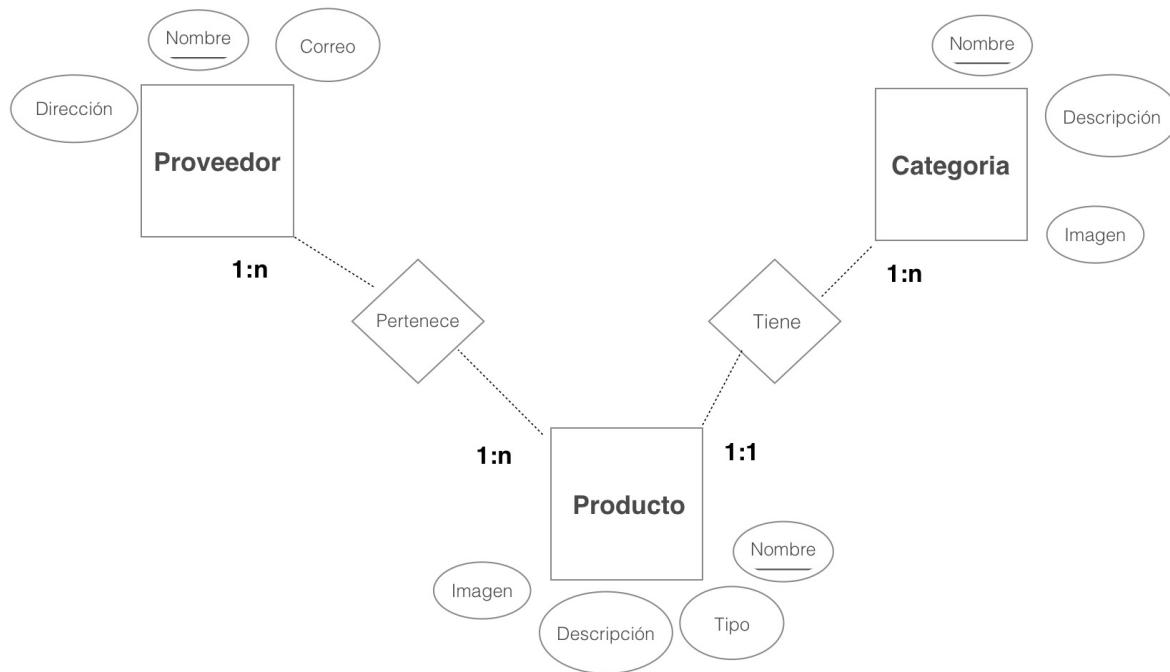


Figura 5.9: Diagrama conceptual de la base de datos realizada para el catálogo.

Una vez teníamos bien definidas las entidades que necesitamos, se realizó un diseño lógico. Para observar bien las tablas que necesitamos crear y sus atributos, ver figura 5.10.

Se puede apreciar que necesitaremos una tabla por entidad y una tabla extra para las relaciones entre Proveedores y Productos, ya que se trata de una relación de muchos a muchos. En el siguiente capítulo se explicará cómo se ha creado la base de datos en Django.

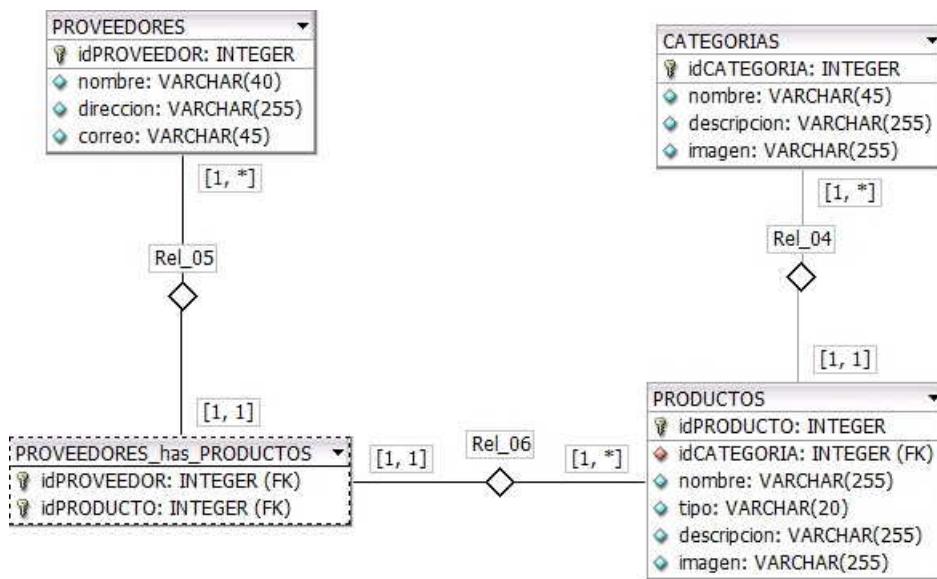


Figura 5.10: Tablas de la base de datos tras el diseño lógico.

Capítulo 6

Implementación, pruebas y documentación

6.1. Promoción creación de perfiles

Este, aunque pueda parecer un apartado muy simple, es de los más importantes, debido a que hoy en día todo el mundo tenemos una marca en internet. La idea es que estos perfiles web pueden llegar a ser completamente independiente de la imagen real. Por esto el hecho de ser tú quien controle estos perfiles te da más opciones de que esa marca virtual sea lo más positiva posible.

La comunicación es una característica innata al ser humano. En estos últimos años este mundo de la comunicación ha crecido sobre todo gracias a las plataformas sociales, como son Facebook y Twitter. Por otro lado las empresas necesitan darse a conocer y para eso conviene aparecer en un lugar destacado en el mayor buscador de internet, Google.

Debido a ello decidimos crear una cuenta en una app de Google, llamada Google My Business [13]. Esta app nos ofrece aparecer en las búsquedas de Google de forma completamente gratuita. Entre otras cosas nos ofrece 2 de las ideas que mas importaban a la empresa: aparecer en Google Maps y que al buscar algo relacionado con “materiales de construcción Castellón” o búsquedas parecidas se mostrase un anuncio en el margen derecho con información de la empresa, como podemos ver en las figuras 6.2 y 6.3.

Además Google My Business nos ofrece mostrar el anuncio en todas las plataformas, ya sean teléfonos móviles, tabletas y ordenadores. Tener este abanico multiplataforma es una gran ventaja competitiva. La figura 6.1 nos muestra las ventajas de estar registrado en Google My Business, como son aparecer en la web, aparecer en el Google Maps, destacar en Google+, mostrar la información correcta y aparecer en todos los dispositivos.



Figura 6.1: Ventajas de estar en Google My Business.

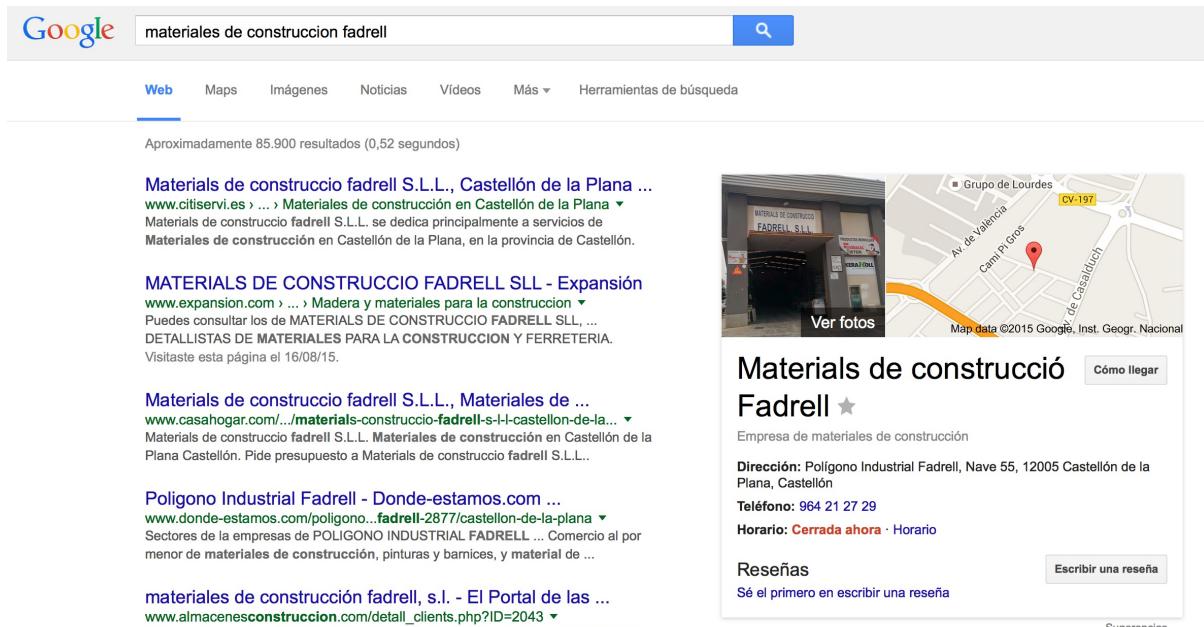


Figura 6.2: Publicidad en Google lograda con Google My Business.

Para obtener estas ventajas es suficiente con tener una cuenta de google, y contestar un pequeño cuestionario. Como curiosidad cabe comentar que el método que usa Google para cerciorarse de que el lugar donde has indicado que está localizada la empresa es correcto, es enviando una carta a la empresa por correo ordinario. Con esto a las pocas semanas la empresa ya tenía más de 2000 visitas, definidas como pulsaciones en el anuncio que hemos creado, a través del enlace de publicidad de Google. Además en la carta de Google, ofrecían a la empresa descuentos para obtener más publicidad, aunque al final no se utilizó, ya que en ese momento la idea era darse a conocer a coste cero.

Google My Business nos ofrece además una gran oportunidad de comunicarnos con los clientes. Entre sus muchas opciones está la de iniciar conversaciones con un cliente en particular, o la de responder a observaciones. El hecho de incluir una imagen y que esta aparezca en los perfiles de todas las personas que nos siguen es otra manera de darle vida a la empresa.

También tenemos la opción de realizar un video-chat, si fuera necesario, ya sea para recibir un pedido de un cliente, o por si el cliente prefiere tener un trato más “Personal” y empezar una conversación con este sistema. Esto nos ofrece multitud de ventajas como poder ver en directo donde tiene que llevarse el pedido y poder seleccionar cual es el mejor transporte dependiendo del lugar, y de sus cualidades, como pueden ser el terreno, dimensiones. etc.

Además, al estar todo en una misma cuenta de Google, podemos modificar cualquier dato de la empresa desde cualquier dispositivo con conexión a internet. Por último podemos ser informados de cada reseña que nos dejen los clientes.

Por otro lado se crearon 2 perfiles en las redes sociales Facebook y Twitter. Estas apps fueron instaladas en el móvil de la empresa para hacer un uso más cotidiano y familiar. Cuando esto fue instalado, necesitamos realizar varias reuniones con los miembros de la empresa para explicarles el funcionamiento y la importancia que esto conlleva.

Facebook ha cambiado sus registros para que estén orientados a personas y no se puedan crear perfiles para una empresa o una página web. Para poder crear una página de Facebook

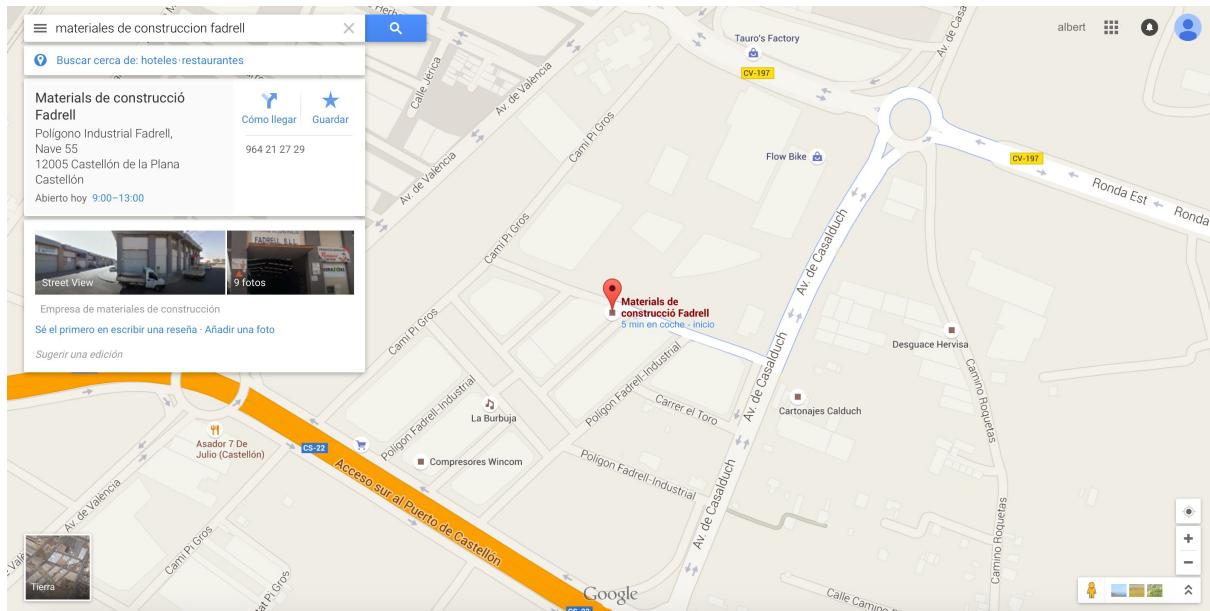


Figura 6.3: Ubicación de la empresa en Google Maps, mostrándonos la situación exacta con un icono.

a una empresa es necesario hacerlo desde una cuenta personal, siendo esta la que hará de administrador del nuevo perfil de la empresa.

Se decidió que la cuenta se creara a nombre de uno de los miembros de la empresa. Fue necesario guiarle en el proceso de creación y formarle en su uso y administración. La idea es que pueda seguir actualizándola y administrándola en el futuro.

Twitter por su parte sí que permitía usar el correo de la empresa para la creación de un perfil, y fue bastante simple y rápido.

Las funcionalidades de ambos perfiles web para la empresa estaban bien definidos.

- Facebook: pensado para publicar fotos de la empresa, de los clientes, de los eventos que se realizan, reuniones etc. El objetivo principal es el de promocionar la empresa con imágenes para fortalecer el contacto con los clientes.
- Twitter: orientado para promocionar artículos que se encuentren en oferta, recordar las fechas de reuniones, etc. La idea era usarlo para promoción no tanto de la empresa, sino de los productos y de los eventos en los que organiza la empresa.

6.2. Integración de la venta online en el portal

Al empezar con el proyecto y tener que usar un software nuevo del cual no teníamos ningún tipo de idea previa, tuvimos que aprender desde cero cómo funcionaba y se programaba la herramienta. No sabíamos por ejemplo cómo manejaba Django o Django-cms internamente el acceso a las bases de datos, y tampoco cómo se crearían las tablas necesarias para el catálogo que teníamos en mente. Aun sabiendo lo que queríamos hacer de antemano, necesitábamos estudiar primero a fondo este framework.

Por todo esto antes de empezar con la implementación decidimos estudiar la documentación online. Como podemos ver en la figura 6.5 este era el aspecto de la web de django-cms que hemos



Figura 6.4: Perfil de la empresa en Facebook.

usado para la construcción del portal web. Como la idea inicial era la de poder realizar compras, decidimos empezar por ver qué opciones teníamos a nuestro alcance que fueran compatibles con django-cms.

Como se puede ver en la figura 6.6 el propio django-cms nos ofrece este servicio, mediante **django-SHOP**.

Describiremos a continuación los distintos intentos realizados para integrar la venta online en nuestro portal. Antes de empezar a explicar las diversas herramientas que al final no funcionaron, cabe decir que la idea era conseguir previamente al desarrollo completo de un proyecto que nos permitiese hacer ventas online, pensábamos en hacer un pequeño proyecto con esta funcionalidad, para cerciorarnos de como funcionaba y luego ya desarrollarlo completamente. Se pretendía conseguir un pequeño boceto o una primera versión muy simple y esquemática.

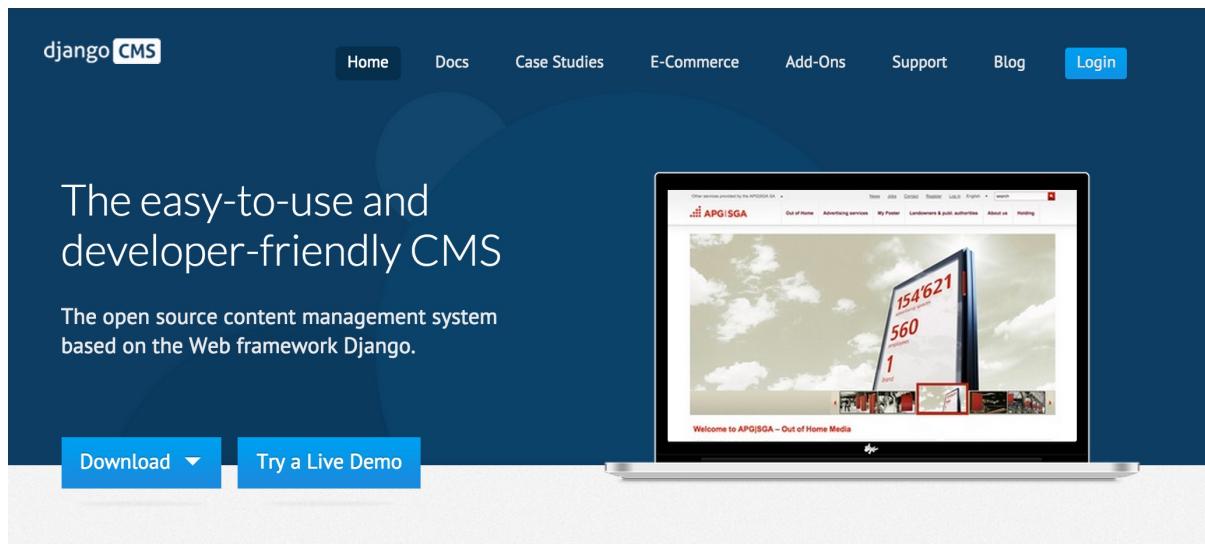


Figura 6.5: Web antigua de Django-cms.

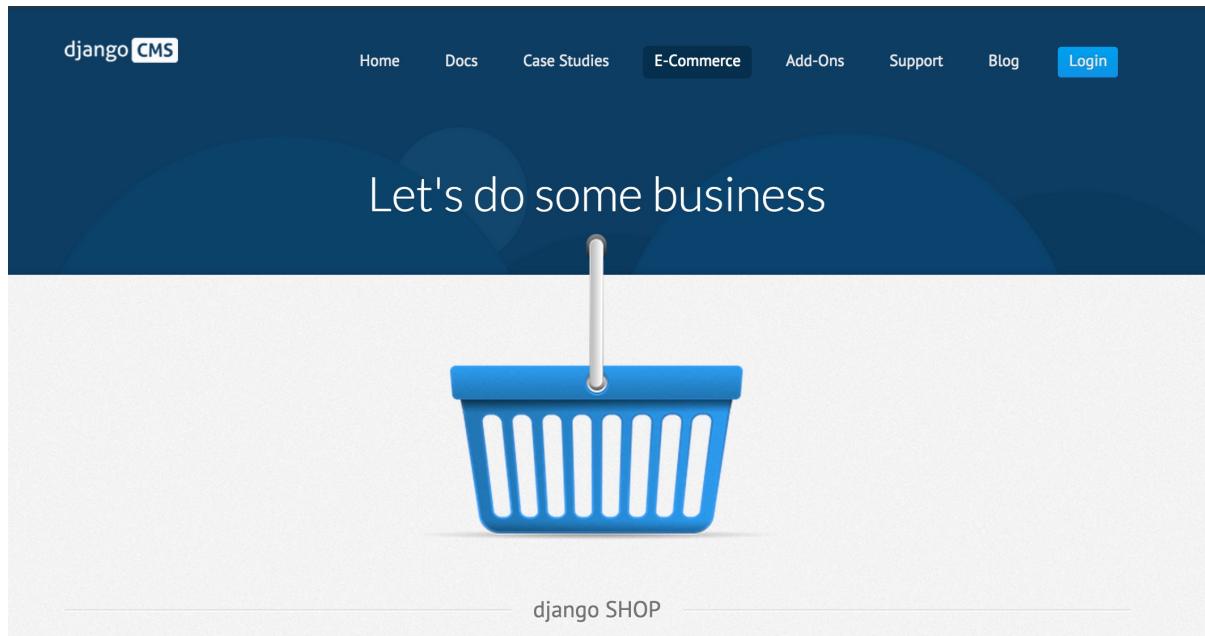


Figura 6.6: Referencia a django SHOP.

Resumimos los principales problemas que surgieron al intentar instalar y utilizar Django-SHOP:

- El manual estaba pensado para ser desarrollado por completo en un sistema operativo Linux y en aquel momento aun estábamos usando Windows.
- Algunas de las órdenes indicados en el manual de Django-SHOP, resultaban no funcionar o no hacer exactamente lo que decían.

- Tuvimos problemas de compatibilidad entre versiones de Python: Djagno-SHOP todavía usaba Python2.7 y nosotros estábamos utilizando Python3.4.
- Descubrimos que el proyecto de desarrollo de Django-SHOP estaba estancado desde hacía 4 años.

Al mismo tiempo que se realizaban las pruebas en Django-SHOP, fuimos trabajando con Django-cms, del cual teníamos varios proyectos que utilizábamos para realizar diversas pruebas. Pero Django-cms evolucionaba muy rápido y nos encontramos con diferentes problemas en nuestros proyectos desarrollados con Django-cms:

- Al evolucionar el framework, todo lo realizado hasta ese momento en Django-cms quedó obsoleto.
- La versión estable de Django para este framework pasó de ser la 1.6.8 a ser la 1.7.7, y a los pocos días evolucionó a la versión Django=1.7.8.
- El gran problema de esta continua evolución fue que la forma de trabajar y configurar el framework, variaba demasiado, lo cual complicaba el aprendizaje del mismo.

De un día para otro cambiaron el aspecto de la web y la opción de e-commerce desapareció como podemos observar al comparar las figuras 6.5 y 6.7.

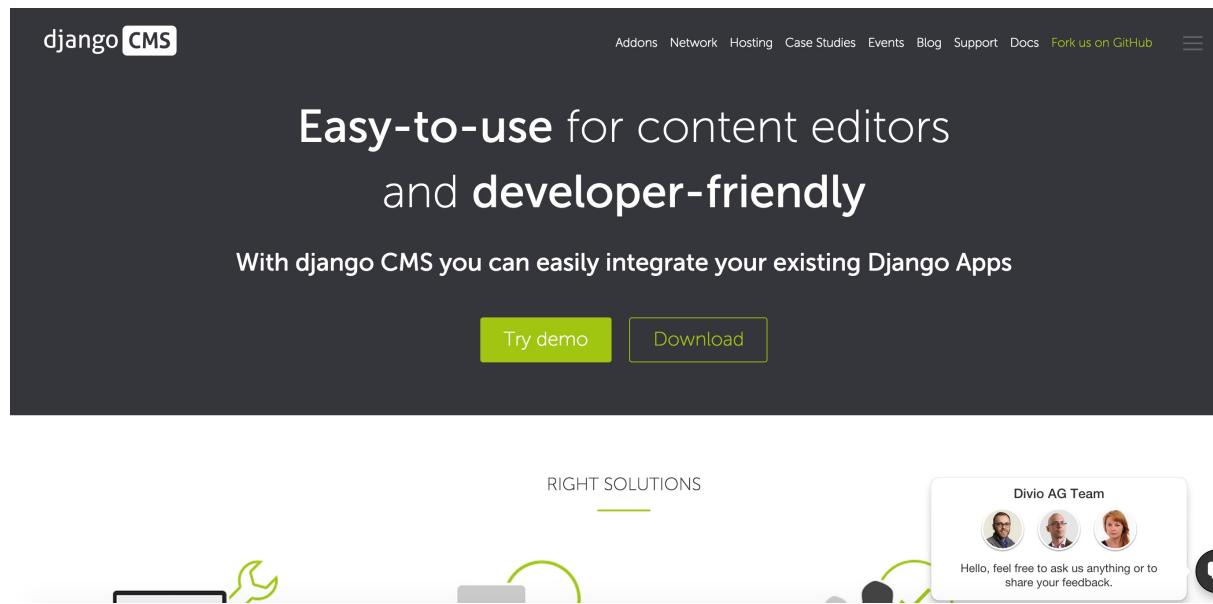


Figura 6.7: Nueva web de Django-cms.

Otro problema fue que al evolucionar el framework, evolucionaba el manual. La forma de hacer las cosas variaba lo cual resultaba molesto y complejo, además de que dificultaba y retrasaba el aprendizaje.

Tras este descubrimiento tuvimos que realizar una reunión con el tutor para sopesar qué otras opciones teníamos.

Se acordó con el tutor intentar trabajar con el **Django-OSCAR** que se utilizaba en algunos de los ejemplos del manual de Django-cms como aplicación de venta a integrar en un portal.

Django-OSCAR es un framework comercial, diseñado para la construcción de webs de ventas. Está estructurado para que todas las partes de la funcionalidad básica puedan ser personalizadas para adaptarse a las necesidades del proyecto.

Además ofrecía en su web distintos ejemplos, ver figura 6.8. Entre los ejemplos nos encontrábamos 3 que instalamos y probamos de forma independiente en entornos virtuales adaptados a sus necesidades. A la hora de intentar integrar uno de estos proyectos de ejemplo desarrollado en Django-OSCAR en uno realizado con Django-cms aparecieron los siguientes problemas:

- Problemas de compatibilidades entre distintos componentes del entorno de desarrollo.
- En concreto uno de los problema de versiones que resultaban ser más graves era que las versiones de Django que se utilizaban tanto en el proyecto realizado con Django-OSCAR, como en el proyecto desarrollado con Django-cms, resultaban ser versiones incompatibles entre sí. Con cambios demasiado grandes como para intentar adaptar uno de los 2 proyectos.

The screenshot shows a web browser displaying the 'Oscar Demo' website. At the top, there is a header with language selection ('español'), a search bar ('Ir'), and a login link ('Inicio de sesión o registro'). Below the header, the main navigation bar includes links for 'Explorar la tienda', 'Book', 'Ofertas', 'Buscar' (search), and 'Ver carrito' (view cart). The total cart value is shown as 'Total carrito: 0,00 £'. The main content area features a category title 'Dairy & Eggs'. On the left, there is a sidebar with a search bar labeled 'BUSCAR CATEGORÍAS' and a list of categories: 'Dairy & Eggs' (selected) and 'Book'. Two products are listed: 'Organic Eggs' (image not available, price 100,00 £, status 'No disponible') and 'Example Book' (image available, price 6,00 £, status 'No disponible').

Figura 6.8: Ejemplo portal web realizado con django-OSCAR.

Por otro lado, pensando en que el error de compatibilidades de las versiones de Django-OSCAR con Django-cms pudiese solucionarse en un futuro, intenté modificar los ejemplos que nos proporcionaba Django-OSCAR. Sin embargo, resultaba ser un código demasiado complejo de entender, ya que la curva de aprendizaje de Django ya es bastante elevado y la de Django-OSCAR añade aún mayor complejidad.

Después de estos acontecimientos, tuvimos que reunirnos y ver qué opciones nos quedaban. Como habíamos consumido mucho más tiempo del planificado para esta tarea y tras hablar con la empresa, acordamos que con realizar un pequeño catálogo sería suficiente. La opción de vender no resultaba fundamental para ellos, ya que su idea principal era la de darse a conocer. Elegimos que el framework que seguro que no iba a dar problemas de compatibilidad era Django, ya que habíamos probado anteriormente a añadir un proyecto de Django en uno de Django-cms y no había ningún tipo de error.

Tras esta introducción, donde hemos expuesto el camino seguido, y el porqué de nuestras elecciones, abordaré en los siguientes apartados los proyectos que sí que tuvieron frutos. Aunque pueda parecer que tanto Django-SHOP como Django-OSCAR resultaron un fracaso, aprendimos muchas cosas, sobre todo el uso del modelo-vista-controlador específicamente usado sobre un proyecto que se basa en Django, el entender los errores que nos devuelven los frameworks, etc.

6.3. Desarrollo del portal

La mejor manera de entender este proyecto es volver al esquema que representa las distintas capas de software usado y desarrollado que se muestra en la figura 4.1 . En el podemos ver todas las capas que lo forman y podemos observar la profundidad que tiene. En concreto en este capítulo nos centraremos en los últimos dos escalones.

Para entender bien el funcionamiento del framework Django-cms es necesario explicar cómo se implementan las páginas en este framework. Para ello se apoya en su interfaz gráfica y en el uso de contenedores. En estos contenedores nosotros tenemos la capacidad de modificar tanto su contenido como la distribución del mismo. La interfaz gráfica a su vez nos permite añadir o quitar estos contenedores de las páginas.

Antes de empezar con el desarrollo concreto de nuestro proyecto hemos de recordar todos los pasos que hemos explicado en el capítulo 4 hasta llegar al punto en que nos aparece un cuestionario como vimos en la figura 4.4.

Resumiendo, los pasos serían, instalación de Python, instalación del gestor de sistema de base de datos, creación y activación de un entorno virtual, instalación de Django y descargar del instalador de Django-cms.

Será justo en el momento en el que creamos el proyecto para nuestro portal cuando se instalará la versión de Django-cms que queramos. Para ello necesitaremos una carpeta vacía. Una vez le indicamos el nombre nos aparece el cuestionario antes citado.

En nuestro caso las respuestas no fueran exactamente las mostradas en la figura 4.4. Para empezar la base de datos por defecto es SQLite. Sin embargo por cuestiones de seguridad decidimos cambiar para que se usara PostgreSQL. En la elección de Django-cms al igual que Django ha ido evolucionando, pero a un ritmo mucho menor, así que la versión estable es la más recomendada. En la elección de Django, indicamos que queremos la que hemos instalado anteriormente, en nuestro caso 1.7.8. Las siguientes preguntas son para instalar unos u otros módulos, según el proyecto, pero esos serían los más estandarizados y fueron los elegidos. En el idioma elegimos el castellano, y la zona horaria la de Madrid/Spain.

Una vez terminamos con esto tenemos un proyecto que ya podemos arrancar, aunque nos encontraremos con un portal completamente vacío.

A partir de este momento podemos trabajar de 2 formas distintas: accediendo a los archivos y modificándolos manualmente o de forma gráfica, en nuestro caso hemos mezclado un poco de cada para comprender al máximo el funcionamiento de este framework y tener un mayor control sobre el aspecto y funcionamiento del portal.

Para acceder a la versión de administrador, tenemos que introducir al final de la url del portal generado la palabra “/admin”. Con esto nos pedirá un nombre y contraseña que anteriormente hemos acordado durante la instalación de Django-cms, como podemos ver en la figura 6.9.



Figura 6.9: Acceso a Administración de Django-cms.

Una vez en el entorno de administración, como podemos ver en la 6.10 nos muestra todas las opciones que tenemos, y un historial de lo último que hemos realizado. En este caso vemos que hay un apartado que pone “Catalogo” ya que estamos trabajando sobre la versión que lo incluye. Pero aparte de eso y de que el historial de acciones estaría vacío, este sería el aspecto que tiene un proyecto en Django-cms recién creado.

Todas estas acciones que podemos ver en la figura 6.10 son las funcionalidades que podemos variar de forma gráfica o modificando los ficheros vistos en las figuras 5.3 y 5.4, que son los que realmente se modifican, ya sea trabajando gráficamente o directamente sobre los mismos.

Como se ha explicado anteriormente en el capítulo de Entorno de desarrollo, Django utiliza el Modelo-Vista-Controlador [8] para el desarrollo de sus proyectos. Su funcionamiento lo vemos en la figura 6.11

Este patrón de diseño es uno de los más importantes en el desarrollo de proyectos informáticos y está cogiendo mucha fuerza en el mundo de las aplicaciones web.

The screenshot shows the Django CMS administration interface. The top bar is dark grey with the text "Administración de Django". Below it, the main area has a light grey header "Sitio administrativo". The interface is divided into several sections:

- Autenticación y autorización**: Contains links for "Grupos" and "Usuarios", each with "Añadir" and "Modificar" buttons.
- Catalogo**: Contains links for "Categorías", "Productos", and "Proveedores", each with "Añadir" and "Modificar" buttons.
- django CMS**: Contains links for "Grupos de usuarios (página)", "Permisos globales de las páginas", "Placeholder estáticos", "Páginas", and "Usuarios (página)", each with "Añadir" and "Modificar" buttons.
- Sitios**: A section for site management.

On the right side, there is a sidebar titled "Acciones recientes" (Recent Actions) with a table listing recent changes:

Mis acciones	
+ tu	Categoría
+ yo	Proveedor
✗ ladrios	Categoría
✓ Geolite	Proveedor
+ Azir	Proveedor
+ ladrios	Categoría
+ Geolite	Proveedor
✗ Geolite	Proveedor
✗ albert	Proveedor
✗ vvvv	Proveedor

Figura 6.10: Administración de Django-cms.

Lo más importante de este patrón, es que está formado por 3 actores.

- **Modelo:** es el encargado de gestionar y mantener los datos del proyecto.
- **Vista:** es el responsable del interfaz gráfico de usuario y de la detección de eventos que surjan sobre los componentes.
- **Controlador:** es quien hace corresponder la interacción del usuario, en la Vista, con los posibles cambios en el Modelo.

La parte de la Vista está casi completamente dentro de la carpeta templates, y es este uno de los apartados que más hemos modificado manualmente, sin usar el entorno gráfico.

Mucho del código que hay en este directorio está en lenguaje HTML. Dentro de los archivos de la carpeta templates, el principal es “base.html” y modificando este podemos cambiar no solo el contenido y formato, si no que podemos modificar la estructura de la página web. Estos cambios serían a nivel global de todas las páginas que compongan nuestro portal, aunque luego podemos modificar cada página una a una según las necesidades.

Para entenderlo mejor explicaremos que Django-cms en la versión gráfica nos ofrece una vista de la estructura del portal. Como podemos ver en la figura 6.12 tenemos 4 apartados diferentes

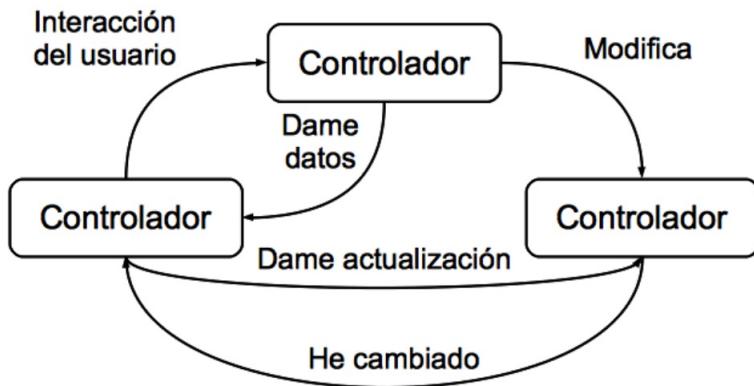


Figura 6.11: Dinámica del patrón MVC

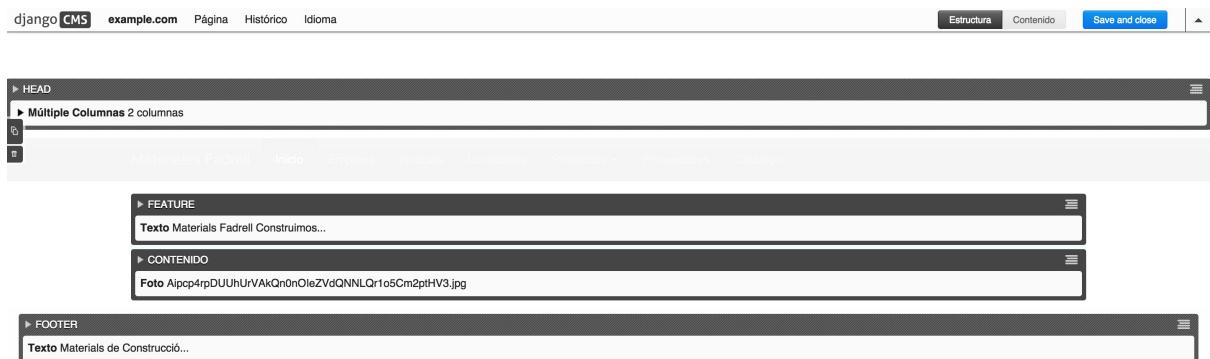


Figura 6.12: Estructura web principal.

para la web, los cuales podemos modificar independientemente entre ellos. Esta estructura en concreto corresponde a la web principal.

Dentro de cada contenedor su contenido puede ser diferente en cada página de nuestro portal web. Además no es necesario que se usen todos los contenedores que se tienen definidos, si no se utilizan simplemente no aparecerá por pantalla ese contenedor en concreto.

La versión por defecto, si no añadimos nosotros el resto, solo tiene un contenedor, denominado “CONTENIDO”. Este contenido es dinámico y como mostraremos más adelante también es posible dividirlo en columnas.

Para cambiar la estructura decidimos hacerlo de manera manual. Pretendíamos añadir un contenedor para la cabecera del portal. Para ello tuvimos que añadir el código que nos muestra la figura 6.13 en el archivo “base.html”. Además, como hemos dicho antes, pensamos que la mejor opción es que fuera estático ya que pretendemos que esa zona sea igual en todas las páginas de nuestra web. Para ello hemos utilizado el lenguaje de programación de plantillas propio de Django. Este lenguaje se basa en el uso de etiquetas del tipo “{ % ... % } ” para describir el comportamiento y contenido de las páginas. En este caso usamos “cms_toolbar” que se encarga de agregar elementos a la barra de herramientas. Esto nos permite modificar desde el frontend, lo cual es de gran ayuda para el usuario. De este modo le indicamos a nuestro proyecto que en la versión gráfica queremos que aparezca un contenedor en la posición de la cabecera, para que así luego el contenido y la distribución puedan hacerse de forma gráfica.

Luego hemos indicado que queremos que este nuevo placeholder/contenedor sea estático y que se llame “head”.

```
{% cms_toolbar %}  
<head>  
|  {% static_placeholder 'head' %}  
</head>
```

Figura 6.13: Cabecera.

Como se pretendía que el pie de página fuera también estático, mostrando siempre la información de la empresa, utilizamos la misma técnica que para la cabecera.

Otro de los aspectos que estaba claro, era que queríamos que hubiera un menú para poder seleccionar qué apartado queremos consultar. Para ello necesitamos otro código en el fichero “base.html”. Además como se ve en la 6.14 le hemos dado estilo añadiéndole Bootstrap[11]. Se trata de un framework que utiliza HTML, CSS y JavaScript para conseguir que nuestro portal tenga un aspecto mucho más agradable y nos aporta funcionalidad adicional, como puede ser el que aparezca un desplegado con las diferentes páginas web.

```
<div id="navbar" class="navbar-collapse collapse">  
| <ul class="nav navbar-nav">  
| |  {% show_menu 0 1 100 100 "menu.html" %}  
| </ul>  
</div>
```

Figura 6.14: Menú.

Después de esto añadimos un poco más de Bootstrap para que se acercara lo máximo posible a lo que la empresa quería, y empezamos con el desarrollo de cada página del portal. Para cada página se ha utilizado una estructura diferente según se requería. La forma básica de definir el contenido es que este puede ser dividido en columnas y estas a su vez en otras columnas. De esta forma podemos darle el formato deseado a cada página web.

Página Principal

Para la página principal decidimos utilizar una imagen grande del local, desde la perspectiva que vería un cliente antes de entrar, para atraer su atención. Además se ha elegido el título corto y sencillo, “Materials Fadrell Construimos tus sueños”, para que sea lo más directo posible, ver figura 6.15 .

Se le planteó a la empresa la idea de añadir enlaces a sus redes sociales, tanto Facebook, como Twitter, pero al ser principiantes en su uso preferimos aplazarlo a que se encontraran más cómodos con él.

Empresa

La siguiente página web, sería la referente a la empresa ver figura 6.16. Como hemos dicho anteriormente buscamos que la información sea directa y sencilla, así que elegimos 3 imágenes de los productos más destacados y entre estos, la información más general posible, pero al mismo tiempo dando todos los datos imprescindibles.

Materials Fadrell

Construimos tus sueños.



Materials de Construcción Fadrell S.L.L.

Polígono Fadrell, Nave 55, 12005 Castellón

Telf. 964 21 27 29 / 615 118 259

Figura 6.15: Página de inicio del portal.

Noticias

En este apartado la empresa pidió que se incluyesen varias fotos. Esto resultó un problema por la distribución, y por el tamaño de las fotos que se nos proporcionó que resultaban ser muy dispares. Esto provocaba grandes diferencias y perdíamos equilibrio. Al final tras muchas pruebas, decidimos intercalar texto con grupos de 3 imágenes, las cuales tuvieron que ser redimensionadas a priori. Como se puede ver en la figura 6.17 este es el resultado obtenido, y en la figura 6.18 vemos cual es la distribución de los contenidos. Nuestro problema viene de que nosotros en Django-cms no podemos predeterminar el tamaño de una imagen, ya que no accedemos directamente al código HTML que se guarda dentro de los contenedores para poder indicarle que tamaño queremos que tenga cada una. Este es el motivo por el cual tuvimos que redimensionar las imágenes antes de subirlas a nuestro portal web, ya que es la única manera de poder hacer las encajar, haciendo que queden uniformes.

Si nos fijamos más en detalle en la figura 6.18 podemos observar que en la distribución que se ha realizado, dividimos el contenedor para que primeramente nos permita introducir texto.



MATERIALES DE CONSTRUCCIÓN FADRELL



**GRUPO
IBRICKS**

[Materiales Fadrell](#) [Inicio](#) **Empresa** [Noticias](#) [Localizame](#) [Productos](#) [Proveedores](#) [Catalogo](#)



Nuestra joven empresa, fue constituida en el año 2012 aunque tenemos 35 años de experiencia en este mundo de la venta de materiales de construcción.

Formamos parte del grupo Ibricks.

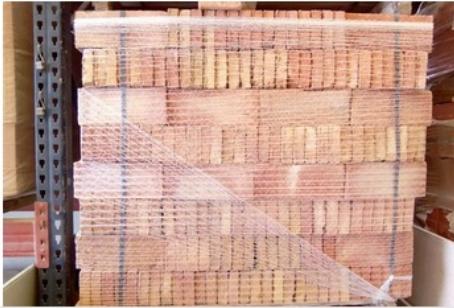
Somos una empresa local de Castellón de la Plana.

Tenemos los mejores materiales y ofrecemos un servicio personalizado para cada cliente.

Para contactar con nosotros puede realizarlo por:

- TELF. 964 21 27 29 / 615 118 259
- materialsfadrell@gmail.com

No dude en contactar con nosotros.





Materials de Construcción Fadrell S.L.L.

Polygono Fadrell, Nave 55, 12005 Castellón

Telf. 964 21 27 29 / 615 118 259

Figura 6.16: Página empresa.

Además nos permite darle formato, utilizando un tipo de letra concreto, un tamaño, que esté todo centrado, etc.

A continuación volvemos ha dividir el contenedor en otro apartado, el cual será dividido en 3 columnas, y dentro de cada una seleccionaremos que queremos una imagen, que deberíamos haber redimensionado con anterioridad. Se puede observar que hemos repetido este proceso 2 veces más para este caso en concreto.

Localízame

Uno de los módulos más destacados y que creímos que podía ser muy útil es el de la API de Google Maps. Con él podemos hacer que aparezca un pequeño mapa de Google, indicando dónde está nuestra empresa localizada, ver figura 6.20. Entre otras acciones nos permite calcular la ruta más rápida entre nuestra empresa y la dirección que introduzcamos por pantalla. Al hacer esto nos redirige a la web de Google Maps , tal y como podemos ver en la figura 6.21 .

Los módulos son tratados tanto por Django como por Django-cms como aplicaciones que nos proporcionan algunas mejoras a nuestro proyecto. En nuestro fichero “setting.py” de la



Materiales Fadrell Inicio Empresa **Noticias** Localizame Productos + Proveedores Catalogo

Kerakoll 2014, Auditorio Castellón



H40, NO LIMITS



CURSO DE FORMACIÓN GEOLITE, HOTEL LUZ



Materials de Construcción Fadrell S.L.L.

Polygono Fadrell, Nave 55, 12005 Castellón

Telf. 964 21 27 29 / 615 118 259

Figura 6.17: Página noticias.

figura 4.6, podemos ver el conjunto de todas las aplicaciones que tenemos en nuestro proyecto. Se incluyen las instaladas por defecto y en la parte final la correspondiente a nuestro catálogo y el propio portal realizado en Django-cms. La figura 6.19 nos muestra la totalidad de nuestras aplicaciones instaladas en nuestro proyecto.

Ofertas

Tenemos también un apartado de ofertas muy sencillo. Cada mes habrá un producto en oferta, el cual querían ir cambiando. En el caso de que quisieran poner en oferta más de un producto también sería posible, anidando y extendiendo la estructura, como hemos visto en imágenes anteriores, 6.22.

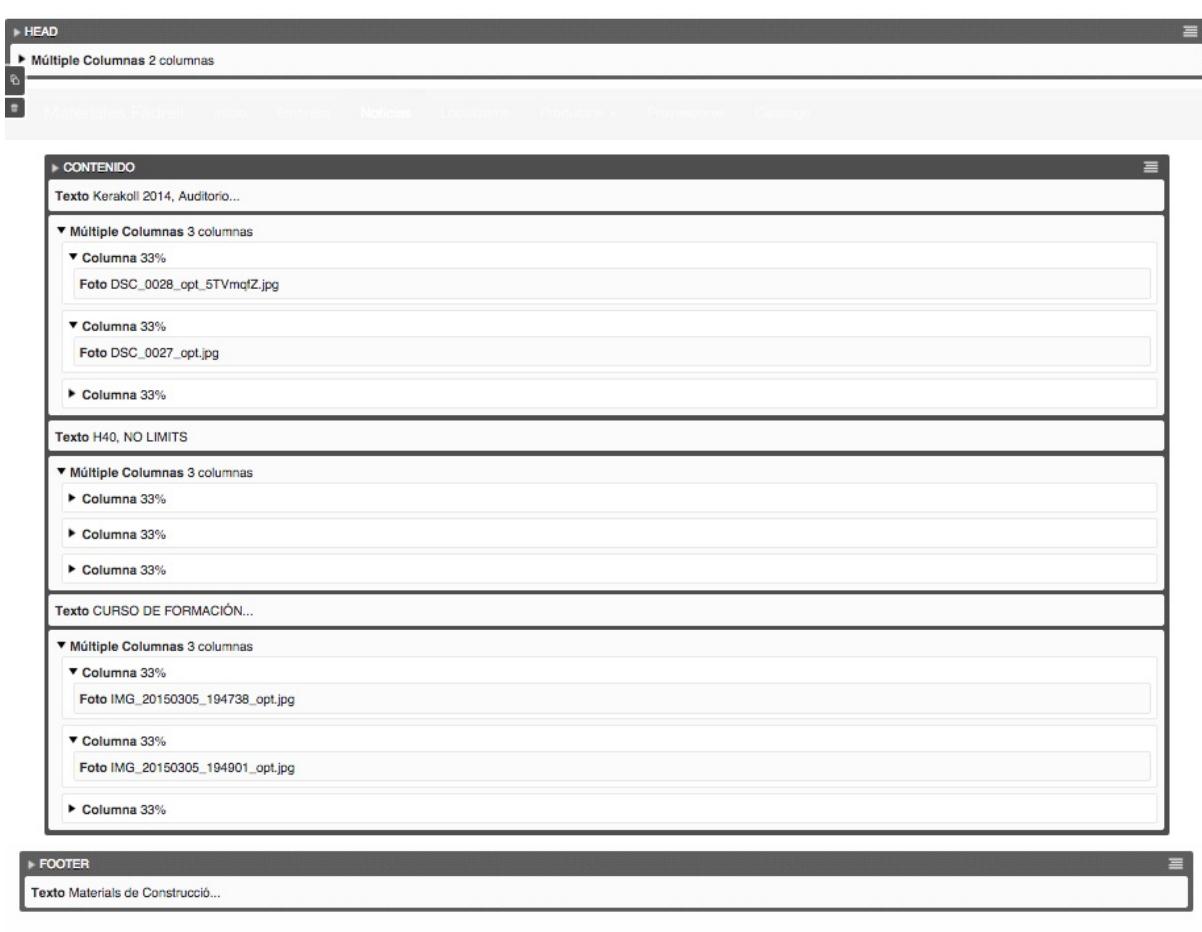


Figura 6.18: Estructura de la página de noticias.

Proveedores

Por último está el apartado de proveedores en el que se muestran las empresas que aportan el material a vender, ver figura 6.23. Se ha elegido poner una imagen del logotipo de cada proveedor, de modo que al hacer click sobre esta, nos llevarán a su web corporativa.

6.4. Implementación del catálogo

En este apartado vamos ha empezar desde el punto al que llegamos en el capítulo 4, en el apartado donde explicamos cómo instalar Django.

Una vez instalado el entorno de desarrollo tal y como se describió en el capítulo 4, usaremos la siguiente orden para crear nuestro proyecto.

```
django-admin.py startproject myproject
```

En este punto nos encontramos justo en el momento que representa la figura 4.3. Dentro de este proyecto desarrollaremos nuestra aplicación catálogo.

Para el desarrollo de una aplicación en Django los pasos que hay que seguir están bastante marcados. Al mismo tiempo que se irá explicando qué se hace en cada paso, se explicará para qué.

```

INSTALLED_APPS = (
    'djangocms_admin_style',
    'djangocms_text_ckeditor',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.admin',
    'django.contrib.sites',
    'django.contrib.sitemaps',
    'django.contrib.staticfiles',
    'django.contrib.messages',
    'cms',
    'menus',
    'sekizai',
    'treebeard',
    'djangocms_style',
    'djangocms_column',
    'djangocms_file',
    'djangocms_flash',
    'djangocms_googlemap',
    'djangocms_inherit',
    'djangocms_link',
    'djangocms_picture',
    'djangocms_teaser',
    'djangocms_video',

    'django_extensions',
    'my_demo',
    'my_demo.apps.myapp.catalogo',
)

```

Figura 6.19: Aplicaciones instaladas en nuestro proyecto

Como nuestra idea inicial es crear una aplicación para luego poder unirla al proyecto desarrollado con Django-cms, esto es lo primero que tenemos que hacer. Crearemos la aplicación. Para ello necesitamos encontrar a la altura del archivo “manage.py” de nuestro proyecto, confirmar que hemos activado el entorno virtual y usar la orden:

```
python manage.py startapp catalogo
```

Esto nos creará un directorio nuevo llamado “catalogo” el cual tendrá la siguiente estructura, ver figura 6.24.

Los archivos nuevos que aparecen son:

- **admin.py**: Se utiliza para añadir contenidos al apartado de administrador. Añadiendo las tareas que deseemos que se puedan acceder de forma gráfica.
- **migrations/**: En este directorio nos encontramos información que referencia a la base de datos y será la que use el sistema para saber si ha habido alguna actualización en los modelos.
- **models.py**: En este archivo crearemos los modelos, que normalmente representan una tabla de la base de datos. Además contendrá la información de los atributos esenciales y el comportamiento de los datos.



Localízame



Calcular la manera más rápida de ir:

Su dirección: Calcular la ruta

Materials de Construcción Fadrell S.L.L.

Polígono Fadrell, Nave 55, 12005 Castellón

Telf. 964 21 27 29 / 615 118 259

Figura 6.20: Página de localización de la empresa.

- **tests.py**: Se utiliza para escribir tests, los cuales luego podremos ejecutar, para comprobar el comportamiento de la aplicación.
- **views.py**: En este archivo crearemos las vistas, una vista es una función de Python que toma una petición web y devuelve una respuesta web. La respuesta puede ser desde devolver un HTML, redireccionar, ERROR 404, etc.

En este punto lo primero que tenemos que hacer es crear nuestro modelos, para ello accedemos a nuestro archivo “models.py”. Con la ayuda de los diagrama entidad-relación de la figura 5.9 y el esquema lógico de la figura 5.10, podemos tener una primera idea de lo que tenemos que ir implementando.

Un modelo es la representación de los datos de nuestra aplicación. Contiene los atributos básicos y el comportamiento de los datos que serán almacenados. Por lo general, cada modelo se representará en una tabla de la base de datos.

En cuanto a la implementación, la forma de crear clases es muy parecida a la que realizaríamos en un proyecto cualquiera desarrollado con Python. Como podemos ver en la figura 6.25, aunque el formato que se sigue es muy parecido a Python, tenemos que indicarle que nos guarde cada dato, con unas características pensadas para una base de datos, ya que será el

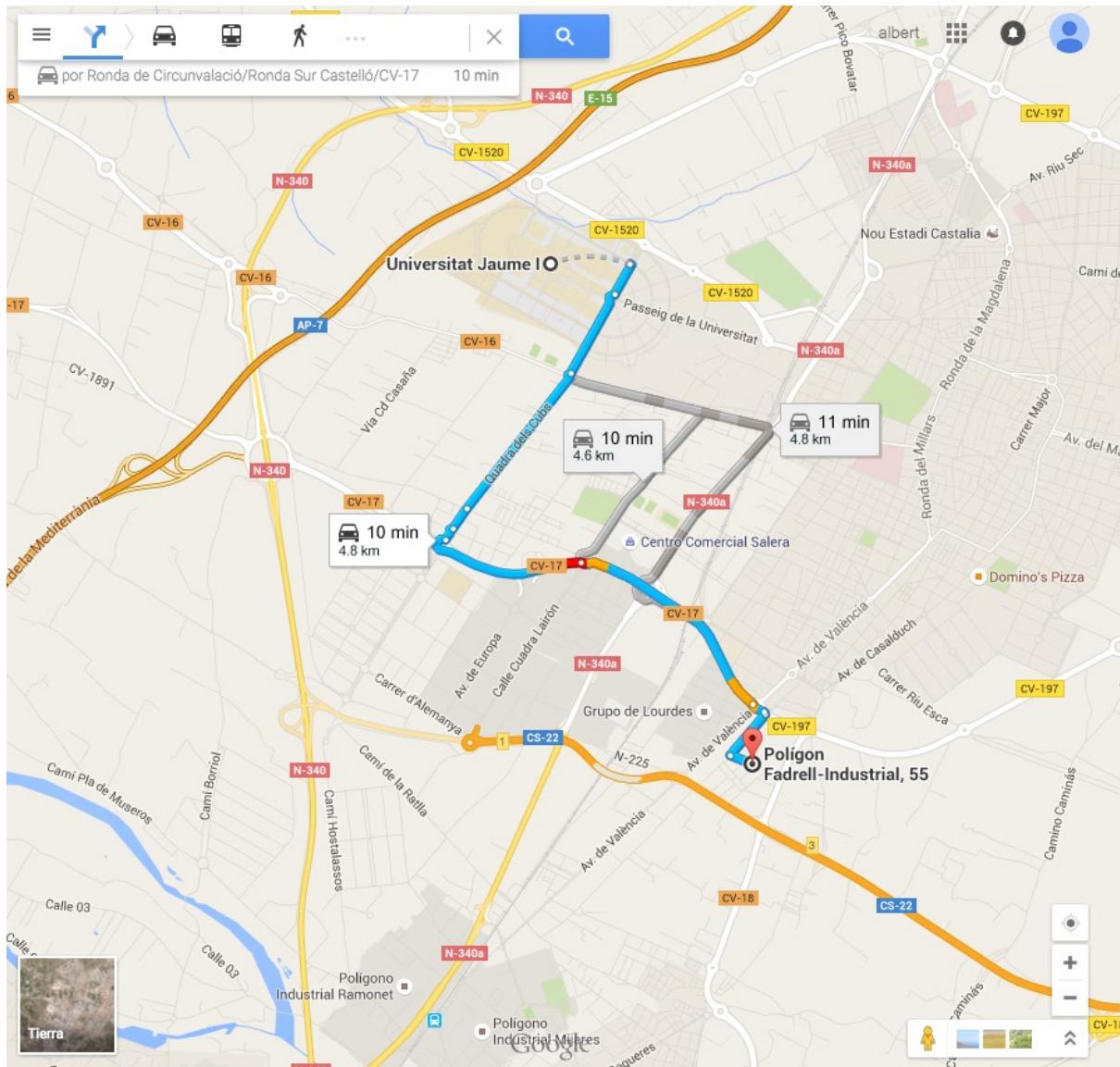


Figura 6.21: Página de Google Maps con la ruta desde la UJI a la empresa.

propio Django el que con ayuda de los modelos, nos creará las tablas necesarias que representen nuestro sistema.

La forma de crear un modelo es creando una clase. que hereda de “`models.Model`”. A continuación indicaremos los atributos que tendrá nuestro modelo. Este es uno de los puntos más potentes del framework ya que utilizando su sintaxis nos crea al mismo tiempo las clases a nivel de bases de datos y a nivel de una clase de Python. Por último sobrescribiremos los métodos de escritura `__str__` y de guardado `save`, adaptándolo a nuestras necesidades.

Es muy importante en este punto explicar que en Django si utilizamos algún tipo de clase o función que no está a nuestro alcance deberemos importarla. Para ello necesitaremos escribir al principio del archivo todos los imports que sean necesarios. En este caso en concreto sería, ver figura 6.26.

The screenshot shows the homepage of the Materials de Construcción Fadrell website. At the top left is the logo 'MC MATERIALS DE CONSTRUCCIÓN FADRELL' with a blue square icon containing a house and a wrench. To the right is the 'GRUPO IBRICKS' logo with three orange vertical bars. A black navigation bar below the header contains links: 'Materiales Fadrell', 'Inicio', 'Empresa', 'Noticias', 'Localízame', 'Productos', 'Proveedores', and 'Catalogo'. Below the navigation, a promotional text reads: 'Este mes tenemos en oferta la Llana biflex Inox COLOTOOL para microcemento.' To the right of the text is a small image of the product, a stainless steel flexible lanyard. Below the text is another line: 'Para más información no dude en contactar con nosotros.' At the bottom of the page, a light gray footer box contains the company name 'Materials de Construcción Fadrell S.L.L.', its address 'Polígono Fadrell, Nave 55, 12005 Castellón', and its phone number 'Telf. 964 21 27 29 / 615 118 259'.

Figura 6.22: Página de ofertas.

Cabe destacar que en las tablas fruto del diseño lógico, teníamos una relación de muchos a muchos y esto había dado lugar a una tabla extra. Sin embargo, como vemos en la figura 6.27 tenemos simplemente creadas las clases correspondientes a las 3 entidades del diagrama entidad-relación: Producto, Categoría y Proveedor. Esto es gracias a que se lo podemos indicar en la clase Producto, diciendo que tenemos una relación de muchos a muchos con la clase Proveedor. Automáticamente se crea esa tabla extra que se necesita con todos los atributos que considera que puedan ser necesarios. La crea internamente, para poder usarla como nosotros queremos, pero si no accedemos a la base de datos no podemos saber que esa tabla ha sido creada. Por otro lado investigando un poco vimos que se podían crear “diccionarios” que se utilizan para poder implementar en nuestras vistas relaciones entre varias opciones. Y era tan simple como hacer lo que vemos en la figura 6.28.

Para que estos cambios se vean reflejados en el proyecto, necesitaremos indicarle en el fichero “settings.py” del directorio “myproject” que hemos añadido una nueva aplicación, tal y como se muestra en la figura 6.29.

Ahora necesitaremos que la base de datos se actualice. Para ello le indicaremos que queremos que busque si hay algún cambio en la base de datos, mediante la siguiente orden:

```
python manage.py makemigrations catalogo
```

Esto convertirá las clases de nuestro modelo en tablas de la base de datos mediante las correspondientes sentencias CREATE en la base de datos. Sin embargo no nos los actualizará con el contenido, Para ello una vez creados los modelos usaremos la orden:

```
python manage.py migrate catalogo
```

Si queremos que actualice esa aplicación en concreto pero en el caso de que no le indicamos nada, revisará todas las aplicaciones.

En este punto nos encontramos con que tenemos unos modelos a los cuales no podemos acceder desde la aplicación. Para ello es necesario entrar en modo administrador a nuestro proyecto. A diferencia de Django-cms en Django tenemos que indicarle explícitamente que queremos que nos cree un usuario para poder acceder a la versión de administrador. Para ello usaremos la orden:



Figura 6.23: Página de proveedores.

```

catalogo/
    __init__.py
    admin.py
    migrations/
        __init__.py
    models.py
    test.py
    views.py

```

Figura 6.24: Estructura de una aplicación en Django.

```

class Proveedor(models.Model):
    nombre=models.CharField(max_length=200)
    direccion=models.CharField(max_length=200)
    correo=models.CharField(max_length=200)
    slug = models.SlugField(unique=True)
    def __str__(self):
        return self.nombre

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.nombre)
        super(Proveedor, self).save(*args, **kwargs)

```

Figura 6.25: Código del modelo del proveedor.

```

from django.db import models
from django.template.defaultfilters import slugify
from PIL import Image

```

Figura 6.26: Ejemplo de los imports necesarios para la creación de los modelos.

```
python manage.py createsuperuser
```

Estando siempre a la altura del fichero “manage.py”.

Nos pedirá que introduzcamos un nombre de usuario y una contraseña, que deberemos repetir para confirmar que no nos hemos equivocado.

Para acceder a la versión de administrador deberemos arrancar el proyecto con el comando “python manage.py runserver”. Si no hemos indicado otro puerto, para acceder a la versión de administrador por defecto sería a través de la URL “<http://127.0.0.1:8000/admin>”. Tras introducir el nombre y la contraseña nos encontraremos con algo parecido a la figura 6.30.

Para poder ver nuestros modelos en el administrador de Django tenemos que modificar el fichero “admin.py”, de la figura 6.31. Nos encontramos con código comentado, que hace referencia a la versión antigua que se utilizaba para registrar los modelos en el administrador en las primeras versiones de Django que utilizamos. Para poder registrar lo que queríamos en el administrador, como podemos observar en la actualidad es mucho más simple, introduciendo una simple etiqueta que sería la que haría todo el trabajo de registrar el modelo (`@admin.register(modelo)`). Al configurarlo de este modo la versión de administrador de Django cambia y nos muestra lo que vemos en la figura 6.32. Desde esta web podremos insertar, borrar y modificar los objetos que queramos introducir en nuestra base de datos.

De los 3 modelos que tenemos vamos a centrarnos en el de Productos que es el que tiene un poco de todo, para no repetirnos demasiado. Una de las principales ventajas que nos ofrece este framework es que nos proporciona mucha seguridad, ya que si intentamos introducir un producto, y nos olvidamos de algún dato, el nos lo muestra y nos indica en un pequeño texto que pasa. Además si hemos diseñado bien la base de datos, al tener claves ajena con los otros modelos, nos muestra un desplegable con las opciones que tenemos. De este modo nos aseguramos que los datos se reciben, como se espera.

En el caso del Proveedor para el que podíamos tener más de uno para el mismo Producto, nos indica que podemos seleccionar los que sean, mostrando todos los que tenemos ya insertados en la base de datos en una pequeña tabla. Para la variable Tipo que habíamos definido como

```

class Producto(models.Model):
    nombre=models.CharField(max_length=200)
    categoria=models.ForeignKey('Categoria')
    proveedor=models.ManyToManyField(Proveedor)
    tipo=models.CharField(max_length=1, choices=PRODUCTO_TIPOS)
    descripcion= models.CharField(max_length=200)
    imagen= models.ImageField(upload_to="img/")
    slug = models.SlugField(unique=True)

    def __str__(self):
        return self.nombre

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.nombre)
        super(Producto, self).save(*args, **kwargs)


class Categoria(models.Model):
    nom=models.CharField(max_length=200)
    slug = models.SlugField(unique=True)
    descripcion= models.CharField(max_length=200)
    imagen= models.ImageField(upload_to="img/")

    def __str__(self):
        return self.nom

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.nom)
        super(Categoría, self).save(*args, **kwargs)

```

Figura 6.27: Modelos implementados en Django para el catálogo.

```

PRODUCTO_TIPOS=
    ('N', 'nacional'),
    ('I', 'importado'),
)

```

Figura 6.28: Ejemplo de creación de tipos de datos seleccionables.

un seleccionable, nos lo muestra con un desplegado con las opciones que le habíamos indicado, como muestra la figura 6.33.

Para poder visualizar todo esto en nuestro portal necesitamos crear las vistas. En nuestro caso hemos tenido que crear 2 vistas, una que nos muestre la totalidad de las categorías, y otra para que pulsando en una de estas nos redireccione a otra página que nos muestre sólo los productos de esa categoría.

En este punto sería donde conectaríamos todo lo realizado hasta el momento, para comprender como funcionan las vistas. Primero deberemos crear 2 archivos HTML que pondremos en el directorio “/templates”, que serán nuestras plantillas. Se llaman plantillas porque realmente

```

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'catalogo',
    'django_markdown',
)

```

Figura 6.29: Añadir una aplicación al settings.

The screenshot shows the Django administration interface. At the top, it says "Django administration" and "Welcome, adrian. Documentation / Change password / Log out". Below that, it says "Site administration". There's a sidebar on the left with "Auth" selected, showing "Groups" and "Users" with "Add" and "Change" buttons. To the right, there's a "Recent Actions" box which is currently empty, and a "My Actions" box which also says "None available".

Figura 6.30: Versión de administrador de Django.

el contenido que se genera es dinámico, lo cual nos proporciona grande ventajas, respecto a los archivos HTML estáticas.

Como podemos ver en la figura 6.34, tenemos un bucle en la línea 17 que recorre una lista de objetos. Serán estos objetos los que le añadiremos a la vista que crearemos a continuación. Además en las líneas 19 y 20 vemos que accedemos a los atributos de la clase Categoría, que será el tipo de modelo que le indicaremos que tiene que enviar. En resumen tendremos que crear una vista que devuelva una lista de objetos del tipo Categoría para poder recorrerla y de cada objeto acceder a sus atributos.

Para crear la vista, deberemos modificar el fichero “views.py”, en él crearemos las vistas necesarias.

Como observamos en la figura 6.35, la forma de crear una vista consiste en crear una clase con un nombre que queramos a la cual le indicamos que vamos ha devolver un “ListView”, que sería como decir que vamos a devolver una lista de objetos. Seguidamente vemos en la línea 11 que vamos ha utilizar una plantilla que en nuestro caso se llama “index.html”, y en la línea 12 que el tipo de objetos ha devolver serán Categorías como se había previsto.

La siguiente vista tiene la peculiaridad de que en ella realizamos una búsqueda y además al mismo tiempo recibiremos un dato en la plantilla, para que esta cree el contenido dinámicamente de los productos que forman parte de una categoría. Para ello se ha creado un método como vemos en la línea 19 de la figura 6.35, que nos devuelve un subconjunto de los Productos que cumplan que la categoría a la que pertenecen es la que le pasamos a través de la URL.

La idea que nosotros perseguimos es la de que al acceder a la aplicación nos muestre el listado de todas las categorías. El cliente seleccionará una de ellas, y usando la URL le redireccionamos a otra página que nos mostrará sólo los productos de esa categoría seleccionada.

```

from django.contrib import admin
from .models import Producto,Categoria,Proveedor
from django_markdown.admin import MarkdownModelAdmin


@admin.register(Producto)
class ProductoAdmin(MarkdownModelAdmin):
    pass


@admin.register(Categoría)
class CategoiaAdmin(MarkdownModelAdmin):
    pass


@admin.register(Proveedor)
class ProveedorAdmin(MarkdownModelAdmin):
    pass


# class ProductoAdmin(admin.ModelAdmin):
#     prepopulated_fields ={'slug': ('nombre',)}
#     list_display=('nombre', 'categoria', 'tipo')
#     search_fields=['nombre']


# class CategoiaAdmin(admin.ModelAdmin):
#     prepopulated_fields ={'slug': ('nom',)}
# Register your models here.

# admin.site.register(Producto, ProductoAdmin)
# admin.site.register(Categoría, CategoiaAdmin)

```

Figura 6.31: Registrar modelos en el administrador.

The screenshot shows the Django administration interface. At the top, a blue header bar reads "Django administration". Below it, a white header bar reads "Site administration". The main content area is divided into sections:

- Authentication and Authorization** (blue header): Groups (Add, Change), Users (Add, Change).
- Catalogo** (blue header): Categories (Add, Change), Products (Add, Change), Proveedors (Add, Change).
- Recent Actions** (gray header): A list of recent actions with icons:
 - + wwwww Product
 - + YYYY Product
 - + kkkkk Proveedor
 - ccccc Product
 - dddddd Product

Figura 6.32: Versión de administrador de Django con el catálogo.

Para hacer esto Django utiliza un formato muy especial para trabajar con las URLs. Como vemos en la figura 6.36, este sería el contenido del fichero “urls.py”. En primer lugar se incluye la URL de administración. Nosotros nos centraremos en las líneas 9 y 10. Su forma de actuar con las URLs, es que el framework va comparando la que queremos acceder con todas las que tengamos en este archivo, si alguna coincide, entonces llama a la vista que le indica. En nuestro caso en la línea 9 tenemos indicando que, si el contenido de la URL es simplemente el servidor

Add producto

Please correct the errors below.

Nombre:	sadf
This field is required.	
Categoría:	<input type="text"/> +
Proveedor:	aaa kkkk <input type="button"/>
Hold down "Control", or "Command" on a Mac, to select more than one.	
This field is required.	
Tipo:	<input type="text"/>
This field is required.	
Descripción:	<input type="text"/>
This field is required.	
Imagen:	<input type="button"/> Seleccionar archivo nada seleccionado
This field is required.	
Slug:	<input type="text"/>

[Save and add another](#) [Save and continue editing](#) [Save](#)

Figura 6.33: Pantalla de inserción de productos.

y el puerto, quiere decir que no hemos introducido nada más a la URL de base, entonces nos llamará a la vista “IndexView.as_view()”. En la línea 10 indicamos que si la dirección tiene el formato `http://localhost:8000/c/cat-id` se ejecutará el método ProductosCat y este recibirá el identificador de la categoría que queremos mostrar. El resultado será una vista conteniendo solo los productos de dicha categoría.

El framework proporciona un lenguaje basado en expresiones regulares precedidas por el carácter ‘r’ para representar conjuntos de URL. Al mismo tiempo permite asociar acciones a las URLs accedidas.

En este momento ya tendríamos la aplicación completamente funcional. Al arrancar el servidor nos mostraría lo que vemos en la figura 6.37, y si clicamos en una imagen nos llevaría a la otra plantilla que vemos en la figura 6.38. Si nos fijamos en las URLs de las figuras veremos que ha funcionado correctamente.

6.5. Integración del catálogo en el portal

Por último hemos de integrar esta aplicación realizada con Django en nuestro proyecto realizado en Django-cms. La forma de realizarlo es bastante sencilla, consiste en añadir al “settings.py” del proyecto en Django-cms el nombre de la aplicación y en la aplicación crear un archivo llamado “cms.app.py” el cual pondremos a la altura de “models.py” y “views.py” dentro del directorio “catalogo”. Como vemos en la figura 6.39 lo que hacemos es indicarle dónde tenemos la ruta con las URLs de la aplicación y el nombre con el cual queremos que la registre en nuestro portal.

6.6. Desplegado del portal

Mientras finalizábamos la implementación del portal, empezamos con el trabajo de búsqueda del hospedaje.

En la propia web de Django-CMS nos encontramos un apartado que nos ofrece diferentes webs que ya soportan Django.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <!-- Latest compiled and minified CSS -->
6      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
7      <title>Catalogo</title>
8  </head>
9  <body>
10     <div class="container">
11         <div class="jumbotron">
12             <h2>Catalogo</h2><br />
13             <p class="bg-warning">Catalogo</p>
14
15         </div>
16         {% for a in object_list %}
17             <div class="col-md-3">
18                 <h6> {{a.nom}}</h6>
19                 <a href="/c/{{a.id}}"></a>
20             </div>
21             {% endfor %}
22             <div class="row">
23             </div>
24         </div>
25     </div>
26 </body>
27 </html>
28

```

Figura 6.34: Plantilla principal para el catálogo.

```

1  from django.shortcuts import render
2  from django.views.generic import ListView, DetailView
3  from .models import Producto, Categoria
4
5  from django.http import HttpResponseRedirect
6  from django.shortcuts import render_to_response
7  from django.template import RequestContext
8  from django.views.generic.base import RedirectView
9
10
11 class IndexView(ListView):
12     template_name = 'index.html'
13     model=Categoria
14
15
16 class ProductosCat(ListView):
17
18     template_name='productoscat.html'
19     model=Producto
20
21     def get_queryset(self):
22         print(self.args)
23         print(self.kwargs)
24         return Producto.objects.filter(categoria=self.kwargs['cat_id'])
25

```

Figura 6.35: Creación de vistas.

Tuvimos que hacer una pequeña investigación para ver cuántas de la media docena de webs que nos aconsejaban se adaptaban a nuestras necesidades. En la web nos aconsejaban las siguientes páginas web para contratar un servidor: Aldryn, site5, nine.ch, DjangoEurope, Hosting4Django y ungleich.

```

1  from django.conf.urls import patterns, include, url
2  from django.contrib import admin
3  from catalogo.views import IndexView, ProductoDetailView, ProductosCat
4
5
6  urlpatterns = patterns('',
7      # Examples:
8      # url(r'^$', 'myproject.views.home', name='home'),
9      # url(r'^blog/', include('blog.urls')),
10
11     url(r'^admin/', include(admin.site.urls)),
12     url(r'^$', IndexView.as_view()),
13     url(r'^c/(?P<cat_id>\d+)/$', ProductosCat.as_view()),
14

```

Figura 6.36: Contenido de urls.py

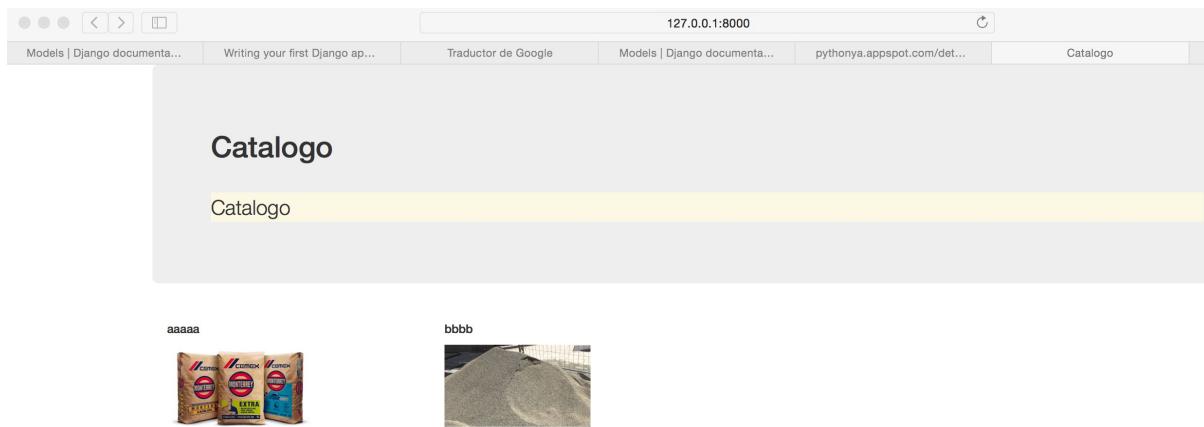


Figura 6.37: Página principal del catálogo.

Al ir mirando una a una, encontramos diferentes problemas, ya que muchos de los servidores web que nos recomendaban estaban limitados al uso de MySQL, que es otro sistema de gestión de bases de datos. En nuestro caso el hecho de usar PostgreSQL resultó finalmente ser un inconveniente en este punto, ya que la elección del hospedaje quedó muy limitada.

Otros problemas fueron la compatibilidad de nuestro sistema al completo, ya que al evolucionar tan rápido Django, muchos de estos servidores no daban soporte todavía a las versiones que habíamos utilizado tanto de Django-cms para el portal web y para la aplicación desarrollada en Django. Por otro lado el usar Python 3.4 también nos restringió la búsqueda, ya que en muchos servidores solo daban soporte a Python 2.7.

Por todo esto la única web que sí que nos daba soporte para todas nuestras necesidades era Aldrynn.com. Aunque intentamos desplegar nuestro portal en este servidor, en este punto de desarrollo del proyecto habíamos sobrepasado ya las 300 horas previstas. Se había logrado leer la documentación y se había descargado el software necesario para poder desplegar el portal pero no llegamos a completar este proceso.

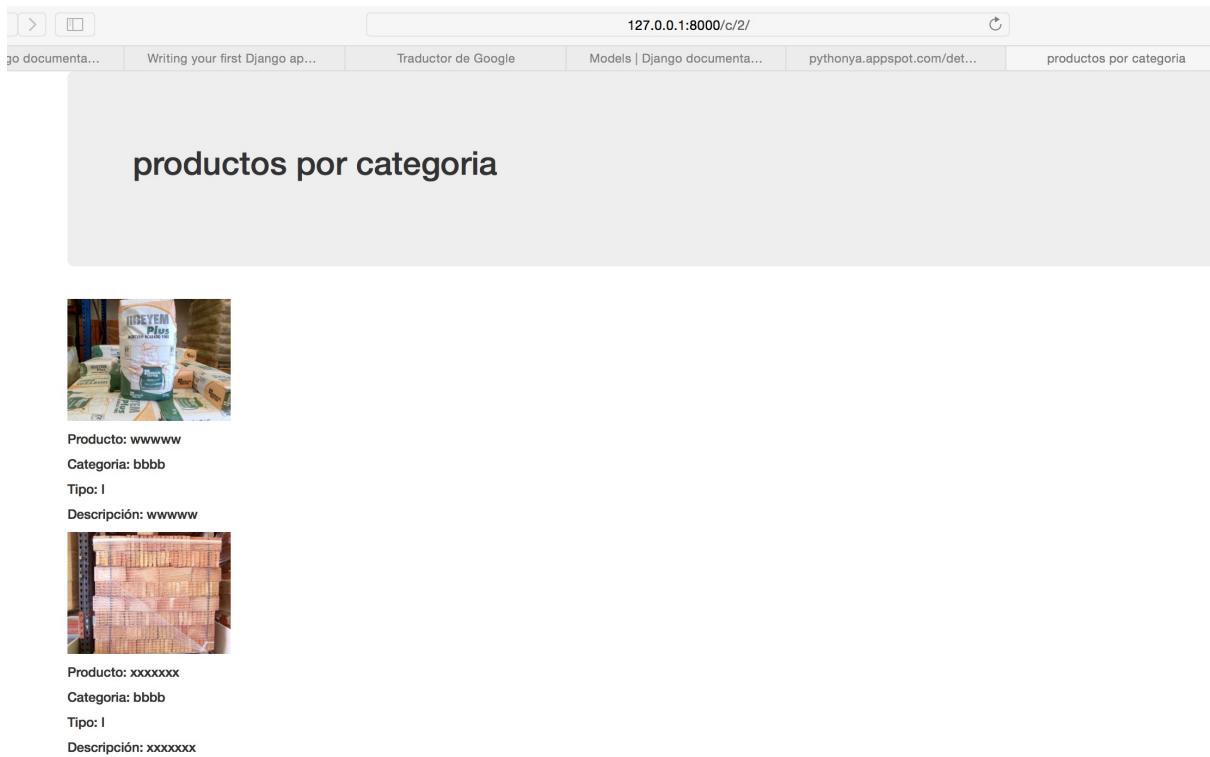


Figura 6.38: Productos por categoría.

```

1  from cms.app_base import CMSApp
2  from cms.apphook_pool import apphook_pool
3  from django.utils.translation import ugettext_lazy as _
4
5  class MyApphook(CMSApp):
6      name = _("catalogo")
7      urls = ["my_demo.apps.myapp.urls"]
8
9  apphook_pool.register(MyApphook)
10
11

```

Figura 6.39: Contenido del fichero cms.app.py.

6.7. Documentación y formación

Tal y como puede verse en la figura 3.2 una de las tareas planificadas inicialmente consistía en escribir una pequeña documentación: una guía para el gestor de los contenidos del portal de cómo funcionaba la web.

Se planificó la escritura de un pequeño manual de administrador, ya que la idea principal de la empresa era poder cambiar las ofertas por temporadas. Para hacer estos cambios es necesario modificar la web. El manual debería contener un ejemplo práctico de este proceso, pero también sería aconsejable explicar un poco todo lo que se puede conseguir con el uso del administrador de Django-cms, para tener una idea general.

Nuestro manual debería contener un ejemplo práctico de como modificar una oferta paso a paso. Estos cambios no resultan nada triviales para una persona que no ha usado nunca Django-cms y que además no tiene un contacto directo con la programación web, como es el caso de

los miembros de la empresa. Por esto el manual tenía que ser lo más sencillo posible. El uso de imágenes favorecería el aprendizaje, aunque debería explicarse todas y cada una de las posibles opciones.

El manual debe apoyarse sobre el uso de la parte gráfica que nos ofrece el framework Django-cms, ya que de no ser así, deberíamos explicar HTML básico, sólo para entender lo que ya está programado y esto sería muy largo. Además el propio Django-cms nos ofrece su propio lenguaje de programación, que se usa junto el HTML para conseguir modificar el comportamiento de nuestro portal web. Añadir todo esto en un manual, escrito para que una persona sin ningún tipo de conocimiento previo de informática o programación es un trabajo demasiado grande y complejo ahora mismo.

La idea que se tenía sobre el manual en el momento de la propuesta técnica era la de un documento muy corto que permitiera poder realizar algunos cambios puntuales en el portal web. Esto se habló antes de utilizar este framework y ver que para poder hacer un uso correcto del mismo se necesitan bastantes conocimientos para entender bien lo que se quiere realizar.

Hoy por hoy la empresa tampoco veía como una necesidad el tener un manual de administrador. En un futuro si la empresa quiere realizar cambios y no sabe como hacerlos, tal vez acudan a mí para realizar esos cambios o para la creación del manual de administrador.

En definitiva la elaboración del manual se tuvo que cambiar por reuniones explicativas a todos los miembros de la empresa, ya que el tiempo jugaba en nuestra contra. Además pensamos que una prueba en vivo de cómo funciona todo sería mucho más didáctico. Para ello una vez el portal era funcional, nos reunimos con todos para mostrarles como:

- Acceder a la versión Administrador del portal.
- Borrar y añadir nuevas páginas al portal.
- Modificar el contenido de un contenedor.
- Cambiar la distribución de un contenedor.
- Añadir, modificar y borrar tanto productos, como categorías o proveedores.

Como el trato con el personal era continuo siempre que surgía alguna duda, ellos preguntaban y se explicaba. Hubiera sido mejor realizar una documentación pero fue imposible, con el tiempo que teníamos y la cantidad de problemas que fueron surgiendo.

Capítulo 7

Conclusiones

Podemos diferenciar las conclusiones técnicas de las personales.

En cuanto a lo que hemos conseguido desarrollar durante la estancia en prácticas decir que he conseguido crear un portal web desde cero, la creación de cuentas en diversas redes sociales y la optimización en las búsquedas de Google. Estos eran los objetivos empresariales, que quedarían cumplidos. Por otro lado en los objetivos técnicos, se han cumplido casi todos: diseñar, estructurar y crear un portal web para la empresa, diseñar e implementar un pequeño catálogo de los diferentes materiales y la formación del personal en el uso tanto del portal web como de las diferentes redes sociales. Los 2 objetivos que no he podido realizar han sido la integración del proceso de ventas de artículos, ya que como he explicado hubieron muchos problemas de compatibilidades. Por otro lado tenemos la escritura de un manual de administrador, que no se pudo realizar por falta de tiempo en la estancia.

Al ser la primera vez que usaba un framework para la creación de un portal web, descubrí su complejidad. En mi caso tengo la sensación de que el framework de Django-CMS ha sufrido una gran evolución en muy poco tiempo y ha coincidido con mi época de las prácticas, y esto no ha jugado a mi favor. Es más, ha sido un gran inconveniente.

Después de usar mucho el framework, me doy cuenta que es imposible aprender todo sobre él en un tiempo tan limitado como son 300 horas. Su curva de aprendizaje es altísima, pero al mismo tiempo es muy potente y esto lo justifica. Aunque se apoya en un lenguaje de programación, Python, que yo ya había usado en diversas asignaturas durante la carrera se podría decir que tanto Django como Django-CMS es un mundo en sí mismo.

Uno de los problemas más importantes, fue la documentación disponible, ya que esta es escasa y confusa. Al seguir los manuales muchas veces daban por sobreentendidos muchos pasos, que en un primer momento te hace pensar que algo estás haciendo mal. Y buscas problemas en cosas que en muchas ocasiones eran “tonterías”.

El desarrollo de este proyecto ha sido una de las experiencias donde más he aprendido sobre informática y sobre todo del trato con la gente. Ya que, aunque yo solo tenía un trato directo con los miembros de la empresa, veía diariamente el trato que tenían ellos con los clientes, y creo que es una de las cosas más importantes que hay que aprender para en un futuro integrarme bien en el mundo laboral.

Personalmente al principio cuesta el hecho de que todo es nuevo, tanto el entorno, como los compañeros, las herramientas que he ido usando, el tamaño del proyecto, con la responsabilidad que eso conlleva. Sin embargo con el tiempo he ido aprendiendo todo lo que he podido y comparándome conmigo mismo antes y después de la estancia en prácticas puedo decir que he crecido académica y personalmente.

Creo que la ayuda del tutor ha sido fundamental para llegar hasta aquí. Al no tener un informático en la empresa, cualquier duda tenía que plantearla al tutor, y esto creo que también ha ralentizado un poco el proyecto. Aunque solo sea por casos como que tienes un error y no eres capaz de verlo, y buscando información puedes “perder” toda la mañana, cuando hablando con el tutor son 5 minutos. Intentar resolver el problema por tu cuenta no es perder tiempo, pero no avanzas en el proyecto todo lo que esperabas, o que tenías previsto.

El camino ha sido un poco duro, pero creo que ha valido la pena, ya que te sientes bien al ver que al principio no sabías nada de todo este mundo de Django, y ahora he llegado ha realizar un portal web, bastante completo y complejo.

También me gustaría agradecer a la empresa Materiales de Construcción Fadrell por hacer que mi primer contacto con el mundo laboral sea tan fácil. Agradecerles sobre todo el esfuerzo por intentar comprender el proyecto al detalle, ya que aunque no son informáticos, me iban preguntando diariamente por el desarrollo del proyecto.

Aprender todo esto, creo que puede favorecer que en un futuro próximo mi búsqueda de un trabajo. No solo con proyectos hechos con Django, sino para la creación de cualquier web, o el uso de algún CMS al cual creo que me podré adaptar rápidamente.

Bibliografía

- [1] <http://gestwin.com>
- [2] <http://www.djangoproject-cms.org/en>
- [3] <https://www.djangoproject.com>
- [4] <https://www.drupal.org/>
- [5] <https://www.python.org>
- [6] <http://www.boe.es/boe/dias/2015/05/20/pdfs/BOE-A-2015-5578.pdf>
- [7] CAY S. HORSTMANN, *Big Java*, 4th edition.
- [8] ÓSCAR BELMONTE FERNÁNDEZ, CARLOS GRANELL CANUT, MARÍA DEL CARMEN ERDOZAIN NAVARRO, *Desarrollo de proyectos informáticos con tecnología JAVA*
- [9] http://www.seg-social.es/Internet_1/Trabajadores/index.htm
- [10] <http://django-book.mkaufmann.com.ar/chapter05.html>
- [11] <http://getbootstrap.com>
- [12] <http://mlgdiseno.es/flat-design-la-tendencia-web-en-este-2013>
- [13] www.google.com/business