ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE COMPUTADORES

APLICACIÓN WEB PARA LA GESTIÓN Y PROMOCIÓN DE CAMPAÑAS

WEB APPLICATION FOR CAMPAIGN MANAGEMENT AND PROMOTION

Realizado por
Pedro Guillén Aroca
Tutorizado por
Carlos Rossi Jiménez
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Febrero 2015

Fecha defensa:
El Secretario del Tribunal

Resumen: El proyecto que aquí se presenta ha consistido en la realización de una ***aplicación web para la gestión y promoción de campañas***, en particular el análisis, diseño e implementación de una herramienta para la promoción turística e integrada en el portal de *andalucia.org*.

La aplicación desarrollada se estructura en dos subsistemas, uno de administración y otro de gestión de los diferentes *microsites* creados para las campañas existentes en la plataforma. Al subsistema de administración tienen acceso el *administrador* y el *gestor*. El *administrador* dota al *gestor* de permisos para gestionar una campaña. Las campañas se componen de una serie de elementos (*plantillas*, *documentos e ítems*, *concursos*, *eventos e invitaciones*, *integración con redes sociales*) que ayudan a construir su *microsite*. Además, se ha desarrollado un módulo de estadísticas, integrado con Google Analytics, que aporta información analítica sobre una campaña a los responsables de la misma.

Para la realización del proyecto se han utilizado diferentes herramientas y tecnologías. Para el análisis y diseño de la aplicación se ha utilizado el lenguaje UML y la herramienta MagicDraw, y para su desarrollo Python y Django.

Este proyecto es parte de un acuerdo de colaboración entre la *Universidad de Málaga* y la *Empresa Pública para la Gestión del Turismo y del Deporte de Andalucía* perteneciente a la *Consejería de Turismo y Comercio de la Junta de Andalucía*.

Palabras clave: Campaña, promoción, Python, Django.

Abstract: The project presented here has consisted in the execution of a ***web application for campaign management and promotion***, specifically, the analysis, design and implementation of a tool aiming to promote tourism and has been integrated into the web page andalucia.org.

The developed application is structured in two subsystems, one for administration and other for the management of different microsites created for each campaigns within this platform. *Administrator* and *manager* have access to the administration subsystem. The *administrator* provides the permissions at *manager* to manage a campaign. The campaigns consist of a set of elements (templates, documents and items, contest, events and invitations, social networks integration) that conforms the *microsite.* In addition, we have developed a statistics module, integrated with Google Analytics, which provides an analytical information about a campaign for their managers.

Different tools and technologies have been involved in the development of this project. On one hand, MagicDraw tool for the application analysis and design, and on the other hand,,Python and Django for its development.

This project is part of a partnership agreement between the *Universidad de Málaga* and *Empresa Pública para la Gestión del Turismo y del Deporte de*

Andalucía, belonging to *Consejería de Turismo y Comercio de la Junta de Andalucía*.

Keywords: Campaign, promotion, Python, Django.

Índice.

Capítulo 1. Introducción

Actualmente y en la historia reciente, un sector importante dentro de la economía de cada país es el turismo. Cada día la competencia en este sector es mayor, y una herramienta muy importante para ganar afluencia de turistas es la promoción a través de campañas publicitarias.

Con el auge de las nuevas tecnologías y el fácil acceso a la información a través de internet, el modelo de promoción de un producto está cambiando. Cada vez más, el consumidor es el que elige qué información quiere recibir, dónde y cuándo. Por ello, es importante estar presente en diferentes canales que conduzcan al consumidor a tu producto.

Muchas empresas invierten grandes cantidades de dinero en promocionarse a través de internet. Muchos son los sistemas para promocionar un producto: envío masivo de mails, concursos, redes sociales, retransmisión de eventos en directo, microsites, etc. La creación de este tipo de campañas, de promociones muy específicas por internet, consume una gran cantidad de recursos económicos y humanos.

Uno de los grandes problemas, además del coste que supone una campaña para una empresa, es la gestión de la misma. El uso de herramientas externas para el envío de mails, usar las redes sociales para crear concursos, encuestas u otras herramientas que nos puedan servir para promocionar un producto, generan una gran pérdida de tiempo y una gestión tediosa por la descentralización de los datos. Si pudiéramos gestionar todos los datos desde una misma herramienta evitaríamos esa descentralización, teniendo en todo momento control sobre los datos y el seguimiento de una campaña promocional en una misma plataforma y su promoción por diferentes medios de comunicación.

Entre los elementos más importantes de la promoción de un producto nos encontramos con los siguientes:

- *Newsletters*: permiten el envío masivo con información sobre el producto que se quiere promocionar.

- *Teaser*: anticipo a una campaña, en el cual la información está fragmentada sin llegar a desvelar el producto a promocionar, con la finalidad de crear un foco de atención en el cual se crea cierta intriga para los usuarios finales.
- *Microsite*: apartado dentro de una web, que por si mismo tiene entidad de web y que está personalizada para la promoción de un producto
- Difusión por *redes sociales*: extender a través de las redes sociales la promoción de los productos, compartiendo información desde la web promocional a sitios como Facebook, Twitter o Instagram.
- *Concursos*: permiten publicitar el producto y atraer a un mayor número de usuarios.
- *Encuestas*: frecuentemente usadas para hacer llegar el producto al usuario y tener un reporte de su aceptación. También ayudan a mejorar la experiencia del usuario con respecto al producto.
- *Estadísticas*: reportan información sobre el éxito de la campaña y su aceptación.

Plataformas como *Acumbamail* [1] o *Mailchimp* [2]**,** permiten el envío masivo de correo, permitiendo la personalización del newsletter a enviar y proporcionando estadísticas sobre el éxito de la campaña mail. Dichas plataformas disponen de diferentes tipos de suscripciones, algunas de ellas son gratuitas pero con ciertas limitaciones, tanto de información y uso, como en la cantidad de usuarios y envíos permitidos al mes.

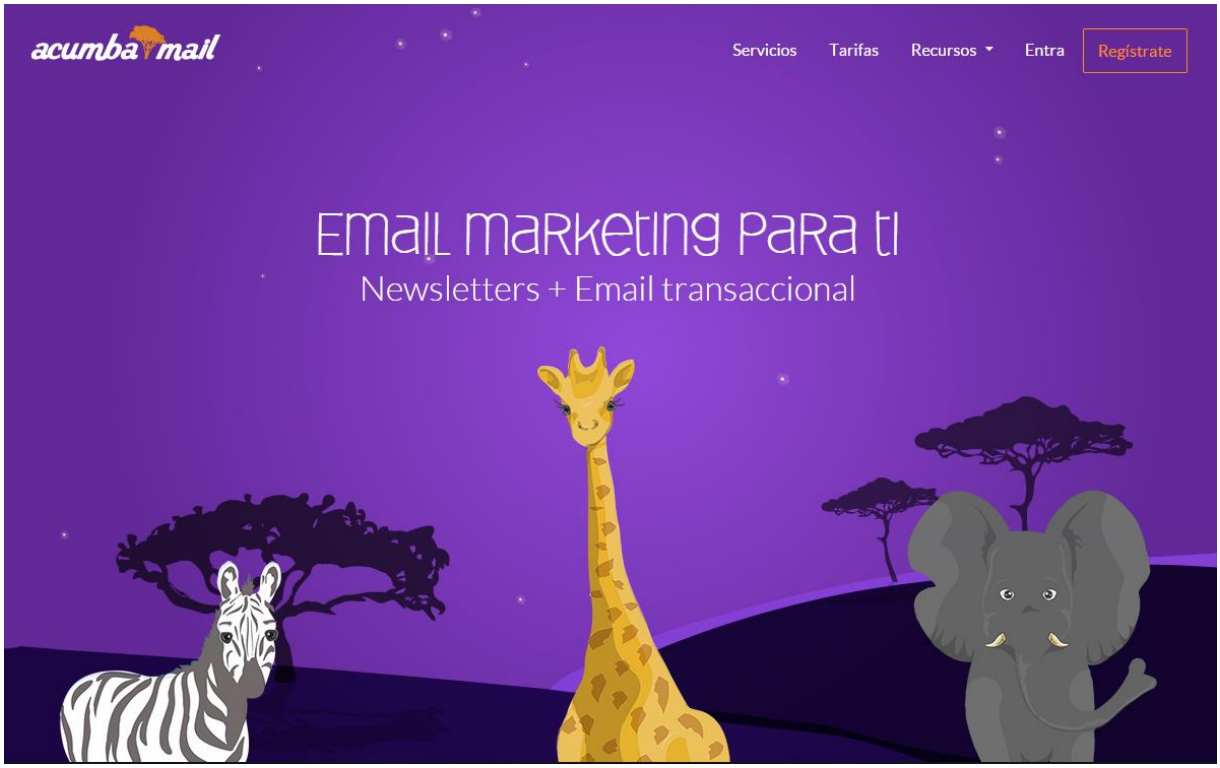

Figura 1.1. Página principal de Acumbamail.

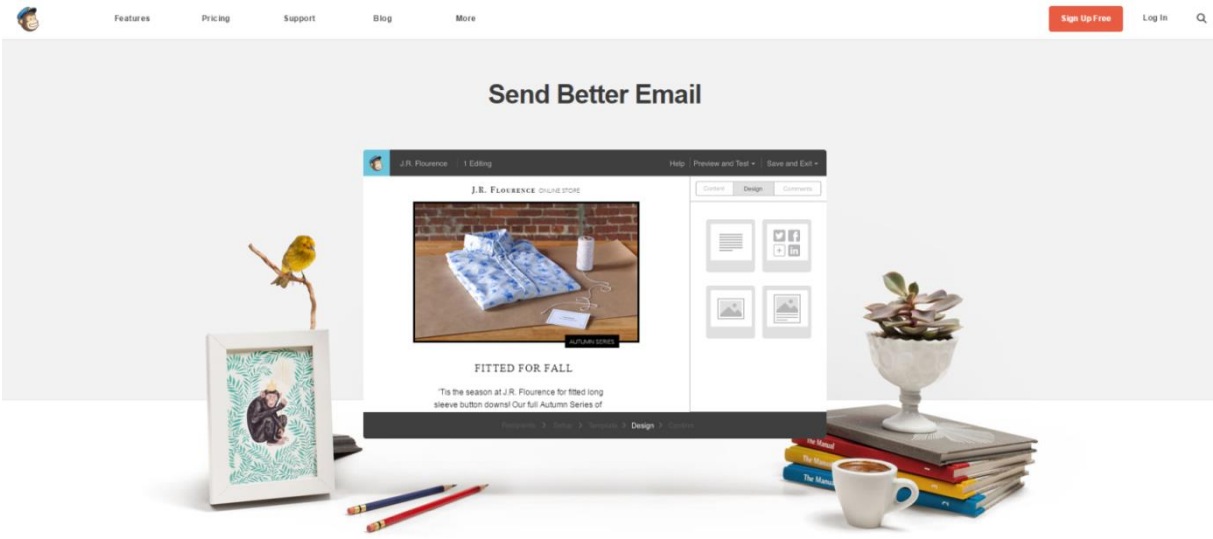

Figura 1.2. Página principal de MailChimp.

Existen también, plataformas como *Woobox* [3]**,** *Bloonder* [4] *o Wildfire by Google* [5] que disponen de un mayor uso de recursos para la promoción de nuestro producto. Estas plataformas permiten la creación de microsites, concursos a medida, personalización de redes sociales, encuestas y otras posibilidades para su interacción ya sea a través de redes sociales como Facebook, a través de blog o páginas web. Estos servicios también ofrecen estadísticas sobre la evolución de la promoción. Aunque algunos aspectos en *Woobox* son gratuitos, estos no cubren todas las necesidades de promoción, personalización e interacción con redes sociales, que en todas las plataformas tienen algún coste en función de la temporalidad y del número de usuarios de nuestras redes sociales.

Figura 1.3. Página principal de Bloonder.

Figura 1.4. Página principal de Woobox.

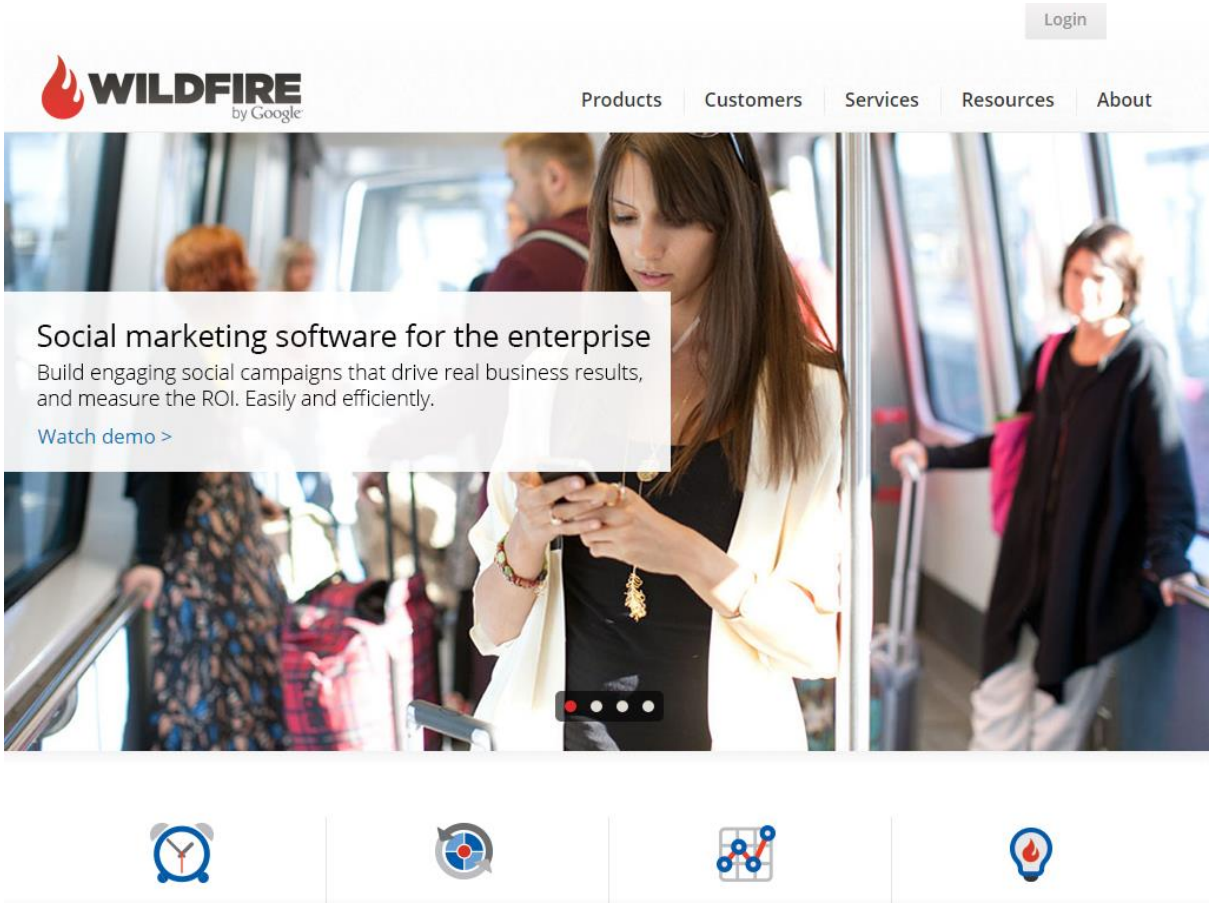

Figura 1.5. Página principal de WildFire by Google.

Las redes sociales juegan un papel fundamental para hacer que una promoción sea viral con menos esfuerzo. Éstas sirven de apoyo como se puede ver

en plataformas como *Woobox* o *Bloonder*. Entre las redes sociales con más influencia y con mayor número de usuarios destacamos *Facebook* [6], *Twitter* [7], *Instagram* [8] *y Youtube* [9]. Cada una de ellas con un cometido principal:

- *Facebook*: es la red social que permite más posibilidades para compartir información en texto, audio y vídeo, tanto de la propia plataforma como de otras diferentes, incluyendo la sincronización de publicaciones con otros medios.
- *Twitter*: enfocado a lanzar pequeñas dosis de información en un texto de 140 caracteres, al cual le podemos adjuntar imagen estática, imagen animada o vídeo.
- *Instagram*: dedicado principalmente a compartir imágenes o fotografías.
- *YouTube*: plataforma de vídeo de Google, que permite subir vídeos, crear tu propio canal de vídeos e incluso la retransmisión en directo de eventos en streaming.

Además de las redes sociales, existen otras plataformas diferentes como *Google Analytics* [10], *Adobe Analytics* [11] o *KISSmetrics* [12] que también sirven de apoyo a la gestión de la promoción de un producto, con cuya integración se pueden obtener estadísticas sobre nuestro sitio web, número de visitas, tiempo del usuario en nuestra web, etc. Entre ellas la opción más popular y económica es Google Analytics. Adobe Analytics ofrece una opción más personalizada pero de mayor coste. Y finalmente KISSmetrics es un caso intermedio entre las dos anteriores, ofreciendo un coste más competitivo en relación a su personalización.

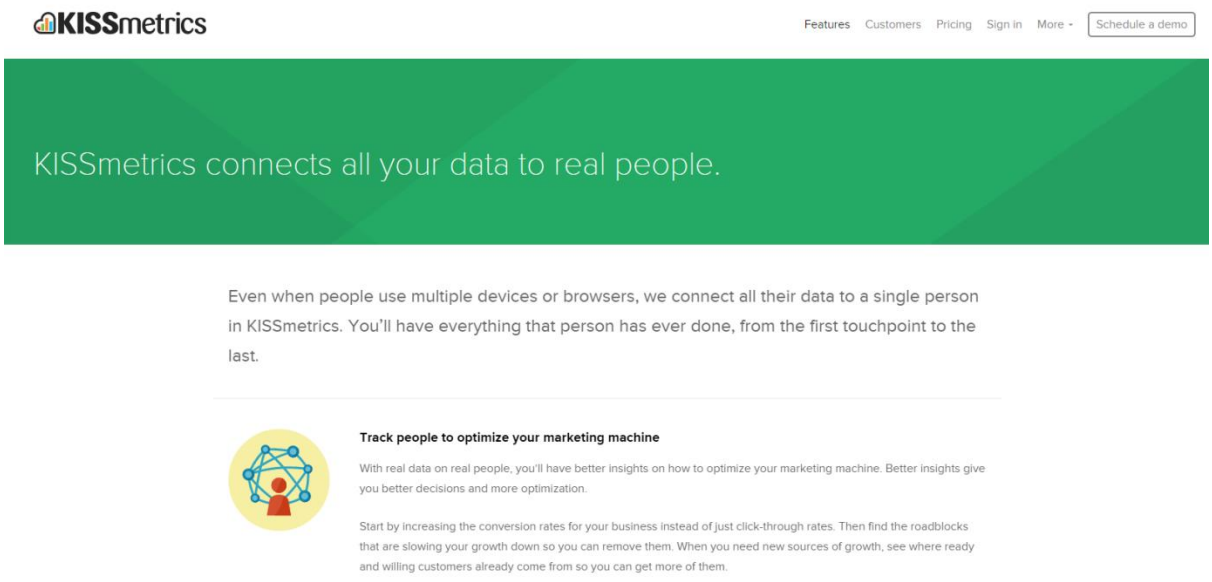

Figura 1.6. Página principal de KISSmetrics.

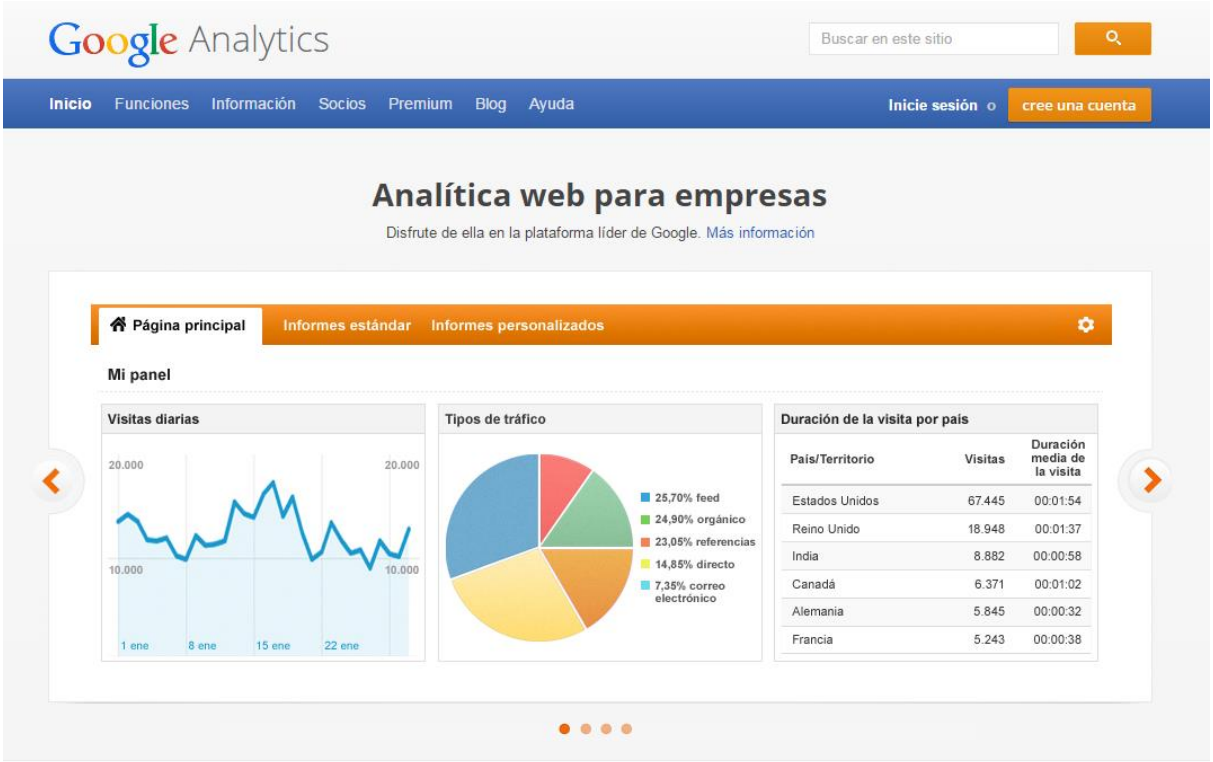

Figura 1.7. Página principal de Google Analytics.

Como se ha expuesto, entre todas las herramientas comentadas, no encontramos ninguna opción que combine los mejores aspectos de todas ellas para ofrecer una solución óptima en la gestión integral de campañas.

1.1. Objetivo del Trabajo Fin de Grado

El objetivo de este proyecto es realizar una aplicación flexible para crear campañas altamente personalizadas y con una gran cantidad de herramientas que permitan gestionar y agilizar la promoción de un producto. Todo ello con un ahorro de coste económico y de tiempo en la gestión de la misma.

Además, como objetivos a alto nivel, se pretende que el trabajo fin de grado sirva para profundizar sobre los conocimientos para la promoción digital, recursos necesarios y estrategia comercial. Así como ampliar los conocimientos hasta ahora adquiridos tanto en la formación académica como laboral.

Otro de los objetivos, es la especialización en el desarrollo de aplicaciones web con tecnologías como son Python [13] y Django [14], así como HTML5 [15], CSS3 [16] o JavaScript.

Con todo lo comentado hasta este punto, se busca crear una aplicación web que estará integrada dentro de la plataforma “andalucia.org” de la Empresa Pública para la Gestión del Turismo y del Deporte de Andalucía perteneciente a la Consejería de Turismo y Comercio de la Junta de Andalucía, y permitirá a empresas externas crear campañas promocionales y que éstas mismas sean gestionadas por

dicha empresa pública. Entre las funcionalidades destacables de la aplicación y que son objetivo principal de este proyecto, destacamos:

- Gestión y *administración de campañas*.
- Gestión de *microsite* de una campaña y sus apartados correspondientes.
- Gestión de *teaser* asociados a una campaña.
- Gestión de envío de *newsletters* de la campaña.
- Gestión de *eventos* asociados a una campaña.
- Gestión e integración de *concursos y encuestas.*
- Gestión de *usuarios* asociados a una campaña.
- Integración con *Google Analytics.*
- Integración con *redes sociales*.

1.2. Descripción del sistema

El sistema a implementar es una aplicación que, integrada dentro de la web andalucia.org y gestionada por la Empresa pública para la Gestión del Turismo y del Deporte de Andalucía (TURASA), nace con la idea de homogeneizar dentro de la propia estructura de la empresa, la gestión de las campañas turísticas y a la vez proporcionar una herramienta propia a otras empresas que son licitadas para la gestión de las mismas.

El sistema está desarrollado con Python y Django como principales tecnologías, y también con el uso de HTML5, CSS3, Javascript, Jquery y ZURB Foundation. Es una aplicación cliente-servidor basado en un modelo-vista-controlador.

La aplicación es desplegada en los servidores propios de TURASA, usando un servidor Linux sobre una distribución CentOS [17].

El sistema permite asignar a un usuario o a varios como responsables de una campaña. Dichos usuarios son los responsables de crear, editar y gestionar la campaña.

Una vez creada una campaña, existe la posibilidad de enviar correos electrónicos a los usuarios registrados o no en la plataforma, para atraer tráfico al microsite de la campaña online. Normalmente la campaña comenzará con un teaser para su propia promoción. Se podrán incluir cualquier tipo de elementos multimedia e incluso streaming en directo.

Una vez comenzada una campaña, ésta estará formada por una serie de documentos en el panel de administración, los cuales derivan en las vistas que conforman el microsite generado para la campaña y que son accesibles para todos los usuarios.

Cada documento que genera una vista, dispone de elementos multimedia, personalización de estilos, posibilidad de organizar la visualización de los elementos que lo componen, etc. Existirá también la posibilidad de crear concursos y eventos asociados a las campañas.

Durante el transcurso y el tiempo que esté accesible una campaña, se realizará un seguimiento de los usuarios que acceden e interaccionan durante la evolución de la misma. Dicha información será accesible por el gestor de la campaña en el panel de administración.

Con esta descripción general, el usuario encargado de gestionar una campaña tendrá un sistema mediante el cual gestionar la promoción de un producto de forma específica y personalizada.

El desarrollo de este sistema ha sido implementado haciendo uso de la metodología SCRUM. A continuación se detallan los *sprints* desarrollados, coste en tiempos y una breve descripción de su contenido:

- 1º sprint (3 semanas): *Análisis y Diseño* del sistema. Se han analizado los requisitos del cliente (TURASA), aplicaciones similares del mercado y trabajos que habían sido realizados por otras empresas para realizar las campañas de promoción turística de Andalucía. Posteriormente se han creado y diseñado los diferentes diagramas UML para el desarrollo del sistema.
- 2º sprint (3 semanas): *Desarrollo*. Se han implementado las clases y métodos necesarios para crear el núcleo principal de la aplicación, la gestión de *microsite* y *teaser*.
- 3º sprint (3 semanas): Continuación del desarrollo. En esta iteración se ha implementado la gestión de *concursos* y *encuestas*.
- 4º sprint (3 semanas): Continuación del desarrollo. En esta iteración se ha implementado la gestión de *usuarios*, *eventos* y envío de *newsletters*.
- 5º sprint (2 semanas): Continuación del desarrollo. En esta iteración se ha implementado la integración con *Google Analytics* y *redes sociales*.
- 6º sprint (2 semanas): *Pruebas* y corrección de errores. Se ha procedido realizar pruebas y correcciones generales.
- 7º sprint (1 semana): *Despliegue* del sistema y puesta en producción. Se ha generado la documentación necesaria para su puesta en marcha en los servidores de TURASA y su integración con el portal “andalucia.org”.

Figura 1.8. Página principal de andalucia.org.

1.3. Organización del Trabajo Fin de Grado

La memoria del presente trabajo fin de grado se divide en una serie de capítulos que se comentan a continuación.

En el capítulo 2 (*tecnologías y herramientas utilizadas*), se profundiza en los principales medios utilizados para el desarrollo del proyecto.

En el capítulo 3 (*análisis, diseño e implementación de la aplicación*), se da una descripción exhaustiva de la aplicación, delimitando el alcance del proyecto y detallando el diseño (casos de uso y diagramas UML) e implementación del mismo.

En el capítulo 4 (*conclusiones y futuros trabajos*), se presentan las conclusiones sobre la realización de la aplicación, comentando en qué grado se han conseguido los objetivos iniciales y mostrando posibles líneas de futuro para la expansión de la aplicación.

En los *apéndices* se incluyen documentos adicionales acerca de la instalación y configuración de la aplicación en el entorno de producción, así como el manual de usuario para su manejo.

Por último, en la *bibliografía* se citan las fuentes de información consultadas durante la realización del proyecto.

Capítulo 2. Tecnologías y herramientas utilizadas

En este capítulo daremos una visión general de las distintas tecnologías y herramientas empleadas en este proyecto.

La principal herramienta para el desarrollo de este proyecto ha sido el framework Django [14], un framework en Python [13] para el desarrollo de aplicaciones web. Para complementar el desarrollo con esta herramienta hemos usado las tecnologías HTML5 [15], Javascript [18], Jquery [19], CSS3 [16] y Foundation5 [20] para el desarrollo del frontend y PostgreSQL [21] como base de datos. Además, para llevar a cabo las diferentes tareas asociadas al proceso de desarrollo de software se han utilizado las siguientes herramientas complementarias:

- Para agilizar su desarrollo hemos utilizado el editor Pycharm [22] creado por JetBRAINS. Pycharm es nuestro editor de código para Python, HTML5, Javascript y CSS3.
- Para interactuar con la base de datos se ha optado por pgAdmin [23], un gestor de base de datos para PostgreSQL.
- Para la gestión del código y su versionado se ha utilizado Mercurial y como herramientas visual TortoiseHg [24].
- La aplicación ha sido modelada previamente con el lenguaje UML [25] usando la herramienta MagicDraw [26].
- Durante su desarrollo se ha aplicado la metodología ágil SCRUM [27] y como herramienta para el control de las tareas Taiga [28].
- Para la puesta en producción del producto se ha utilizado el servidor de aplicaciones Nginx [29].

2.1. Python

Python [13] es un lenguaje de programación interpretado, con una sintaxis limpia, elegante y con un tipado dinámico. Este lenguaje es multiplataforma y soporta la programación orientada a objetos, la programación imperativa y la programación funcional.

Este lenguaje de programación fue creado por Guido van Rossum a finales de los años ochenta y lo llamó así por los cómicos británicos Monty Python.

Figura 2.1. Logotipo de Python.

Aunque ya hemos nombrado algunas de sus características, entre las destacables cabe mencionar:

- Simple: el lenguaje y la sintaxis que usa Python es sencilla, lo que permite centrarse más en la solución del problema que en la sintaxis del lenguaje.
- Fácil de aprender: al disponer de una sintaxis muy simple, es muy sencillo de aprender.
- Libre y de código abierto: Python es un claro ejemplo de FLOSS (*Free and Open Source Software*), gracias a la disponibilidad de su código fuente, se permite su estudio y modificación con la idea de que cualquier persona pueda mejorar el lenguaje.
- Lenguaje de alto nivel: Evita tener que manejar elementos de bajo nivel como la reserva de memoria, centrándonos en la solución del problema.
- Portable: Los programas realizados con Python pueden ser exportados a otras plataformas como Windows, Linux o Macintosh, entre las más populares.
- Interpretado: No requiere compilación del programa.
- Orientado a objetos: permite la programación orientada a objetos.
- Integración: puedes incluir código Python en otros programas no necesariamente del mismo lenguaje.
- Librerías extendidas: Python dispone de una extensa librería con métodos y funciones que abrevian el desarrollo de un programa.

En Python se incluye un modo interactivo o un modo consola donde podemos lanzar el código de Python para poder probar código o realizar operaciones.

La versión de Python escogida para el desarrollo de este proyecto viene impuesta por las especificaciones del propio producto, la versión escogida es la 2.7.

2.2. Django

Django [14] es un framework de código abierto desarrollado en Python para la creación de aplicaciones web.

Django está basado en el patrón de diseño *Model Template View*, algo parecido a lo que conocemos como *Modelo Vista Controlador* y que tiene su equivalencia con respecto a él (ver Figura 2.2 y Figura 2.3).

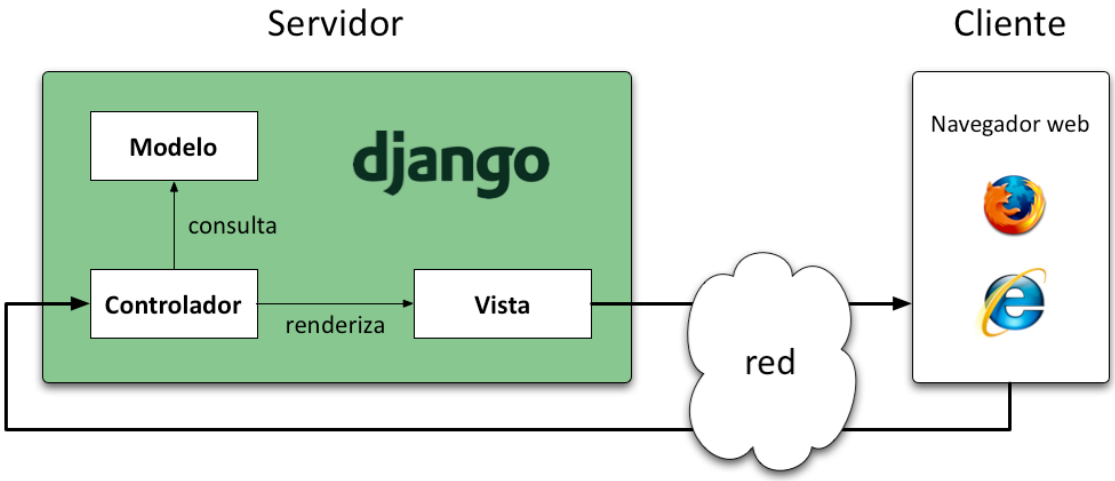

Figura 2.2. Modelo Vista Controlador en Django.

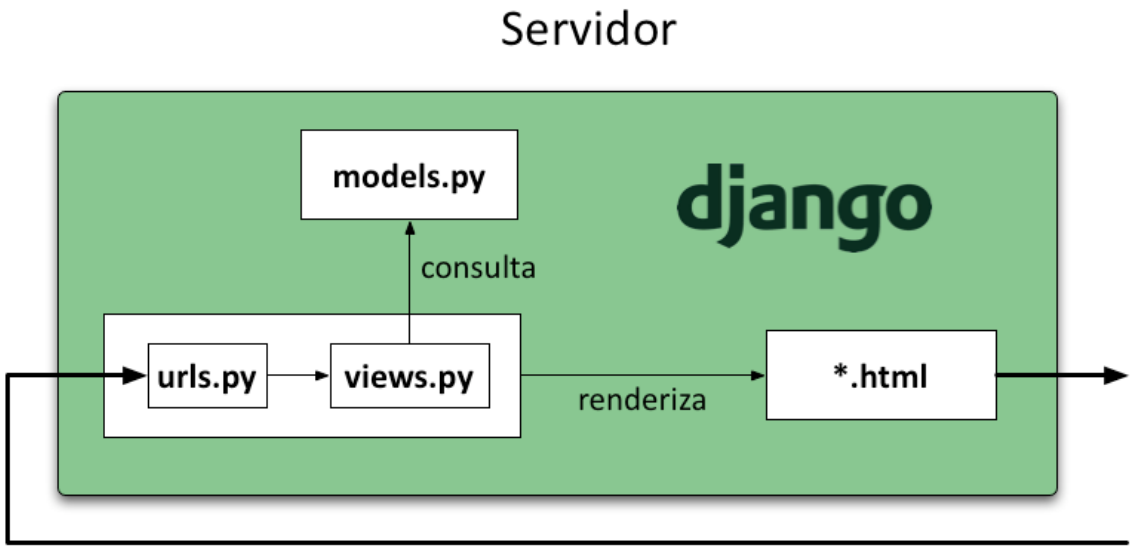

Figura 2.3. Equivalencia entre Modelo Vista Controlador y Template View Model. Donde Template (*.html) es la Vista, View (urls.py y view.py) es el Controlador y Model (models.py) es el Modelo.

Este framework nace en el otoño de 2003 a partir de la creación de varias páginas dedicadas a diarios de noticias de la World Company de Lawrence, Kansas.

La decisión de realizar este framework, surge de la necesidad de Adrian Holovaty y Simon Willison por adaptar sus diarios a continuos cambios en reducidos espacios de tiempo.

En el verano del año 2005, la mayoría de los sitios de World Online estaban desarrollados con Django, y fue entonces cuando el equipo de World Online (Adrian Holovaty, Simon Willison y Jacob Kaplan-Moss) decidió liberar como código abierto el framework. Por lo que en Julio de 2005 fué liberado y llamado Django en honor a el guitarrista de jazz Django Reinjardt.

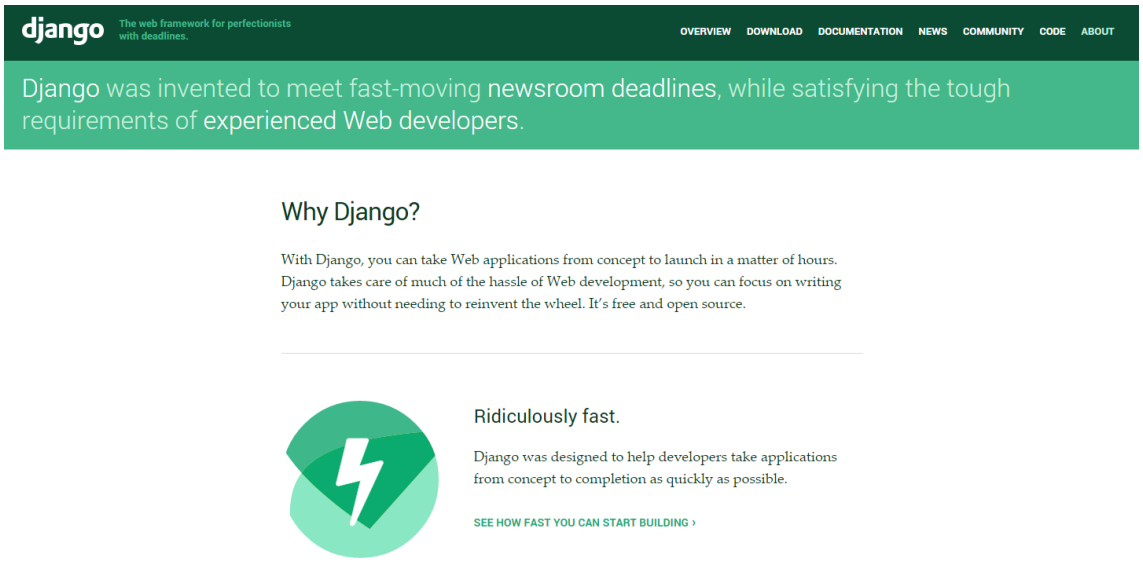

Figura 2.4. Página principal de Django.

Entre las premisas destacables de Django se encuentran:

- Acoplamiento débil. Cada capa es independiente y desconoce completamente a las demás.
- Hacer mucho con poco código.
- Desarrollos rápidos.
- DRY, *Don´t Repeat Yourself.*
- *Explicit is better than implicit.*
- Consistencia.
- Eficiencia, seguridad, flexibilidad y simplicidad.

De las características más importantes que nos ofrece el framework Django, podemos encontrar:

- Una interfaz de administración para administrar los modelos de datos sin necesidad de programar nada.
- API para trabajar con base de datos
- Incorpora un ORM, *Oriented Relational Model* (mapeador objeto-relacional)
- Sistema de plantillas basado en un lenguaje de etiquetas con herencia.
- Un *dispatcher* de URLs basado en expresiones regulares.
- Sistema de middleware para añadir funcionalidades adicionales.
- Soporte multilenguaje
- Documentación extensa.

El gestor de campañas es una aplicación Django, que se acopla dentro el resto de aplicaciones que conforman la plataforma de la Comunidad Turística Andaluza de andalucia.org

2.3. HTML5

HTML5 [15] es la última versión de HTML (Hyper Text Markup Language), en ella se han incluido nuevos elementos, atributos y comportamientos adaptándose a las nuevas tecnologías y las necesidades de los nuevos sitios web.

Figura 2.5. Logotipo de HTML5

HTML5 es el resultado de la colaboración entre el World Wide Web Consortium (W3C) y el Web Hypertext Appliction Technology Working Group (WHATWG).

Entre las nuevas características que aporta HTML5, podemos destacar:

- Elemento Canvas, que permite generar gráficos estáticos y animaciones.
- Elementos de audio y vídeo.
- Almacenamiento de datos “offline”
- Elementos asociados con el contenido: article, footer, header, nav o section.
- Nuevos controles: calendar, date, time, email, url, search, etc.

Aunque HTML5 aún no es un estándar oficial, en general es bien soportado por las últimas versiones de los navegadores más populares como son Firefox, Chrome o Internet Explorer.

HTML5 nos ha ayudado a definir el sistema de *templates* por defecto que contiene la aplicación.

2.4. JavaScript

JavaScript [18] es un lenguaje interpretado. Este tipo de lenguajes son ideales para trabajar en aplicaciones de desarrollo web, ya que los propios navegadores son los que interpretan, y por tanto ejecutan, los programas escritos en dicho lenguaje.

Las dos principales características de JavaScript son, por un lado, que es un lenguaje basado en objetos, y por otro, que es un lenguaje orientado a eventos, debido por supuesto al tipo de entornos de ventanas en los que se utiliza. Esto implica que gran parte de la programación en JavaScript se centra en describir objetos (con sus variables de instancia y métodos de "clase") y escribir funciones que respondan a ciertos eventos (movimientos del ratón, pulsación de teclas, etc.).

JavaScript no es un lenguaje de propósito general: no permite un control absoluto sobre los recursos del ordenador. Cada programa en JavaScript sólo tiene acceso al documento HTML en el que va inmerso. En la Figura 2.6 podemos ver un ejemplo de una función en JavaScript.

```
<script>


//gup : el parametro name indica el nombre del atributo del que queremos conseguir el valor, si no existe devuelve vacio

function gup( name )
{

  name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
  var regexS = "[\\?&]"+name+"=([^&#]*)";
  var regex = new RegExp( regexS );
  var results = regex.exec( window.location.href );
  if( results == null )
    return "";
  else
    return results[1];
}
</script>
```

Figura 2.6. Código de una función en JavaScript.

2.5. CSS3

CSS3 [16] es la última versión de CSS. Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal que es usado para definir el diseño de presentación de una aplicación o de un documento estructurado, escrito en HTML o XML (y por extensión en XHTML). El encargado de formular la especificación de las hojas de estilo es el W3C (World Wide Web Consortium), que sirve de estándar para los agentes de usuario o navegadores. La idea que hay detrás del desarrollo de un CSS es separar la estructura de una aplicación de su diseño.

La manera tradicional de emplear las hojas de estilo en cascada (CSS) con una página HTML tiene 3 diversas formas:

- CSS *in-line*. El diseño se incrusta en el código de la aplicación, como una propiedad más.
- CSS *interna/embebida*. Se integra dentro de la aplicación, aunque en un lugar propio.
- CSS *externa/ligada*. Se trata de un fichero externo a la aplicación que lo contiene, separando completamente el diseño de la implementación.

2.6. Foundation

Foundation [20] es un framework en el cual tenemos una colección de herramientas creadas por ZURB, para crear sitios y aplicaciones web de forma más rápida y eficiente.

ZURB consigue hacer con su framework que tu aplicación web se adapte al dispositivo en el que se carga o muestra, haciéndola “responsive”. Para ello hace uso de plantillas HTML, clases CSS, formularios, botones, navegación y otros componentes que debemos de usar en nuestro sitio web. También hace uso de algunas extensiones JavaScript como es JQuery.

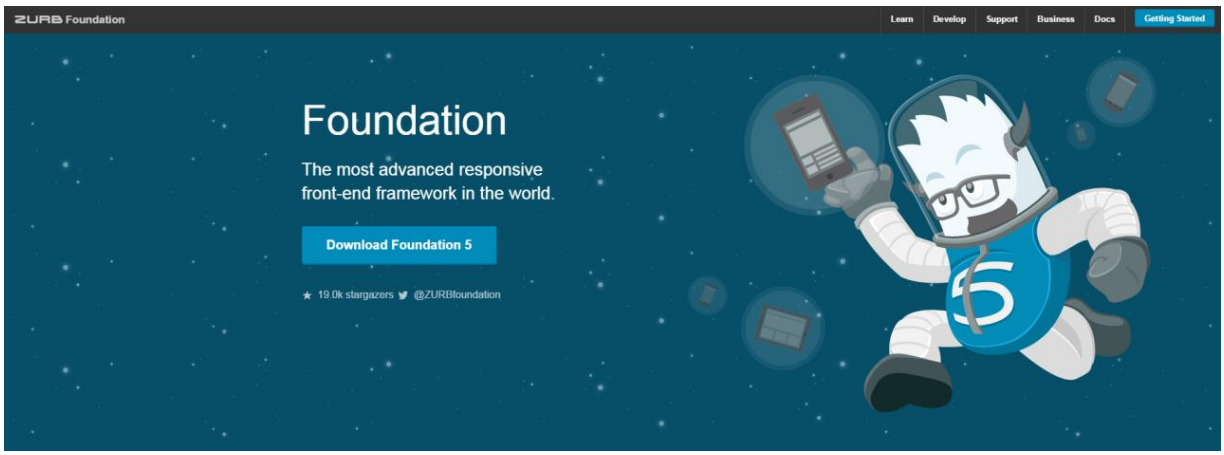

Figura 2.7. Página principal de ZURB Foundation

Entre sus características generales, destacamos:

- Comportamiento estable.
- Mobile-first. La filosofia es diseñar primero para dispositivos móviles.
- Creado con Sass (*Syntactically Awesome Style Sheets*) [30].
- OOCSS (Object Oriented CSS).
- Pocas dependencias de terceros.
- Open-Source.
- Con plantillas *ready to use*.
- Bien documentado.
- Comunidad extensa.

2.7. PostgreSQL

PostgreSQL [21] es un sistema gestor de base de datos relacional orientado a objetos bajo licencia libre BSD.

Este proyecto es llevado a cabo por la comunidad de PGDG (PostgreSQL Global Development Group).

PostgreSQL usa un modelo cliente/servidor y multiprocesos para garantizar la estabilidad del sistema, por lo cual un fallo en uno de sus procesos no afecta al resto y el sistema continúa funcionando (ver Figura 2.8).

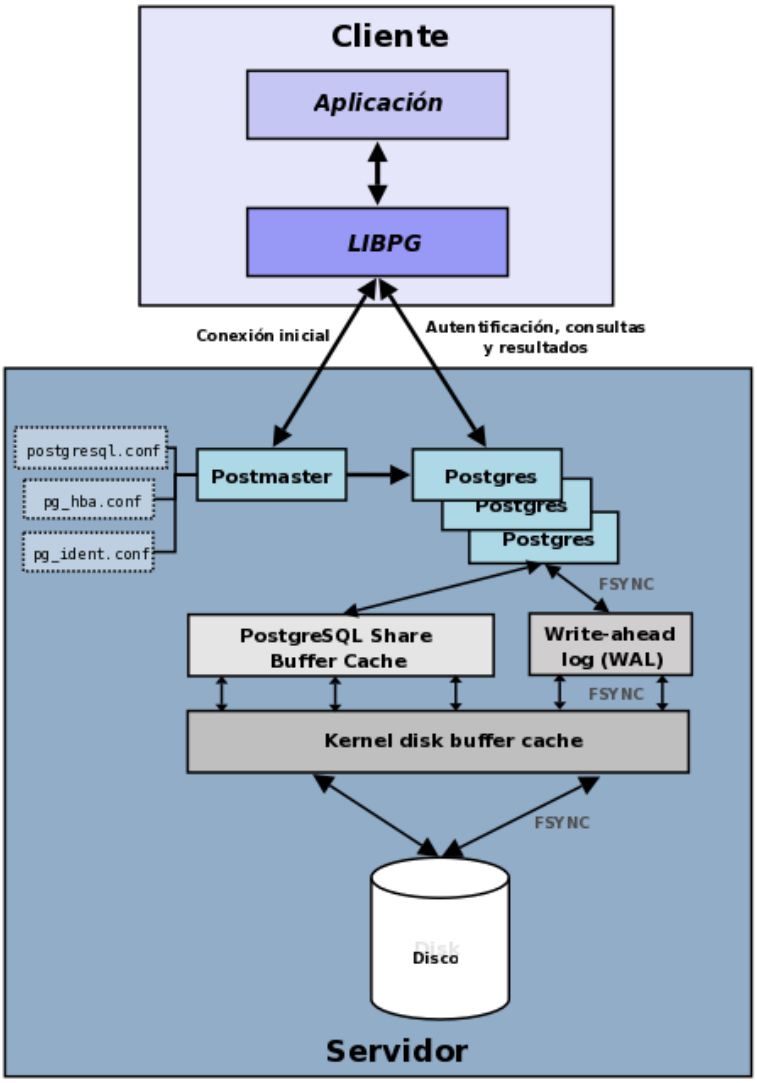

Figura 2.8. Componentes más importantes de un sistema PostgreSQL

Entre las características que podemos destacar de PostgreSQL nos encontramos con:

- Base de datos ACID (*Atomicity, Consistency, Isolation and Durability*).
- Integridad referencial
- *Tablespaces*
- *Nested transactions (savepoints)*
- Replicación asincrónica/sincrónica / *Streaming replication - Hot Standby*
- *Two-phase commit*
- PITR - point in time recovery
- Copias de seguridad en caliente (Online/hot backups)
- Unicode
- Juegos de caracteres internacionales
- Regionalización por columna
- *Multi-Version Concurrency Control* (MVCC)
- Múltiples métodos de autentificación

- Acceso encriptado vía SSL
- Actualización in-situ integrada (*pg_upgrade*)
- *SE-postgres*
- Documentación muy completa
- Licencia BSD
- Multiplataforma, disponible entre otras plataformas para Linux, UNIX y Windows.

A pesar de tener algunos limites como los presentados en la Figura 2.9, es una de las mejores apuestas libres del mercado.

Límite	Valor
Máximo tamaño base de dato	Ilimitado (Depende del sistema de almacenamiento)
Máximo tamaño de tabla	32 TB
Máximo tamaño de fila	1.6 TB
Máximo tamaño de campo	1 GB
Máximo numero de filas por tabla	Ilimitado
Máximo numero de columnas por tabla	250 - 1600 (dependiendo del tipo)
Máximo numero de indices por tabla	Ilimitado

Figura 2.9. Límites de PostgreSQL

2.8. pgAdmin

PgAdmin 3 [23], es la última version de pgAdmin. Esta aplicación nos permite gestionar y administrar base de datos PostgreSQL en un entorno gráfico visual. Está disponible de forma gratuita y es código abierto. Además, está disponible para las distintas plataformas como Lonux, Mac OSX o Windows.

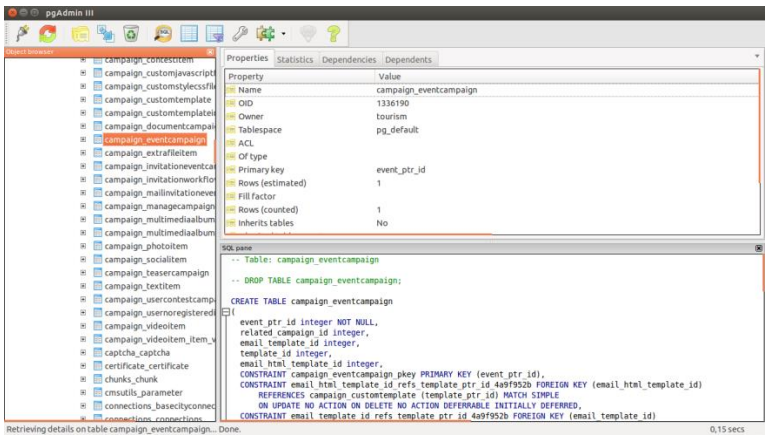

Figura 2.10. Interfaz de pgAdmin 3

PgAdmin permite conectarnos con las bases de datos que se están ejecutando en nuestro sistema. De ésta forma, accede a todas las funcionalidades de la base de datos como son las consultas, manipulación y gestión de datos, pudiendo incluso realizar otras operaciones más avanzadas.

2.9. Nginx

Nginx [29] es un servidor web (ver arquitectura en la Figura 2.11) que también hace de proxy inverso, y como proxy para protocolos de correo electrónico.

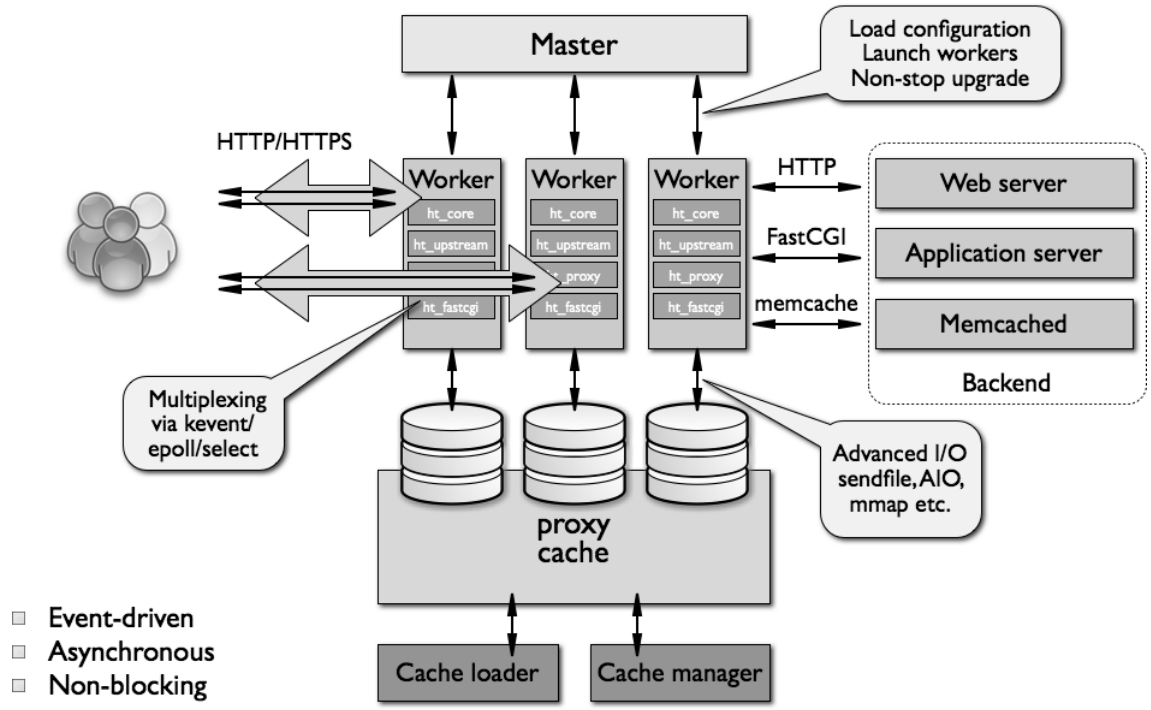

Figura 2.11. Diagrama de la arquitectura de Nginx

Nginx fue creado por Igor Sysoev y es distribuido bajo licencia BSD. Entre sus características destacamos:

- Servidor de archivos estáticos, índices y autoindexado.
- Proxy inverso con opciones de caché.
- Balanceo de carga
- Tolerancia a fallos
- Soporte de HTTP sobre SSL
- Soporte para FastCGI con opciones de caché.
- Servidores virtuales basados en nombre y/o en dirección IP
- Streaming de archivos FLV y MP4.
- Soporte para autenticación
- Compatible con IPv6
- Soporte para protocolo SPDY
- Compresión gzip
- Habilitado para soportar más de 10.000 conexiones simultáneas.
- Proxy SMTP, POP3 e IMAP
- Soporta STARTTLS
- Soporta SSL.

2.10. Pycharm

Pycharm [22] es un IDE desarrollado por la compañía JetBrains, y está basado en IntelliJ IDEA, un IDE que la compañía tiene enfocado a JAVA y que también es base para Android Studio, entre otras herramientas para desarrollo.

Pycharm es un IDE pesado y cargado de herramientas y utilidades, lo cual facilita las labores de programación para un desarrollo rápido de las aplicaciones.

Pycharm ha sido ideado para desarrollar aplicaciones principalmente con Python. Entre sus características cabe destacar:

- Autocompletado, resaltado de sintaxis, herramientas de análisis y refactorización
- Integración con frameworks para el desarrollo de aplicaciones con Python: Django, Flask, Pyramid o Web2Py.
- Integración con frameworks JavaScript: JQuery y AngularJS.
- Depurador de Python y JavaScript.
- Integración con lenguajes de plantillas: Django Template, Jinja2 o Mako.
- Soporta las versiones de Python 2.x y 3.x, Pypy, IronPython y Jython.
- Compatible con SQLAlchemy(ORM), Google App Engine y Cython.
- Soporte para un modo VIM
- Soporta sistemas de control de versiones como Git, CVS o Mercurial.

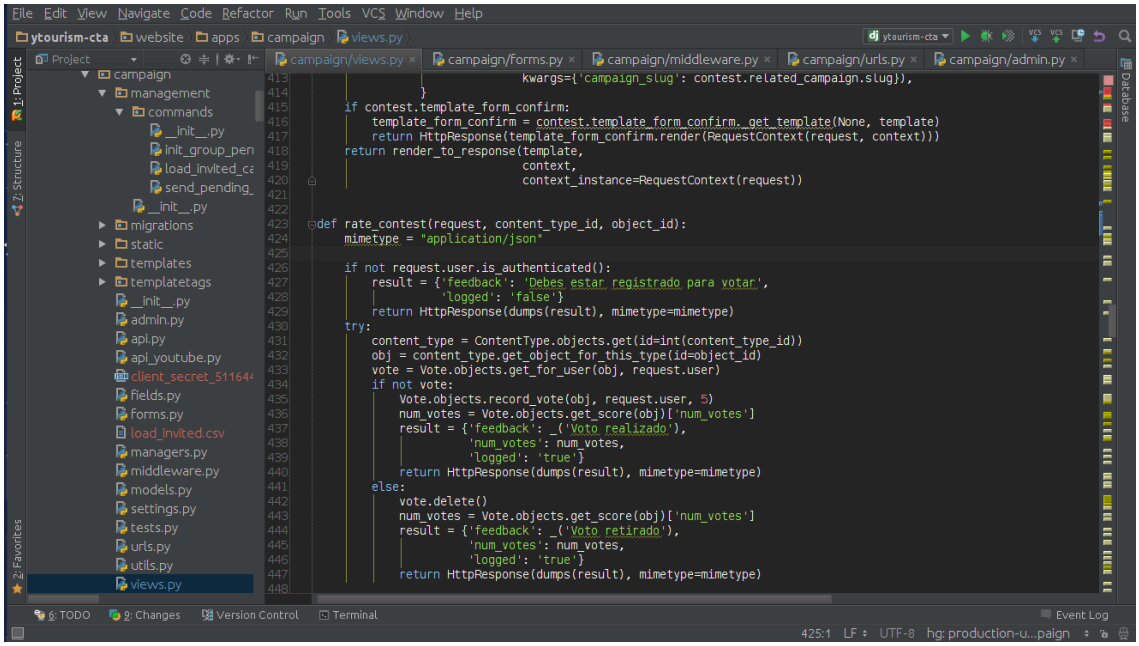

Figura 2.12. Interfaz del IDE Pycharm

Pycharm es un IDE multiplataforma, que lo podemos encontrar para Linux, MacOS X y para Windows. Y para el cual, existen dos versiones una profesional y otra de distribución gratuita para la comunidad.

2.11. Mercurial

Mercurial es un sistema de control de versiones de código. Este sistema es multiplataforma (Linux, Windows y Mac OS X) y ha sido implementado en su mayor parte en Python. Fué creado por Matt Mackall y el código se encuentra disponible bajo licencia GNU GPL 2.

Mercurial nos permite de forma sencilla y mediante línea de comandos, gestionar el versionado de nuestro código. Entre las características de mercurial hay que destacar:

- Sistema distribuido
- Rendimiento y escalabilidad
- Gestión robusta de archivos de texto y binarios
- Opciones avanzadas de ramificación e integración entre ramas.

2.12. TortoiseHg

TortoiseHg [24] es un cliente visual para el sistema de control de versiones de Mercurial. Es multiplataforma y puede ser usado también por línea de comandos.

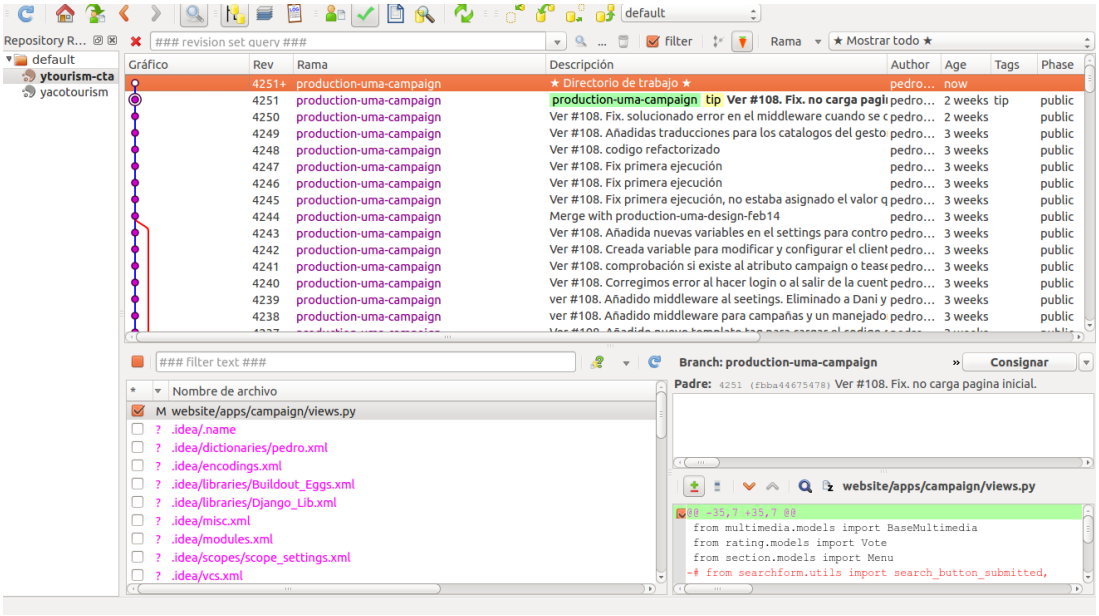

Figura 2.13. Interfaz de TortoiseHg

Entre sus características destacamos:

- Explorador de repositorios.
- Dialogo para realizar *commit*.
- Soporte para las herramientas visuales de *diff/merge*.
- Sincronización con otros repositorios del mismo proyecto.
- Interfaz gráfica intuitiva para manejar Mercurial.

2.13. UML

UML (Unified Modeling Language) [25] es un lenguaje para el modelado de sistemas software. Dicho lenguaje es un estándar que nos permite de forma gráfica definir un sistema, detallar sus aspectos, y documentar y construir el mismo.

Podemos diferenciar entre tres tipos de diagramas UML: de *estructura*, de *comportamiento* y de *interacción*. Estos tipos de diagramas muestran diferentes aspectos del sistema representado.

- Diagramas de *estructura*:
  - Diagrama de clases
  - Diagrama de objetos
  - Diagrama de componentes
  - Diagrama de estructura compuesta
  - Diagrama de paquetes
  - Diagrama de despliegue
- Diagramas de *comportamiento*:
  - Diagrama de casos de uso
  - Diagrama de actividades
  - Diagrama de estado
- Diagramas de *interacción*:
  - Diagrama de secuencia
  - Diagrama de colaboración/Diagrama de comunicación
  - Diagrama de tiempo

En este proyecto nos hemos ayudado de UML para especificar la aplicación que hemos creado.

2.14. MagicDraw

MagicDraw [26] es una herramienta gráfica para el diseño y análisis de UML, que se utiliza para el modelado, documentación, construcción y mantenimiento de sistemas software orientados a objetos.

Gracias a esta herramienta es posible realizar diagramas de casos de uso, diagramas de interacción, de clases, de estado, de despliegue, etc., de forma rápida y sencilla.

En este proyecto se ha usado para realizar el análisis y el diseño UML y también para su correspondiente documentación.

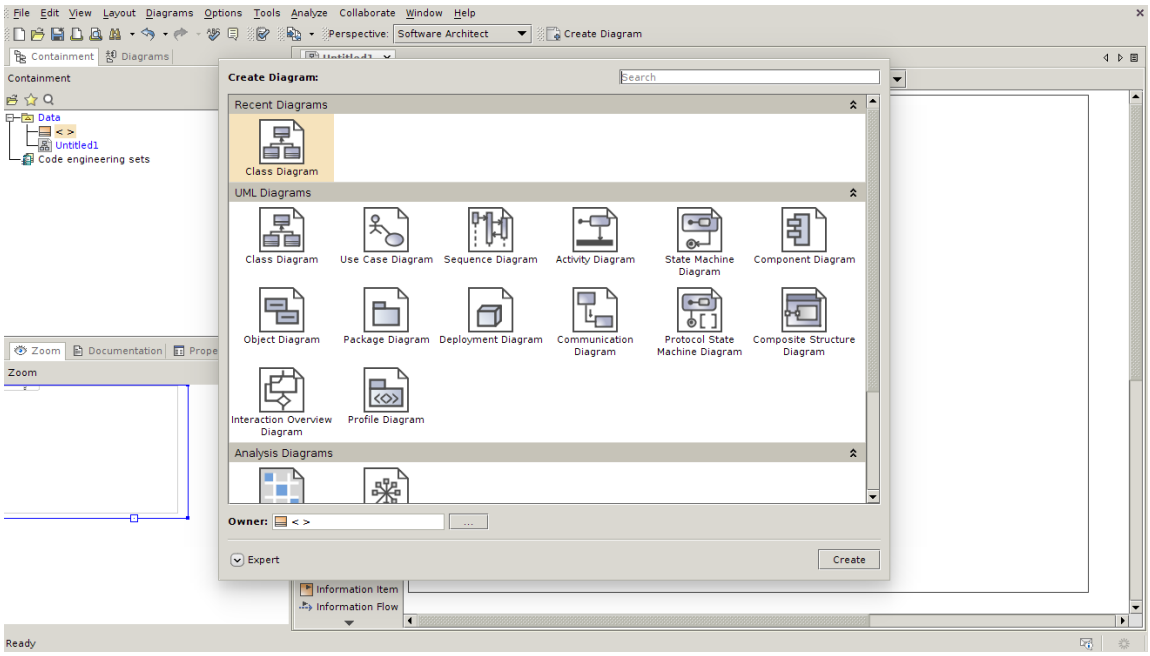

Figura 2.14. Interfaz de MagicDraw para crear diagramas UML.

2.15. Taiga

Taiga [28] es una herramienta que nos permite gestionar el seguimiento de nuestro producto software bajo una metodología ágil SCRUM y KANBAN.

Este gestor de proyectos ha sido desarrollado con Django y Python para la parte *backend* y con AngularJS para el *frontend*.

Es un producto de código abierto con repositorio en Github, con lo cual nos ofrece la posibilidad de instalarlo en nuestros propios servidores.

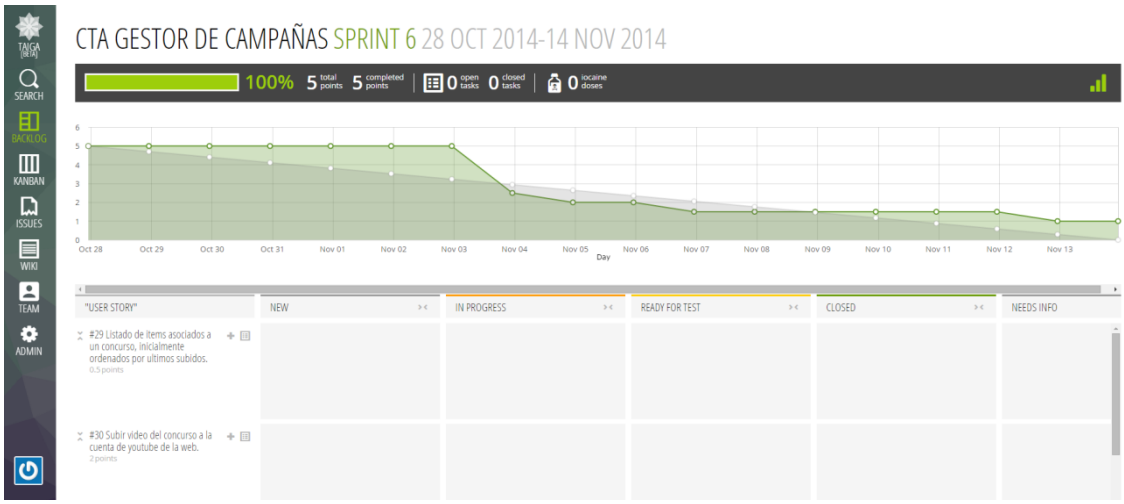

Figura 2.15. Gráfica y pizarra de tareas de un sprint en Taiga

Capítulo 3. Análisis, Diseño e Implementación de la Aplicación

El diseño de una aplicación es un paso fundamental que permite que el tiempo de implementación y verificación de la misma disminuya. Un buen diseño asegura el desarrollo de un sistema estable, robusto y eficiente, fácil de comprender, modificar y ampliar.

En este capítulo se pretenden explicar las diferentes decisiones que han sido tomadas en el diseño e implementación de la aplicación, y de los requisitos que han conducido a tomar estas decisiones. En primer lugar se describe el funcionamiento general del sistema y los requisitos que especifican el comportamiento deseado. En segundo lugar se explican las decisiones tomadas en el diseño y que dan base al correcto funcionamiento de la aplicación. Y finalmente se describen algunos aspectos destacados de la implementación de la aplicación.

3.1. Requisitos del sistema

A lo largo de este apartado se hace un análisis exhaustivo de los requisitos de la aplicación, describiendo todas las posibles funcionalidades que presentará, la utilidad de las mismas y los resultados esperados tras su realización. Todo ello está acompañado de los *diagramas de casos de uso* que facilitan la elaboración del producto software.

Para este cometido la aplicación se divide en diferentes escenarios posibles. Cada uno de ellos se corresponde con la interacción de los diferentes elementos o componentes del sistema y los posibles perfiles de usuarios: *administrador*, *gestor*, *usuario registrado y usuario no registrado.*

En la sección 3.1.1 se hace una introducción sobre el funcionamiento general del sistema para tener una idea de las necesidades del mismo, y en la sección 3.2 se habla de la gestión del sistema y de los distintos elementos con los que interactúan los perfiles de usuarios del mismo.

3.1.1. Funcionamiento resumido del sistema

Para poder delimitar cuáles son las finalidades del proyecto, es fundamental comprender cuál es el comportamiento básico del sistema *Gestor de Campañas* y los diferentes subsistemas y actores que se pueden dar en él.

El sistema se compone de dos subsistemas principales, la interfaz de administración y el *microsite* de la campaña. Este último encapsula dentro de él las siguientes funcionalidades: *documento*, *menú*, *teaser*, *concursos*, *eventos e invitaciones* y *estadísticas*. Los actores que pueden interaccionar con estos diferentes escenarios son el *administrador*, *gestor*, *usuario registrado* y *usuario no registrado*.

Figura 3.1. Caso de uso: subsistemas administración y microsite.

El subsistema de administración tiene acceso restringido y sólo los usuarios de tipo *administrador* y *gestor* podrán acceder a ella. Dentro del subsistema de administración, el *administrador* es quién concede permisos para la gestión del *microsite* al *gestor*. Tanto *administrador* como *gestor* tienen permisos para gestionar el sistema *Gestor de Campañas*.

Desde el subsistema de administración se puede gestionar el *microsite* que es creado con el fin de realizar una campaña. Dicho *microsite* está compuesto por diferentes elementos, entre los que destacamos: *documentos*, *menú*, *plantillas*, *teaser*, *concurso*, *eventos e invitaciones*. También en el subsistema de administración podremos consultar estadísticas sobre la campaña del *microsite*. Definimos de forma resumida cada uno de los elementos:

- *Documentos*: nos permiten gestionar el contenido a mostrar en una página del *microsite*.
- *Menús*: permiten gestionar menús para acceder a los contenidos generados dentro del *microsite*.
- *Teaser*: es un *documento* especial, acotado en el tiempo, que permite promocionar una campaña de forma previa.
- *Concurso*: permite a los usuarios participar dentro del *microsite* añadiendo texto y elementos multimedia.
- *Eventos e invitaciones*: permite gestionar los eventos que se dan para una campaña dentro del *microsite*, y permiten el envío de invitaciones y difusión de los mismos.
- *Estadísticas*: permite consultar estadísticas sobre las interacciones de los usuarios con el *microsite* y todos sus componentes.

Desde los diferentes elementos que componen el *microsite*, se permite la integración con otras plataformas y redes sociales como YouTube.

El *microsite* creado es accesible por los *usuarios registrados* y los *usuarios no registrados*. Aunque a diferencia de los *usuarios no registrados*, los *usuarios registrados* pueden interaccionar participando en los *concursos*. Tanto *usuarios registrados* como *no registrados* tienen acceso a todos los *documentos* que componen el *microsite*, así como sus respectivos *menús*, *teaser* y *eventos e invitaciones*.

3.2. Casos de uso

En este apartado se van a documentar los casos de uso. Éstos se van a documentar, aportando una descripción textual de cada uno de ellos, y usando diagramas de actividad en los casos de uso más complejos.

3.2.1. Casos de uso del subsistema administración del Gestor de Campañas

El sistema *Gestor de Campañas* dispone de una interfaz de administración que permite gestionar y configurar campañas para la promoción de productos. Esta gestión es llevada a cabo por el *administrador* y el *gestor*.

En la interfaz de administración disponemos de los siguientes elementos:

- *Permisos de gestión de campañas*
- *Plantillas*
- *Microsite*. Este incluye a su vez:
    - *Documentos*
    - *Menú*
    - *Teaser*
    - *Concurso*
    - *Eventos e invitaciones*
    - *Estadísticas*

Permisos de Gestión de Campañas

Los *permisos de campaña*, son gestionados exclusivamente por el *administrador*, que es quién concede los permisos de *gestor* de campañas a un *usuario registrado* en el sistema.

Un *gestor* podrá gestionar más de una campaña, pero sólo aquellas campañas a las que se les haya concedido permisos.

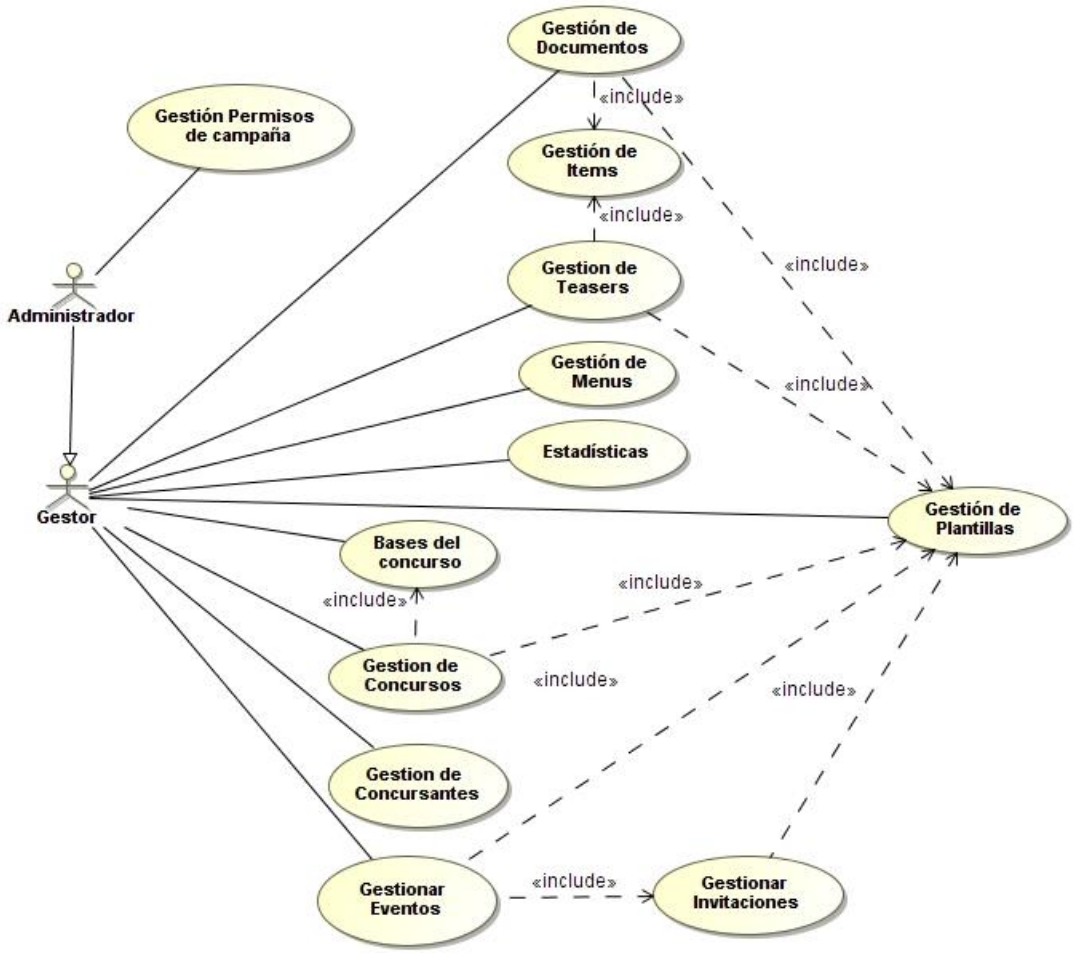

Figura 3.2. Caso de uso: administrador y gestor.

Plantillas

Para poder personalizar al máximo una campaña, las *plantillas* nos permiten definir plantillas en HTML usando el mismo lenguaje que Django usa para su sistema de plantillas.

Para complementar ésta personalización también se permite el uso y la adición de elementos que complementen al HTML, como son el CSS, imágenes y ficheros JavaScript.

Microsite y documentos

El elemento central de la campaña es el *microsite*. El *microsite* está compuesto por el conjunto de *documentos*, *plantillas*, *menús*, *concursos*, *teasers* y *eventos e invitaciones*. La configuración de estos, dan como resultado el *microsite*.

El elemento principal de un *microsite* es el *documento*, éste nos permite la gestión de los contenidos de una página del *microsite*. Los elementos disponibles para configurar dentro de un *microsite* son:

- Nombre: nombre para identificar el documento.

- Slug: URL asociada al documento.
- Cuerpo: contenido central del documento.
- Contenidos para el posicionamiento: etiquetas meta para título, descripción y palabras clave.
- Contenidos multimedia: fotos, álbumes multimedia y vídeos.
- Buscador: buscador que se alimenta de contenidos del Portal principal sobre el que cuelga el *microsite*.
- Sección relacionada: permite relacionar secciones del *portal* principal con el *microsite*.
- Estados de la publicación: establece el estado y su visibilidad de forma pública. Los tres estados disponibles son: *borrador*, *publicado* y *pendiente*.
- Etiquetas: define etiquetas para identificar al documento.
- Plantilla: permite escoger una *plantilla* de las disponibles para mostrar el contenido.
- Ítems: permite añadir contenido extra al documento. Estos permiten darle versatilidad a un documento pudiendo añadir y configurar tantos como sean necesarios. Entre los ítems disponibles encontramos:
    - *Ítem de texto*: añade contenido de texto al *documento*.
    - *Ítem de concurso*: añade un concurso al *documento*.
    - *Ítem de video*: añade videos al *documento*. El poder añadir varios *ítems de videos*, nos permite la posibilidad de agruparlos y diferenciarlos.
    - *Ítem de foto*: añade una foto al *documento*.
    - *Ítem de audio*: añade contenido de audio al *documento*.
    - *Ítem de fichero extra*: añade cualquier tipo de fichero para su descarga en el portal. Pensado sobre todo para añadir ficheros en formato PDF.
    - *Ítem de álbum multimedia*: añade álbumes multimedia al *documento*. El poder añadir varios ítems de álbum multimedia, nos permite la posibilidad de agruparlos y diferenciarlos.
    - *Ítem de carrusel*: añade un carrusel al *documento*.

En cada *documento* del *microsite*, se gestiona la información y el contenido que aparece en una página del mismo. Cada uno de los *documentos* compone las diferentes páginas que se pueden visualizar dentro del *microsite*.

Menú

Para navegar dentro de una campaña, es fundamental disponer de un menú de navegación, donde configurar la jerarquía de navegación dentro del *microsite*.

En el *microsite* se permite configurar dicha jerarquía de navegación de modo que se distinguen tres tipos de menús:

- *Menú principal*: listado de enlaces para uso como menú principal del *microsite*.
- *Menú secundario*: listado de enlaces del portal, para su uso como menú de menor relevancia que el principal, dentro del *microsite*.
- *Menú recomendamos*: listado de enlaces del portal, para añadir enlaces extras y acceso a contenido adicional.

Dentro de cada enlace de un menú se permite definir si dicho elemento es padre de otro elemento del menú, pudiendo crear una jerarquía en forma de árbol dentro de cada elemento del menú.

Teaser

Teaser es un *documento*, dónde podremos seleccionar un rango de fechas en el cual será accesible. El *slug* de un *teaser* colgará directamente de la URL principal del portal.

Concurso

Un *concurso* estará definido por los siguientes elementos:

- Nombre: nombre asignado al *concurso*.
- Slug: URL para la página del *concurso*.
- Fecha de inicio y fin: Define el rango temporal en el que es posible participar.
- Tipo de concurso: permite definir si el concurso es de vídeo, texto e imagen, pudiendo realizar todas las posibles combinaciones entre estos elementos. Además se permite definir si los vídeos serán publicados en el perfil YouTube del portal. También se permite la opción de poder etiquetar las respuestas por el usuario que concursa.
- Definir etiquetas del formulario y plantillas de forma personalizada: permite definir cada una de las etiquetas que componen el formulario y asignar una plantilla personalizada para mostrar el contenido del *concurso*.
- Personalización de textos de errores, validación y de aceptación del *concurso*.

El *concurso* podrá ser añadido a cualquier *documento*, mostrando el comportamiento de la Figura 3.3

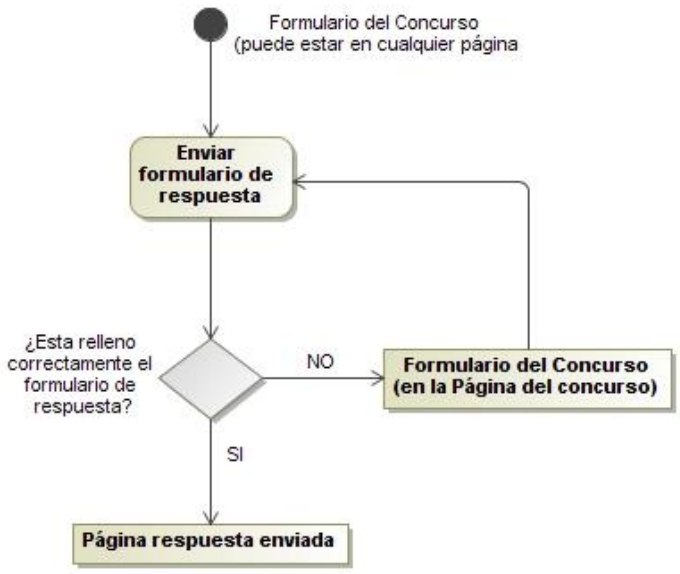

Figura 3.3. Diagrama de flujo par el formulario de un concurso

Eventos e invitaciones

Se pueden definir cuantos *eventos* se precisen dentro de una campaña. Los *eventos* nos ayudan a distinguir momentos especiales dentro de una campaña. Estos *eventos* provienen del portal, pero a ellos se les añade una funcionalidad extra para su inmersión en el *Gestor de Campañas*. Con ello podremos tener *eventos* relacionados con una campaña, con la funcionalidad extra de añadirle una *plantilla* para personalizar su aspecto y además añadir *invitaciones* para difundir su información vía mail tanto a *usuarios registrados* como a *usuarios no registrados*.

Dentro de un evento se pueden generar un listado de *invitaciones*. Cada *invitación* se compone de:

- Datos de la invitación: contiene información que identifica la *invitación*.

- *Usuarios registrados*: dispone de un listado de *usuarios registrados*, los cuales pueden ser seleccionados e invitados.
- *Usuarios no registrados*: se pueden añadir usuarios que no estén registrados en la plataforma, y en el listado creado de *usuarios no registrados*, seleccionar los que se deseen para añadirlos a la *invitación*.
- Usuarios invitados: listado de usuarios (tanto *usuarios registrados* como *no registrados*) invitados al evento. En el listado se registra si el sistema ha enviado el mail al usuario, si este acepta o rechaza la *invitación* y si el usuario añade algún comentario sobre el mismo.
- Flujo de trabajo de la *invitación*: define los pasos que sigue el proceso de invitación. En él se definen los siguientes aspectos:

- Envío de mail: permite activar el envío de mail, para que el propio sistema de envíos de portal lo envíe. También permite enviar la *invitación* a partir de una fecha.
- Mail de invitación: define el mail que se envía para invitar a los usuarios al evento, personalizando el asunto del mail y la *plantilla* del mismo.
- Mail de invitación aceptada: es el mail que se envía a un usuario que ha aceptado la invitación al evento, personalizando el asunto del mail y la *plantilla* del mismo.
- Mail de invitación rechazada: es el mail que se envía a un usuario que ha rechazado la invitación al evento, personalizando el asunto del mail y la *plantilla* del mismo.
- Mail compartir invitación: es el mail enviado a un usuario que no ha recibido invitación, y el cual ha sido invitado por otro que si la ha recibido.
- Página inicial de una invitación: se define con la personalización de una *plantilla* y un *documento* para complementar su información si se requiere. En esta página inicial se selecciona si se acepta o rechaza la invitación o si se comparte.
- Página para añadir un comentario de una invitación: se define con la personalización de una *plantilla* y un *documento* para complementar su información si se requiere. En esta página permite añadir un comentario que es registrado en el sistema para el usuario que lo realiza.
- Página para compartir una invitación: se define con la personalización de una *plantilla* y un *documento* para complementar su información si se requiere. En esta página se permite añadir correos electrónicos de usuarios y enviarle una invitación del evento.
- Página final de una invitación: se define con la personalización de una *plantilla* y un *documento* para complementar su información si se requiere.

Estos elementos que componen el flujo de trabajo se comportan de la siguiente forma:

1. Proceso de *envío del mail de invitación* a los usuarios con enlace a la página principal para aceptar o rechazar la invitación.
2. *Página inicial de la invitación* para aceptar o rechazar la misma. Si la invitación es aceptada por el usuario vamos al *punto 3*, y si la invitación es rechazada vamos al *punto 4*. En dicha página también se puede invitar a otros usuarios, si se pulsa sobre invitar a otros usuarios, vamos al punto *5*.
3. Al *aceptar la invitación*, se envía el mail de aceptación de la invitación al usuario y pasa a la página para añadir un comentario. Una vez añadido, vamos al *punto 6* y final del flujo de trabajo de la invitación.

4. Al *rechazar la invitación*, se envía el mail de rechazo de la invitación y vamos al *punto 6* y final del flujo de trabajo de la invitación.
5. En la página de invitación enviaremos la invitación al evento a otros usuarios y volveremos al *punto 2.*
6. *Página final de la invitación.*

En la Figura 3.4 podemos observar de forma gráfica el flujo de trabajo de una *invitación*.

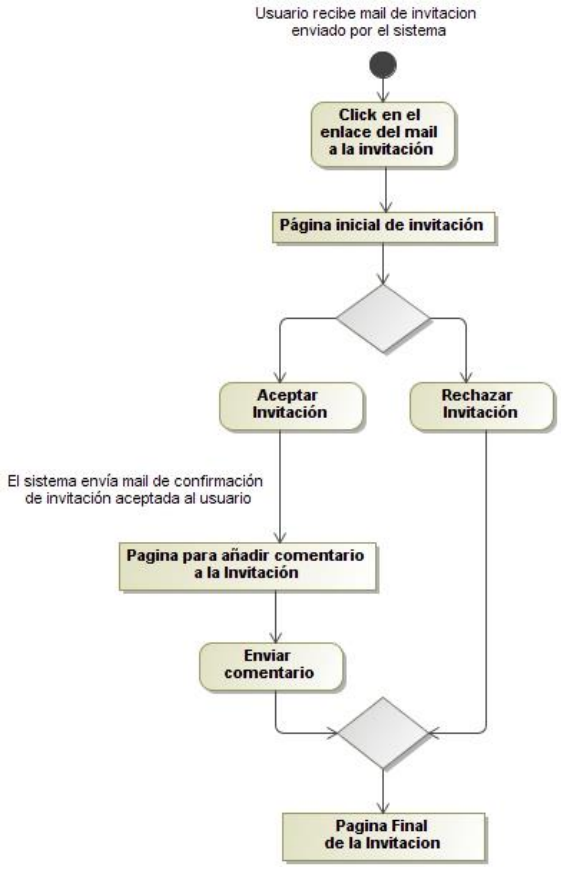

Figura 3.4. Diagrama de flujo de trabajo de una invitación a un evento.

Estadísticas

Se añaden estadísticas recogidas por la plataforma de Google Analytics sobre la campaña. Este información estará definida por el usuario en la plataforma anteriormente comentada.

3.2.2. Casos de uso del subsistema microsite

Las campañas añaden interacción con el usuario en el *microsite* mediante las *invitaciones* y los *concursos*, además del contenido descargable. Todo tipo de usuario podrá navegar a través los contenidos que hayan sido publicados en el *microsite*. Adicionalmente, el *administrador* y el *gestor* podrán visualizar el contenido no publicado dentro del *microsite*. Aunque tanto *usuarios registrados* como *usuarios*

no registrados tienen acceso al *microsite*, sólo los primeros pueden interaccionar con los *concursos*.

Eventos e invitaciones

Los *usuarios registrados* y los *usuarios no registrados* que hayan recibido una invitación, pueden interactuar con el *microsite* realizando las siguientes acciones:

- Aceptar invitación.
- Rechazar invitación.
- Añadir comentario a la invitación.
- Compartir invitación.

Figura 3.5. Caso de uso: Acciones de un usuario con las invitaciones a un evento.

Concursos

Para poder participar en un *concurso* hay que ser un *usuario registrado* en el sistema. El usuario puede realizar las siguientes acciones con respecto a un *concurso*:

- Participar: puede participar rellenando su respuesta, ello puede implicar añadir un texto, foto, video e incluso etiquetar la respuesta.
- Votar respuestas: puede añadir un voto para valorar la respuesta de otros usuarios que han participado en el *concurso*.

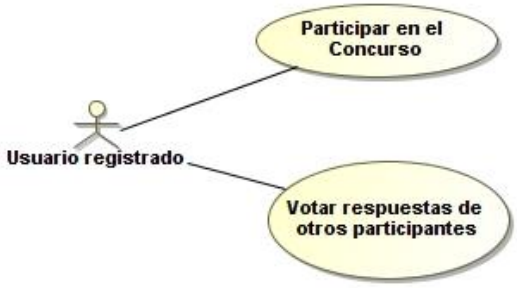

Figura 3.6. Caso de uso: Interacción de un usuario registrado con un concurso.

3.3. Diseño

El proceso de diseño, como se ha mencionado al comienzo de este capítulo, es una de las fases más importantes en el desarrollo de un software. En los siguientes apartados de este punto se van a tratar de explicar las decisiones de diseño más relevantes que se han tomado.

3.3.1. Arquitectura

La arquitectura es una de las decisiones de diseño más importantes. En nuestro caso hemos optado por dotar a la aplicación de una *arquitectura web*, la cual ha sido impuesta por su integración dentro del portal “andalucia.org”.

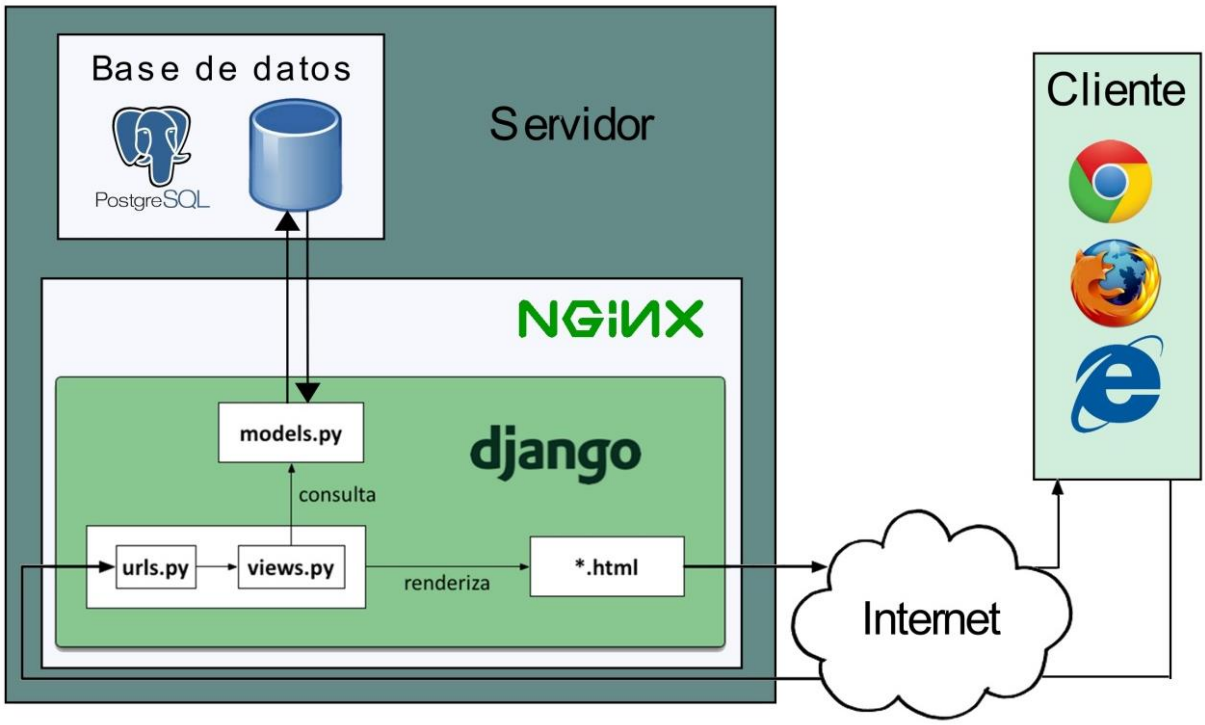

Figura 3.7. Arquitectura cliente-servidor en este proyecto

La arquitectura desarrollada en este trabajo fin de grado está conformada por la siguiente serie de componentes, visibles en la Figura 3.7:

- *Cliente:* aplicación que se ejecuta en un navegador Web y que se encarga de iniciar las peticiones al servidor.
- *Aplicación servidora Gestor de Campañas:* Aplicación desarrollada en Python/Django que es la encargada de realizar las acciones sobre el servidor. El servidor lee las peticiones del cliente, las procesa y envía una respuesta.

- *Sistema gestor de base de datos:* Será el encargado de administrar la información que intercambian cliente y servidor. El servidor hace uso de una base de datos en PostgreSQL para sus funciones.
- *Nginx:* Servidor web que aloja la aplicación servidora *Gestor de Campañas*.
- *HTTP:* Es un protocolo de comunicación que permitirá conectar a *cliente* y *servidor*.

3.3.2. Estructura de la aplicación

A lo largo de este punto se verá cómo se ha descompuesto la aplicación en diferentes clases, viendo las funcionalidades que aporta cada una y las decisiones de diseño tomadas en ellas para su realización.

En la Figura 3.8 se muestra la estructura de directorios del proyecto de la aplicación. A continuación se describe brevemente la estructura de directorios, que está compuesta por:

- *management*: directorio que contiene comandos para el envío de mails y la carga masiva de usuarios no registrados para el envío de invitaciones.
- *migrations*: directorio que contiene las migraciones realizadas en la base de datos para su despliegue.
- *static*: directorio que contiene ficheros estáticos para el uso de la aplicación.
- *templates*: directorio que contiene las plantillas de las vistas que se usan en la aplicación.
- templatetags: directorio que contiene los template_tags implementados en el proyecto. Estos, son métodos que pueden ser usados en las plantillas y que añaden funcionalidad a las mismas para el manejo de datos.
- *admin.py*: contiene las clases que componen la interfaz de administración del *Gestor de Campañas*. Estas permiten personalizar la interfaz por defecto que Django tiene establecida.
- *api_youtube.py*: API que permite la comunicación con la API de YouTube para la subida de vídeos de forma automática desde la aplicación a la plataforma de vídeos.
- *fields.py*: contiene la definición de campos personalizados para su uso en formularios o donde se requieran.
- *forms.py*: contiene la definición de las clases que componen los formularios usados en el proyecto.
- *managers.py*: contiene la declaración de clases para el manejo de conjuntos de datos.
- *middleware.py*: capa de la aplicación que procesa las peticiones recibidas y detecta si pertenecen o no a la aplicación del *Gestor de Campañas* para procesarlas.
- *models.py*: contiene las clases, con sus respectivos atributos y métodos, que componen los modelos de la aplicación.

- *settings.py*: establece los parámetros de configuración de la aplicación.
- *tests.py*: contiene las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación.
- *urls.py*: establece los patrones para definir las URLs de la aplicación y los métodos que cargan las correspondientes vistas.
- *utils.py*: se disponen de métodos que son usados en las diferentes partes de la aplicación.
- *views.py*: contiene los métodos que controlan y cargan las vistas.

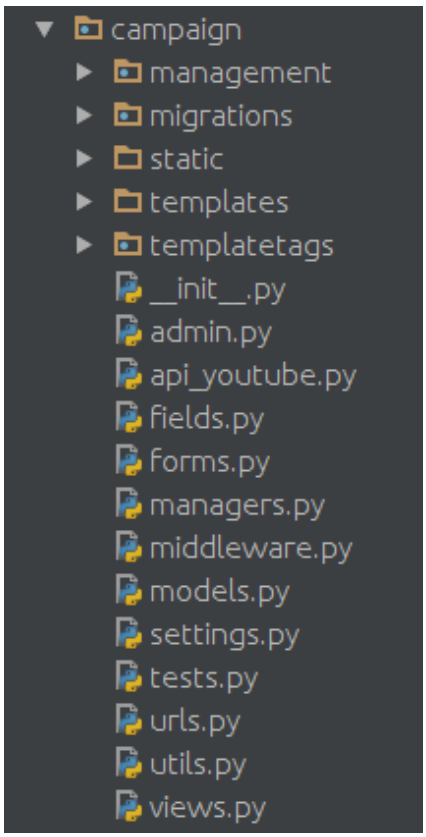

Figura 3.8. Estructura de directorios.

En los siguientes sub-apartados, se explica el diseño realizado en la aplicación, detallando los aspectos más importantes del mismo. Ésta, ha sido implementada como parte del portal de la web de la Comunidad Turística Andaluza "andalucia.org". Por ello, encontramos herencia de clases que pertenecen a otras aplicaciones, así como dependencias con otras aplicaciones externas que han facilitado su diseño e implementación.

Diseño de clases

En la Figura 3.9, Figura 3.10, Figura 3.11, Figura 3.12 y Figura 3.13 podemos ver las diferentes clases que componen el *Gestor de Campañas*. Entre ellas podemos diferenciar:

Campaign: Clase que hereda de la clase *Section* de la aplicación *section* del portal andalucia.org. Esta clase es la que permite crear una campaña y gestionar todo lo que ella conlleva.

DocumentCampaign: Clase que hereda de la clase *Document* del portal andalucia.org. Gestiona el contenido de cada una de las vistas que componen el *microsite* de una campaña. Al resto de atributos para gestionar el contenido que hereda de *Document*, se le añaden el atributo *related_campaign* que lo relaciona con *Campaign* y el atributo custom_template que lo relaciona con CustomTemplate para asignarle una Plantilla personalizada.

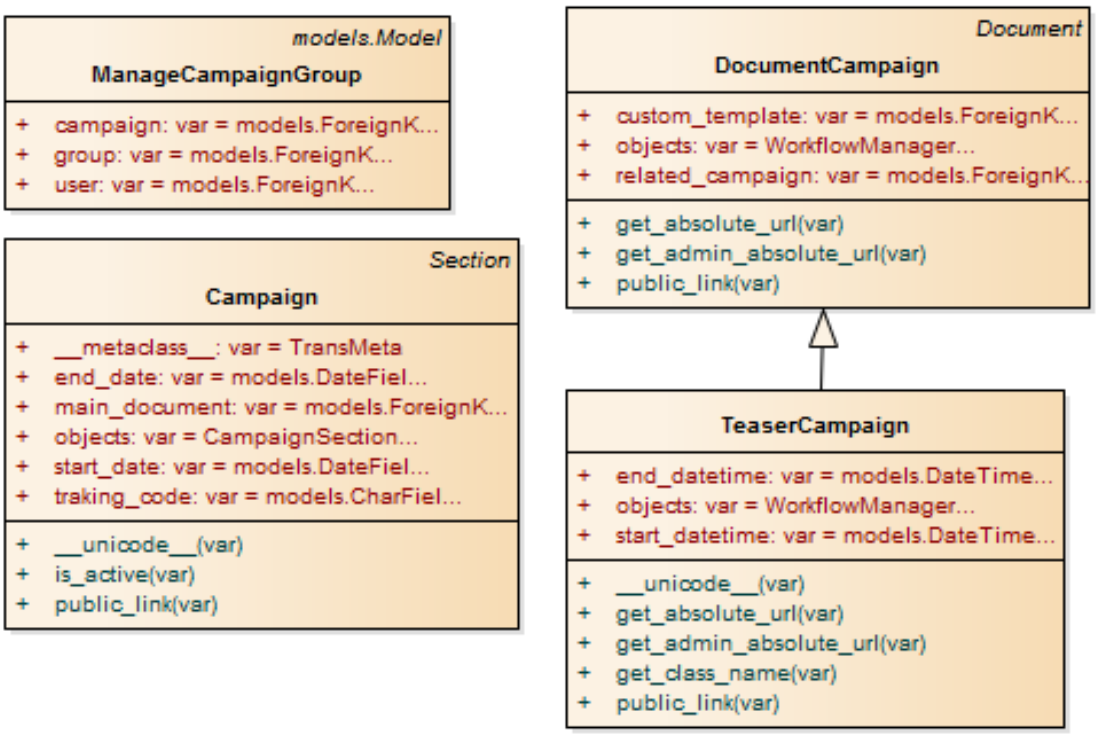

Figura 3.9. Diagrama de clases (1 de 5)

TeaserCampaign: Clase que hereda de *DocumentCampaign* y que está diseñada para añadir *teasers* a una campaña. A este documento especial se le añade una fecha de inicio y de fin (*start_datetime* y *end_datetime* respectivamente).

ManageGroupCampaign: permite relacionar un usuario con el grupo de permisos para poder editar una campaña. Donde el atributo *campaign*, *user* y *group,* son la campaña, usuario y grupo de permisos, respectivamente.

BasesContestCampaign: contiene atributos para definir bases legales para los *concursos*. Permite añadir un fichero descargable con las bases, además de establecer un título y descripción para las mismas.

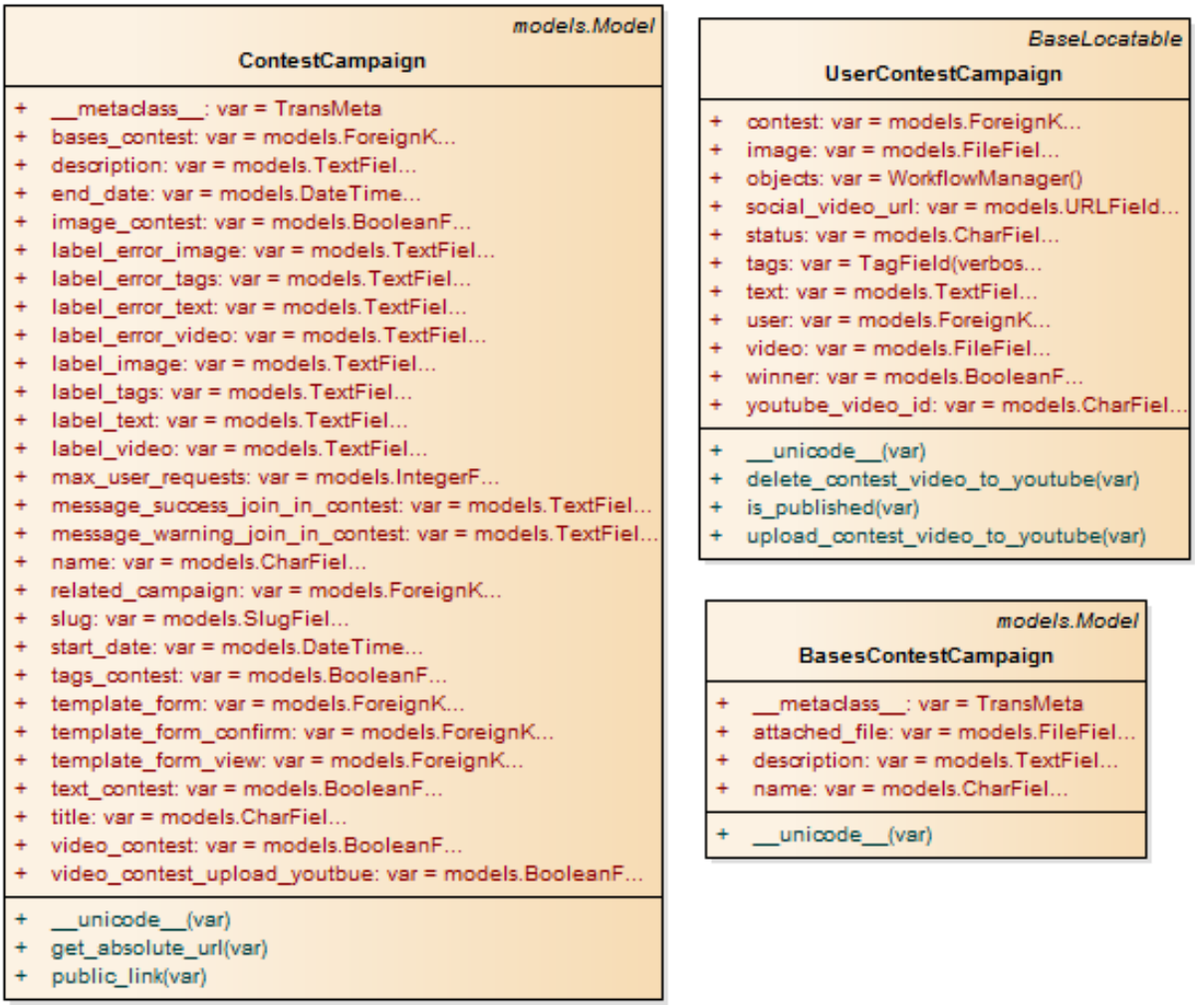

Figura 3.10. Diagrama de clases (2 de 5)

ContestCampaign: define el *concurso* de una campaña. Entre los atributos más destacables se encuentran: *start_date* para marcar el comienzo del *concurso*; *end_date* para marcar el final; *related_campaign* para indicar en que campaña se encuadra; *bases_contest* hace referencia a las bases del concurso *BasesContestCampaign*; *video_contest*, *image_contest* y *text_contest* para indicar si el *concurso* admite respuesta de video, imagen y texto, respectivamente; *video_contest_upload_youtube* es un booleano para indicar si se sube el video a la plataforma YouTube una vez sea publicado; *tags_contest* para indicar si la respuesta ha de ser etiquetada. También contiene atributos para controlar el número de respuestas máximas, el aspecto o el contenido del formulario del *concurso* y sus vistas que hacen referencia a la clase *CustomTemplate*.

UserContestCampaign: hereda de *BaseLocatable*, que permite añadir la localización. Entre sus atributos incluye: *user,* para definir el usuario concursante; *contest*, para indicar a que *concurso* corresponde la respuesta; *video*, *image* y *text* contienen el video, imagen y texto según el *concurso* lo requiera; *youtube_id*, el identificador de YouTube si el video ha sido publicado en la misma; *status*, para

indicar el estado de publicación de la respuesta del *concurso*; *tags*, las etiquetas de que han sido definidas para esa respuesta; y *winner*, que indica si la respuesta es ganadora del *concurso*.

CustomTemplate: esta clase hereda del paquete “django-dbtemplates”, y nos permite añadir código HTML en el atributo *custom_html*. Además, permite añadir estilos personalizados con *custom_css.* Esta *plantilla* podrá estar relacionada o no con una campaña, indicando así su ámbito de disponibilidad para una campaña en exclusiva o para todas.

CustomStyleCssFile, CustomTemplateImage y CustomJavascriptFile: son clases que a través del atributo *custom_template* están relacionadas con una plantilla, y que permiten añadir ficheros de tipo CSS, imagen y JavaScript, para ser usados en su plantilla relacionada.

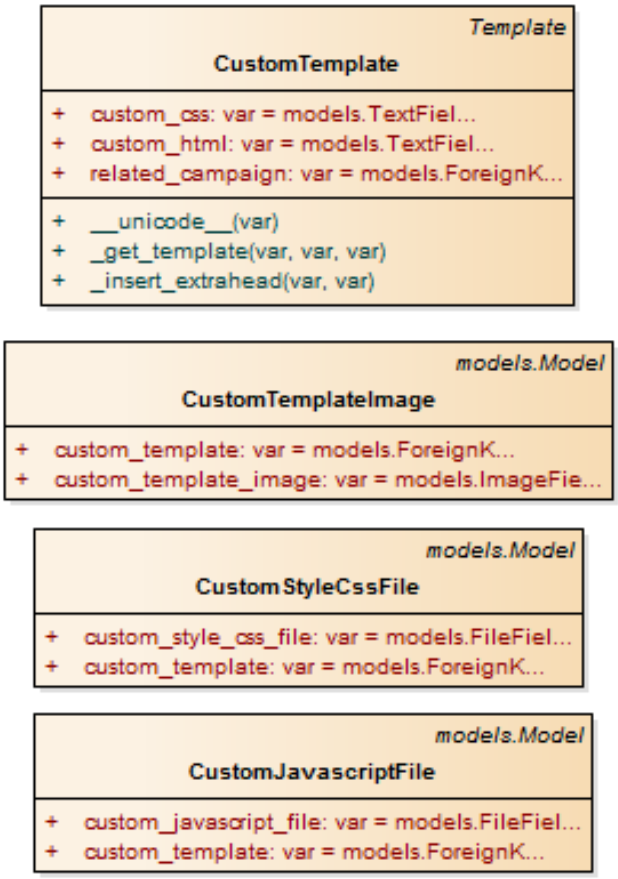

Figura 3.11. Diagrama de clases (3 de 5)

BaseItem: hereda de las clases *RealInstanceMixin* y *LinkProvider*, que nos añaden métodos y atributos requeridos, como son utilidades para las instancias que hereden de *BaseItem* o campos para añadir enlaces a contenidos como podría ser un *DocumentCampaign*. *BaseItem*, es una clase que contiene elementos comunes que define un ítem, entre ellos contiene un atributo *related_document*, con el cual se relaciona un ítem con un *CampaignDocument*, para extender la gestión de sus

contenidos. También contiene los atributos: *order*, para definir como se ordenan los ítems; *status*, para indicar si su contenido está publicado o no; *url*, URL externa para añadir un enlace; *ga_tracker*, para añadir una etiqueta que identifique el ítem en Google Analytics; *title_link*, titulo para identificar la URL externa; y *alternate_text*, que está definido para ser usado como un texto alternativo.

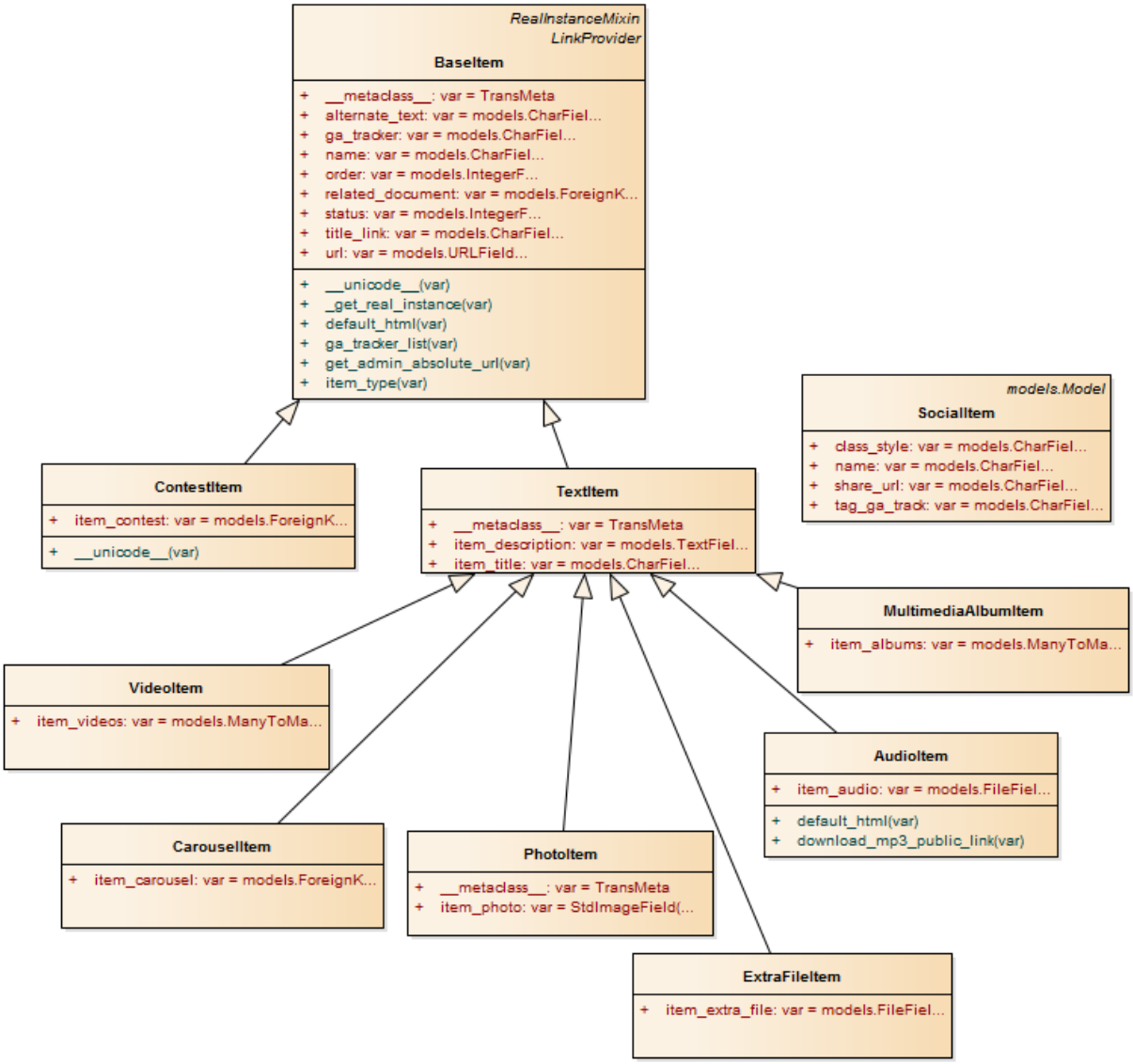

Figura 3.12. Diagrama de clases (4 de 5)

TexItem: hereda de BaseItem, incluyendo los atributos *item_title* y *item_description*, para añadir un título y descripción al ítem respectivamente. A su vez heredan de este:

- *VideoItem*: permite seleccionar un conjunto de videos del portal, tiene una relación de muchos a muchos con la clase *Video*.
- *PhotoItem*: permite añadir una imagen.
- *AudioItem*: permite añadir un fichero de audio.
- *ExtraFileItem*: permite añadir ficheros extra.

- *MultimediaAlbumItem*: permite seleccionar un *MultimediaAlbum* de los existentes en el portal.
- *CarouselItem*: permite seleccionar un *Carousel* de los existentes en el portal.

SocialItem: permite definir patrones para compartir contenido en redes sociales. Está compuesto por los atributos: *name*, para identificar la plataforma con la que se comparte; *class_style*, para definir estilos al ítem; *share_url*, para definir un patrón con la URL que comparte el contenido; *tag_ga*, para establecer la etiqueta que nos identifica que se ha compartido un contenido en Google Analytics.

Figura 3.13. Diagrama de clases (5 de 5)

EventCampaign: hereda de la clase *Event*, contenida en el portal. Esta nos permite definir *eventos* dentro de una campaña. Con el atributo *related_campaign*, relacionamos un evento con una campaña. Los *eventos* también nos permiten definir las *plantillas* que muestra el evento o las que se usarán para enviar notificaciones a los usuarios del evento.

InvitationEventCampaign: controla el sistema de *invitaciones* a un evento. Esta clase contiene un atributo *name* para identificar las *invitaciones* al *evento* y otro atributo *related_event* para identificar el *EventCampaign* con el cual está relacionado.

InvitationWorkflow: permite definir el flujo de trabajo que sigue las *invitaciones* a un *evento*. Este flujo de trabajo se relaciona con un sistema de *invitaciones* (*InvitationEventCampaign*) mediante el atributo *related_invitation_event*. Entre los atributos para definir este flujo de trabajo de las *invitaciones* a un *evento*, tenemos:

- *activate_send_mail* y *activate_send_mail_date*: indican que el sistema puede enviar la invitación y a partir de qué fecha se envían los mails, respectivamente.
- *subject_email*, *invitation_email_template* e *invitation_email_template_html*: son los atributos que definen el asunto y la *plantilla* que muestra la invitación en el mail, también se adjunta la *plantilla* en formato HTML completo.
- *answer_subject_email*, *answer_invitation_email_template* e *answer_invitation_email_template_html*: son los atributos que definen el asunto y la *plantilla* que son enviados como respuesta a una invitación a través de mail, también se adjunta la *plantilla* en formato HTML completo en el mismo.
- *answer_subject_email_accepted*, *answer_invitation_accepted_email_template* e *answer_invitation_accepted_email_template_html*: son los atributos que definen el asunto y la *plantilla* que son enviados como respuesta a una aceptación de una invitación a través de mail, también se adjunta la *plantilla* en formato HTML completo en el mismo.
- *answer_subject_email_declined*, *answer_invitation_declined_email_template* e *answer_invitation_declined_email_template_html*: son los atributos que definen el asunto y la *plantilla* que son enviados como respuesta a un rechazo de una invitación a través de mail, también se adjunta el template en formato HTML completo en el mismo.
- *share_subject_email*, *share_invitation_email_template* e *share_invitation_email_template_html*: son los atributos que definen el asunto y la *plantilla* de un mail que es enviado cuando un usuario comparte la invitación a un evento, también se adjunta la *plantilla* en formato HTML completo en el mismo.
- *invitation_startworkflow_template*, *invitation_commentworkflow_template*, *invitation_shareworkflow_template*, *invitation_endworkflow_template*: son las *plantillas* correspondientes con la vista principal de la invitación, la vista para añadir un comentario cuando se acepta una invitación, la vista para compartir la invitación y la vista que se muestra cuando se termina el flujo de trabajo de la invitación. Opcionalmente, disponemos de los atributos *document_startworkflow_template*, *document_commentworkflow_template*,

document_shareworkflow_template y *document_endworkflow_template*, que añaden contenido extra a las *plantillas* mencionadas anteriormente.

UserNoRegisterInvitedCampaign: clase que añade usuarios que no están registrados en la plataforma y a los cuales se les quiere enviar una invitación. Estos vendrán definidos por los atributos: *full_name*, para definir el nombre completo; *email*, para establecer el mail al cual se va a enviar la invitación; *position*, posición dentro de la empresa; *company*, compañía a la que pertenece; y *related_campaign*, para establecer la campaña con la cual están relacionados.

MailInvitationEventCampaign: clase que está relacionada con *InvitationEventCampaign*, y con la cual controlamos las *invitaciones* que han sido creadas para su envío y la interacción de los usuarios con las mismas. Para ello, se compone de los siguientes atributos: *user*, para definir si es *usuario registrado* al que se le envía una invitación; *no_registered_user*, relacionado con *UserNoRegisterInvitedCampaign* para identificar a un *usuario no registrado* en la plataforma; *is_sent*, para indicar si ha sido enviado el mail de invitación; *comment*, este campo contendrá un comentario añadido por el usuario; *will_attend*, para indicar si el usuario acepta la invitación; y *changed_data*, que indica que los datos de la invitación han sido modificados.

Diseño de la interfaz de administración

Django nos permite definir mediante clases, que hacen uso de las clases definidas en el diseño descrito, una interfaz de administración. En la Figura 3.14, Figura 3.15 y Figura 3.16, disponemos del diseño de clases que se ha usado para personalizar dicha interfaz. Entre estas clases, listamos las más destacadas y explicamos su funcionalidad:

- *ManageCampaignGroupAdmin*: Añade a la interfaz de administración el acceso a configurar y gestionar el listado de permisos que se le asignan a usuarios para gestionar una campaña.
- *HiddenModelAdmin*: oculta las clases que heredan de ella a los usuarios que no tiene permisos de *superusuario* o *administrador*.
- *CampaignAdmin*: provee una interfaz que contiene los objetos de la clase *Campaign*, y que permite gestionar las campañas.
- *DocumentCampaignRelatedBaseItemModelAdmin*: relaciona la clase *DocumentCampaign* con la clase *BaseItem*, para poder añadir los diferentes elementos que heredan de *BaseItem* a un *DocumentCampaign*. Esta clase añade una pestaña en la interfaz de administración de un *DocumentCampaign* con el listado de ítems (*TextItem*, *VideoItem*, etc.) para poder gestionar los ítems relacionados con el documento.
- *CampaignRelatedDocumentCampaignModelAdmin*: relaciona la clase de *DocumentCampaign* con *Campaign*, permitiendo mostrar en una pestaña de

la interfaz de administración de una campaña, el listado de documentos asociados a la misma para su gestión.

- *CampaignRelatedContestAdmin*: esta clase relaciona la clase *ContestCampaign* con la propia *Campaign*, y muestra en una pestaña de la interfaz de administración de la campaña, el listado de *concursos* relacionados para poder gestionarlos.
- *CampaignRelatedEventCampaignAdmin*: esta clase relaciona la clase *Campaign* con EventCampaign, de tal forma que en la interfaz de administración de *Campaign* se añade una pestaña que contiene un listado de *EventCampaign* relacionados.

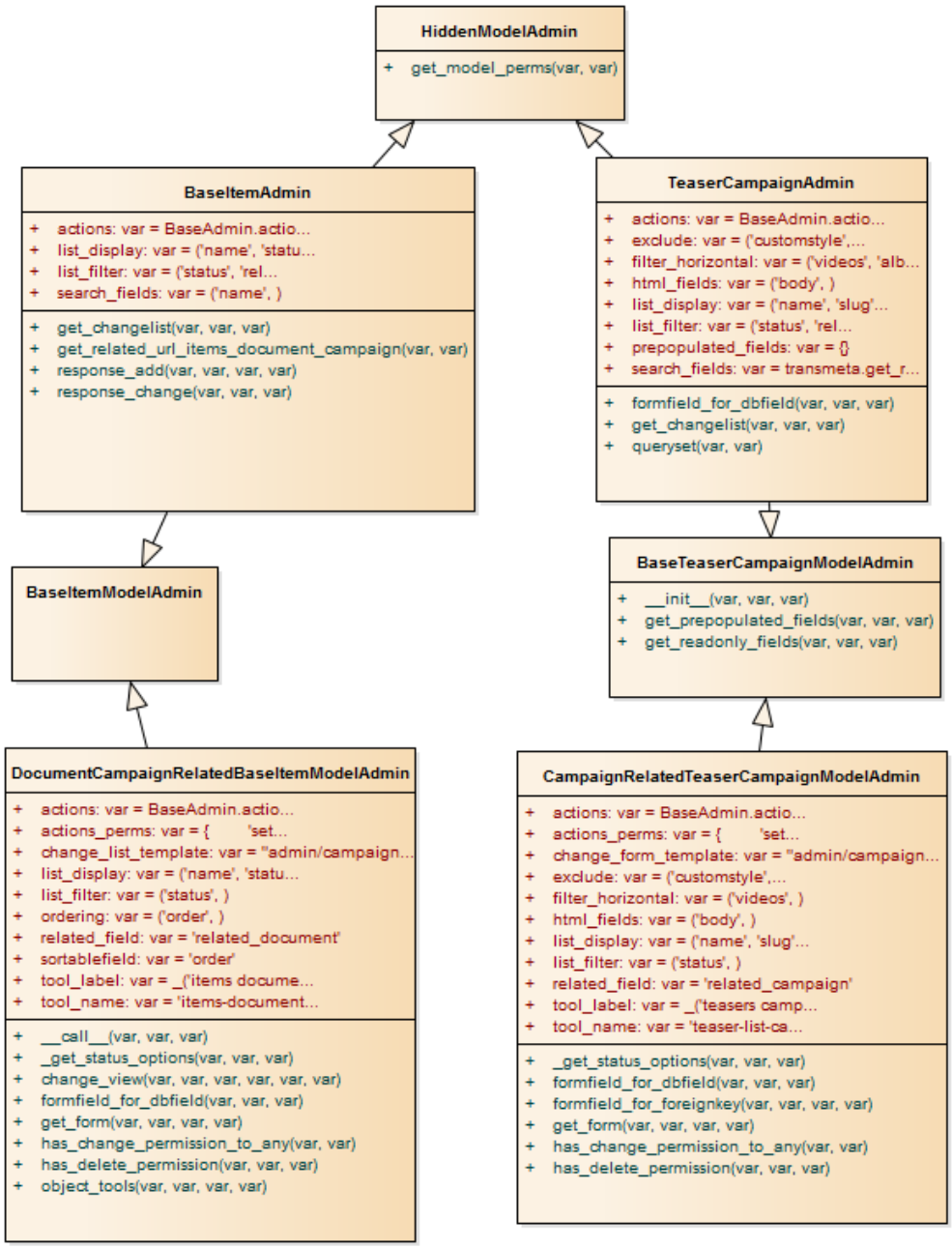

Figura 3.14. Diagrama de clases de las clases relacionadas con la interfaz de administración (1 de 3, ver Figura 3.15 y Figura 3.16)

Figura 3.15. Diagrama de clases de las clases relacionadas con la interfaz de administración (2 de 3, ver Figura 3.14 y Figura 3.16)

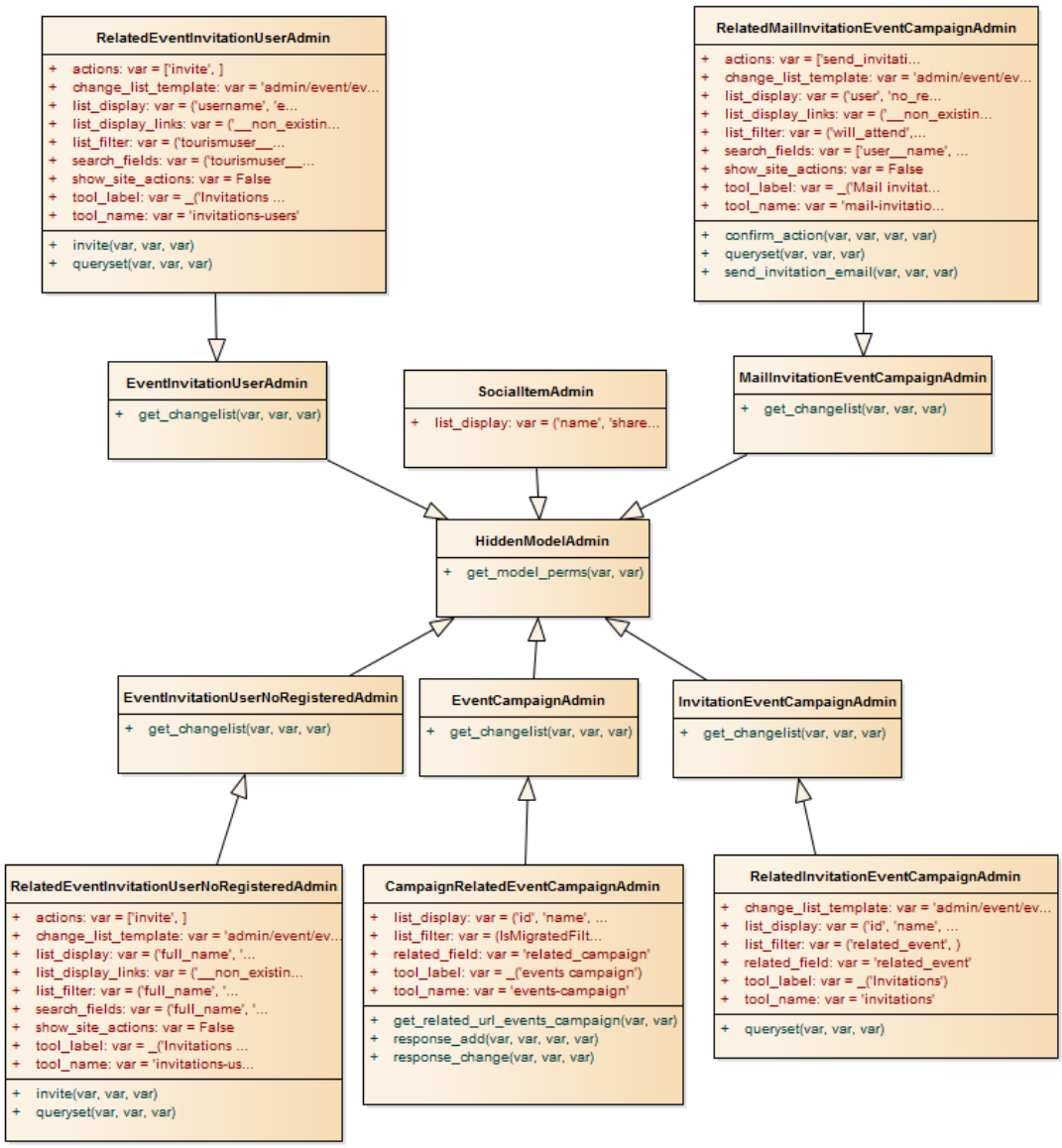

Figura 3.16. Diagrama de clases de las clases relacionadas con la interfaz de administración (1 de 3, ver Figura 3.14y Figura 3.15)

- *CustomTemplateAdmin*: añade una interfaz de administración para gestionar los objetos de tipo *CustomTemplate*.
- *DocumentCampaignRelatedCustomTemplateModelAdmin*: esta clase relaciona la clase *DocumentCampaign* con *CustomTemplate*, de modo que en la interfaz de administración de un documento disponemos de una pestaña donde consultar la *plantilla* que tiene asociada.
- *InvitationEventCampaignRelatedInvitationWorkflowModelAdmin*: esta clase relaciona la clase *InvitationEventCampaign* con *InvitationWorkflow*, de modo que en la vista de invitación de un evento mostramos una pestaña para configurar el flujo de trabajo de la invitación.
- *RelatedEventInvitationUserAdmin* y *RelatedEventInvitationUserNoRegisterdAdmin*: añaden a la interfaz que

gestiona las *invitaciones*, las pestaña para añadir las *invitaciones* de *usuarios registrados* en la plataforma como de *usuarios no registrados*.

- *RelatedInvitationEventCampaignAdmin*: hace referencia a la clase *InvitationEventCampaign* y es mostrada como una pestaña en la interfaz de un evento. Dicha pestaña contiene un listado de las *invitaciones* creadas para dicho evento.
- *RelatedMailInvitationEventCampaignAdmin*: en la interfaz de administración de un elemento de la clase *InvitatiónEventCampaign*, crea una pestaña con un listado de la clase *MailInvitationEventCampaign*, es decir un listado con las *invitaciones* creadas para ser enviadas y relacionadas con dicho evento.
- *CampaignRelatedTeaserCampaignModelAdmin*: añade a la interfaz de administración de un objeto de la clase *Campaign*, una pestaña para gestionar los objetos de la clase *TeaserCampaign*.

El resto de clases no comentadas que aparecen en la Figura 3.14 y Figura 3.15, son clases de menor relevancia para la interfaz de administración y que en su mayoría aportan métodos para la gestión de la misma.

3.3.3. Modelo físico de datos

A continuación se ilustra en la Figura 3.17 (consultar documentación entregada para ver esta figura ampliada y con todos los atributos) el *modelo de base de datos*, en la cual podemos destacar las siguientes tablas de datos:

- *campaign_campaign*: es la tabla que identifica una campaña, con ella se relacionan otras tablas como *campaign_documentcampaign*, *campaign_teaser*, etc.
- *campaign_customtemplate*: en esta tabla se guarda la información sobre las platillas HTML utilizadas para mostrar el contenido albergado en otras tablas. Un ejemplo de tablas que están relacionadas con ésta es *campaign_documentcampaign,* en la que disponemos de una clave foránea a la misma con la finalidad de poder asociarle un aspecto visual diferente al existente por defecto para mostrar el contenido.
- *campaign_documentcampaign*: es una de las tablas más importantes y la cual gestiona el contenido.
- *campaign_baseitem*: es la base de los ítems asociados a *campaign_documentcampaign* y que ayuda a ampliar esa gestión de contenidos, enriqueciendo aún más los contenidos de este último.
- *campaign_campaigninvitationworkflow*: esta tabla aglutina la información para controlar el flujo de trabajo de las *invitaciones* a un *evento*, y está relacionada con *campaign_documentcampaign*, *camapign_eventcampiagn* y *campaign_customtemplate*.

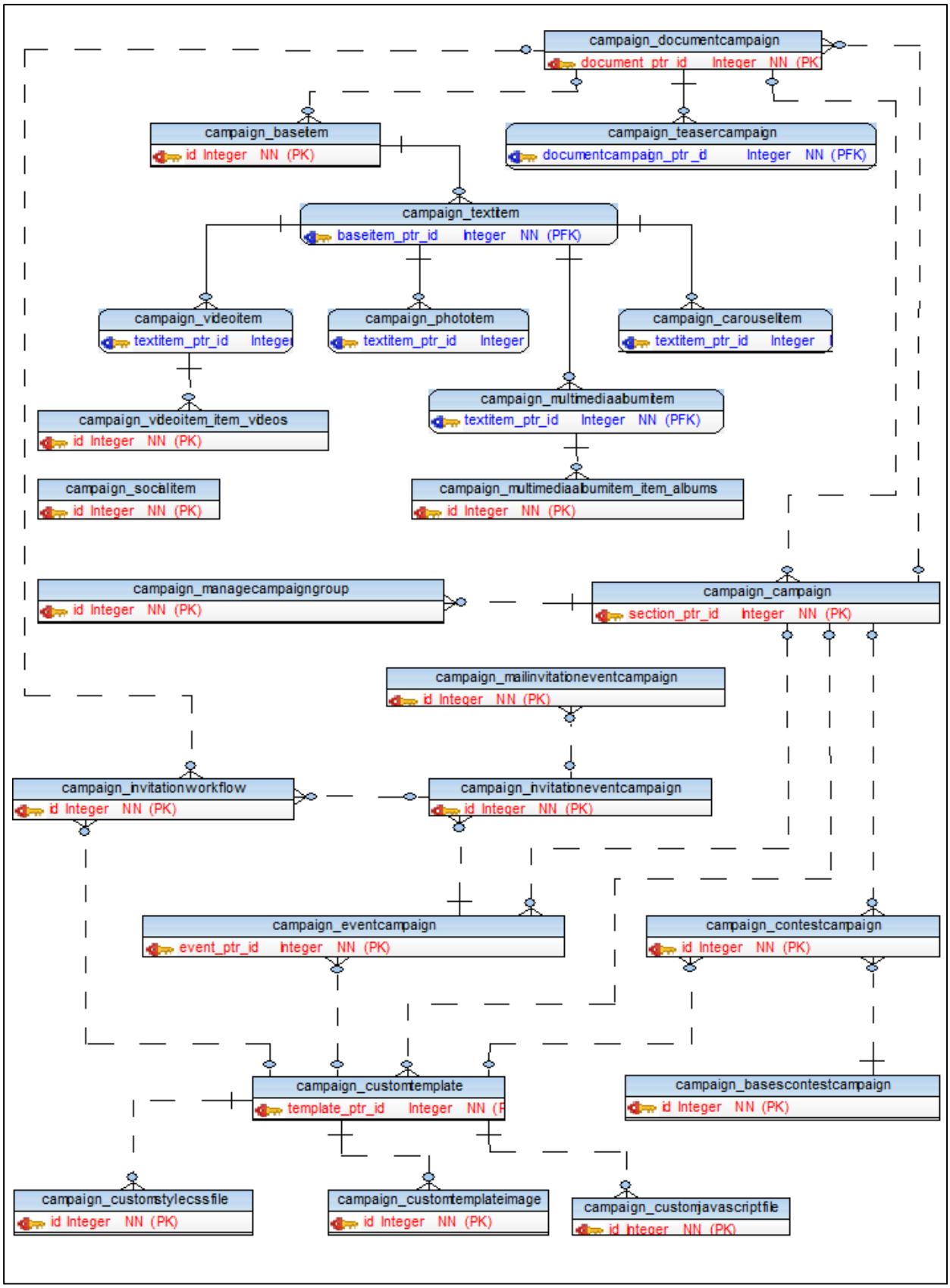

Figura 3.17. Diagrama de datos físicos.

3.4. Implementación

Este apartado no pretende cubrir todos los aspectos de implementación que abarcan el proyecto, se explican más bien casos concretos de implementación que son de gran relevancia para el desarrollo del trabajo fin de grado y resultado de la aplicación final.

3.4.1. Metodología de desarrollo y espacio de trabajo

Durante el desarrollo del trabajo fin de grado se ha aplicado la metodología SCRUM. Al final del apartado 1.2 de esta memoria, están descritas las distintas iteraciones realizadas sobre el producto.

Para llevar un correcto versionado del desarrollo del código, se han utilizado las herramientas Mercurial, TortoiseHg (en local) y RhodeCode (servidor).

Finalmente y para completar el entorno de desarrollo, PyCharm es el IDE de desarrollo que ha sido elegido, el cual ya introducimos en el apartado 2.10, y que por sus cualidades nos ayuda de forma productiva en el desarrollo.

3.4.2. Permisos de Gestión de Campañas

Para poder acceder a la interfaz de administración de Django y gestionar una campaña, habría que acceder al panel de administración de usuarios registrados y desde allí, asignarle los permisos. Pero existiría el problema de que el usuario tendría acceso a todas las campañas, y quizás no es lo correcto, porque en un mismo portal pueden existir distintos *gestores* para distintas campañas, o *gestores* externos a los cuales no se le permite el acceso completo a la plataforma.

Para evitar esto, se ha desarrollado la clase *ManageCampaignGroup*, la cual permite asociar un usuario con una campaña y los permisos que se estimen.

Por defecto, existe un grupo de permisos llamado “Campaign Manager” para ser asignado al *gestor*. Si no se le asigna ningún grupo de permisos en el sistema, éste le otorga los permisos mínimos para gestionar una campaña.

Con todo lo comentado, esta clase nos permite que en las clases que controlan la interfaz de administración, se pueda sobre-escribir el método “queryset” y controlar los elementos a los que tiene acceso el *gestor*. Un *gestor* que esté asociado a una campaña sólo tendrá acceso a dichos elementos. En el código entrega, en el fichero “admin.py” podemos observar el método *filter_related_campaign* que filtra los documentos que son accesibles.

```
def filter_related_campaign(queryset, request):
   managecampaign = ManageCampaignGroup.objects.filter(user=request.user)
   campaigns = [x.campaign.id for x in managecampaign]
   #devuelve los objetos que estan asociados a las campañas en la que el usuario
es participe
   # y los objetos que no estan asociados a ninguna campaña
   return queryset.filter(Q(related_campaign__id__in=campaigns) |
Q(related_campaign=None))

class DocumentCampaignAdmin(BaseDocumentCampaignModelAdmin, BaseAdmin,
DocumentWorkflowBatchActionProvider, HiddenModelAdmin):

   def queryset(self, request):
       qs = super(DocumentCampaignAdmin, self).queryset(request)
       qs = filter_model_name(qs, 'documentcampaign')
       if request.user.is_superuser:
           return qs
       return filter_related_campaign(qs, request)
```

Figura 3.18. Método para filtrar documentos relacionados con una campaña.

3.4.3. Microsite

El *microsite* se define como una sección especial definida dentro del portal en el que se integra, “andalucia.org”. Uno de los requisitos fundamentales, es la flexibilidad en el aspecto gráfico y hacer que los documentos del portal tuvieran más posibilidades a la hora de gestionar y mostrar su contenido.

Sistema de Plantillas

Para poder personalizar el *microsite*, se ha utlizado el sistema de *plantillas* basado en el paquete *django-dbtemplates*, el cual ha sido adaptado y complementado con diferentes métodos que añaden código CSS y JavaScript.
Este sistema de plantillas guarda las mismas en base de datos, y relaciona con ellas ficheros CSS y JavaScript. En el fichero “models.py” tenemos los métodos (*_get_template* y *_insert_extrahead*) que forman la *plantilla* y que cargan el contenido que tenga asociado. El método *_get_template* devuelve el *template* y el método *_insert_extrahead*, añade los ficheros CSS y JavaScript que tenga asociada la *plantilla*.

Para mejorar el aspecto de la edición de código HTML en la interfaz de administración se ha configurado para que acepte las librerías del editor de código online *CodeMirror*. Para ello, se ha integrado la aplicación *django-codemirror*, y se han configurado las variables que aparecen en el fichero “settings.py” de la aplicación. Para poder visualizar el editor en nuestra interfaz de administración, en la clase *CustomTemplateAdmin*, hemos añadido el código para añadir las librerías de *CodeMirror* y además hemos sobre-escrito el método que carga los widgets asociados al formulario.

Extensión de los contenidos en documentos y teaser

La flexibilidad del *microsite*, además de por el sistema de *plantillas*, tiene que venir acompañado por la flexibilidad en la gestión de su contenido. Para ello, además de poder relacionar una *plantilla* con el *documento* de una campaña, se le añade una serie de *ítems* para gestionar y añadir a su contenido. Estos *ítems* pueden ser de diferentes tipos y extienden el contenido de forma ilimitada al documento. Cuando se carga la vista, estos son enviados en una variable llamada "items". Para poder controlarlos dentro de la *plantilla* se han creado diferentes filtros, los cuales en Django se designan como *template_tags*. En la Figura 3.19 podemos observar algunos de los distintos métodos que han desarrollado para acceder a los items y añadir funcionalidad.

```
@register.filter()
def get_item_by_name(items, name):
   item = items.filter(name=name)
   if item:
       return item[0]._get_real_instance()
   return None


@register.filter()
def get_group_items_by_name(items, name):
   return items.filter(name=name)


@register.filter()
def get_real_instance(item):
    return item._get_real_instance()


@register.filter()
def random_items(items):
   item = random.choice(items)
   return item._get_real_instance()
```

Figura 3.19. Filtros para acceder en las plantillas al contenido de los items.

3.4.4. Concursos

Para un usuario poder participar en un *concurso*, tiene que rellenar un formulario. Este formulario ha tenido que ser configurado previamente en la interfaz de administración. Así pues hay que contralar dichos aspectos de su configuración, ello lo hacemos sobre-escribiendo ciertos métodos que se definen en la clase que controla el formulario. En el fichero "forms.py" se define la clase *UserContestCampaignForm* que se instancia para crear un formulario para el *concurso*. En este, se definen campos no obligatorios como *video*, *image* y *text*, y en los métodos *clean_video*, *clean_image* y *clean_text*, según los campos booleanos

video_contest, *image_contest* y *text_contest*, controlamos si son requeridos debido a la configuración del *concurso*.

3.4.5. Eventos y Sistema de invitaciones

Los *eventos* de una campaña tienen la posibilidad de enviar invitaciones vía mail. Para proceder al envío de estas invitaciones de forma masiva, usamos el paquete *django-rq* que permite encolar procesos y trabajos para lanzarlos en segundo plano. Para poder usar el paquete *django-rq* tenemos que tener instalado un servidor *Redis*. Desde la interfaz de administración donde se gestionan las *invitaciones*, pueden ser lanzados los trabajos que hacen que se envíen en segundo plano las *invitaciones*. En el fichero "admin.py" en la clase RelatedMailInvitationEventCampaignAdmin se define el método send_invitation_email con el cúal se procede al envío de *invitaciones*.

3.4.6. Patrones de URLs y Middleware

Para poder recibir las diferentes peticiones, acciones y cargar vistas, hay que definir los patrones de las URLs que los identifican. Estos patrones son añadidos al fichero "urls.py", cuyos patrones de URLs cargan los métodos definidos en el fichero "views.py". Estos patrones pueden entrar en conflicto con otros patrones alojados en el portal en el cual se integra la aplicación, pero para que esto no ocurra, se ha creado una clase *Middleware*, que procesa previamente estos patrones y decide si la petición es para procesar una URL que pertenece a la aplicación del *Gestor de Campañas* o es para otra aplicación también integrada en el portal. En el fichero "middleware.py" podemos ver el código que implementa la clase *CampaignMiddleware*.

3.4.7. Comandos del Gestor de Campañas

Los *comandos* y *acciones* nos permiten automatizar ciertas tareas. En el caso de los comandos, pueden ser ejecutados desde la consola o automatizados mediante un *cron* en el servidor.

Para el *Gestor de Campañas* se han desarrollado varios comandos, entre ellos el que permite automatizar el envío de correos de *invitaciones* pendientes de enviar. En la Figura 3.20, podemos ver el código que se ejecuta cuando se escribe en una consola linux "./python manage send_pending_event_invitations_campaign ".

```
class Command(BaseCommand):
   help = u"Send pending invitations to events of campaings to users"
   option_list = BaseCommand.option_list + (
       make_option('-m', '--max-to-send', default='100', dest='max_to_send'),
       make_option('-s', '--subject', default='Invitation', dest='subject'),
   )
   def handle(self, *args, **options):
       max_to_send = int(options.get('max_to_send'))
       subject = options.get('subject', '')
       activate('es')  # all the emails will be in spanish
       for invitation in
MailInvitationEventCampaign.objects.filter(is_sent=False)[:max_to_send]:
           invitation_user_email=''
           if invitation.user:
               invitation_user_email = invitation.user.email
           elif invitation.no_registered_user:
               invitation_user_email = invitation.no_registered_user.email
           print u'Enviando invitacion del evento %d al usuario %s' %
(invitation.related_invitation_event.related_event.id, invitation_user_email)
           _send_invitation_email(invitation)
           time.sleep(5)
```

Figura 3.20. Clase que ejecuta el comando send_pending_event_invitations_campaign

3.4.8. API de YouTube

Como requisito de la aplicación, se ha desarrollado la posibilidad de poder subir vídeos a YouTube, procedentes de respuestas de los *concursos* y cuando estos sean publicados en el portal. Para poder conseguir dicho control, en la clase que se define el *concurso* disponemos de un atributo que indica si los videos son subidos a YouTube. Cuando se produce la publicación de un video que es respuesta a un *concurso*, la acción que cambia el estado de una respuesta a pública, consulta si el *concurso* debe subir el vídeo a YouTube, y si es cierto, entonces procede a la comunicación con la dicha plataforma y la subida.

El método está implementado dentro del fichero “api_youtube.py”, y en él se gestiona la autenticación de la conexión con la Api de YouTube (versión 3) y el método de subida.

3.4.9. API de Google Analytics

Para completar la gestión de una campaña, es importante tener referencia del éxito de la misma, ello lo podemos comprobar obteniendo estadísticas del *microsite*.

Se ha desarrollado una vista en la que se pueden consultar las estadísticas relacionadas con el *track* de Google Analytics en la campaña.

En el fichero “api_analytics.py” encontramos desarrollado los métodos de autenticación y como se obtiene información sobre una campaña que dispone de un identificador de Google Analytics.

Capítulo 4. Conclusiones y Futuros Trabajos

En este punto se han de recordar los objetivos que se indicaron en el capítulo inicial para reflexionar en qué grado se han alcanzado las metas que fundamentan este trabajo fin de grado.

Además, se dará una lista de posibles ampliaciones para expandir las funcionalidades de la aplicación y líneas de trabajo futuras.

4.1. Conclusiones

Volviendo al primer capítulo hay que recordar que la finalidad de este proyecto es la realización de un *Gestor de Campañas* para el cual se han explorado las posibilidades que ofrece internet para la promoción online de un producto.

Para dar sentido a la aplicación, ésta se enmarcó dentro del contexto de sistemas que gestionan campañas publicitarias online con los diferentes mecanismos que también aportan las redes sociales y herramientas de analítica web, entre ellas encontramos AccumbaMail, Woobox, Bloonder o Google Analytics. Con los productos analizados, se estudió la estrategia y requisitos que se necesitaban para poder crear una herramienta que gestionara campañas para promocionar el turismo en Andalucía y que se integrara dentro de la plataforma de la Comunidad Turística Andaluza “andalucia.org”.

En la solución desarrollada, se han incluido necesidades cubiertas por otras herramientas para realizar las promociones turísticas pero integrándolas dentro de la plataforma “andalucia.org”, pero con la ventaja de tener los datos centralizados, obteniendo un mayor control sobre la campaña. Por todo ello, la solución aportada ha mejorado la eficiencia de gestión de una campaña, supliendo a otras herramientas que se usaban de forma externa y complementaria.

Las tecnologías utilizadas para desarrollar esta aplicación han facilitado su rápido desarrollo y la realización de tareas complejas. Entre las citadas en capítulo 2, cabe destacar principalmente Python y Django. Esta combinación ha sido en gran parte la causa del éxito en el desarrollo, ya que de ambas tecnologías podemos encontrar una extensa documentación, complementada con una gran cantidad de

paquetes implementados por la comunidad y que nos han servido para extender nuestro desarrollo con mayor número de funcionalidades, además de las implementadas.

Con respecto a la metodología utilizada, podemos destacar que SCRUM nos ha permitido llevar el desarrollo de forma controlada, marcando metas por cada sprint, haciendo que no se disparen los tiempos y costes de implementación, así como añadiendo dinamismo al desarrollo.

A todo lo comentado, me gustaría añadir lo complicado que resulta resumir en unas líneas todo lo que abarca y aporta el trabajo fin de carrera y este proyecto en concreto. Éste me ha aportado varios aspectos que culminan mi formación académica en la titulación de Grado en Ingeniería de Computadores. Me ha ayudado a ver la importancia que un trabajo fin de grado tiene a la hora de completar la formación del alumno y realizar todas las fases que conlleva el desarrollo de un proyecto software. Éste ha resultado ser una gran experiencia a nivel académico, personal y profesional ya que ha sido realizado en gran parte en la Universidad de Málaga en un acuerdo de colaboración con la Consejería de Turismo y la Empresa pública para la Gestión del Turismo y del Deporte de Andalucía. Gracias a este proyecto también me he podido especializar en el área de las aplicaciones web con las tecnologías Python y Django que sin duda marcarán la continuación de mi camino dentro del ámbito profesional. También destacar que la idea de este proyecto es que fuese un proyecto aprovechable y así lo ha sido, ya que la Empresa pública para la Gestión del Turismo y del Deporte de Andalucía la usa para la gestión de sus campañas promocionales para ferias internacionales de turismo y otros eventos.

Para finalizar, decir que este proyecto puede ser el complemento de muchos otros proyectos e invito a todo el mundo que quiera a continuar y compartir el camino que yo he comenzado aquí.

4.2. Trabajos futuros

Una vez estudiadas las posibilidades que proporciona la aplicación se pueden ver futuras mejoras y ampliaciones para el producto software. A continuación se listan y explican brevemente los futuros trabajos y mejoras para la aplicación:

- *Comercialización de productos.* Cuando se promociona un producto online, la mayoría de usuarios que acceden a esa promoción pueden estar interesados en adquirir dicho producto. Por ello, sería interesante integrar diferentes plataformas de pago como puede ser PayPal y crear un catálogo de productos junto a un carrito de la compra para poder comercializarlos online.
- *Eventos profesionales, mensajería y estadísticas.* En muchas ocasiones, existen eventos dentro de una campaña para comerciar u ofrecer productos a empresas profesionales. La idea sería poder establecer contactos entre dos nuevos tipos de usuarios, la figura del *usuario comercial* y la del *usuario profesional*. De forma que puedan quedar registradas dentro de la plataforma

el contacto entre ambos usuarios y el éxito final de la comercialización. Para registrar este contacto se crearía un sistema de mensajería privada dentro de la aplicación, además de ofrecer un contacto directo a través de chat. La comercialización de los productos se podría reflejar en un panel donde se mostrasen las estadísticas para estudiar el éxito de un *evento* profesional y de la aceptación y comercialización con los profesionales del sector.

- *Integración con Instagram y Twitter para concursos.* Actualmente la plataforma cuenta con la integración de YouTube en el sistema de concursos. Pero sería interesante también la posibilidad de integrar otras redes sociales para los concursos como son Instagram y Twitter. Instagram nos permitiría integrar concursos de imágenes y Twitter estaría más indicado para concursos de texto. La realización de estos concursos se podrían realizar definiendo un *hashtag*, que permita a la aplicación detectar estas entradas y mostrarlas en el *microsite*, permitiendo el registro de las respuestas dentro del portal.
- *Creación de una API de campañas.* Otra posible forma de poder dar más publicidad a una campaña, sería la creación de una API que nos permita alimentar con nuestro contenido otras plataformas. También podría servir para crear un portal específico para un producto y alimentar su contenido con los contenidos gestionados desde el gestor de campañas.
- *Versionado del sistema de plantillas.* Incluir un sistema de versionado para las *plantillas* basadas en el paquete “django-dbtemplates”, puede ayudarnos a recuperar versiones antiguas o depurar errores de cambios posteriores realizados en las *plantillas*. Esto se podría realizar gracias al paquete “django-reversion”.
- *Módulo de encuestas y estadísticas.* Aunque el sistema de concurso se podría usar a modo de encuesta, éste sólo aceptaría una pregunta. Por ello es interesante crear un módulo específico para realizar encuestas en las que el *gestor* pueda añadir una o más preguntas al usuario y el tipo de respuesta (seleccionable, multi-respuesta, etc.). Estas encuestas se enviarían vía mail para que participasen los Usuarios registrados y los Usuarios no registrados en el portal. Las encuestas estarían disponibles durante un tiempo específico. Además, se crearía una vista donde poder consultar las estadísticas a modo de resumen de las encuestas realizadas por los usuarios.
- *Test con Selenium*. Generación de una batería de pruebas, de forma que se cubran todos los aspectos de la aplicación con las herramientas que proporciona Selenium [31].

Apéndice A. Instalación y configuración de la aplicación

La aplicación *Gestor de Campañas* es parte de la aplicación del portal de "andalucia.org". En los puntos siguientes de este apéndice se explican los requisitos, dependencias y pasos a seguir para configurar e integrar la aplicación dentro del portal anteriormente mencionado.

A.1. Requisitos del sistema

La aplicación requiere una serie de paquetes que son necesarios y que se listan a continuación:

- *Django* (versión 1.4.2)
- *South* (versión 0.7.6)
- *reportlab* (versión 2.5)
- *django-rosetta-yaco* (versión 0.6.12)
- *django_rq* (0.5.1)
- *django-dbtemplates* (versión 1.3)
- *django-codemirror-widget* (versión 0.4.0)
- *oauth2* (versión 1.5.167 o superior)
- *google-api-python-client* (versión 1.2)

El portal en el que está integrada la aplicación hace uso del paquete *zc.buildout* [32] para realizar el despliegue de la aplicación. De modo que para realizar este despliegue, añadimos los paquetes necesarios para que estos sean instalados al ejecutar el *buildout*. Estos se añaden en el apartado de "eggs" del fichero *buildout.cfg* (ver Figura de Apéndice A.1).

```
[eggs]
recipe = zc.recipe.egg
eggs =
   …
   Django==1.4.2
   South==versión 0.7.6
   reportlab==2.5
   django-rosetta-yaco==0.6.12
   django_rq==0.5.1
   django-dbtemplates==1.3
   django-codemirror-widget==0.4.0
   oauth2>=1.5.167
   google-api-python-client==1.2
   django-codemirror-widget==0.4.0
   django-dbtemplates==1.3
```

Figura de Apéndice A.1. Código en el fichero *buildout.cfg*

A.2. Configuración general

En el fichero settings.py de la aplicación general, hay que configurar ciertos parámetros de los paquetes añadidos:

- Para el paquete “django-codemirror-widget”, añadimos al fichero “settings.py” el código de la Figura de Apéndice A.2.

```
CODEMIRROR_PATH = 'codemirror'
CODEMIRROR_MODE = 'django'
CODEMIRROR_THEME = 'default'
CODEMIRROR_CONFIG = { 'lineNumbers': True, }
CODEMIRROR_JS_VAR_FORMAT = "%s_editor"
```

Figura de Apéndice A.2. Parámetros y configuración del paquete *django-codemirror-widget*

- Para el paquete “django-dbtemplates”, añadimos al fichero “settings.py” los valores de la Figura de Apéndice A.3.

```
DBTEMPLATES_ADD_DEFAULT_SITE = True
DBTEMPLATES_AUTO_POPULATE_CONTENT = True
DBTEMPLATES_CACHE_BACKEND = CACHE_BACKEND
DBTEMPLATES_USE_CODEMIRROR = True
DBTEMPLATES_USE_TINYMCE = False
DBTEMPLATES_MEDIA_PREFIX = path.join(STATIC_URL, "dbtemplates/")
TEMPLATE_LOADERS += ('dbtemplates.loader.Loader',)
DBTEMPLATES_CACHE_BACKEND = 'memcache://127.0.0.1:11211'
CACHES = CACHES + { 'dbtemplates' : dict(
    BACKEND='django.core.cache.backends.memcached.MemcachedCache',
    LOCATION = ['127.0.0.1:11211'],
   ),}
```

Figura de Apéndice A.3. Parámetros y configuración del paquete *django-dbtemplates*

- Para configurar el *middleware* del sistema, añadimos al fichero "settings.py" los valores de la Figura de Apéndice A.4

```
MIDDLEWARE_CLASSES = MIDDLEWARE_CLASSES + (
   'campaign.middleware.CampaignMiddleware',
)
```

Figura de Apéndice A.4. Configuración del *Middleware*

- Para configurar el paquete django-rosetta-yaco, añadimos al fichero "settings.py" los valores de la Figura de Apéndice A.5

```
TRANSMETA_LANGUAGES = LANGUAGES
TRANSMETA_DEFAULT_LANGUAGE = 'en'
TRANSMETA_MANDATORY_LANGUAGE = 'es'
TRANSMETA_VALUE_DEFAULT = ''
```

Figura de Apéndice A.5. Parámetros del paquete *django-rosetta-yaco*

- Para configurar la aplicación del *Gestor de Campañas* dentro del sistema, añadimos al fichero "settings.py" los valores de la Figura de Apéndice A.6.

```
INSTALLED_APPS = INSTALLED_APPS + (
   'dbtemplates',
   'campaign'
)
```

Figura de Apéndice A.6. Aplicaciones instaladas en el proyecto.

A.2.1. Parámetros de configuración de la aplicación "Gestor de Campañas"

Dentro de la aplicación del *Gestor de Campañas*, encontramos otro fichero de configuración en la ruta "campaign/settings.py". En él, encontramos el resto de parámetros que se usan dentro de la aplicación. Para completar la configuración de este fichero tenemos que configurar los parámetros de autenticación para la comunicación con la API de YouTube y Google Analytics. Ésta última configuración, la podemos ver en el Apéndice B.

A.3. Despliegue

Para realizar el despliegue del proyecto junto con la aplicación del *Gestor de Campañas*, tenemos que copiar la aplicación dentro de nuestro proyecto Django (en el mismo lugar dónde estén el resto de aplicaciones instaladas) y tener configurado todos los parámetros del apartado A.2 y A.2.1. Una vez realizado esto, tenemos que abrir un terminal de consola y seguir los siguientes pasos:

1. Ejecutar el comando para lanzar el *buildout*.

   ```
   $ ./buildout –N
   ```

2. Actualizar la base de datos, haciendo uso del paquete *South*. Para ello ejecutamos en consola:

   ```
   $ ./python manage.py migrate
   ```

3. Actualizar los ficheros estáticos para el editor de código del paquete *django-codemirror-widget* lanzando el comando:

   ```
   $ ./python manage.py collectstatic -v3
   ```

   ```
   [pregunta si desea sobreescribir los ficheros, responder 'yes']
   ```

4. Lanzamos el comando para inicializar los grupos de permisos del *Gestor de Campañas*:

   ```
   $ ./python manage.py init_group_permissions
   ```

5. Iniciamos la aplicación:

   ```
   $ python manage.py runserver
   ```

Apéndice B. Configuración de la API de YouTube y de Google Analytics

La aplicación *Gestor de Campañas* contiene partes que se comunican con otras plataformas e intercambian información. En concreto, hablamos de las plataformas de YouTube y de Google Analytics. En los siguientes apartados se explica cómo se configura el portal para aceptar dicha comunicación.

B.1. Publicación de vídeos del Gestor de Campañas en YouTube.

En el *Gestor de Campañas*, se permite automatizar la subida de vídeos a YouTube. Esto se produce cuando son subidos y publicados como respuesta a un *concurso*. Para obtener dicho resultado hay que configurar la variable *YOUTUBE_JSON_OAUTH2_CLIENTE_SECRET* que está en el fichero "campaign/settings.py". A esta variable hay que asignarle la ruta y el nombre de un fichero en formato JSON que nos permite usar la API de YouTube en su versión 3 como una aplicación instalada en local. Esta forma de interacción es conocida como "flujo de aplicación instalada", en el enlace https://developers.google.com/youtube/v3/guides/authentication#installed-apps encontramos más información sobre los diferentes flujos de trabajo con la API de Youtube versión 3.

Para obtener el fichero JSON que necesitamos y configurar la variable *YOUTUBE_JSON_OAUTH2_CLIENTE_SECRET*, seguimos los siguientes pasos:

1. Entramos en https://console.developers.google.com/
2. Si no tenemos activada la API de Youtube v3, Realizar el paso 3. Sino pasar al paso 4.
3. Activamos al *API de Youtube v3* para el proyecto que tengamos generado. Para ello entramos en https://console.developers.google.com/project y

entramos en el proyecto que queremos activar la API de YouTube. En el apartado de APIs y autenticación, hacemos click en APIs, buscamos en el listado YouTube Data API v3 y en la columna del final le damos a activar.

4. Si no tenemos credenciales de Oauth llamada “ID de cliente para aplicaciones nativas”, realizamos los pasos 5, 6 y 7. Si ya tenemos dichas credenciales, pasamos al paso 8
5. En APIs y autenticación, hacemos click en Credenciales. Una vez allí, hacemos click sobre el botón “Crear ID de cliente nuevo”.
6. En el formulario que se abre, en TIPO DE APLICACION seleccionamos “Aplicación instalada”, y en TIPO DE APLICACIÓN INSTALADA seleccionamos “Otros”.
7. Esto nos creará una nueva entrada en el apartado de credenciales de Oauth llamada “ID de cliente para aplicaciones nativas”.
8. Sobre la entrada de “ID de cliente para aplicaciones nativas” pulsamos sobre el botón que tiene asociado “Descargar JSON”.
9. Copiamos el fichero descargado en nuestra aplicación.
10. En el fichero “campaign/settings.py” asignamos el nombre del fichero a la variable.

Por ejemplo, Si nuestro fichero se llama *client_secret_x_.apps.googleusercontent.com.json*, la configuración para el fichero “campaign/settings.py” quedaría de la siguiente manera:

```
YOUTUBE_JSON_OAUTH2_CLIENT_SECRET =
client_secret_x_.apps.googleusercontent.com.json
```

B.2. Estadísticas del Gestor de Campañas con Google Analytics.

El visor de estadísiticas requiere configurar una variable que existe en el fichero “campaign/settings.py” para poder visualizar las estadísticas del *microsite* de una campaña. Esta variable se llama *ANALYTICS_JSON_OAUTH2_CLIENTE_SECRET*, y al igual que su homóloga *YOUTUBE_JSON_OAUTH2_CLIENTE_SECRET*, hay que asignarle la ruta y nombre de un fichero en formato JSON que nos permite usar la API de Google Analytics como aplicación instalada en local.

Para obtener el fichero en formato JSON y configurar la variable *ANALYTICS_JSON_OAUTH2_CLIENTE_SECRET,* hay que seguir los mismos pasos que se han seguido en el apartado B.1 del Apéndice B, a excepción de:

- En lugar de comprobar si está activa la API de YouTube y activarla, en este caso hay que comprobar que la API que tiene que estar activa es la API correspondiente a Google Analytics.

- Si ya tenemos configurada la API de YouTube, y usamos la misma cuenta para Google Analytics y YouTube, entonces no será necesario obtener el fichero en formato JSON y podremos asignar el mismo valor a *ANALYTICS_JSON_OAUTH2_CLIENTE_SECRET* que el valor que le hemos asignado a *YOUTUBE_JSON_OAUTH2_CLIENTE_SECRET*.

Apéndice C. Manual de Usuario

Para hacer un buen uso y aprovechar todas las posibilidades de un producto software es necesario consultar el manual de usuario. Dentro éste, siempre se encuentran el modo y los pasos a seguir para realizar determinadas operaciones y entender el funcionamiento del sistema. Aunque una aplicación debe ser intuitiva en su manejo, siempre ha de ir acompañada de un manual de usuario para poder resolver dudas.

El manual de usuario de la aplicación desarrollada se puede encontrar junto a la documentación entregada para este trabajo fin de grado en formato PDF con el nombre de “Manual de Usuario: Gestor de Campañas.pdf”. En éste se detallan:

- Características generales.
- Administración de permisos para gestionar una campaña.
- Gestión de bases de los concursos.
- Compartir contenido con otras plataformas o redes sociales.
- Gestión de una campaña, donde se incluye la gestión de *documentos*, *teaser*, *ítems* que amplían el contenido, visor de *estadísticas*, *menús* del *microsite*, *concursos*, respuestas de los *concursantes*, y *eventos e invitaciones*.
- *filters* y *template_tags* para gestionar la funcionalidad del contenido las *plantillas*.
- La formación de URLs del gestor.
- Variables que nos encontramos dentro del contexto de las *plantillas* que carga cada una de las URLs.
- Modelo de datos, para poder consultar los atributos y el tipo de objeto de las variables que pueden ser usadas dentro del contexto de una *plantilla*.

Referencias

[1] Acumbamail. [Online]. https://acumbamail.com/

[2] MailChimp. [Online]. http://mailchimp.com/

[3] Woobox. [Online]. https://woobox.com/

[4] Bloonder. [Online]. http://bloonder.com/

[5] Wildfire by Google. [Online]. http://www.wildfireapp.com/

[6] Facebook. [Online]. https://www.facebook.com/

[7] Twitter. [Online]. https://twitter.com/?lang=es

[8] Instagram. [Online]. http://instagram.com/

[9] YouTube. [Online]. https://www.youtube.com/

[10] Google Analytics. [Online]. http://www.google.es/intl/es/analytics/

[11] Adobe Analytics. [Online]. http://www.adobe.com/es/solutions/digital-analytics.html

[12] KISSmetrics. [Online]. https://www.kissmetrics.com/

[13] Python. [Online]. http://www.python.org

[14] Django. [Online]. http://www.djangoproject.com/

[15] HTML5. [Online]. http://www.w3.org/TR/html5/

[16] CSS3. [Online]. http://www.w3.org/Style/CSS/

[17] CentOS. [Online]. http://www.centos.org/

[18] Javascript.

[19] Jquery. [Online]. http://jquery.com

[20] ZURB Foundation. [Online]. http://foundation.zurb.com/

[21] Postgresql. [Online]. http://www.postgresql.org.es/

[22] PyCharm. [Online]. http://www.jetbrains.com/pycharm/

[23] pgAdmin. [Online]. http://www.pgadmin.org/

[24] TortoiseHg. [Online]. http://tortoisehg.bitbucket.org/

[25] UML. [Online]. http://www.uml.org

[26] MagicDraw. [Online]. http://www.nomagic.com

[27] SCRUM. [Online]. http://www.scrum.org

[28] Taiga. [Online]. https://taiga.io/

[29] Nginx. [Online]. http://nginx.org/

[30] Sass (Syntactically Awesome Style Sheets). [Online]. Syntactically Awesome Style Sheets

[31] Selenium. [Online]. http://docs.seleniumhq.org

[32] Jacob Kaplan-Moss. (2009, Apr.) jacobian.org. [Online]. http://jacobian.org/writing/django-apps-with-buildout/

[33] Redmine. [Online]. http://www.redmine.org

[34] Rhodecode. [Online]. https://rhodecode.com/

[35] Documentación. Creación de filters y template_tags. [Online]. https://docs.djangoproject.com/en/1.4/ref/templates/builtins/

[36] Documentación de Django - Sistemas de Plantillas. [Online]. https://docs.djangoproject.com/en/dev/topics/templates/

[37] Mark Lutz's, *Learning Python*.: O'Reilly Media, 2013.

[38] Henrik Kniberg, *Scrum and XP from the trenches*, Lulu.com, Ed., 2007.

[39] Adrian Holovaty and Jacob Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right (Expert's Voice in Web Development)*, 2nd ed.: Apress, 2009.

[40] Daniel Greenfeld and Audrey Roy, *Two Scoops of Django: Best Practices for Django 1.6*.: Two Scoops Press, 2014.