GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

Desarrollo de una plataforma web de gestión de Ibeacon y Eddystone

Autor:

Iván CHINCHILLA MIJAS

Supervisor:

Rafael AMORÓS

Tutor académico:

Gabriel RECATALÁ

Fecha de lectura: 7 de Noviembre de 2016

Curso académico 2015/2016

Resumen

Este proyecto ha consistido en el desarrollo de una aplicación web para la gestión de ibeacons y eddystone (balizas), lo que permite llevar un seguimiento y mantener la información asociada a cada uno de ellos. Se han llevado a cabo las fases de análisis, diseño e implementación propias de ingeniería de software. Mediante esta aplicación los usuarios pueden asociar a cada baliza un contenido propio, identificándolo mediante una zona y/o campaña, para que, mediante tecnología bluetooth, pueda enviárselo a los clientes mediante una aplicación móvil.

Palabras clave

Desarrollo web, Java, Spring Boot, Gestión.

Keywords

Web development, Java, Spring Boot, Management.

Índice general

Índice de figuras **7**

1. Introducción **9**

1.1. Justificación y motivación del proyecto 9

1.2. Objetivos del proyecto 10

1.3. Descripción del proyecto 11

2. Planificación del proyecto **13**

2.1. Metodología de trabajo 13

2.2. Planificación temporal 17

2.3. Incidencias y problemas 19

3. Entorno tecnológico **21**

3.1. IntelliJ IDEA 22

3.2. Apache Maven 23

3.3. Spring Boot 24

3.4. Twitter Bootstrap 26

3.6. MySQL 27

3.5. Trello 27

4. Desarrollo de la aplicación **29**

4.1. Análisis 29

4.1.1. Casos de uso 29

4.1.2. Requisitos funcionales 31

4.1.3. Requisitos de datos 36

4.1.4. Requisitos de software y hardware 37

4.1.5. Prototipos de la interfaz 38

4.2. Diseño e implementación 42

4.2.1. Arquitectura de la aplicación 42

4.2.2. Diagrama de clases 43

4.2.2.1. Patrón MVC 43

4.2.2.2. Patrón DAO 45

4.2.2.3. Patrón Observador 46

4.2.3. Interfaz de usuario 46

4.3. Validación y verificación 53

4.3.1. Pruebas unitarias 54

4.3.2. Pruebas de GUI y navegación 55

4.3.3. Pruebas de usuario 56

5. Conclusiones **57**

Bibliografía **59**

Índice de figuras

1.1. Logo empresa acogida 10

1.2. Arquitectura del sistema 12

2.1. Ciclo de vida de Scrum 14

2.2. Tareas y Diagrama de Grantt 18

3.1. IntelliJ IDEA 23

3.2. Parte del fichero pom.xml 24

3.3. Estructura proyecto Spring Boot 25

3.4. Plantilla Bootstrap del proyecto 26

3.5. Diagrama de clases 27

3.6. Tablero Kanban del proyecto 28

4.1. Diagrama de casos de uso 30

4.2. Diagrama de secuencia 31

4.3. Prototipo: login 39

4.4. Prototipo: Panel de control 39

4.5. Prototipo: zonas 40

4.6. Prototipo: campañas 40

4.7. Prototipo: contenidos 41

4.8. Prototipo: configuración de la cuenta 41

4.9. Arquitectura 42

4.10. Modelo Vista Controlador 44

4.11. Diagrama de clases DAO 45

4.12. Patrón Observador 46

4.13. Login 48

4.14. Panel de control 48

4.15. Zonas creadas 49

4.16. Añadir zona 50

4.17. Contenidos creados 50

4.18. Añadir contenido 51

4.19. Campañas creadas 52

4.20. Añadir campaña 52

4.21. Configuración de la cuenta 53

4.22. Informe de test 55

4.23. Test de Zona 55

Capítulo 1

Introducción

El desarrollo del proyecto que se presenta en este documento es el resultado del trabajo realizado para la asignatura *EI1054 Prácticas Externas y Trabajo Final de Grado* del Grado en Ingeniería Informática en la Universidad Jaume I de Castellón.

En este capítulo introductorio describimos la motivación que nos ha llevado al desarrollo del proyecto, así como sus objetivos y su descripción.

1.1. Justificación y motivación del proyecto

Este proyecto ha sido desarrollado en Mestral Telecomunicaciones S.L., empresa especializada en redes y telecomunicaciones con una experiencia del personal de la empresa, en algunos casos, de 25 años. Si bien en los inicios dedicó su labor profesional a la informática, fue en 1996 cuando apostó por las redes y telecomunicaciones. Siempre ha buscado la innovación como herramienta para ofrecer a sus clientes una tecnología que permitiera un ahorro de costes y una mayor productividad.

Figura 1.1: Logo empresa de acogida.

Este proyecto consiste en el desarrollo de una plataforma web para la gestión de balizas (beacons) que nos servirán para poder enviar información sobre el local propietario de éstas a los clientes registrados en la aplicación. El propósito de este sistema tiene como finalidad ayudar a las empresas a promocionar sus productos y ofertas. Las balizas son dispositivos de bajo consumo que emiten una señal broadcast. Utilizan conexión bluetooth de bajo consumo para transmitir datos a dispositivos móviles sin necesidad de una sincronización de los aparatos.

Cada vez que una baliza se sincronice con un aparato móvil, el usuario recibirá la información que esté asociada a la baliza. Para acceder a la información de cada baliza, así como a la del usuario, utilizaremos una base de datos. Esta base de datos estará gestionada desde la plataforma web. La aplicación móvil sólo se encargará de, a partir de la señal de la baliza, buscar la información que tiene asociada y mostrársela al cliente.

1.2. Objetivos del proyecto

El objetivo de este proyecto es el desarrollo de una aplicación web que permita llevar un control de las balizas de las que disponemos y la información que se le asocia a cada una de ellas. Además, los usuarios de esta aplicación serán capaces de llevar un estudio del comportamiento y de las estadísticas.

Mediante este sistema se pretende:

- Facilitar la gestión de las balizas que disponemos.
- Acceder y mantener la información asociada a las balizas.
- Crear y gestionar campañas de marketing mediante la interacción con la base de datos.

Para desarrollar este sistema, se ha realizado un proceso íntegro de ingeniería de software que a continuación será detallado. En primer lugar, se va a explicar la fase de recopilación de requisitos y análisis; en segundo lugar, se describirá la arquitectura, el diseño a nivel de componentes e interfaz de usuario; finalmente, la fase de implementación y pruebas.

1.3. Descripción del proyecto

En este proyecto se ha desarrollado de forma conjunta un sistema compuesto por una aplicación web y una aplicación móvil. Mi tarea era desarrollar la aplicación web y, aparte, crear de forma conjunta con un compañero de prácticas la base de datos. La aplicación web se apoya en una base de datos para recoger todos los datos referentes a cada una de las balizas y usuarios, así como todos los contenidos y campañas que tienen asociadas. La aplicación móvil se encargará de detectar los beacons y mostrar el contenido asignado a estos, para ello realizará consultas a la base de datos cada vez que detecte uno.

Los usuarios que van a utilizar esta aplicación son los usuarios de la empresa que estarán registrados en la aplicación. Cada cliente debe tener su propio usuario y contraseña para poder acceder a la gestión de la información.

En la Figura 1.2 podemos observar la arquitectura de todo el sistema a desarrollar.

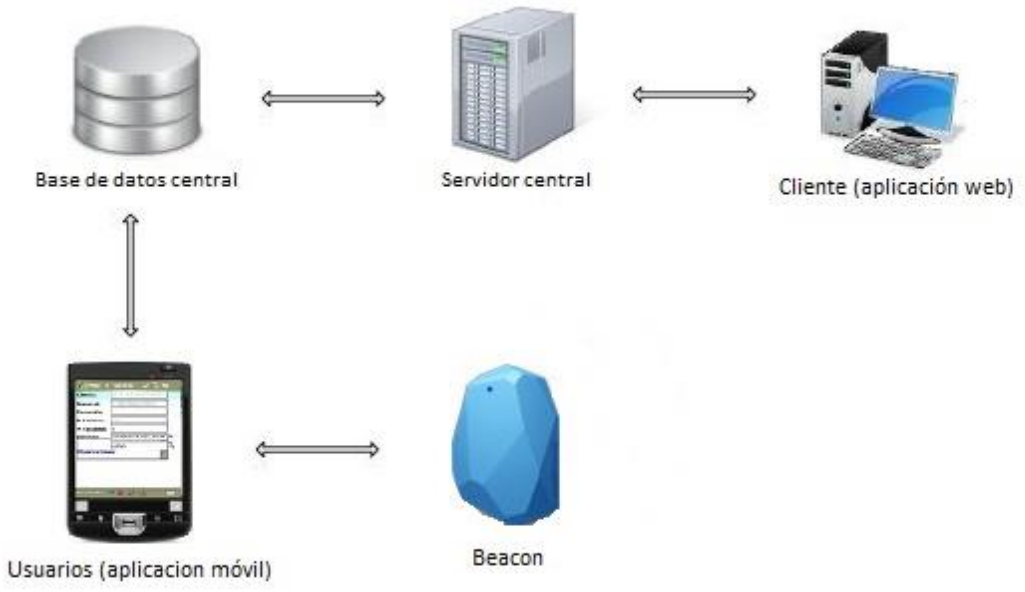

Figura 1.2. Arquitectura del sistema.

Las funcionalidades que permitirá realizar la aplicación a los usuarios son las siguientes:

- Autenticación de usuarios para utilizar la aplicación.
- Crear nuevas zonas a las que se le asignan balizas.
- Crear nuevos contenidos asociados a las zonas que tenemos creadas.
- Crear nuevas campañas de marketing asociadas a las zonas que tenemos creadas.
- Elaboración de estadísticas y estudios de comportamiento.

Para desarrollar este sistema será necesario diseñar e implementar una base de datos que persistirá la información de las balizas, usuarios y contenidos, entre otros, en la aplicación web.

Además, el diseño de la aplicación pretende ser agradable e intuitivo, ya que el objetivo es facilitar al usuario la gestión de toda la información relevante a las balizas.

Capítulo 2

Planificación del proyecto

En este apartado se presenta la metodología utilizada para el desarrollo del proyecto, justificando su uso y comparándola con otras estudiadas a lo largo de la carrera. También se mostrará la planificación de las distintas tareas del proyecto, analizándolas para presentar una estimación de la duración y coste de las mismas.

2.1. Metodología de trabajo

Este proyecto ha sido llevado a cabo mediante el uso de una metodología ágil de desarrollo de proyectos software. Se ha tomado esta decisión por los motivos siguientes:

- Efectividad y flexibilidad ante cambios de requisitos.
- Metodología basada en iteraciones de una a cuatro semanas.
- Al final de cada iteración, se dispone de un producto funcional.
- Evitar problemas como retrasos de tiempo y complejidad.

Entre las distintas metodologías que existe, en este proyecto se ha utilizado la que probablemente sea la más conocida: SCRUM[1]. Se ha elegido esta metodología ya que se adecúa a las características del proyecto.

La Figura 2.1 muestra el ciclo de vida de esta tecnología, donde:

- El **Product Backlog** es la pila de requisitos que debe implementar el sistema. Se compone de funcionalidades y tareas que debe realizar el programador que está en constante evolución durante todo el ciclo de vida, hasta el cierre del sistema.

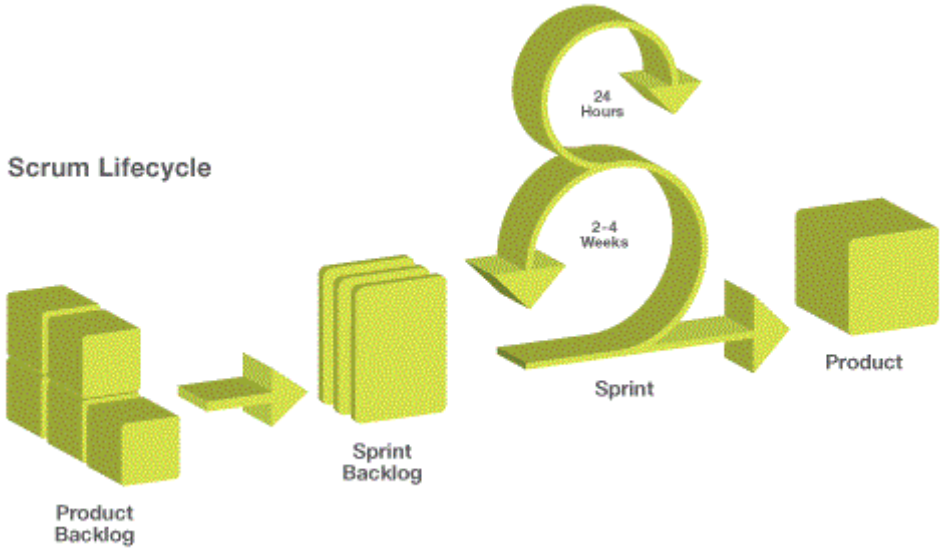

Figura 2.1: Ciclo de vida de Scrum.

- El **Sprint Backlog** es la pila de tareas que se van a desarrollar en el Sprint actual. Para extraer estas tareas del *Product Backlog* se pueden llevar a cabo técnicas de priorización de tareas, según las necesidades del usuario y la complejidad de éstas. Para llevar el seguimiento del estado de estas tareas, se ha utilizado la técnica *Kanban*[2], que se describe posteriormente.

- El **Sprint** es la iteración propiamente dicha, en nuestro caso, al realizar los informes semanalmente, tiene una duración de 7 dias. En esta iteración, los programadores desarrollan las tareas definidas en el *Sprint Backlog* y, al finalizar uno de éstos, el resultado es un producto funcional que el cliente es capaz de ejecutar y tiene un valor añadido respecto a la iteración.

Además, al inicio y fin de cada iteración se realizan reuniones de planificación de la entrega y construcción del *Backlog*, e inspección del incremento integrado en cada *Sprint*. La metodología también define que se han de realizar reuniones diarias de 15 minutos aproximadamente para identificar problemas y obstáculos para resolverlos lo antes posible.

El equipo de desarrollo de *SCRUM* está formado por los siguientes roles:

- **Product Owner:** Decide la funcionalidad del producto y es el responsable de priorizar las tareas a desarrollar en cada iteración. Representa el usuario del sistema.

- **Scrum Manager:** Motiva y coordina al equipo y es responsable de detectar los problemas que pueden surgir durante el proceso.

- **Team Scrum:** Crean el producto en sí y son un equipo multidisciplinar de programadores, *testers*, analistas, etcétera.

Todos los miembros del equipo de desarrollo han de conocer con detalle la visión del P*roduct Owner* y han de colaborar regularmente y de manera directa con este.

Aunque este proyecto se ha basado en una metodología ágil, en cada iteración se ha realizado un proceso de ingeniería de software clásico y, por claridad, este proyecto se ha documentado de esta forma; el apartado de desarrollo de la aplicación se divide en cuatro fases bien diferenciadas:

- **Análisis de requisitos.** En esta fase, se estudian las necesidades del cliente y qué espera que el sistema ofrezca. También, se identifican los distintos actores que estarán involucrados así como qué rol van a tomar en el sistema. Como resultado, se obtiene un diagrama de casos de uso, una recopilación de requisitos de usuario formalizada y una serie de prototipos sencillos de la interfaz de usuario.

- **Diseño e implementación.** En esta fase tiene lugar el diseño de la arquitectura, que ofrece una visión externa del sistema, con todos los componentes que lo forman, a alto nivel. Además, se realiza el diseño a nivel de componentes, diseño de clases, identificación de patrones de diseño, etc. Finalmente, consultando los prototipos obtenidos en la fase anterior, se obtiene el diseño real de la interfaz de usuario y se lleva a cabo la construcción propiamente dicha del sistema.

- **Validación y verificación.** Para lanzar el sistema a producción es necesario probarlo en distintos escenarios para asegurar su correcto funcionamiento. Se han llevado a cabo pruebas unitarias, en las que se prueba la lógica interna de un componente independiente de otros, y pruebas de usuario para asegurar el correcto funcionamiento y calidad del sistema.

- **Despliegue o implantación.** En esta fase, el producto es lanzado en un entorno de producción. En este caso, esta fase se ha dividido en dos: lanzar el producto para un grupo reducido de usuarios y estos reportar errores, en caso de encontrarlos, y finalmente, una vez el sistema funciona correctamente durante un tiempo establecido, realizar el lanzamiento masivo para todos los usuarios.

Como se ha dicho anteriormente, en este proyecto se han puesto en práctica metodologías ágiles, por tanto, el desarrollo de estas cuatro fases no ha sido lineal. Para cada iteración se ha llevado a cabo un pequeño proceso de ciclo de vida clásico formado por estas fases. De esta forma, al finalizar cada *sprint*, se obtiene una parte funcional del sistema que el usuario puede validar.

2.2. Planificación temporal

En este apartado se muestran las tareas a realizar para llevar a cabo este proyecto. Debido a que el tiempo total de estancia por parte del alumno en la empresa debe ser de unos 38 días, ya que son 300 horas a ocho horas por jornada.

En la Figura 2.2 se pueden observar estas tareas junto a un diagrama de *Gantt*, donde se muestra la duración y la disposición en el tiempo de cada una. Estas tareas son las típicas de un desarrollo ciclo de vida clásico. En cambio, el proyecto se ha desarrollado mediante una metodología ágil, por lo que este diagrama no es representativo del desarrollo, ya que Scrum no permite realizar una planificación de todo el proceso de desarrollo del proyecto.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Proyecto Final de grado**	**38 días**	**mié 01/06/16**	**vie 22/07/16**
2	**Inicio**	**3 días**	**mié 01/06/16**	**vie 03/06/16**
3	Definir el proyecto con el tutor y el supervisor	1 día	mié 01/06/16	mié 01/06/16
4	Definir el método de trabajo y la documentación	1 día	jue 02/06/16	jue 02/06/16
5	Definir el formato y los estándares de trabajo	1 día	vie 03/06/16	vie 03/06/16
6	**Planificación del proyecto**	**3 días**	**lun 06/06/16**	**mié 08/06/16**
7	Definir tareas y estimar el tiempo	1 día	lun 06/06/16	lun 06/06/16
8	Redactar y documentar la propuesta técnica	2 días	mar 07/06/16	mié 08/06/16
9	Entregar la propuesta técnica	0 días	mié 08/06/16	mié 08/06/16
10	**Documentación**	**5 días**	**jue 09/06/16**	**mié 15/06/16**
11	Informarse del funcionamiento de los beacons	1 día	jue 09/06/16	jue 09/06/16
12	Documentarse sobre el lenguaje y las tecnologías para el desarrollo de la plataforma	2 días	vie 10/06/16	lun 13/06/16
13	Documentarse sobre el funcionamiento de la base de datos	2 días	mar 14/06/16	mié 15/06/16
14	**Desarrollo técnico del proyecto**	**27 días**	**jue 16/06/16**	**vie 22/07/16**
15	**Crear la base de datos**	**4 días**	**jue 16/06/16**	**mar 21/06/16**
16	Diseñar y crear la base de da	3 días	jue 16/06/16	lun 20/06/16
17	Insertar datos iniciales en la base de datos	1 día	mar 21/06/16	mar 21/06/16
18	**Desarrollo de la plataforma web**	**15 días**	**mié 22/06/16**	**mar 12/07/16**
19	Programación de la plataforr	12 días	mié 22/06/16	jue 07/07/16
20	Pruebas	3 días	vie 08/07/16	mar 12/07/16
21	**Diseño de la plataforma web**	**4 días**	**mié 13/07/16**	**lun 18/07/16**
22	Definir secciones y menús del backend	3 días	mié 13/07/16	vie 15/07/16
23	Configurar el backend	1 día	lun 18/07/16	lun 18/07/16
24	**Informes**	**3 días**	**mar 19/07/16**	**jue 21/07/16**
25	Redactar informes	3 días	mar 19/07/16	jue 21/07/16
26	Entregar informes	0 días	jue 21/07/16	jue 21/07/16
27	**Puesta en marcha**	**1 día**	**vie 22/07/16**	**vie 22/07/16**
28	Implantación	1 día	vie 22/07/16	vie 22/07/16
29	Entrega final	0 días	vie 22/07/16	vie 22/07/16

Figura 2.2. Tareas y Diagrama de Gantt

En cuanto a la planificación del desarrollo del proyecto, se realizaron 5 sprints en los que se fueron desarrollando las diferentes partes del proyecto. Los sprints se dividieron de la siguiente forma:

- **Sprint 1:** Durante el primer sprint se desarrolló el sistema de login y la parte correspondiente para mostrar las zonas que se crean.

- **Sprint 2:** En el segundo sprint se desarrolló el resto de las funcionalidades relacionadas con las zonas: añadir, editar y eliminar.

- **Sprint 3:** Durante el tercer sprint se empezó a desarrollar la parte que muestra y añade contenidos.

- **Sprint 4:** En el cuarto sprint se terminó de desarrollar las funcionalidades referentes de los contenidos: editar y borrar, también se realizó la parte que muestra las campañas que tenemos creadas.

- **Sprint 5:** En el último sprint se desarrolló la gestión de las campañas, las funcionalidades para añadir, editar y eliminar las campañas.

2.3. Incidencias y problemas

Se produjeron dos incidencias importantes en el inicio del proyecto, una por retraso en la adjudicación de un equipo de trabajo por parte de la empresa y otra por una mala estimación de horas en la tarea de documentación por parte del cliente, ya que, a pesar de tener un perfil técnico, no poseía un perfil cercano a las tecnologías utilizadas en el proyecto. Estas incidencias afectaron al desarrollo del proyecto, retrasando el tiempo de finalización del mismo.

La primera incidencia se produjo al inicio de la estancia y fue causada por un retraso de un envío, que al final no se produjo, de un equipo para poder empezar a desarrollar el proyecto. Cosa que al final se resolvió utilizando mi propio equipo de trabajo (ordenador portátil).

La segunda incidencia fue causada por la falta de documentación para informarme acerca de la tecnología que se iba a gestionar en la aplicación web, además de la falta de conocimiento que tenía sobre el lenguaje y las tecnologías de desarrollo que iba a utilizar.

Capítulo 3

Entorno tecnológico

Para el desarrollo del proyecto, se necesitan una serie de herramientas y tecnologías que permitirán la total puesta en marcha de la aplicación.

Las herramientas y tecnologías son las siguientes:

- IntelliJ IDEA[3]: Es un ambiente de desarrollo integrado (IDE) para el desarrollo de programas informáticos.

- Apache Maven[4]: Esta herramienta de línea de comandos permite la creación de proyectos Java con una determinada estructura de paquetes y la gestión de dependencias automática sin necesidad de incluir los ficheros necesarios.

- Spring Boot[5]: Es una parte de Spring, un framework para el desarrollo de aplicaciones en Java, que nos permite crear diferentes tipos de aplicaciones de una manera rápida y sencilla. Sus características principales son que provee una serie de elementos que nos permiten desarrollar diferentes tipos de aplicaciones de forma casi inmediata.

- Twitter Bootstrap[6]: Es un framework HTML, CSS y JavaScript para el desarrollo de sitios y aplicaciones web.

- MySQL[7]: Es un sistema de gestión de bases de datos relacionales, multihilos y multiusuario.

- Trello[8]: Herramienta web que permite realizar el seguimiento de un proyecto, permitiendo implementar tableros Kanban de manera sencilla.

Estas herramientas han sido escogidas porque son consideradas bastante prácticas para la realización del proyecto tanto por mi parte cómo por parte del supervisor de la empresa. En las siguientes secciones se explica en profundidad cada una de estas tecnologías y herramientas.

3.1. IntelliJ IDEA

IntelliJ IDEA es un IDE (Entorno Integrado de Desarrollo) multiplataforma para el desarrollo de programas informáticos. Esta desarrollado por JetBrains (anteriormente conocido como IntelliJ) y tiene dos ediciones: Community Edition (Gratuito) y Ultimate Edition (De pago). Para el desarrollo de este proyecto se utilizó la edición Ultimate Edition.

Para realizar las pruebas necesarias en nuestro proyecto utilizamos el framework JUnit, que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

En la Figura 3.1 tenemos la ventana de nuestro proyecto en IntelliJ IDEA.

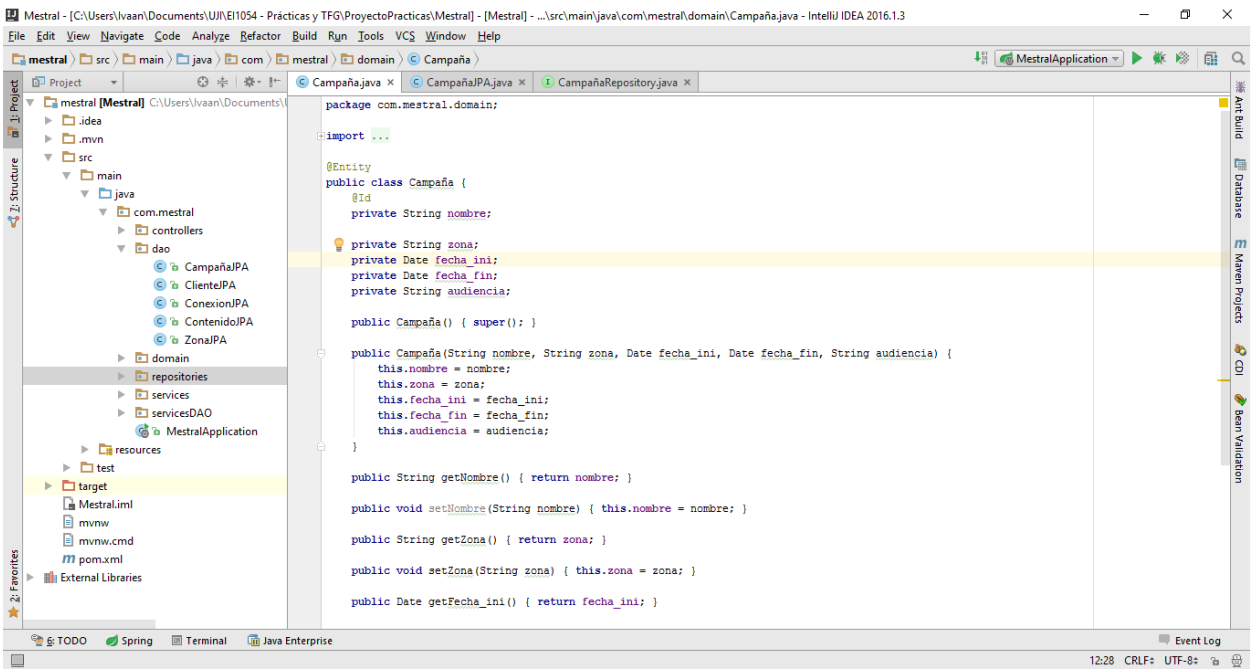

Figura 3.1. IntelliJ IDEA.

3.2. Apache Maven

Maven es una herramienta para la gestión y la construcción de proyectos software escritos en lenguaje Java.

Esta herramienta facilita algunos procesos en el desarrollo como la resolución de dependencias, el nombrado de versiones o el generado de informes. Además, permite la gestión de perfiles, de esta forma, se podría disponer de una configuración distinta para cada perfil.

Toda la configuración del proyecto referente a Maven, ya sean perfiles, dependencias u otros se encuentra en un fichero XML (pom.xml) que se encuentra alojado en el directorio raíz del proyecto.

En la Figura 3.2 podemos observar una parte del fichero pom.xml, donde se muestran diferentes dependencias que necesitamos importar para utilizar Spring Boot.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.5.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

Figura 3.2. Parte del fichero pom.xml.

3.3. Spring Boot

Spring Boot es un nuevo módulo de la plataforma Spring cuyo objetivo es simplificar la creación de aplicaciones y servicios listos para ejecutarse. Sus objetivos son los siguientes:

- Ofrecer una forma muy sencilla de arrancar desarrollos Spring.
- Disponer de funcionalidad out-of-the-box en función de la naturaleza del proyecto.
- Dotar a las aplicaciones de herramientas que permitan su monitorización y auditoría.
- No necesita generación de código ni configuración XML.

A continuación, en la Figura 3.3 podemos ver la estructura de directorios de nuestro proyecto basado en Spring Boot.

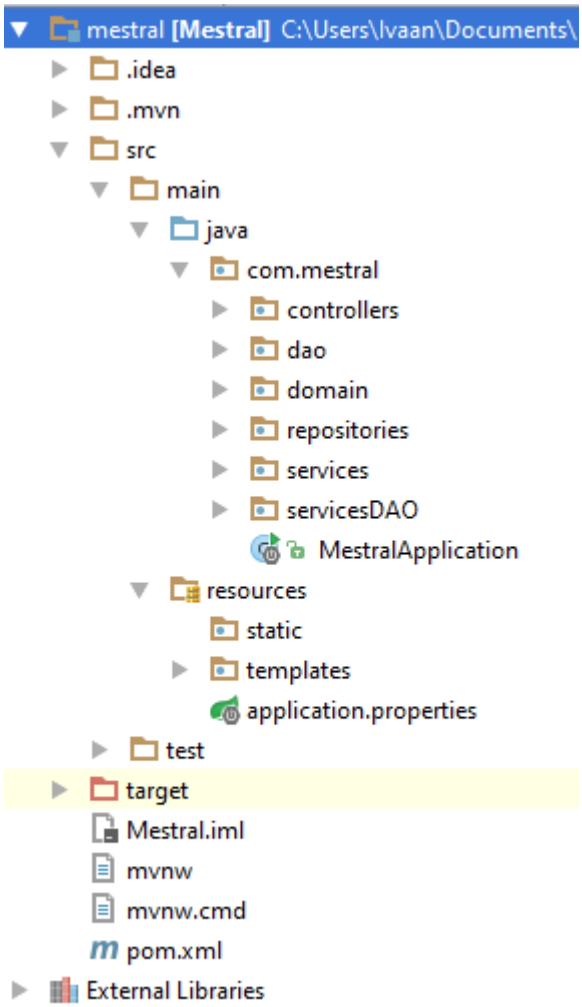

Figura 3.3. Estructura proyecto Spring Boot.

3.4. Twitter Bootstrap

Twitter Bootstrap es un framework de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

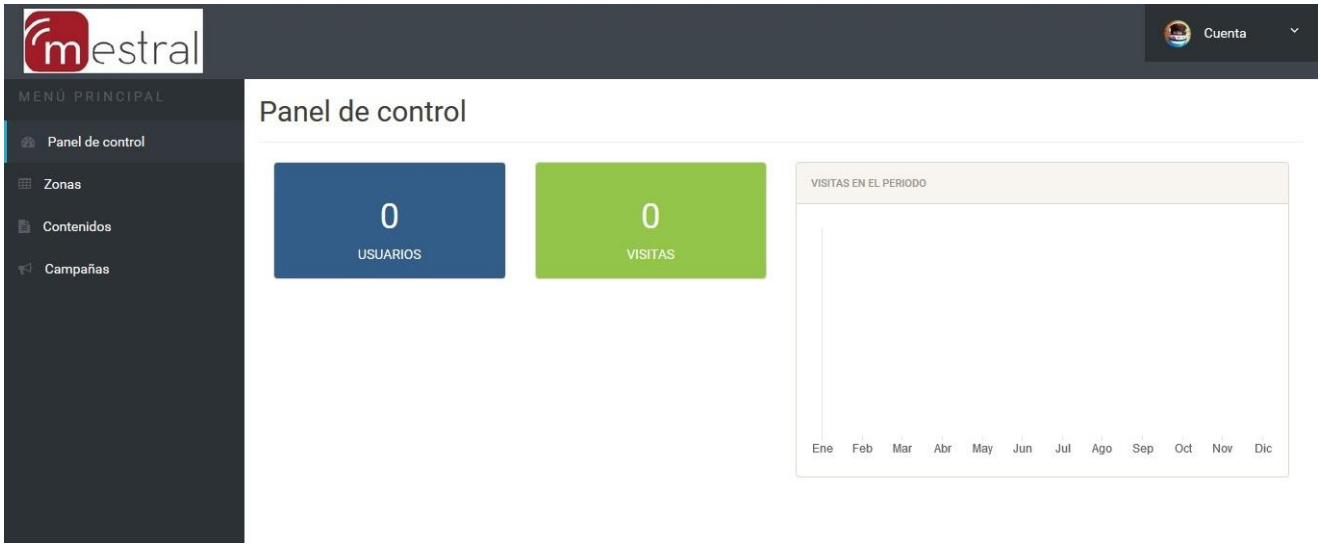

Figura 3.4. Plantilla Bootstrap del proyecto.

Como podemos observar en la Figura 3.4 hemos utilizado diferentes objetos del framework, como el menú de la parte de la izquierda, el botón para añadir un contenido, la tabla central donde se observan los diferentes contenidos o el botón desplegable de la parte de arriba a la derecha.

3.5. MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

En la Figura 3.5 observamos el esquema de la base de datos creada para el proyecto.

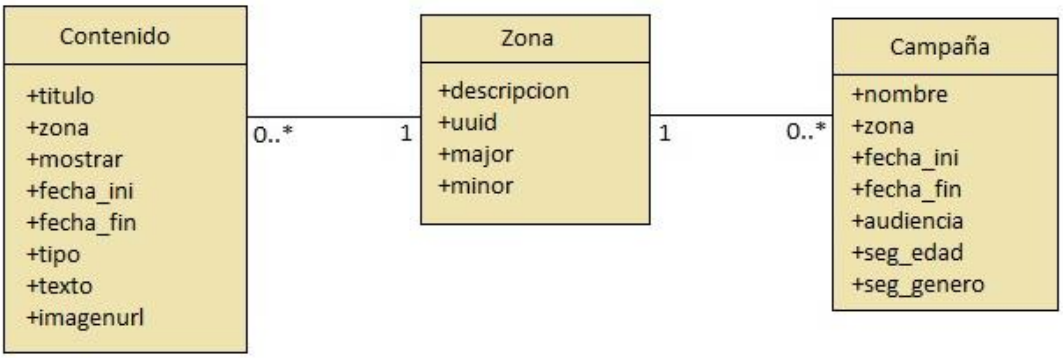

Figura 3.5. Diagrama de clases.

3.6. Trello

Trello es una herramienta web de la compañía Trello Inc. que permite al usuario gestionar las tareas de un proyecto de cualquier ámbito. Está fuertemente enfocado a proyectos que se llevan a cabo utilizando metodologías ágiles, ya que la página principal de la herramientas es un tablero Kanban donde el usuario puede generar tareas y clasificarlas según su estado: pendiente, en proceso o resuelta. Además permite al usuario crear sprints y asignar tareas a cada uno de ellos.

En la Figura 3.6 observamos el tablero Kanban de nuestro proyecto en una etapa concreta del mismo.

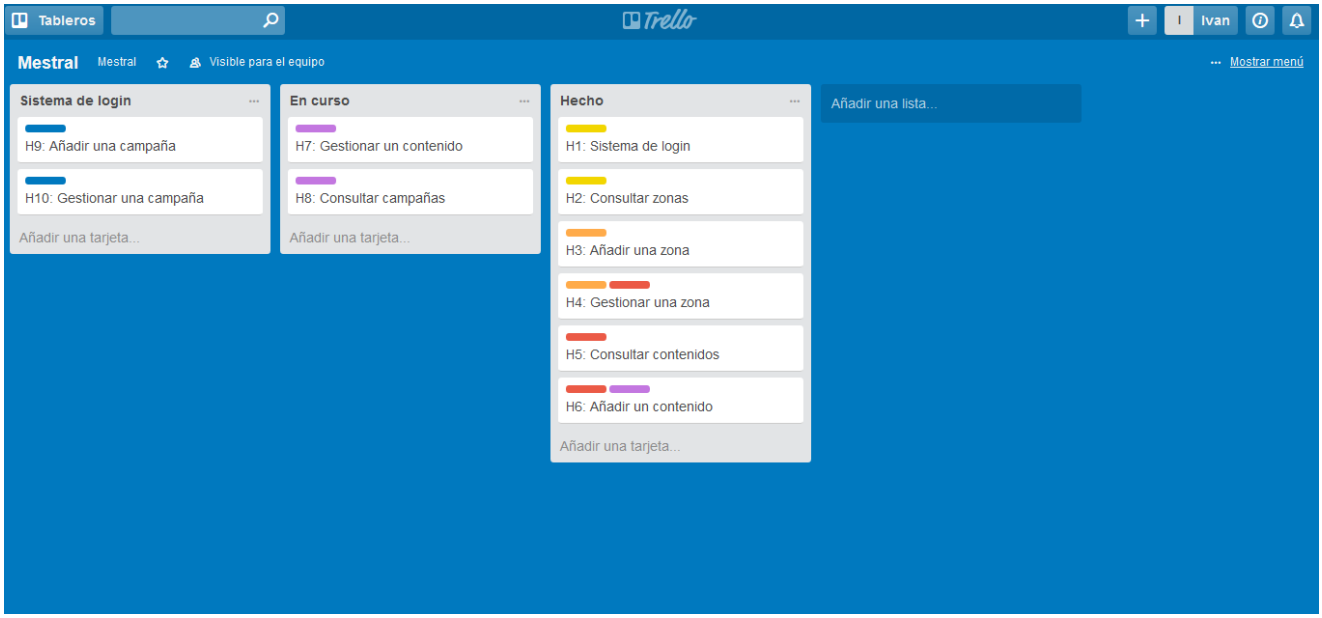

Figura 3.6. Tablero Kanban del proyecto.

Capítulo 4

Desarrollo de la aplicación

En los siguientes apartados se va a especificar la recopilación de requisitos de usuario, seguido del análisis del entorno y actores involucrados, diseño a nivel de arquitectura, componentes e interfaz gráfica y, finalmente, la implementación propiamente dicha del software.

4.1. Análisis

En esta fase se define formalmente todos los elementos que constituyen el contexto del problema. Para ello, se va a presentar los requisitos funcionales del sistema y, como consecuencia de estos, los requisitos de software y hardware. Previamente, se mostrará un diagrama de casos de uso para apreciar qué actores realizan qué determinadas acciones sobre el sistema.

4.1.1. Casos de uso

Los casos de uso son una herramienta para el análisis de proyectos software. Haciendo uso de ellos se puede observar gráficamente la interacción entre los actores involucrados y el propio sistema. En este proyecto, se puede identificar solo un actor: El administrador del sistema.

La Figura 4.1 muestra el diagrama de casos de uso, con el actor y las acciones que realiza.

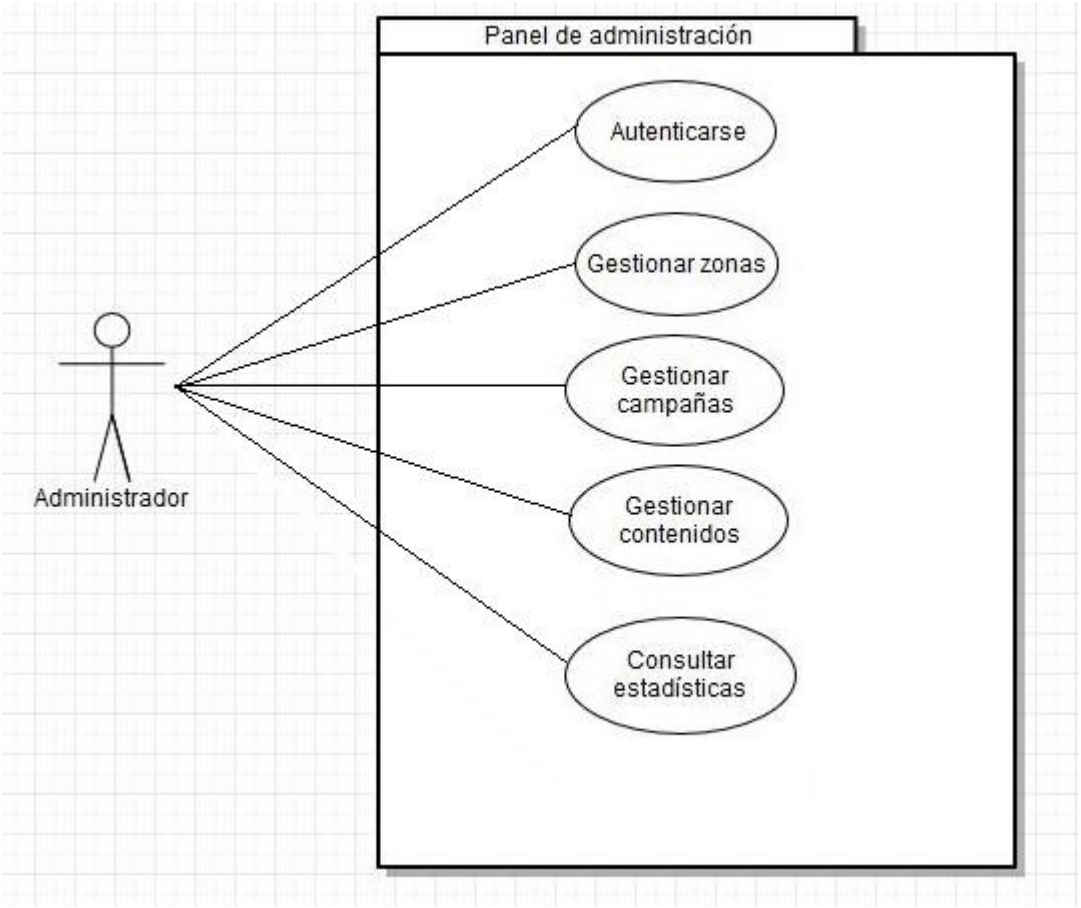

Figura 4.1. Diagrama de casos de uso.

Como se puede apreciar, se muestra al actor principal, el administrador, de nuestro proyecto. El rectángulo que se puede observar representa el alcance del sistema, es decir, aquellas acciones que se realizan dentro del contexto de la aplicación.

Para realizar todas estas acciones, la aplicación se apoya en una base de datos donde se guarda toda la información relacionada con los usuarios y los beacons.

A continuación, en la Figura 4.2 podemos observar la secuencia de acciones que ocurren en el caso de uso de gestionar zonas cuando queremos agregar una zona nueva. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo.

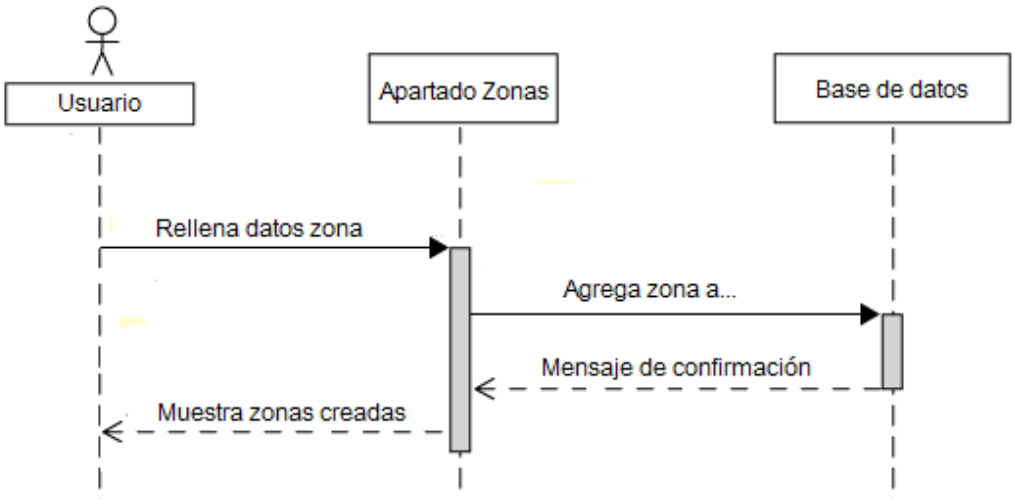

Figura 4.2. Diagrama de secuencia

4.1.2. Requisitos funcionales

El objetivo de los requisitos funcionales de usuario es definir el alcance del sistema, es decir, las características, a nivel funcional, que debe satisfacer el software. La definición formal de estos requisitos es necesaria para comprender el problema a resolver.

A continuación se muestran las tablas que definen estos requisitos. Cada requisito contiene un identificador, nombre, descripción y una secuencia de pasos que definen su ciclo de vida.

Requisito funcional	
Código	FR01
Nombre	Autenticarse
Descripción	El sistema ha de permitir a los usuarios autenticarse para acceder a este. Si el acceso es satisfactorio, el sistema guarda los datos para realizar este proceso automáticamente las próximas veces.

Secuencia normal	
1	Introducir usuario.
2	Introducir contraseña.
3	Seleccionar opción de aceptar.
Importancia	Media
Comentarios	El sistema debe mostrar un mensaje de error en caso de que los datos introducidos sean erróneos.

Requisito funcional	
Código	FR02
Nombre	Crear zona
Descripción	El sistema ha de permitir a los usuarios crear nuevas zonas para asignarles ibeacons.

Secuencia normal	
1	Seleccionar la pestaña zonas del menú.
2	Seleccionar el botón añadir zona.
3	Introducir la información de la zona.
Importancia	Alta
Comentarios	El sistema también ha de permitir editar y borrar las zonas creadas.

Requisito funcional	
Código	FR03
Nombre	Borrar zona
Descripción	El sistema ha de permitir a los usuarios borrar zonas creadas cuando ya no sean de utilidad.

Secuencia normal	
1	Seleccionar la pestaña zonas del menú.
2	Seleccionar el botón borrar zona.
Importancia	Media
Comentarios	El sistema debe eliminar la zona de la lista de zonas del sistema.

Requisito funcional	
Código	FR04
Nombre	Editar zona
Descripción	El sistema ha de permitir a los usuarios editar una zona guardada.

Secuencia normal	
1	Seleccionar la pestaña zonas del menú.
2	Seleccionar el botón editar zona.
3	Introducir la nueva información de la zona.
Importancia	Media
Comentarios	Una vez editada se mostrará en la lista de zonas con la información actualizada.

Requisito funcional	
Código	FR05
Nombre	Consultar zonas
Descripción	El sistema ha de permitir a los usuarios consultar la lista de zonas que hay guardadas en el sistema.

Secuencia normal	
1	Seleccionar la pestaña zonas del menú.
2	Visualizar la lista de zonas del sistema.
Importancia	Media
Comentarios	Este apartado permite consultar directamente la información de las zonas.

Requisito funcional	
Código	FR06
Nombre	Crear campaña
Descripción	El sistema ha de permitir a los usuarios crear nuevas campañas.

Secuencia normal	
1	Seleccionar la pestaña campañas del menú.
2	Seleccionar el botón añadir campaña.
3	Introducir la información de la campaña.
Importancia	Media
Comentarios	El sistema también ha de permitir editar y borrar las campañas creadas.

Requisito funcional	
Código	FR07
Nombre	Borrar campaña
Descripción	El sistema ha de permitir a los usuarios borrar campañas creadas cuando ya no sean de utilidad.

Secuencia normal	
1	Seleccionar la pestaña campañas del menú.
2	Seleccionar el botón borrar campaña.
Importancia	Media
Comentarios	El sistema debe eliminar la campaña de la lista de campañas del sistema.

Requisito funcional	
Código	FR08
Nombre	Editar campaña
Descripción	El sistema ha de permitir a los usuarios editar una campaña guardada.

Secuencia normal	
1	Seleccionar la pestaña campañas del menú.
2	Seleccionar el botón editar campaña.
3	Introducir la nueva información de la campaña.
Importancia	Media
Comentarios	Una vez editada se mostrará en la lista de campañas con la información actualizada.

Requisito funcional	
Código	FR09
Nombre	Consultar campañas
Descripción	El sistema ha de permitir a los usuarios consultar la lista de campañas que hay guardadas en el sistema.

Secuencia normal	
1	Seleccionar la pestaña campañas del menú.
2	Visualizar la lista de campañas del sistema.
Importancia	Media
Comentarios	Este apartado permite consultar directamente la información de las campañas.

Requisito funcional	
Código	FR10
Nombre	Crear contenido
Descripción	El sistema ha de permitir a los usuarios crear nuevos contenidos.

Secuencia normal	
1	Seleccionar la pestaña contenidos del menú.
2	Seleccionar el botón añadir contenido.
3	Introducir la información del contenido.
Importancia	Alta
Comentarios	El sistema también ha de permitir editar y borrar los contenidos creados.

Requisito funcional	
Código	FR11
Nombre	Borrar contenido
Descripción	El sistema ha de permitir a los usuarios borrar contenidos creados cuando ya no sean de utilidad.

Secuencia normal	
1	Seleccionar la pestaña contenidos del menú.
2	Seleccionar el botón borrar contenido.
Importancia	Media
Comentarios	El sistema debe eliminar el contenido de la lista de contenidos del sistema.

Requisito funcional	
Código	FR12
Nombre	Editar contenido
Descripción	El sistema ha de permitir a los usuarios editar un contenido guardado.

Secuencia normal	
1	Seleccionar la pestaña contenidos del menú.
2	Seleccionar el botón editar contenido.
3	Introducir la nueva información del contenido.
Importancia	Media
Comentarios	Una vez editado se mostrará en la lista de contenidos con la información actualizada.

Requisito funcional	
Código	FR13
Nombre	Consultar contenidos
Descripción	El sistema ha de permitir a los usuarios consultar la lista de contenidos que hay guardados en el sistema.

Secuencia normal	
1	Seleccionar la pestaña contenidos del menú.
2	Visualizar la lista de contenidos del sistema.
Importancia	Media
Comentarios	Este apartado permite consultar directamente la información de los contenidos.

Requisito funcional	
Código	FR14
Nombre	Consultar estadísticas
Descripción	El sistema ha de permitir a los usuarios consultar las estadísticas de las distintas zonas creadas

Secuencia normal	
1	Seleccionar la pestaña panel de control del menú.
2	Visualizar las estadísticas de cada zona y las visitas.
Importancia	Media
Comentarios	Este apartado permite consultar directamente las estadísticas de las distintas zonas y detalles.

4.1.3. Requisitos de datos

Para satisfacer los requisitos de usuario expuestos anteriormente, es necesario definir las entidades y atributos que se van a almacenar, es decir, los requisitos de datos.

A continuación, se muestran las tablas que representan cada entidad y las propiedades que contendrán cada una de estas.

Requisito de datos	
Código	DR01
Nombre	Zona
Datos	Para cada zona, el sistema debe almacenar una descripción o nombre y los datos del beacon asociado a esa zona, UUID, major y minor.
Comentarios	Cada zona debe tener asociada un beacon diferente.

Requisito de datos	
Código	DR02
Nombre	Campaña
Datos	Para cada campaña, el sistema debe almacenar el nombre, la zona a la que pertenece, la fecha inicial y final, la audiencia a la que va dirigida y, si se quiere, añadir segmentaciones por rango de edad o género.
Comentarios	Las segmentaciones no son obligatorias, solo se introducen si se seleccionan.

Requisito de datos	
Código	DR03
Nombre	Contenido
Datos	Para cada contenido, el sistema debe almacenar el título, la zona, cuando lo quiere mostrar e información especial de cada contenido.
Comentarios	Podemos elegir el tipo de contenido que queremos crear.

4.1.4. Requisitos software y hardware

La aplicación que se describe en este proyecto ha sido testada sobre varios ordenadores con sistema operativo Windows (7, 8 y 10) y Ubuntu. Los requisitos para acceder a la aplicación web son:

- Ordenador con sistema operativo Windows 7 o superior y sistemas GNU/Linux.

- Procesador Dual Core o superior.
- Conexión a Internet.

Los navegadores web que se han usado son Google Chrome, Mozilla Firefox y Microsoft Edge. Para probar la aplicación en un servidor local hemos utilizado Apache Tomcat[9].

4.1.5. Prototipos de la interfaz

Para terminar con la fase del análisis, se han creado prototipos de las interfaces de usuario para las funcionalidades más significativas de la aplicación: l*ogin*, panel de control, zonas, campañas, contenidos, usuarios y configuración de la cuenta. Estos prototipos no representan el aspecto definitivo de las pantallas, sino que su objetivo es plasmar una idea más visual de la aplicación para permitir la verificación de las funcionalidades, sin entrar en aspectos de diseño. Para realizar estos prototipos hemos utilizado la herramienta Paint[10].

Las figuras 4.3, 4.4, 4.5, 4.6, 4.7 y 4.8 muestran los prototipos para las pantallas de *login*, panel de control, zonas, campañas, contenidos y configuración de la cuenta.

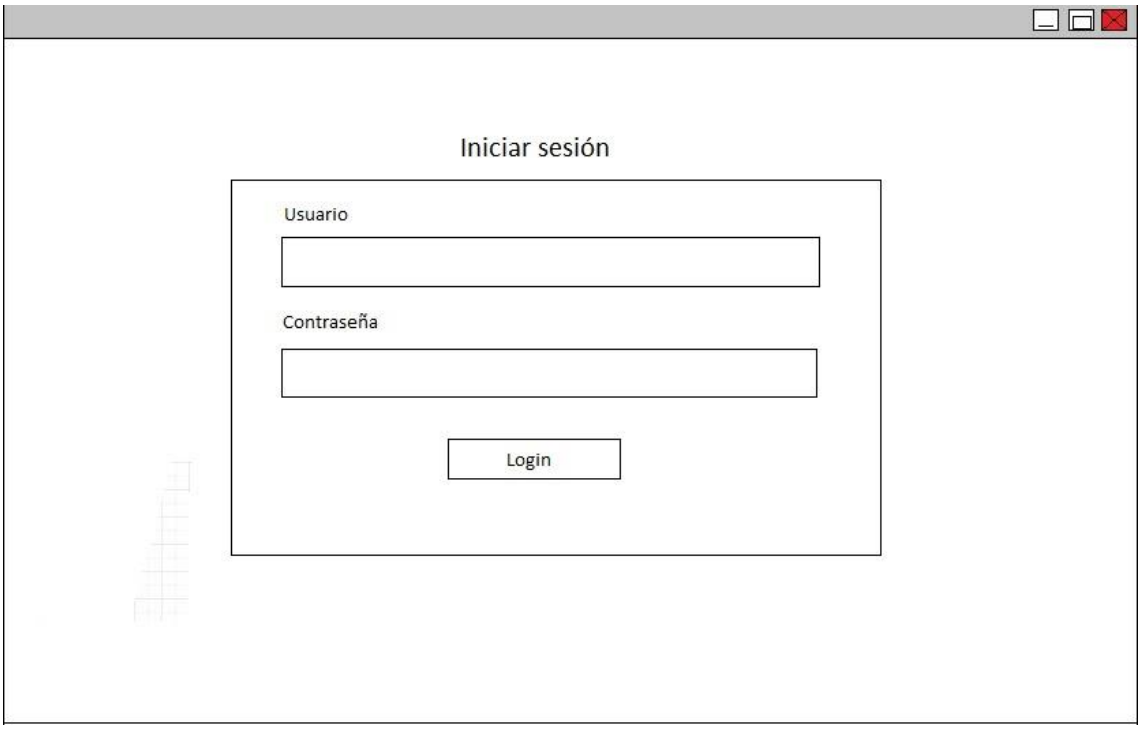

Figura 4.3. Prototipo: login.

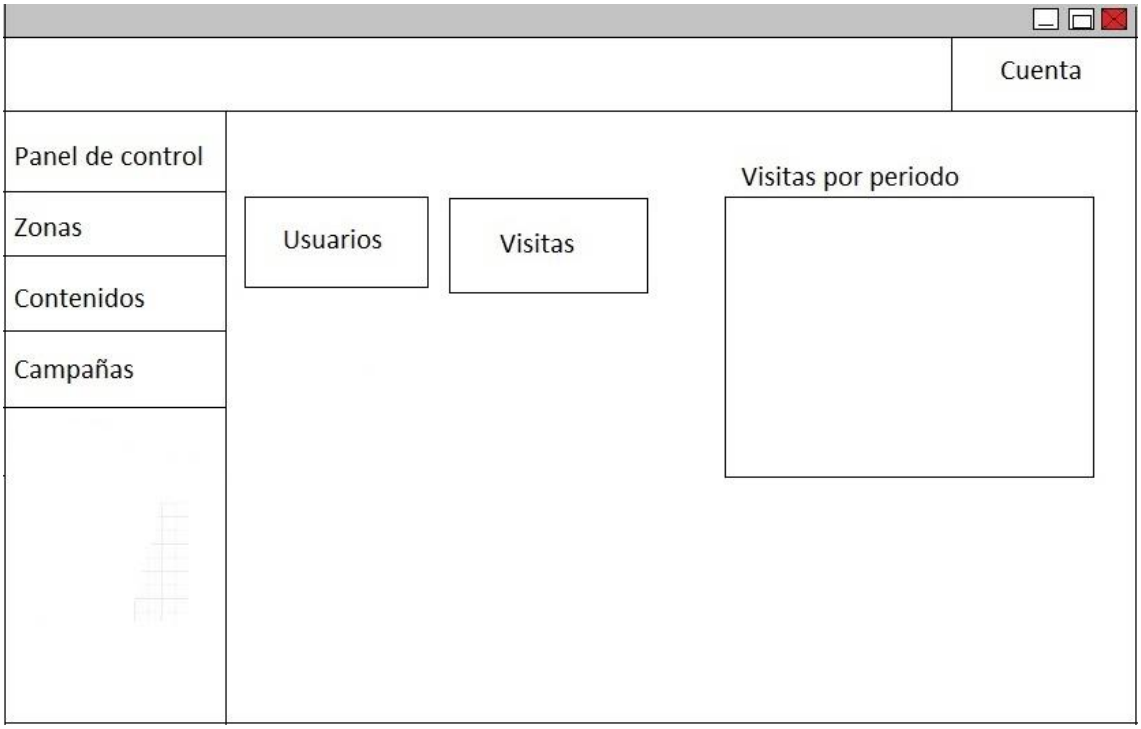

Figura 4.4. Prototipo: panel de control

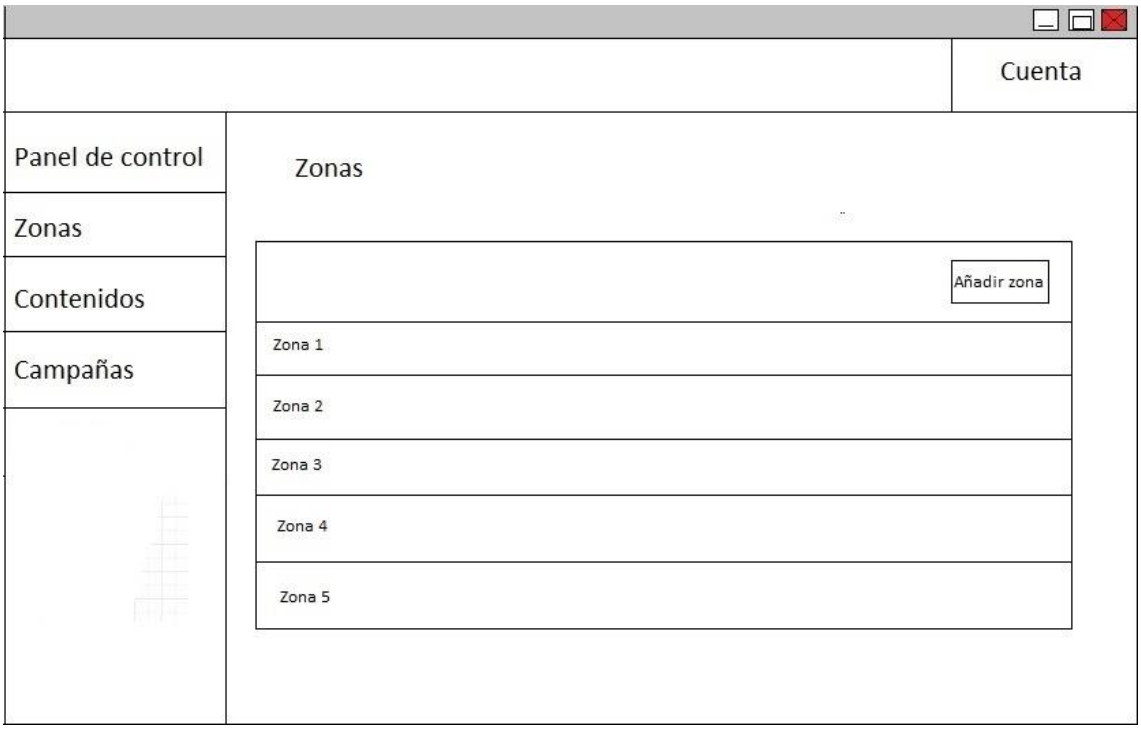

Figura 4.5. Prototipo: zonas.

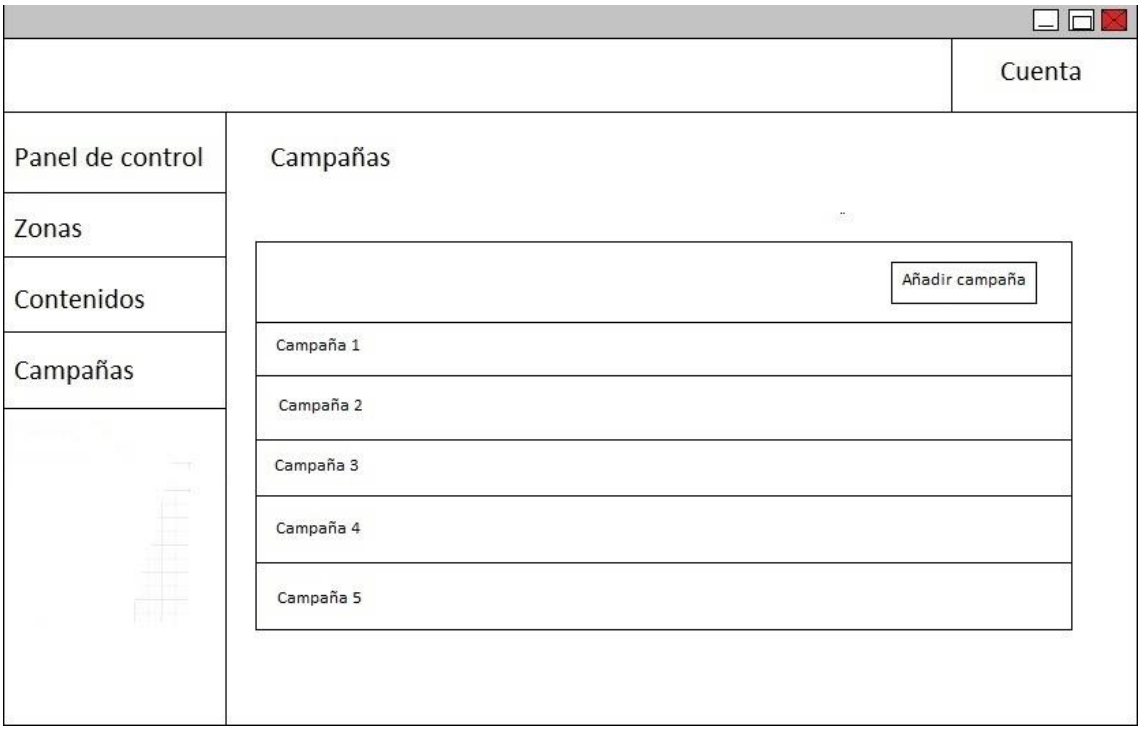

Figura 4.6. Prototipo: campañas.

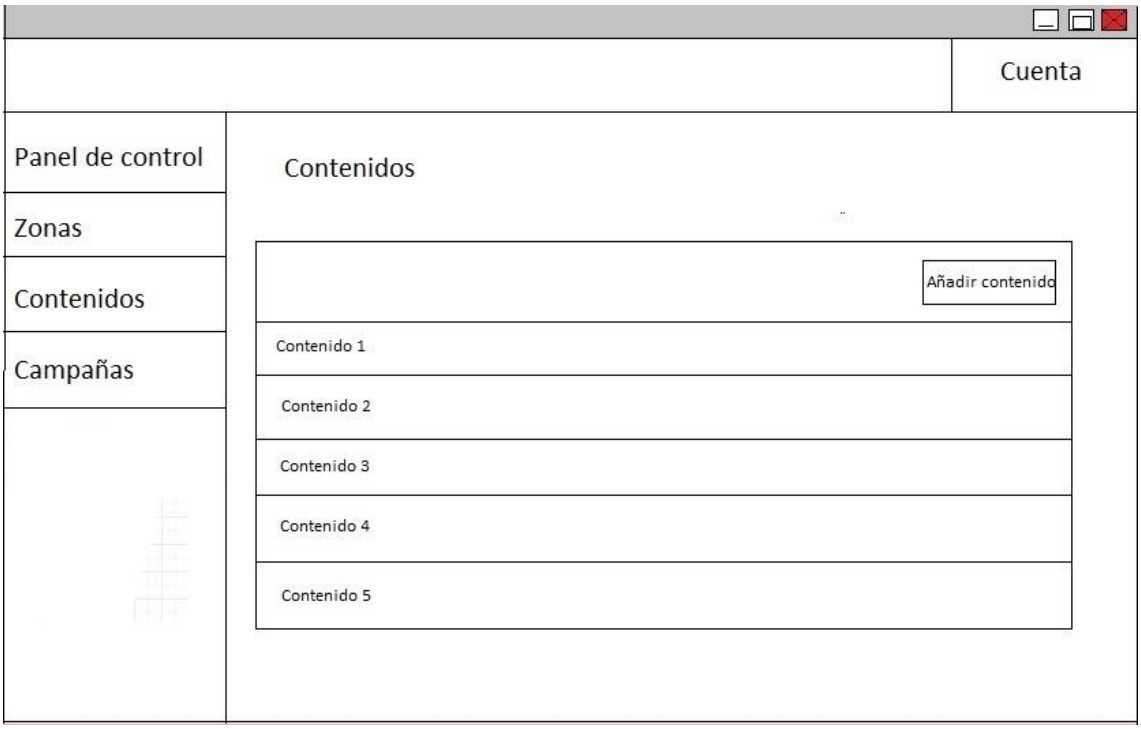

Figura 4.7. Prototipo: contenidos.

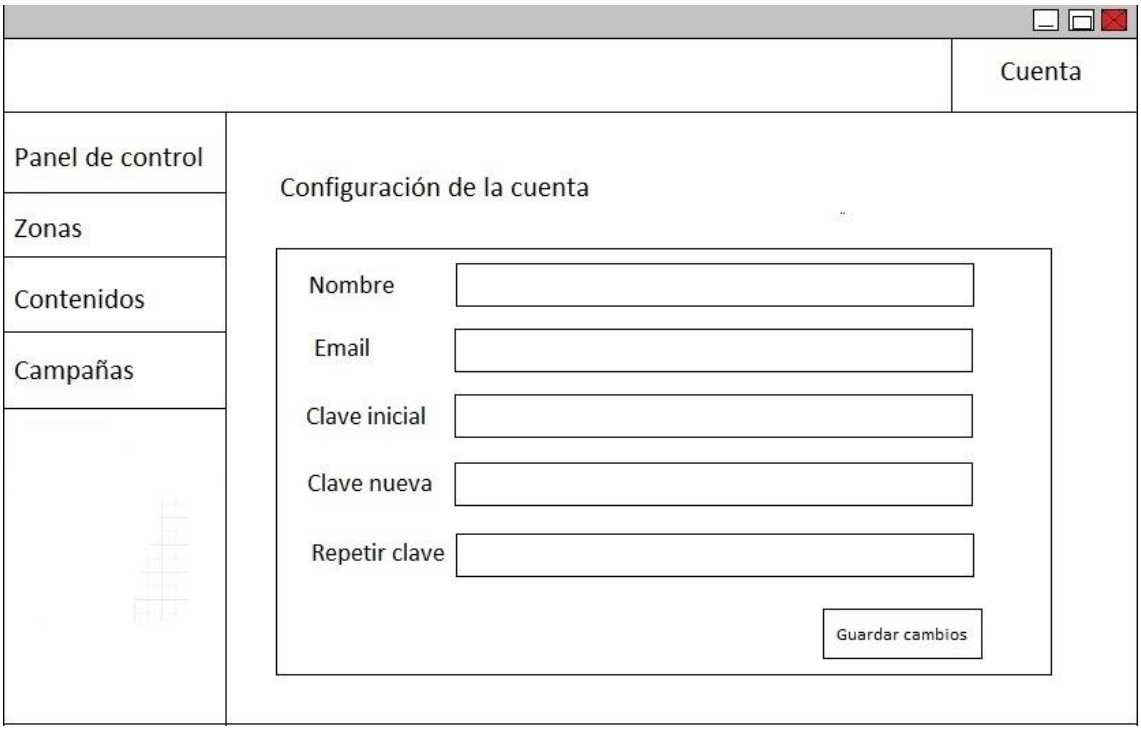

Figura 4.8. Prototipo: configuración de la cuenta.

4.2. Diseño e implementación

Una vez terminada la fase de análisis, se procede a modelar el sistema a nivel de arquitectura y componentes de software. La etapa de diseño permite a los desarrolladores tener una idea más clara y concreta de cómo será el sistema, reduciendo la abstracción respecto a la fase anterior.

4.2.1. Arquitectura de la aplicación

El primer paso en la fase de diseño es definir la arquitectura del sistema a alto nivel. La arquitectura es una especificación de la estructura a nivel global del sistema. Se centra en aspectos de más alto nivel que los algoritmos, funciones o tipos de datos. Estos últimos forman parte del diseño a nivel de componentes.

La figura 4.9 muestra todos los módulos que están dentro del alcance del sistema, así como los que no están pero cumplen alguna función en todo el ciclo de vida de los partes de trabajo.

Figura 4.9. Arquitectura

En este diagrama encontramos tres módulos distintos:

- **Base de datos central:** Esta es la base de datos central de la empresa. Contiene la información de usuarios, ibeacons, zonas, etc…

- **Servidor central:** Esta es la máquina que se encargará de la comunicación entre el cliente (aplicación) y los datos que se encuentran en la base de datos central. Como se ha explicado anteriormente, esta comunicación se realizará mediante un servicio web REST[11].

- **Cliente:** Este módulo compone el sistema que se está diseñando en este proyecto. El cliente realizará peticiones y enviará información al servidor central para que esta sea procesada y persistida en la base de datos central.

4.2.2. Diagramas de clases

En este apartado se van a mostrar algunos diagramas relativos al diseño a nivel de componentes. Esta etapa del diseño se centra en la organización de las clases e interfaces y cómo se relacionan entre sí. Debido a la complejidad del proyecto, no se pueden mostrar todos los diagramas de clases. Por ello, se mostrarán aquellos que tienen más importancia, ya sea porque se adaptan a un patrón de diseño predefinido o porque simplemente se crea conveniente mencionar.

4.2.2.1. Patrón MVC (Modelo Vista Controlador)

El patrón de diseño o arquitectónico Modelo Vista Controlador[12] es el patrón que engloba toda la aplicación. Todos los componentes del sistema se pueden ubicar dentro de uno de estos tres módulos. El objetivo de este patrón es separar la lógica de negocio, de la interfaz de usuario y los datos del modelo. A continuación, se definen los tres módulos que componen este patrón:

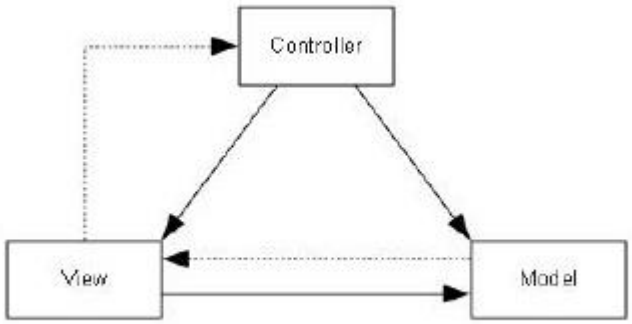

Figura 4.10. Modelo Vista Controlador.

- **Modelo:** Este componente tiene que ver con las representaciones creadas de los datos que va a usar la aplicación. Cada elemento debe tener una representación bien definida y única, con el objetivo de no repetir código y evitar información redundante. Un componente del modelo podría ser la clase *Zona*, ya que es una representación de una entidad de información. La base de datos también se encuentra dentro de este componente.

- **Vista:** Es el componente con el que el usuario interactúa: la interfaz de usuario o GUI. En una implementación pura de MVC, la vista no tiene ningún estado y no realiza acciones, sino que las delega en el controlador.

- **Controlador:** Este componente es el encargado de gestionar la lógica de la aplicación, responder a eventos, peticiones de la vista por parte del usuario, etc.

Gracias a este patrón, la aplicación es más escalable, extensible y capaz de intercambiar alguno de los tres componentes, mientras se cumpla la misma interfaz. Además, al estar los componentes bien separados, es fácil repartir las tareas de desarrollo y de diseño de interfaz entre los participantes en el proyecto.

4.2.2.2. Patrón DAO (Data Access Object)

Data Acces Object (DAO)[13] es un patrón de diseño que proporciona una interfaz para acceder a los datos, ya sean en base de datos un fichero o cualquier otro medio de almacenamiento. Este patrón oculta al programador que usa la interfaz la forma en que se almacenan los datos. Esto permite más abstracción entre la capa de aplicación y la capa de persistencia.

En esta aplicación se han desarrollado cuatro clases DAO. Cada una de estas permite el acceso a datos de diferente ámbito. Las interfaces son las siguientes:

- **ZonaDAO:** Mediante esta interfaz se accede a los datos relativos de las zonas.
- **CampañaDAO:** Mediante esta interfaz se accede a los datos relativos de las campañas.
- **ContenidoDAO:** Mediante esta interfaz se accede a los datos relativos de los contenidos.
- **ClienteDAO:** Mediante esta interfaz se accede a los datos relativos de los usuarios de la aplicación.

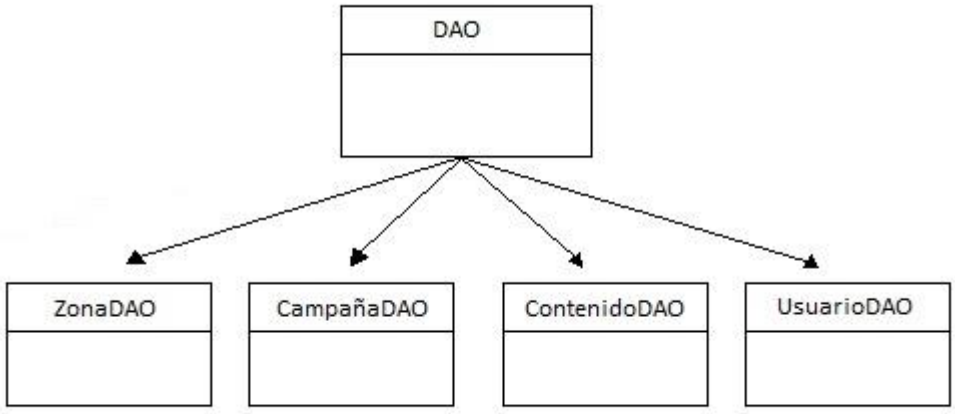

Figura 4.11. Diagrama de clases DAO.

4.2.2.3. Patrón Observador

En el patrón observador[14], un componente (observador) se suscribe a otro (observado), el cual notifica cuando el cambia su estado, esto permite crear una cadena de notificaciones entre varios componentes automáticamente.

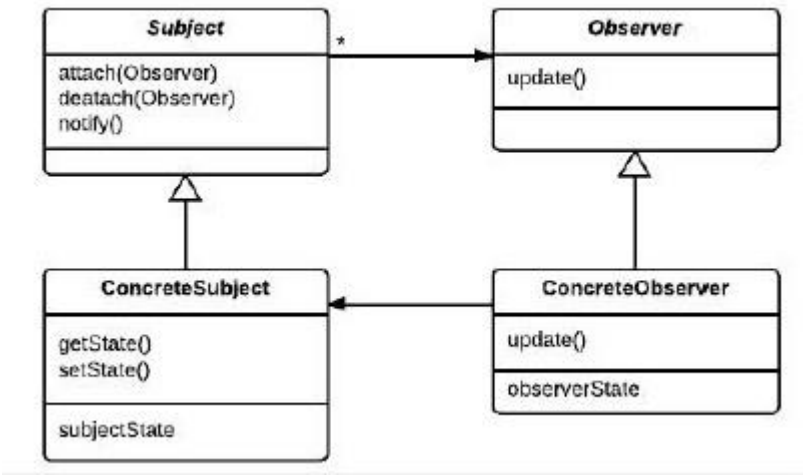

Figura 4.12. Patrón observador

En este caso, hemos utilizado este patrón para notificar los cambios del modelo de datos a la interfaz gráfica.

4.2.3. Interfaz de usuario

Bootstrap ofrece la posibilidad de diseñar la interfaz mediante plantillas y diferentes elementos de diseño basados en HTML y CSS. Idependientemente de la forma en que se realice, cada pantalla se compone de un fichero HTML que representa el aspecto gráfico de la interfaz.

Para diseñar la interfaz se han considerado los siguientes aspectos:

- **Consistencia:** La interfaz ha de mantener una consistencia entre sus diferentes ventanas, ya que una interfaz consistente permite al usuario entender mejor como funcionarán las cosas, incrementando su eficiencia.

- **Jerarquía visual:** Se ha diseñado la interfaz de una manera que permite al usuario centrarse en lo más importante a la hora de realizar cada tarea. El tamaño, color y posicionamiento de cada elemento hacen más sencillo entender la interfaz.

- **Proporcionar FeedBack:** La interfaz ha de comunicarse con el usuario cuando sus acciones han sido realizadas de manera correcta, incorrecta o anda perdido.

- **Reflejar estado:** En todo momento la interfaz debe reflejar el estado en que se encuentra. No hay que obligar al usuario a recordar.

- **Diseño simple:** Se ha optado por un diseño simple sin muchos elementos innecesarios que distraigan la atención del usuario.

4.2.3.1. Login

La Figura 4.13 muestra la pantalla de login. El usuario que vaya a utilizar la aplicación debe estar dado de alta previamente en la base de datos central.

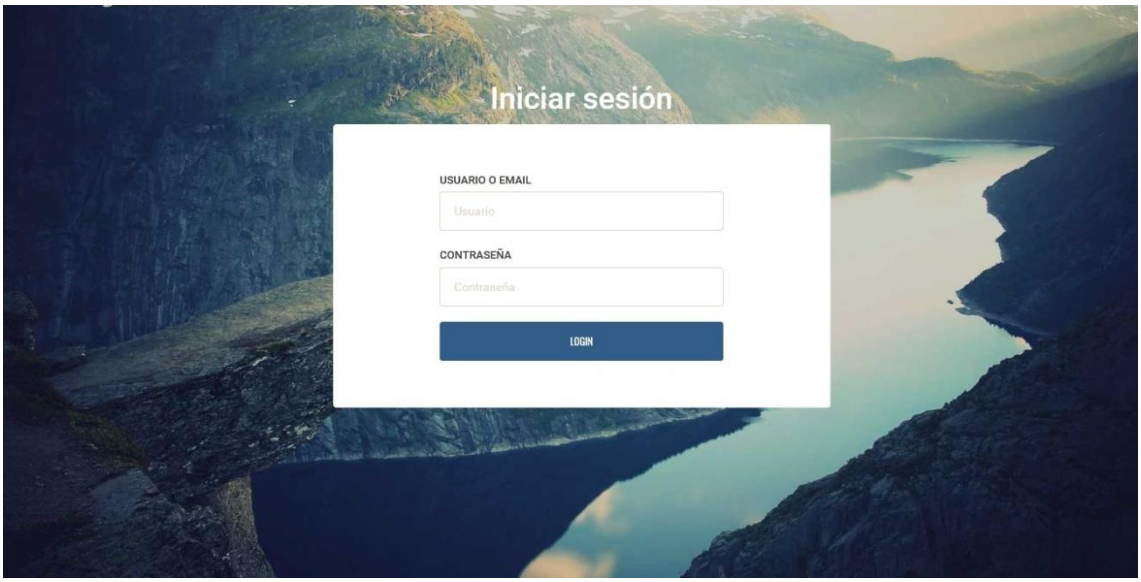

Figura 4.13. Login.

4.2.3.2. Panel de control

La Figura 4.14 corresponde a la pantalla de panel de control. Esta pantalla se muestra por defecto después de que el usuario haya realizado el login correctamente. Cuenta con un menú para poder navegar a las otras pantallas, además muestra unas estadísticas básicas.

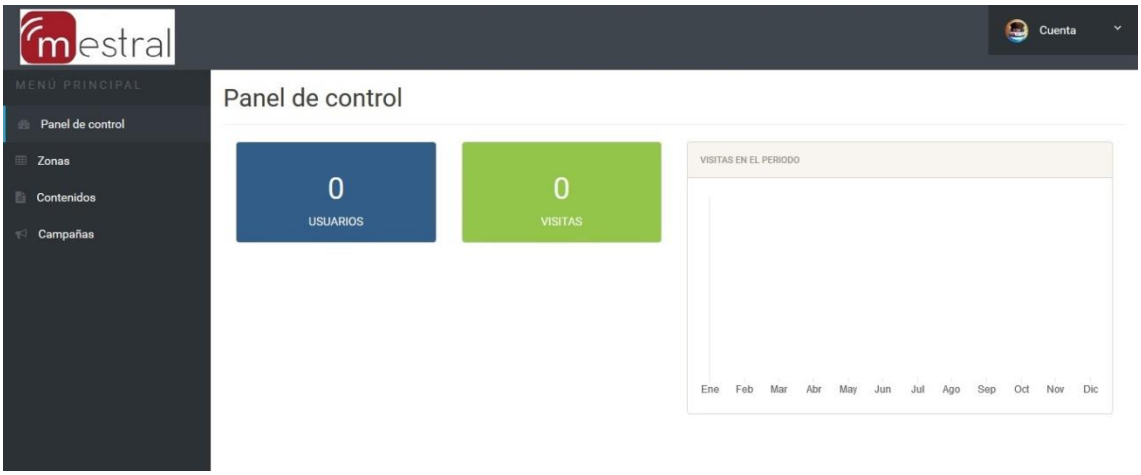

Figura 4.14. Panel de control

4.2.3.3. Gestión de zonas

La pantalla de gestión de zonas contiene dos componentes, una tabla para visualizar las zonas que tenemos creadas y un formulario desplegable.

En la Figura 4.15 podemos observar la tabla con las diferentes zonas creadas con un botón que utilizaremos para que se despliegue el formulario para añadir una zona nueva.

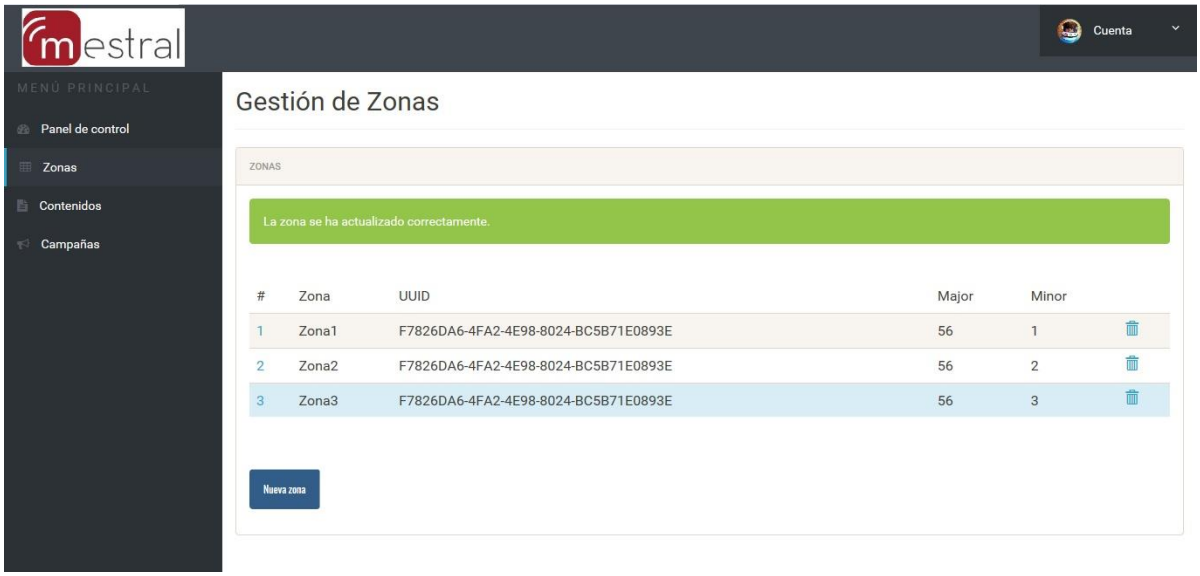

Figura 4.15. Zonas creadas.

Al presionar el botón de "Añadir Zona", se despliega el formulario para poder añadir una nueva zona. En la Figura 4.16 podemos observar los diferentes campos que debemos rellenar.

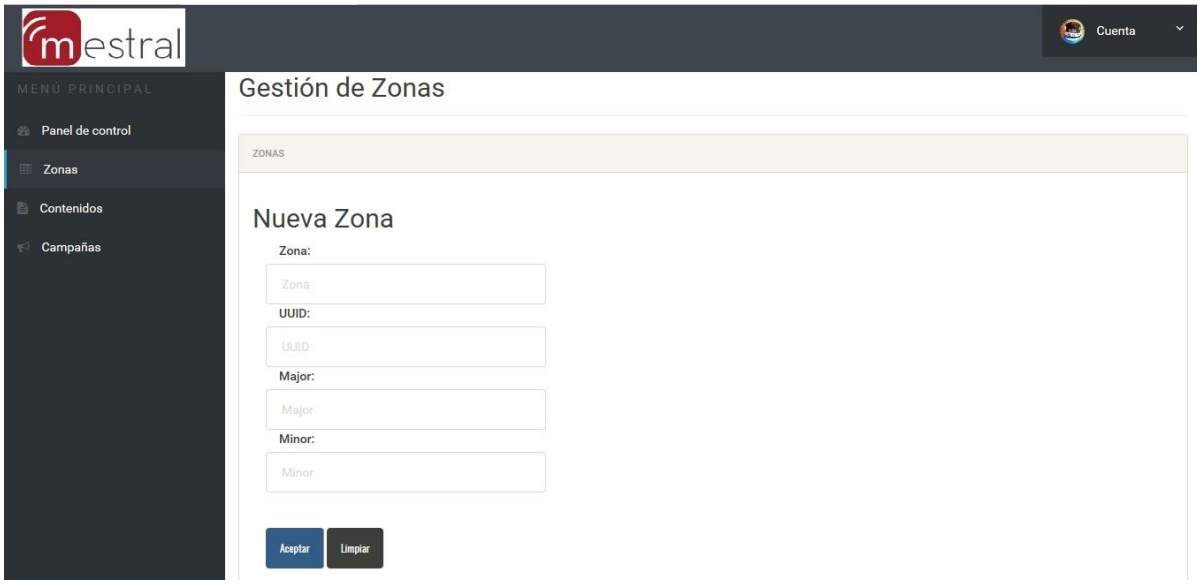

Figura 4.16. Añadir zona.

4.2.3.4. Gestión de contenidos

La pantalla de gestión de contenidos también contiene dos componentes, igual que la pantalla de gestión de zonas.

En la Figura 4.17 podemos observar la tabla con los contenidos creados.

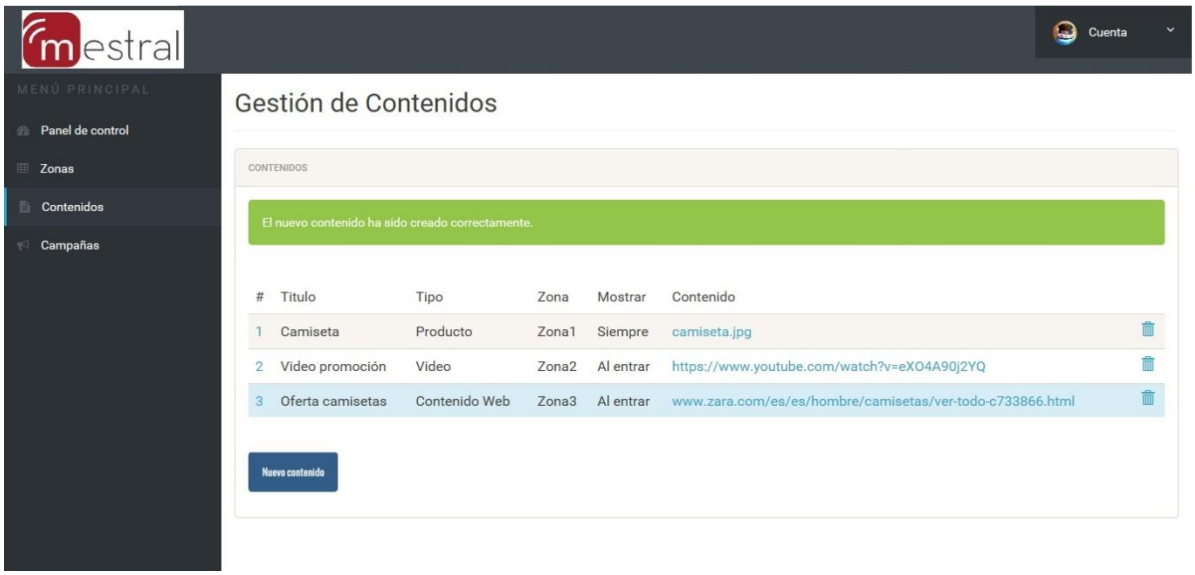

Figura 4.17. Contenidos creados.

Igual que en la pantalla de gestión de zonas, al presionar el botón "Añadir Contenido", se despliega el formulario que podemos observar en la Figura 4.18, en este caso, como hay diferentes tipos de contenidos podemos navegar por las pestañas para poder crear el tipo de contenido que deseamos.

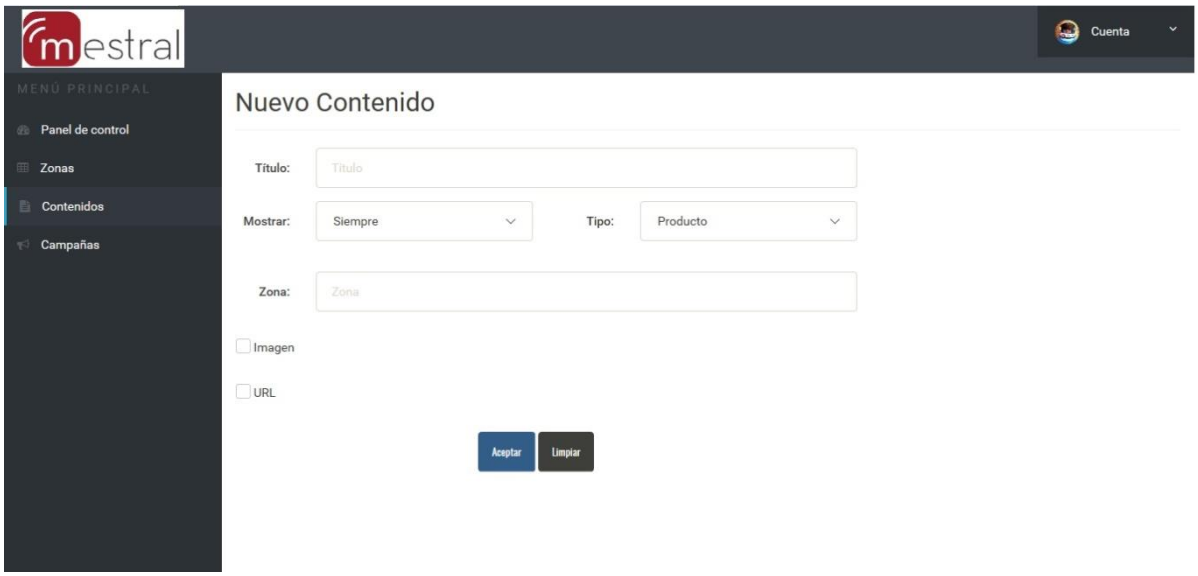

Figura 4.18. Añadir contenido.

4.2.3.5. Gestión de campañas

La pantalla de gestión de campañas sigue el mismo esquema que las dos pantallas anteriores, consta de una tabla para visualizar las campañas y de un formulario desplegable para crear una nueva. En la figura 4.19 observamos las campañas que tenemos creadas.

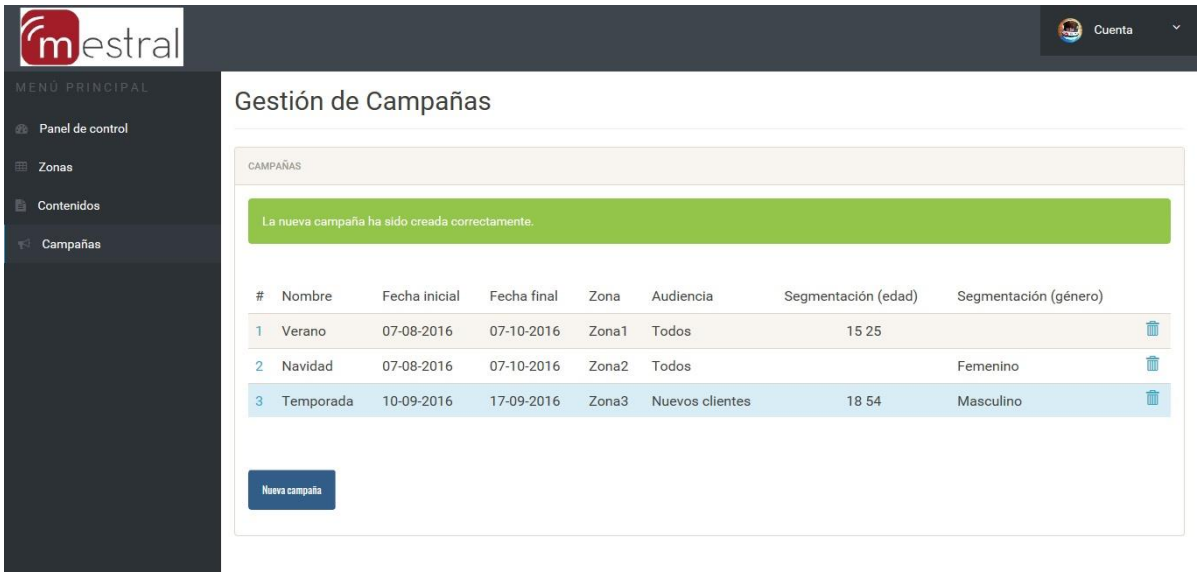

Figura 4.19. Campañas creadas.

En la Figura 4.20 observamos el formulario desplegable que nos sirve para poder añadir nuevas campañas publicitarias.

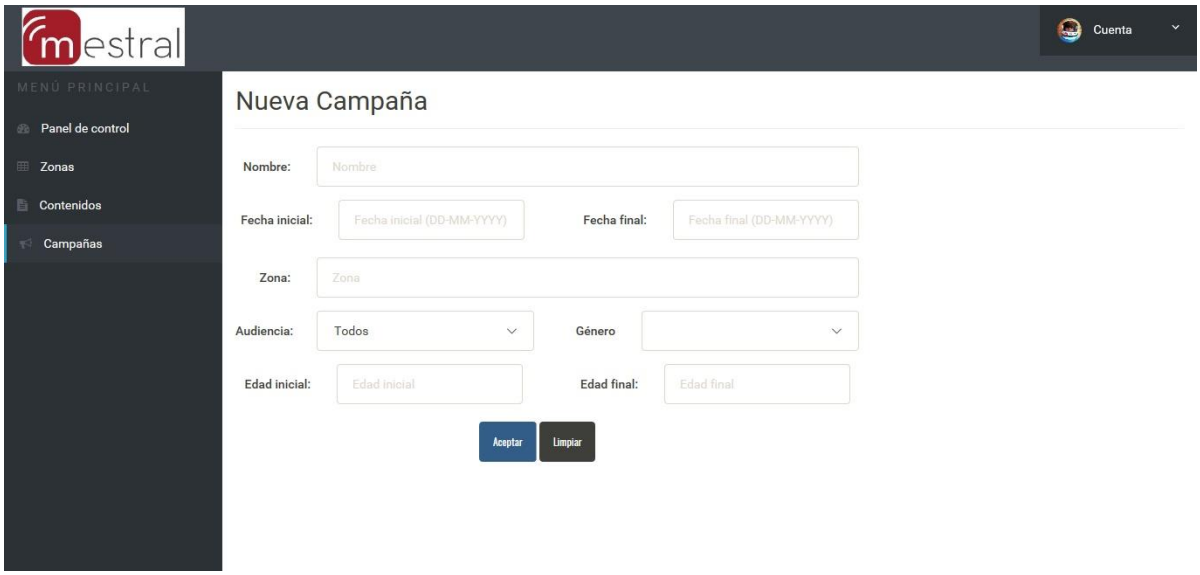

Figura 4.20. Añadir campaña.

4.2.3.6. Configuración de la cuenta

En la Figura 4.21 observamos una pantalla con un formulario para que el usuario pueda cambiar sus datos, como la contraseña.

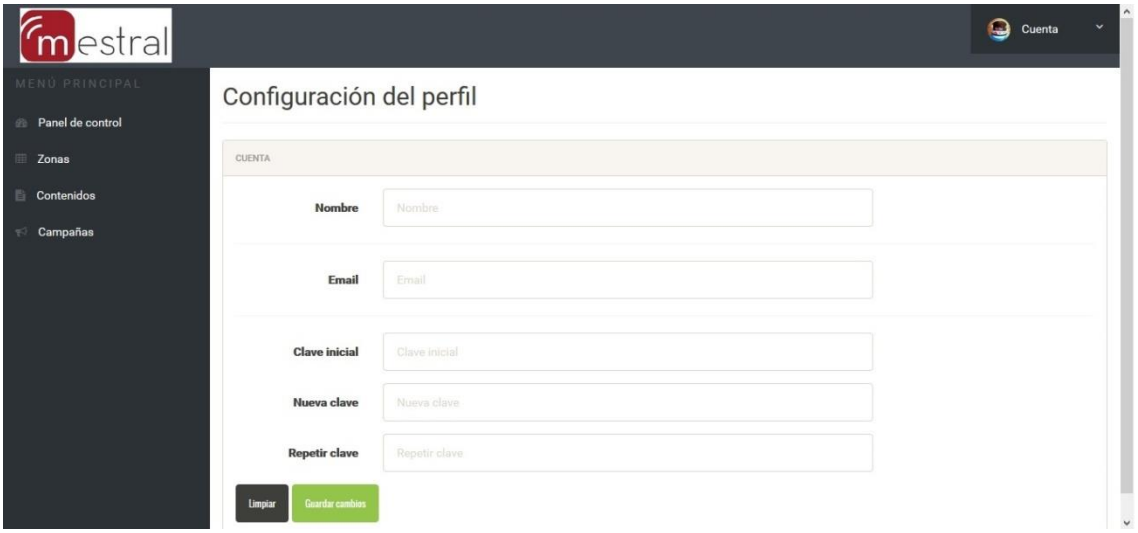

Figura 4.21. Configuración de la cuenta.

4.3. Validación y verificación

El objetivo de la fase de validación y verificación del sistema es comprobar el funcionamiento de este a todos los niveles. Esta fase es esencial en el proceso de desarrollo de software ya que proporciona información sobre la calidad del sistema que se está desarrollando. Existen varios tipos de pruebas según su alcance o según sea el tipo de componente que se está probando:

- Pruebas unitarias.
- Pruebas de GUI y navegación.
- Pruebas de usabilidad.

- Pruebas de rendimiento.

- Pruebas de aceptación.

El objetivo de todo desarrollo software no es llevar a cabo todos los tipos de pruebas, sino para cada caso concreto hay que realizar un estudio con el objetivo de definir qué pruebas son útiles y viables, teniendo en cuenta variables como el tiempo o el presupuesto.

En este proyecto se han realizado pruebas unitarias, de rendimiento y sobretodo de usabilidad, ya que se está desarrollando una interfaz gráfica y este es uno de los aspectos más importantes que hay que tener en cuenta. Además se han seguido las reglas heurísticas propuestos por Nielsen para garantizar una interfaz de calidad del agrado del cliente.

4.3.1. Pruebas unitarias.

Las pruebas unitarias son aquellas que se encargan de comprobar el correcto funcionamiento de cada módulo o componente que constituye el sistema. La prueba de módulo es independiente de los demás, de esta forma nos aseguramos de que cada uno de los módulos trabaja como se espera por separado.

Para desarrollar las pruebas unitarias en este proyecto se ha utilizado la herramienta *JUnit* [15], un *framework* libre y de código abierto que permite definir casos de pruebas unitarias para programas escritos en Java.

En la Figura 4.22 podemos observar el informe de resultados después de ejecutar los tests de la Figura 4.23.

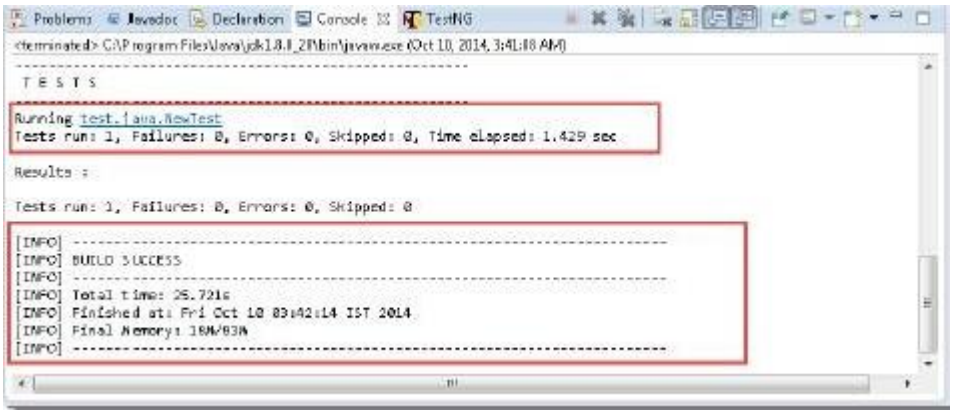

Figura 4.22. Informe de test.

```java
public class ZonaTests {

    ZonaService zonaService = new ZonaService();
    private List<Zona> zonas;

    private static String ZONA_DESCRIPTION = "Mercadona";
    private static int PRODUCT_COUNT = 2;

    @Test
    public void testGetZonas() {
        List<Zona> zonas = zonaService.getAllZonas();
        assertNotNull(zonas);
        assertEquals(PRODUCT_COUNT, zonaService.getAllZonas().size());

        Zona zona = zonas.get(0);
        assertEquals(ZONA_DESCRIPTION, zona.getDescripcion());
    }
```

Figura 4.23. Test de Zona.

4.3.2. Pruebas de GUI y navegación.

Para verificar la interacción del usuario con el sistema realizamos pruebas de GUI y navegación. El objetivo es asegurar que la interfaz en la parte front-end cumple los requisitos del diseño y que la navegación entre páginas funciona correctamente. Estas pruebas consisten en verificar página a página todo el sistema. Las repetimos varias veces para que no se nos escape ningún detalle.

4.3.3. Pruebas de usuario

Las pruebas de usuario se basan en la observación de cómo un grupo de usuarios lleva a cabo una serie concreta de tareas encomendadas por el evaluador, analizando los problemas de usabilidad que se encuentran.

Aun cuando el diseñador tenga amplios conocimientos sobre usabilidad, resulta recomendable evaluar el diseño con usuarios. Esto se debe a que, conforme más tiempo dedica un diseñador a un proyecto, menor es su perspectiva y más difícilmente detectará posibles problemas.

Podemos decir que gran parte de lo que el diseñador percibe cuando mira su propia obra, es una construcción mental; ve aquello que tiene en mente, no aquello que sus usuarios tendrán ante sus ojos.

En este proyecto se han realizado las pruebas con 4 usuarios (familiares y amigos) ajenos al proyecto. Cada usuario ha realizado sobre prototipos de alta fidelidad una serie de tareas típicas que un usuario real llevaría a cabo: Iniciar sesión en el sistema, crear una zona nueva, cambiar su configuración, cerrar sesión, etc.

El resultado de las pruebas fue bastante satisfactorio, aunque a medida que se han ido realizando estas pruebas, se han encontrado errores como la falta de ayuda en la interfaz o la escasa prevención de errores.

Gracias a las pruebas realizadas a lo largo del proyecto y a los prototipos de la interfaz, se han detectado y corregido estos errores, lo cual ha favorecido al desarrollo del proyecto y a alcanzar los objetivos antes del tiempo estimado.

Capítulo 5

Conclusiones

En este Trabajo de Fin de Grado se ha llevado a cabo el desarrollo de una aplicación web enfocada en el ámbito de la gestión y administración de ibeacons y siguiendo el diseño proporcionado por el cliente. Una vez terminado el desarrollo del sistema podemos afirmar que hemos conseguido el objetivo. Repasando los requisitos y objetivos concretos que establecimos hemos logrado la consecución de todos ellos.

Para poder iniciar el desarrollo en sí de la aplicación, ha sido necesario un estudio concreto de la tecnología Spring Boot. Aparte de la implementación propiamente dicha, se ha realizado una fase de análisis, ya que los requisitos de usuario y la lógica del sistema no estaban muy bien definidos al principio de la estancia en la empresa. Además, puesto que lo más importante de la aplicación es la parte gráfica, ha sido necesario profundizar en aspectos de diseño, componentes gráficos y usabilidad.

Todo el sistema está desarrollado sobre el patrón de diseño o arquitectónico Modelo Vista Controlador o MVC, cuyo objetivo es aislar la lógica de negocio del sistema de los datos persistentes y la interfaz de usuario. El uso de este patrón es indispensable si se quiere ampliar la funcionalidad del sistema en un futuro. También, el empleo de paquetes para reestructurar la localización de las clases según su cometido es esencial para la reutilización de componentes.

Cómo conclusiones sobre la experiencia personal, cabe decir que al principio de la estancia surgieron algunos problemas relacionados con los equipos de trabajo y con los objetivos del proyecto, ya que desde la empresa no tenían una idea clara de lo que querían ni de los recursos necesarios para lograrlo. Una vez se solucionaron estos problemas pude disfrutar de la valiosa experiencia que resulta estar en un entorno real de trabajo de cara al futuro. A pesar de que el Grado en Ingeniería Informática es una carrera con un enorme componente práctico, no se trabaja en entornos reales, dónde el estrés es mucho mayor. La integración con otros miembros en un equipo de desarrollo es interesante, a pesar de que el desarrollo estaba completamente a mi cargo, contar con la supervisión y apoyo del resto del equipo es realmente interesante, ya que se puede contar con distintas opiniones acerca de algunos apartados del desarrollo, como la estructura de navegación o el diseño de pantallas.

Finalmente, como aprendizaje personal, la realización de este proyecto me ha proporcionado conocimientos sobre una tecnología actual como es Spring Boot. Además, se me ha permitido poner en práctica, en un entorno empresarial real, muchos conocimientos adquiridos durante la carrera, tales como la metodología ágil SCRUM, patrones de diseño, evaluación de interfaces, testing, etc.

Bibliografía

[1] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 6, septiembre, 2016: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[2] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 8, septiembre, 2016: https://es.wikipedia.org/wiki/Kanban_(desarrollo)

[3] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 9, septiembre, 2016: https://es.wikipedia.org/wiki/IntelliJ_IDEA

[4] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 9, septiembre, 2016: https://es.wikipedia.org/wiki/Maven

[5] Gracia, L.M. (5 septiembre 2013). *Un poco de java:* https://unpocodejava.wordpress.com/2013/09/05/un-poco-de-spring-boot/

[6] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 13, septiembre, 2016: https://es.wikipedia.org/wiki/Twitter_Bootstrap

[7] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 13, septiembre, 2016: https://es.wikipedia.org/wiki/MySQL

[8] *Trello*. Fecha de consulta: 14, septiembre, 2016:

https://trello.com/tour

[9] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 16, septiembre, 2016: https://es.wikipedia.org/wiki/Tomcat

[10] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 16, septiembre, 2016: https://es.wikipedia.org/wiki/Microsoft_Paint

[11] *Spring by Pivotal*. Fecha de consulta: 19, septiembre, 2016: https://spring.io/understanding/REST

[12] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 19, septiembre, 2016: https://es.wikipedia.org/wiki/Modelo-vista-controlador

[13] "Mangrar". (25 julio 2015). *Genbeta: dev*:

http://www.genbetadev.com/java-j2ee/spring-framework-el-patrn-dao

[14] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 25, septiembre, 2016: https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)

[15] *Wikipedia, La enciclopedia libre*. Fecha de consulta: 27, septiembre, 2016: https://es.wikipedia.org/wiki/JUnit