Escuela Técnica Superior de Ingeniería Informática
Grado en ingeniería del software

Aplicación web para la gestión de rutas turísticas
Web application for tourist routes management

Realizado por

ROCÍO ARREBOLA PASCUAL

Dirigido por

JOSÉ MARÍA ÁLVAREZ PALOMO

Departamento

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

Málaga, junio de 2017.

Fecha de la defensa:
El Secretario del Tribunal

Resumen:
En este proyecto se aborda la creación de una aplicación web utilizando Django y los servicios web de Amazon (Amazon Web Services). En esta aplicación se pretende reunir a los viajeros de todo el mundo para que compartan sus experiencias con el resto de usuarios.
La aplicación se alojará en instancias de Amazon EC2, el servidor ofertado por Amazon. Estas instancias se administrarán en AWS Elastic Beanstalk por medio de AWS Auto Scaling, que aumentará o reducirá la capacidad de Amazon EC2 en función al tráfico de carga. Para la base de datos se utilizará una instancia de Amazon RDS, el servidor de bases de datos relacionales de Amazon, con una base de datos MySQL. También se hará un estudio del gasto que supone tener una aplicación alojada en AWS mediante el tablero de facturación que proporciona Amazon.
Para comprobar que AWS Auto Scaling hace un balanceo de carga correcto, se utilizarán herramientas para simular un gran número de peticiones al servidor de la aplicación, de modo que haya que aumentar la capacidad de Amazon EC2.

Palabras clave: Amazon Web Services, Django, aplicación web, Amazon EC2, AWS Elastic Beanstalk, AWS Auto Scaling, Amazon RDS.

Abstract:
This project deals with the creation of a web application using Django and Amazon Web Services. The primary goal of this application is gathering travellers around the world so that they can share their experiences with the rest of users.
This application will be hosted in Amazon EC2 instances, AWS's server. Instances will be managed in AWS Elastic Beanstalk by AWS Auto Scaling, that will increase or reduce server's capacity according to application's traffic. For the database, an Amazon RDS instance will be used, which is the Amazon's relational database server, into this instance a MySQL database will be used. Furthermore, in order to study the costs of hosting a web application in AWS, Amazon's billing dashboard will be used.
For the shake of checking that AWS Auto Scaling is scaling in a proper way, tools to simulate a big number of requests to the application's server will be used, so that it will be necessary to increase Amazon EC2's capacity.

Key words: Amazon Web Services, Django, web appliciation, Amazon EC2, AWS Elastic Beanstalk, AWS Auto Scaling, Amazon RDS.

Índice general

Índice de figuras

Índice de cuadros

Índice de Códigos

CAPÍTULO 1

Introducción

1.1. Motivación

A día de hoy nos encontramos con múltiples aplicaciones muy populares que reúnen a personas con intereses comunes. En estos grupos o foros, los usuarios participan de manera activa aportando su experiencia y su conocimiento e interaccionan entre ellos enriqueciendo la experiencia.

En este proyecto se propone el desarrollo de una aplicación web basada en viajes, pues, aunque existen muchas aplicaciones y foros sobre viajes, no existe un repositorio común para todos los aficionados a viajar, haciendo que la búsqueda de rutas y lugares de interés cuando se pretende ver mundo sea tediosa y el usuario pierda tiempo de manera innecesaria. Aunque esto no solucione el problema de la *infoxicación*, ya que supone añadir una aplicación más, se considera que supone una ayuda para que los usuarios puedan planificar sus viajes de manera más concreta.

1.2. Objetivos

Como se ha mencionado en la sección anterior, este Trabajo de Fin de Grado consistirá en el desarrollo de una aplicación web en la que los usuarios podrán compartir sus experiencias, facilitando a los demás usuarios la planificación de sus viajes, que buscarán rutas acordes a sus preferencias (precio, duración, valoraciones de otros

usuarios, lugares de interés, etcétera).

Las principales funciones que se implementarán serán las siguientes:

- Posibilidad de que el usuario añada rutas hechas por él.
- Posiblidad de que el usuario añada días a las rutas creadas.
- Posibilidad de que el usuario añada lugares de interés.
- Interacción entre usuarios (suscripción a otros usuarios, valoración y comentarios sobre las rutas de otros usuarios).
- Gestión de rutas.

Para próximas versiones de la aplicación se propone la creación de un usuario que tenga el papel de moderador y controle que la información dada es verídica y no es ofensiva. En este proyecto, esto se administrará desde la página del *admin* generada con *Django*.

Respecto a la parte tecnológica, la idea principal consistía en usar los servicios de *Amazon Web Services* para la base de datos, el servidor, la distribución de carga, la monitorización de rendimiento y el uso. Debido a ciertas complicaciones a la hora de subir la aplicación a *Amazon Web Services* finalmente se ha utilizado el servidor interno de *Django* y una base de datos local con el servidor de *MySQL*.

1.3. Django

(*Documentación de django*, s.f.)

1.3.1. ¿Qué es Django?

Django es un *framework* para el desarrollo de aplicaciones Web escrito en *Python*.

Antes de hablar sobre la arquitectura de *Django* es importante hacer la distinción entre proyecto y aplicación en *Django*.

- Una aplicación de *Django* es un conjunto de funcionalidades.
- Un conjunto de aplicaciones conforman un proyecto de *Django*.

Dentro del proyecto, se encontrará un paquete con el mismo nombre que el proyecto que incluirá diversos ficheros que se autogeneran al crear el proyecto. Los más interesantes son:

- *settings.py*: contiene información sobre la configuración del proyecto.
- *urls.py*: en él se indican todas las urls que usará el proyecto Django.
- *wsgi.py*: es el fichero en el que ”se encarga”de realizar la conexión con el servidor.

En este proyecto se desarrollará una única aplicación de *Django*, pues no se considera que sea necesario el desarrollo de ninguna otra más.

El desarrollo de aplicaciones con *Django* sigue el patrón MTV, que viene de *models*, *templates*, *views* y es como se recomienda estructurar las mismas.

Las aplicaciones en *Django* deben tener la siguiente estructura:

- El fichero en el que se implementan los modelos de la aplicación, por convención, es el fichero *models.py*, en él se crearán las clases, que se corresponderán con las tablas de la base de datos. Las clases de dicho fichero heredan de *models.Model* y tienen acceso a la base de datos.
- El fichero en el que se implementan las vistas de la aplicación, por convención, es el fichero *views.py*. Es importante tener en cuenta que cuando se habla de vistas en Django no se habla de interfaz gráfica, sino de controladores, de este modo, el fichero *views.py* será el intermediario entre los modelos y las plantillas. En el caso de esta aplicación, se ha decidido separar las vistas de cada modelo en ficheros diferentes para evitar que el fichero *views.py* sea demasiado extenso, de modo que el código sea más modular. Una vista se define como una función que toma una petición (`request`) y devuelve una respuesta, esta respuesta puede ser de diferentes tipos.
- La interfaz gráfica de la aplicación se creará en los *templates* (plantillas). No hay un fichero específico para declarar estas plantillas, pues habrá una por cada página que tenga la aplicación. Las plantillas se crearán dentro de una carpeta *templates* y tendrán extensión *.html*.

A pesar de seguir el modelo MTV, *Django* no tiene una arquitectura cerrada, por lo que el patrón puede ampliarse. En el caso de este proyecto, se han añadido los siguientes ficheros:

- *forms.py*: en él se crean clases que heredan de *foms.Form*. Dichas clases pueden utilizarse directamente en las plantillas para generar formularios de manera automática, sin tener que escribir código en *html*. De este fichero se hablará con más profundidad en el apartado *Desarrollo* del capítulo *Primera iteración*.
- Ficheros para las pruebas: estos ficheros empiezan por la palabra *tests*. En ellos se crean clases que heredarán de *LiveServerTestCase*. Esta clase inicia el servidor al iniciar un test y lo apaga al finalizar el test, permitiendo el uso de herramientas de automatización de pruebas.
- Un paquete *static* que contendrá diferentes paquetes hijo dentro. Cada paquete hijo contendrá diferentes ficheros. En este proyecto habrá un paquete *css* que contendrá ficheros con extensión *.css* que se usarán para embellecer las plantillas.

1.3.2. Modelos

Como se ha comentado en la sección anterior, los modelos se corresponden con las tablas de la base de datos, los atributos de un modelo son los campos de la tabla, y son los encargados de realizar la conexión con la misma.

Django determina el tipo de columna de la tabla de la base de datos dependiendo del tipo de datos que tenga su correspondiente atributo en el modelo. A diferencia de otros lenguajes, estos tipos de datos no son tipos simples, sino que son clases. Además, dependiendo del tipo de datos, *Django* validará los datos que se intentan insertar en las columnas de la base de datos.

Además, los campos tendrán una serie de opciones dependiendo de si son claves primarias, únicos, si pueden ser vacíos, etcétera. Estas opciones se pasan como parámetros a la hora de definir el tipo de datos del atributo. En caso de que se cree un modelo sin clave primaria, *Django* autogenerará un atributo autoincremental que asumirá el papel de la misma.

Como último apunte, debido a que *Django* está planteado para trabajar con bases de datos relacionales, también es posible definir relaciones entre modelos. Esto se hará creando atributos de tipo *ForeignKey*, para relaciones *uno-a-muchos*, *ManyToManyField*, para relaciones *muchos-a-muchos*.

1.4. *Amazon Web Services (AWS)*

"*Amazon Web Services* (AWS abreviado) es una colección de servicios de computación en la nube (también llamados servicios web) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com."(*Amazon Web Services*, s.f.).

Para reducir la latencia y aumentar la robustez de los servicios, AWS se divide en 11 regiones geográficas, que son los centros de datos donde se alojan los servicios. En

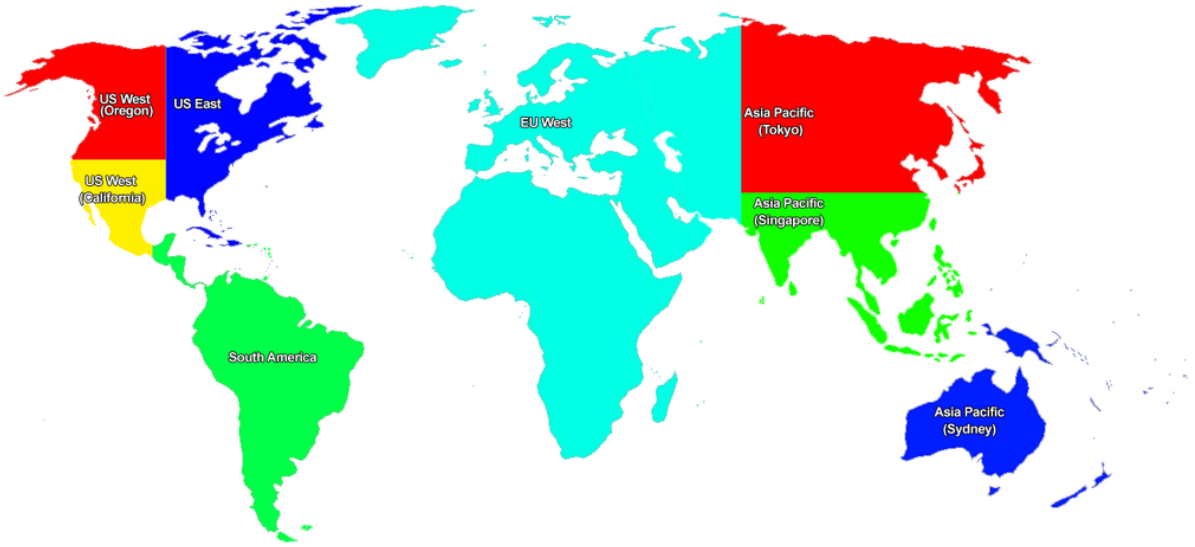

Figura 1.1: Mapa de las regiones geográficas usadas por AWS

nuestro caso, debido a la cercanía geográfica, la aplicación se hospedará en la región de Frankfurt.

Aunque AWS dispone de más de media centena de servicios, en este trabajo de fin de grado se explicarán aquellos que se tenía intención de utilizar a la hora de crear y lanzar el proyecto desarrollado. Los servicios investigados han sido:

- *Elastic Beanstalk (EB)*: para administrar las aplicaciones. Es un PaaS que permite implementar y administrar aplicaciones web en la nube de AWS. Además, proporciona una serie de servicios para facilitar la labor de los desarrolladores.
- *Elastic Compute Cloud (EC2)*: para el servidor de la aplicación. Es un servicio de AWS que permite reservar instancias de servidores alojados en la nube. Como ventaja, existen diferentes tipos de instancias dependiendo de las necesidades

de cada cliente.

- *Identity and Access Management (IAM)*: para el control de acceso a los servicios. “Con IAM es posible dar permiso a otros usuarios para que administren otros recursos propios de tu cuenta de AWS sin necesidad de compartir contraseñas”. (*What Is IAM?*, s.f.)
- *Relational Database Service (RDS)*: para la base de datos de la aplicación.
- *Auto Scaling*: para el autoescalado de EC2. *Auto Scaling* permite incrementar o reducir el número de instancias activas de EC2 dependiendo del número de peticiones que reciba el servidor.
- *Elastic Load Balancing*: para el balanceado de carga.
- *Virtual Private Cloud (VPC)*: que permite el acceso al servidor a través de una red privada.

1.5. *Bootstrap*

(*Bootstrap*, s.f.) Para el desarrollo del *front-end* de la aplicación se ha utilizado *Bootstrap*. *Bootstrap* es un *framework* específico para *front-end* basado en HTML y CSS que permite que los elementos de las páginas web que lo utilizan se ajusten a la pantalla en la que se muestran, haciendo más agradable al usuario la vista de la aplicación sin importar desde dónde la esté utilizando.

CAPÍTULO 2

Requisitos

2.1. Descripción de participantes y usuarios

2.1.1. Perfiles de usuario

Usuario

Representante	Usuario
Descripción	Podrá llevar a cabo operaciones de gestión de rutas tales como la creación, modificación, borrado, visualización o valoración de las mismas. También podrá crear, editar, visualizar y eliminar días dentro de las rutas previamente creadas, además de crear, visualizar, editar y valorar lugares de interés.
Tipo	No hay un tipo de usuario definido.
Responsabilidades	La responsabilidad del usuario con el producto es, meramente, realizar un uso correcto del mismo.
Criterio de éxito	Permite que el resto de usuarios tengan información detallada sobre qué visitar durante un viaje.

2.2. Requisitos funcionales

1. Gestión del perfil de usuario: el usuario podrá gestionar su perfil de usuario, pudiendo registrarse, acceder al sistema, recuperar su contraseña, salir del sistema, modificar el perfil y borrarlo.
   a) Registro: el usuario podrá registrarse en el sistema introduciendo los datos necesarios.
   b) Acceso: el usuario podrá acceder al sistema mediante un nombre de usuario y una contraseña.
   c) Recuperación de la contraseña: el usuario podrá recuperar su contraseña en caso de haberla olvidado.
   d) Salida del sistema: el usuario podrá salir del sistema cerrando la sesión.
   e) Modificación del perfil: el usuario podrá modificar su perfil.
   f) Borrado del perfil: el usuario podrá eliminar su perfil.
2. Gestión de días: el usuario podrá crear nuevos días, visualizarlos, editarlos, borrarlos, añadir y quitar lugares de interés a días.
   a) Creación de días: el usuario podrá crear un nuevo día dentro de una ruta específica creada previamente por él.
   b) Visualización de días: el usuario podrá visualizar los días que estén creados en el sistema.
   c) Modificación de días: el usuario podrá modificar los datos relativos a un día creado previamente por él.
   d) Borrado de días: el usuario podrá borrar un día creado previamente por él.
   e) Añadido de lugares de interés a días: el usuario podrá añadir lugares de interés previamente creados a los días de sus rutas.
   f) Borrado de lugares de interés en días: el usuario podrá quitar lugares de interés que han sido previamente añadidos a un día creado por él.
3. Gestión de rutas: el usuario podrá crear nuevas rutas, visualizarlas, editarlas, borrarlas, seguir rutas, dejar de seguir rutas, valorarlas, eliminar valoraciones hechas en rutas y buscar rutas en el sistema.
   a) Creación de rutas: el usuario podrá crear nuevas rutas.

b) Visualización de rutas: el usuario podrá visualizar las rutas que estén creadas en el sistema.

c) Modificación de rutas: el usuario podrá modificar información relativa a una ruta creada previamente por él.

d) Borrado de rutas: el usuario podrá borrar una ruta creada previamente por él.

e) Seguimiento de rutas: el usuario podrá seguir rutas creadas por otros usuarios.

f) Cancelación del seguimiento de rutas: el usuario podrá dejar de seguir rutas que previamente seguía.

g) Valoración de rutas: el usuario podrá valorar las rutas creadas en el sistema indicando una puntuación y haciendo un comentario.

h) Borrado de valoraciones de rutas: el usuario podrá eliminar valoraciones de rutas previamente hechas por él.

i) Búsqueda de rutas: el usuario podrá realizar búsquedas sobre rutas.

4. Gestión de usuarios: el usuario podrá seguir a otros usuarios, dejar de seguir usuarios, buscar usuarios y visualizar perfiles de usuario.

   a) Seguimiento de usuarios: el usuario podrá seguir a otros usuarios que se encuentren registrados en el sistema.

   b) Cancelación del seguimiento de usuarios: el usuario podrá dejar de seguir usuarios que previamente seguía.

   c) Búsqueda de usuarios: el usuario podrá realizar búsquedas sobre usuarios.

   d) Visualización del perfil: el usuario podrá ver perfiles de usuarios existentes en el sistema.

5. Gestión de lugares de interés: el usuario podrá crear, editar y visualizar lugares de interés, además de valorarlos, eliminar valoraciones previamente hechas en lugares de interés y buscar lugares de interés.

   a) Creación de lugares de interés: el usuario podrá crear nuevos lugares de interés.

   b) Visualización de lugares de interés: el usuario podrá visualizar lugares de interés previamente creados en el sistema.

   c) Modificación de lugares de interés: el usuario podrá modificar lugares de in-

terés sin necesidad de ser el creador de los mismos.

d) Valoración de lugares de interés: el usuario podrá valorar los lugares de interés creados en el sistema indicando una puntuación y haciendo un comentario.

e) Borrado de valoraciones de lugares de interés: el usuario podrá eliminar valoraciones de lugares de interés previamente hechas por él.

f) Búsqueda de lugares de interés: el usuario podrá realizar búsquedas sobre lugares de interés.

6. Monitorización de acciones: el sistema monitorizará las acciones que los usuarios realicen sobre las rutas, incluyendo la creación y modificación de las mismas, añadido de días, modificado de días, añadido de lugares de interés a días y borrado de lugares de interés de días.

2.3. Requisitos no funcionales

REQUISITOS DE ASPECTO

RNF 1. El diseño de la aplicación debe ser *responsive*: para mejorar la experiencia del usuario, la aplicación debe poder adaptar su diseño a cualquier tipo de pantalla.

REQUISITOS DE FACILIDAD DE USO Y APRENDIZAJE

RNF 2. La aplicación debe ser fácil de usar: para mayor comodidad del usuario, el programa debe ser sencillo e intuitivo.

RNF 3. La aplicación debe ser adaptable: debe permitir añadir funcionalidades.

RNF 4. La aplicación debe poder internacionalizarse: la aplicación podrá traducirse a varios idiomas.

REQUISITOS OPERACIONALES

RNF 5. La aplicación debe correr sobre Amazon Web Services.

RNF 6. La aplicación debe estar desarrollada en Django.

2.4. Requisitos de documentación

MANUAL DE USUARIO

La aplicación deberá incluir un manual de usuario que indique, en un lenguaje no técnico, cómo utilizar la misma.

GUÍAS DE INSTALACIÓN Y CONFIGURACIÓN

La aplicación deberá incluir una guía de instalación y configuración escrita en un lenguaje no técnico. Estas guías vendrán incluidas en el manual de usuario.

2.4. Requisitos de documentación

CAPÍTULO 3

Diseño

3.1. Diagrama de clases

Inicialmente, la implementación del proyecto se había pensado con dos tipos de usuarios: el usuario y el administrador. El diagrama de clases inicial era el diagrama de clases de la figura 3.1.

En vista de que no era posible verificar la identidad del usuario cuyo tipo de perfil es *AdministradorLugarInteres*, se decidió hacer diversas modificaciones, quedando el siguiente diagrama de clases (figura 3.2):

Como se puede observar, se han eliminado las clases *AdministradorLugarInteres* y *Evento*, y la relación entre *Usuario* y *LugarDeInteres* pasa de llamarse *sigue* a *crea*. El porqué de estos cambios se explicará más detalladamente en el capítulo 9, *4a iteración*.

Además, se ha añadido una clase *Log*, que almacenará los cambios que realiza un usuario sobre una ruta para mostrarlos posteriormente en los *Timelines* de los usuarios que siguen la ruta cambiada o al usuario que realiza los cambios.

3.1. Diagrama de clases

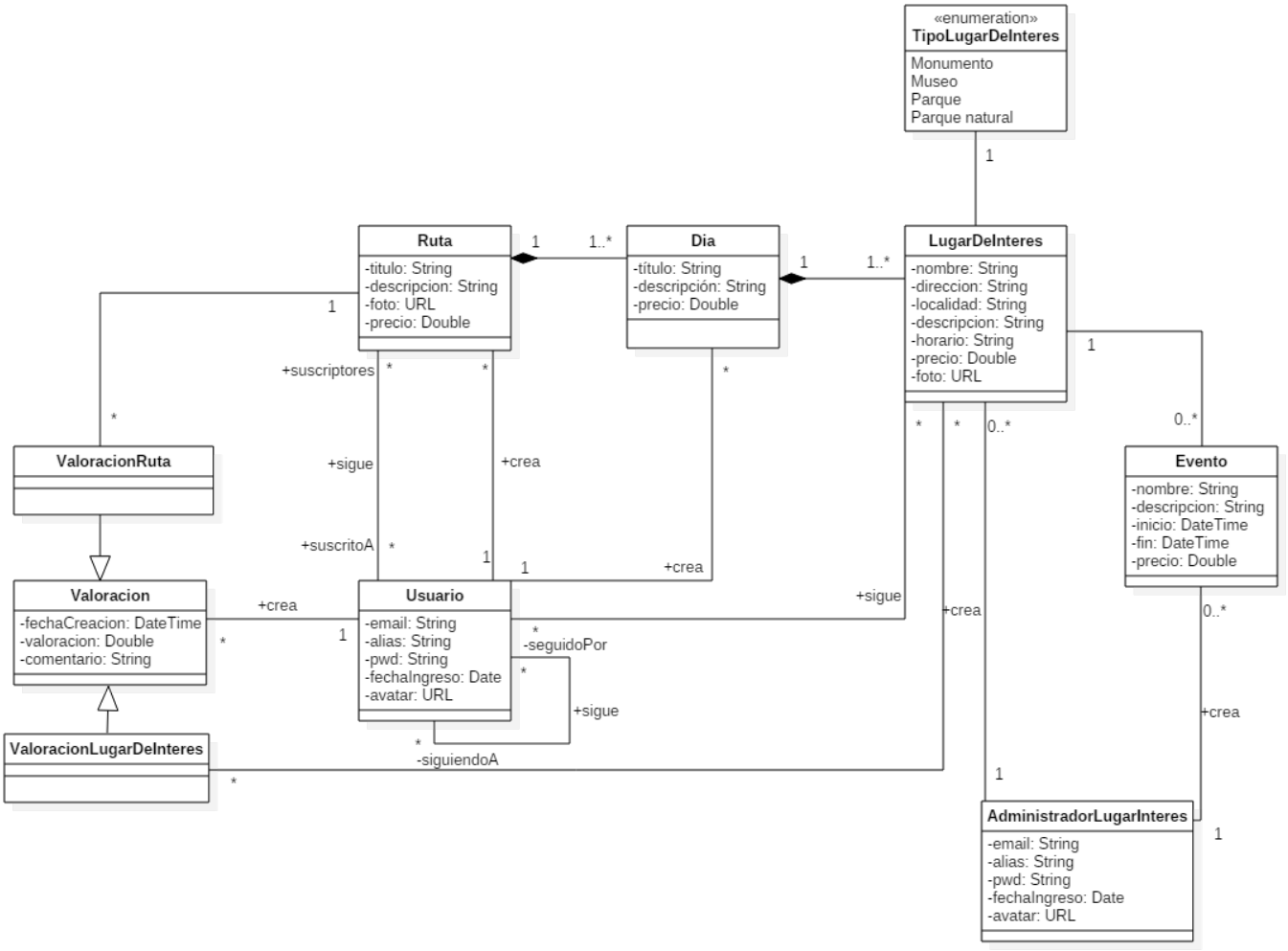

Figura 3.1: Diagrama de clases inicial

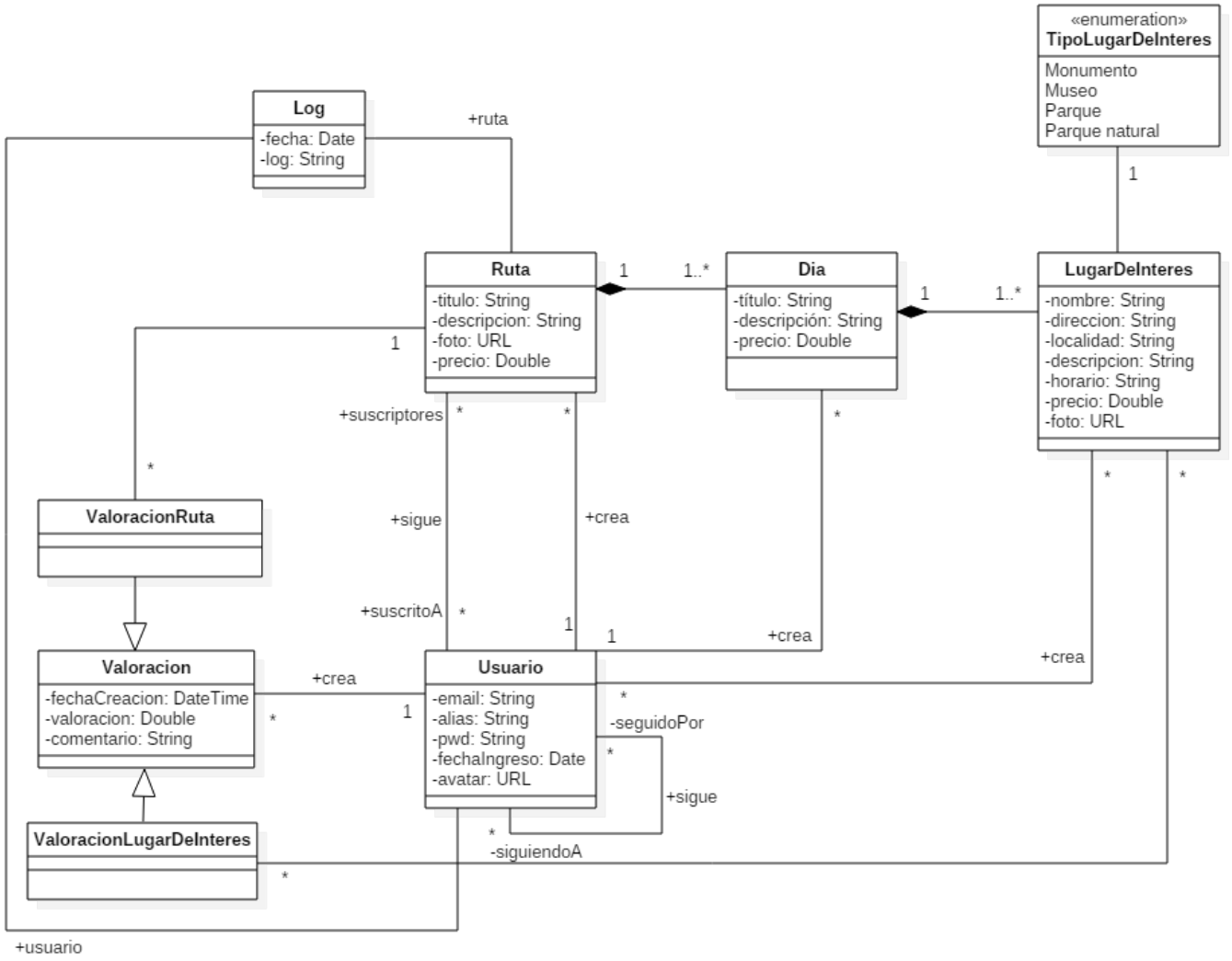

Figura 3.2: Diagrama de clases final

3.1. Diagrama de clases

CAPÍTULO 4

Visión de conjunto

Para facilitar la comprensión de este documento, se hace necesario explicar la visión de conjunto de la aplicación.

En este desarrollo de la aplicación se implementa un único tipo de usuario, que podrá crear rutas turísticas, días dentro de las mismas y añadir lugares de interés a estos.

Las rutas estarán compuestas por días, cada día incluirá una serie de lugares de interés localizados en un mapa y listados por orden en la página de información de cada día.

En la página principal de la aplicación se mostrarán las rutas a las que está suscrito un usuario y las rutas creadas por él, además de un *timeline* que mostrará los cambios más recientes que han realizado los usuarios seguidos por el usuario.

El usuario de la aplicación, después de haber accedido al sistema, podrá realizar las siguientes acciones:

- Crear, editar, visualizar y eliminar rutas.
- Añadir días a rutas.
- Los días que han sido añadidos a rutas podrán ser visualizados, modificados y eliminados.

- Añadir y quitar lugares de interés a días. Ya que, como se ha comentado anteriormente, un día incluye una serie de lugares de interés.
- Crear, visualizar y editar lugares de interés.
- Seguir o dejar de seguir rutas.
- Seguir o dejar de seguir a usuarios.
- Visualizar perfiles de usuarios.
- Valorar rutas y lugares de interés.
- Buscar rutas, usuarios y lugares de interés.
- Editar o eliminar su perfil de usuario.
- Visualizar perfiles de otros usuarios.

CAPÍTULO 5

Primera iteración

5.1. Requisitos implementados

En la primera iteración del desarrollo del proyecto se han decidido implementar los siguientes requisitos funcionales:

- Registro.
- Acceso.
- Creación de rutas.
- Visualización de rutas.
- Modificación de rutas.
- Borrado de rutas.
- Creación de días.
- Visualización de días.
- Modificación de días.
- Borrado de días.

Para facilitar la lectura del documento, los casos de uso de los requisitos implementados vienen desarrollados en el *Apéndice A: Casos de uso*.

5.2. Desarrollo

5.2.1. Modelos

Al ser el primer desarrollo que se hace del proyecto, ha sido necesaria la implementación de los modelos de Django. La implementación de modelos de la primera iteración se hizo acorde al primer diagrama de clases que se diseñó (*Figura 4.1*), habiendo sufrido modificaciones en iteraciones posteriores.

Aparte de las clases que se ven en dicho diagrama, se implementaron las clases *RelacionSeguidos*, *RelacionRutasSeguidas* y *RelacionDiaLugarInteres*; estas clases se utilizan para implementar las relaciones M:M que hay entre algunas tablas. De este modo, la clase *RelacionSeguidos* se corresponde con la relación reflexiva *sigue* entre distintos usuarios; la clase *RelacionRutasSeguidas* implementa la relación *sigue* entre usuarios y rutas; y la clase *RelacionDiaLugarInteres* representa la relación de composición entre la clase *Dia* y la clase *LugarInteres*. La implementación quedó de la siguiente manera:

Código 5.1: Relación `sigue` entre distintos usuarios

```
class RelacionSeguidos(models.Model):
    seguidor = models.ForeignKey(Usuario, related_name='seguidor')
    seguido = models.ForeignKey(Usuario, related_name='seguido')
    class Meta:
        unique_together = ('seguidor', 'seguido')
```

Código 5.2: Relación `sigue` entre usuarios y rutas

```
class RelacionRutasSeguidas(models.Model):
    seguidorRuta = models.ForeignKey(Usuario, related_name='seguidorRuta')
    ruta = models.ForeignKey(Ruta, related_name='ruta')
    class Meta:
        unique_together = ('seguidorRuta', 'ruta')
```

Código 5.3: Relación de composición entre las clases `Dia` y `LugarInteres`

```
class RelacionDiaLugarInteres(models.Model):
    lugaresInteres = models.ForeignKey(LugarInteres,
        related_name='lugaresInteres')
```

```
    dia = models.ForeignKey(Dia, related_name='dia')
    class Meta:
        unique_together = ('lugaresInteres','dia')
```

Cabe destacar el uso de la clase interna *Meta* dentro de cada una de las clases de la relación con el objetivo de definir los metadatos del modelo, en este caso, el objetivo principal de los metadatos es evitar que haya objetos duplicados, esto se indica en el atributo *unique_together* de dicha clase.

En la clase *Ruta* se ha implementado una función llamada *rutasSeguidasUsuario* a la que se le pasa un parámetro *usuario* y que devolverá todas las rutas que sigue ese usuario.

Finalmente, en la clase *Dia* se ha implementado una función *dias_ruta* a la que se le pasa como parámetro *ruta*, esta función devolverá una lista con los días que tiene asociada una ruta.

Tras desarrollar los modelos, es muy importante realizar las migraciones de modo que los modelos pasen a ser entidades en la base de datos. Desde la consola de comandos (y estando en el directorio raíz del proyecto), es necesario ejecutar los siguientes comandos:

```
python manage.py makemigrations
```

Lo que hace este comando es comparar las entidades que hay actualmente en la base de datos con los nuevos modelos y crea un archivo con las migraciones pertinentes. Después de obtener dicho archivo, para realizar los cambios en la base de datos, habrá que ejecutar el comando:

```
python manage.py migrate
```

5.2.2. Vistas

Para esta iteración se han creado ficheros para las vistas del usuario, la ruta y el día. Se ha decidido dividir las vistas en diferentes ficheros para facilitar futuras modifica-

ciones y hacer más sencilla la lectura del código.

views_usuario.py

En este fichero se implementan las vistas necesarias para desarrollar los casos de uso relacionados con el perfil del usuario. Se ha estimado necesario desarrollar los siguientes métodos:

- *init(request)*: carga la página inicial de la aplicación (*login.html*) con su respectivo formulario, también valida que los datos introducidos en el formulario pertenecen a un usuario registrado en el sistema, y que son correctos. La URL asociada a esta vista es una URL vacía, de modo que sea la primera página que se cargue a la hora de acceder a la aplicación en caso de no tener una sesión iniciada en el sistema.
- *registro(request)*: carga una página con un formulario para el registro de un nuevo usuario en el sistema. También se encarga de validar que los campos obligatorios han sido rellenados y que no existan usuarios con el mismo alias o email que se introduce para crear una nueva cuenta y una vez validados los datos (siendo estos correctos), crea un nuevo perfil de usuario.
  Se puede observar que una vez creado el usuario, el alias del mismo se almacena en una variable de sesión. Se ha decidido usar variables de sesión en lugar de *cookies* porque estas últimas no se consideran seguras si no se envían por HTTPS, siendo susceptibles a diversos ataques, como pueden ser ataques de tipo *snooping*[1] y *Man-in-the-middle*. Estos problemas se solucionan (en gran parte) con el uso de sesiones, pues al almacenar toda la información relevante en el servidor, evitan el envío constante de información como ocurre con las *cookies*.
- *inicio(request)*: esta vista carga la página de principal de la aplicación con los datos de las rutas pertenecientes al usuario que está almacenado en la variable de sesión. En caso de no haber ningún usuario almacenado en la variable de sesión, se cargará la página del *login*.

views_ruta.py

En este fichero se han implementado las vistas necesarias para hacer el CRUD de la clase *Ruta*. A grandes rasgos, se puede comprobar que todas las vistas contienen un fragmento de código común, este código controla que el usuario haya accedido

[1] El *snooping* se define como el acceso a datos por parte de terceros de manera no autorizada.

al sistema; en caso de que no haya accedido, la aplicación redirigirá al usuario a la página de *login*. El código es el siguiente:

Código 5.4: Control de sesiones de la aplicación

```
if "usuario" in request.session:
    usuario = models.Usuario.objects.filter(alias =
        request.session["usuario"])[0]
    if usuario == None:
        form = forms.RegistroForm()
        return render(request, 'login.html', {'form': form})
    else:
        # Implementacion de la vista
else:
    form = forms.RegistroForm()
    return render(request, 'login.html', {'form': form})
```

Este código de control de sesiones va a estar presente prácticamente en todas las vistas de la aplicación, a excepción de aquellas relativas a la creación de nuevos usuarios o al registro de los mismos.

También se puede observar que en las vistas *editarRuta(request, id_ruta)* y *borrarRuta(request, id_ruta)* se comprueba que el usuario que intenta realizar la acción sea el creador de dicha ruta; en caso contrario, se le redirigirá a la página principal de la aplicación.

views_dia.py

En este fichero se han implementado las vistas necesarias para hacer el CRUD de la clase *Dia*. Se comprueba a simple vista que la codificación de las vistas relativas al día es muy similar a la codificación de las vistas relativas a la ruta, con una excepción: la vista *anadirDia(request, id_ruta)*, a diferencia de la vista *crearRuta(request)*, comprueba que el usuario que tiene sesión iniciada en el sistema es el creador de la ruta a la que se desea añadir un día; esto se debe a que solo el creador de una ruta puede añadir días a dicha ruta.

5.2.3. Formularios

Debido a las facilidades que ofrece *Django* para la creación de formularios, se ha decidido desarrollar los mismos utilizando el fichero *forms.py*, en este fichero se han implementado clases que heredan de *forms.Form* cuyos atributos se corresponderán con los campos de los formularios que se mostrarán al ejecutar los ficheros *.html*.

En este fichero destaca la definición de un método *clean()* dentro de la clase *UsuarioForm*. Este método viene heredado de *forms.Form* y se utiliza para validar los datos recibidos desde el formulario; en este caso, valida que los campos que hacen referencia a la contraseña y al *email* coincidan.

Código 5.5: Implementación del método `clean()` en la clase `UsuarioForm`

```
def clean(self):
    if (self.cleaned_data.get('password') !=
        self.cleaned_data.get('confirmacionPassword')):
            msg = 'Las contrasenas no coinciden'
            self.add_error('confirmacionPassword', msg)
    if (self.cleaned_data.get('email') !=
        self.cleaned_data.get('confirmacionEmail')):
            msg = 'Los emails no coinciden'
            self.add_error('confirmacionEmail', msg)
```

5.2.4. *Templates*

Dentro del paquete *templates* se encuentran los ficheros de la interfaz gráfica de la aplicación, es decir, los ficheros con extensión *.html*. En esta iteración se han implementado los siguientes:

- *login.html*: en él se ha implementado la página para el *login*. Como se puede comprobar, se muestra un formulario. El formulario usado no se especifica en este fichero sino que se enviará desde la vista. En este caso, el formulario mostrado es *RegistroForm*. Es interesante destacar el uso de la etiqueta *csrf_token* justo después de la apertura del formulario. Esta etiqueta se usa para evitar *Cross Site Request Forgeries*. Esta etiqueta se usa en formularios POST que apuntan a URLs internas.
- La lógica de los ficheros *crearDia.html*, *crearRuta.html* y *registro.html* es similar

a la de *login.html*.

- En el fichero *inicio.html* viene implementada la página principal de la aplicación. En ella se muestra un listado con enlaces a las rutas creadas por el usuario de la sesión.
- Los ficheros *ruta.html* y *dia.html* muestran la información correspondiente a una ruta y a un día, respectivamente.

Para el diseño gráfico de la interfaz gráfica (archivos *.css*), es necesario crear un nuevo paquete llamado *static* en el que se guardarán todos los archivos estáticos (*CSS*, *JavaScript*, imágenes, etc); dentro de este paquete es necesario crear un nuevo paquete llamado *css* para guardar los archivos de este tipo. Para relacionar los *.css* con los *.html*, habrá que escribir la siguiente línea de código dentro de la etiqueta $<$*head*$>$ del archivo *.html*:

```
<link rel="stylesheet" href="{% static 'css/formularios.css' %}">
```

Adicionalmente, se han usado los siguientes componentes de *Bootstrap*:

- *Navbar* para la barra que se encuentra en la parte superior de todas las páginas de la aplicación una vez se ha iniciado sesión.
- Para hacer más visuales algunas opciones, se han utilizado algunos *Glyphicons*, como pueden ser el lápiz que se encuentra al lado del texto *Editar* o la papelera que se encuentra al lado del texto *Borrar*.
- Para mostrar el listado de rutas de un usuario así como los días asociados a una ruta, se ha utilizado el componente *List Group*.

CAPÍTULO 6

Segunda iteración

6.1. Requisitos implementados

En esta iteración se han decidido implementar los siguientes requisitos funcionales:

- Salida del sistema.
- Seguimiento de rutas.
- Cancelación del seguimiento de rutas.
- Búsqueda de rutas.
- Seguimiento de usuarios.
- Cancelación del seguimiento de usuarios.
- Búsqueda de usuarios.
- Visualización del perfil.
- Modificación del perfil.
- Borrado del perfil.

Al igual que en el capítulo anterior, la especificación de los casos de uso de los requisitos implementados se encuentra en el *Anexo A*.

6.2. Desarrollo

6.2.1. Vistas

En esta iteración los cambios más notables se han producido en las vistas, por lo que nos centratemos en las mismas.

views_usuario.py

Se han añadido los siguientes métodos al fichero:

- *logout(request)*: este método se encarga de limpiar la variable de sesión que almacena el nombre de usuario, terminando así con la sesión del mismo e impidiendo que pueda acceder a la aplicación.
- *miPerfil(request)*: este método se encarga de cargar la información del usuario que tiene una sesión iniciada en el sistema en una página. Además de la información básica del perfil, también muestra información sobre las rutas creadas, las rutas seguidas y los usuarios seguidos. Además, en ella se da la opción al usuario de editar o eliminar el perfil.
- *editarPerfil(request)*: carga un formulario con la información básica del usuario que tiene iniciada sesión en el sistema para que este la modifique según estime conveniente.
- *borrarPerfil(request)*: elimina el perfil del usuario que tiene iniciada sesión en el sistema, limpiando también la variable de sesión que almacena el nombre del usuario.
- *perfil(request, id_usuario)*: carga el perfil de un usuario registrado en el sistema. La única diferencia con la vista *miPerfil(request)* es que a esta no se le pasa como parámetro el *id_usuario*, ya que no es necesario.
- *seguirUsuario(request, id_usuario)*: esta vista añade un usuario seleccionado a la lista de seguidos del usuario que tiene iniciada sesión en el sistema.
- *dejarDeSeguirUsuario(request, id_usuario)*: esta vista quita a un usuario seleccionado de la lista de seguidos del usuario que tiene iniciada sesión en el sistema.

views_ruta.py

La vista más destacable del fichero *views_ruta.py* es *busqueda(request)*. En ella, se toma el valor de un formulario de tipo GET y se buscan rutas y usuarios atendiendo a los siguientes criterios:

- El título de la ruta contiene la consulta.
- La descripción del a ruta contiene la consulta.
- El título o la descripción del día contienen la consulta.
- El creador de la ruta contiene la consulta.
- El alias del usuario contiene la consulta.

Esta vista redirigirá a una nueva página que mostrará dos pestañas: una pestaña mostrará las rutas coincidentes encontradas y la otra, los usuarios.

También se han implementado las vistas *seguirRuta(request, id_ruta)* y *dejarDeSeguirRuta(request, id_ruta)*, las cuales no serán comentadas pues son muy similares a las vistas *seguirUsuario(request, id_usuario)* y *dejarDeSeguirUsuario(request, id_usuario)*.

Finalmente, se han modificado las vistas anteriormente implementadas, en ellas ahora se comprueba si el usuario sigue la ruta que cargan.

6.2.2. *Templates*

En esta iteración se ha añadido el *template busqueda.html*, en el que se muestran los datos relativos a la búsqueda realizada.

También se han implementado *templates* relativos al perfil de usuario, como son *perfil.html*, en el que se muestran los datos de un usuario, y *editarPerfil.html*, para la modificación de los datos de un usuario.

Algunos *templates* implementados en la iteración anterior han sido modificados para añadir ciertas funcionalidades; por ejemplo, en *ruta.html* se ha incluido un objeto de tipo *btn-primary* para seguir o dejar de seguir una ruta. Además, se ha modificado la manera de mostrar los días en la página de la ruta, ahora es una tabla.

Una modificación bastante simple pero a la vez bastante significativa es el despliegue de un *alert* usando *JavaScript* para confirmar el borrado de días y rutas.

6.3. Pruebas

Para mejorar la calidad de las pruebas de búsqueda y seguimiento, se han insertado datos en la base de datos desde un archivo con extensión *.txt* utilizando la consola de comandos.

CAPÍTULO 7

Tercera iteración

7.1. Requisitos implementados

En esta iteración del desarrollo del proyecto se han implementado una serie de requisitos funcionales que no se encuentran en el capítulo 3. Esto se debe a que la lista de requisitos del capítulo 2 no es la lista inicial y ha sufrido modificaciones a lo largo del desarrollo.

En este caso, existía otro rol de usuario llamado *Administrador*, que se encargaba de crear, editar, modificar y borrar lugares de interés creados por él. Además, podía crear, modificar y borrar eventos asociados a los lugares de interés creados por él.

A continuación se enumeran y explican los requisitos que se implementaron en esta iteración:

- Registro de administrador: el administrador podrá registrarse en el sistema introduciendo los datos necesarios.
- Acceso de administrador: el administrador podrá acceder al sistema indicando que es un administrador e introduciendo un nombre de usuario y una contraseña.
- Salida del sistema del administrador: el administrador podrá salir del sistema cerrando sesión.
- Creación de lugares de interés: el administrador podrá crear nuevos lugares de

interés.

- Visualización de lugares de interés: el usuario y el administrador podrán visualizar los lugares de interés que estén creados en el sistema.
- Modificación de lugares de interés: el administrador podrá modificar la información relativa a un lugar de interés creado previamente por él.
- Borrado de lugares de interés: el administrador podrá borrar un lugar de interés previamente creado por él.
- Creación de eventos: el administrador podrá crear un nuevo evento dentro de un lugar de interés específico creado por él.
- Visualización de eventos: el usuario y el administrador podrán visualizar los eventos que estén creados en el sistema.
- Modificación de eventos: el administrador podrá modificar información relativa a un evento creado previamente por él.
- Borrado de eventos: el administrador podrá borrar un evento creado previamente por él.

La especificación de los casos de uso, al igual que en los capítulos anteriores, vendrá hecha en el *Anexo A*.

7.2. Desarrollo

7.2.1. Formularios

Se hace necesario mencionar las modificaciones en el fichero *forms.py* antes de explicar las modificaciones en las vistas.

En primer lugar, se crearon dos nuevas clases, una para el formulario de creación y edición del lugar de interés, y otra para el formulario de creación y edición del evento. Se hace interesante mostrar el código del formulario implementado para la creación y edición de eventos, pues ha sido eliminado en posteriores iteraciones (el porqué de estos cambios viene explicado en el siguiente capítulo), por lo que no es posible encontrarla en el código de la aplicación:

Código 7.1: Clase para el formulario de creación/edición de eventos

```
class EventoForm(forms.Form):
    nombre = forms.CharField(max_length=100, label="Nombre")
    descripcion = forms.CharField(max_length=1000, widget=forms.Textarea)
    inicio = forms.DateField(initial=datetime.date.today())
    fin = forms.DateField(initial=datetime.date.today())
    precio = forms.DecimalField(max_digits=8, decimal_places=2)
```

También se hace necesario comentar que el campo *horario* de la clase *LugarInteresForm* sea de tipo *CharField*, en lugar de ser de tipo *TimeField*, se ha decidido implementar de esta manera para dar más libertad al administrador del lugar de interés, pues los horarios pueden variar dependiendo de si es fin de semana, festivo o laboral.

Se añadió también un campo *admin* de tipo *BooleanField* a la clase *UsuarioForm*, de manera que se puedan crear usuarios con el rol de administrador de lugares de interés. Asimismo, también se añadió un campo *administrador*, también de tipo *BooleanField* a la clase *RegistroForm* para que el administrador de lugares de interés pudiera iniciar sesión.

7.2.2. Vistas

Las vistas implementadas en iteraciones anteriores fueron modificadas para evitar que el administrador de lugares de interés accediera a páginas que no le estaban permitidas, como eran la página de inicio de usuario, las páginas de creación, visualización, modificación y borrado de la ruta, la páginas de creación, visualización, modificación y borrado del día y la página de búsqueda. De este modo, el administrador de lugares de interés solo podía acceder a las páginas de creación, visualización, modificación y borrado de lugares de interés y eventos, además de la página de inicio del administrador de lugares de interés, que mostraba únicamente los lugares de interés creados por el mismo.

Se procede ahora a comentar las vistas que han sido creadas específicamente para esta iteración.

views_admin.py

En este fichero solo se ha implementado una única vista: *inicioAdmin(request)* esta vista cargaría todos los lugares de interés (ordenados por nombre) creados por el administrador que tiene iniciada sesión en el sistema y los mostraría en su página principal de la aplicación.

views_lugar_interes.py

Se puede comprobar que en todas las vistas de este fichero se repite el mismo fragmento de código:

Código 7.2: Control de sesiones de la aplicación

```
if "admin" in request.session:
        admin = models.Administrador.objects.filter(alias =
            request.session["admin"])[0]
        if admin == None:
                form = forms.RegistroForm()
                return render(request, 'login.html', {'form': form})
        else:
                # Implementacion de la vista
else:
        form = forms.RegistroForm()
        return render(request, 'login.html', {'form': form})
```

Este código impide que los usuarios que no hayan iniciado sesión en el sistema puedan ver las páginas del mismo. Y, aunque hayan iniciado sesión, si no son de tipo *Administrador*, tampoco podrán acceder a ciertas páginas de la aplicación.

En este fichero se han implementado las vistas necesarias para hacer un CRUD del lugar de interés. Se puede observar que en las vistas *borrarLugarInteres(request, id_lugarInteres)* y *editarLugarInteres(request, id_lugarInteres)* se comprueba que el administrador que desea realizar la acción sea el creador del lugar de interés. En caso de no serlo, se redirigirá a la página principal de la aplicación.

views_evento.py

La implementación de las vistas de este fichero es similar a la del fichero *views_lugar_interes.py*. En él se desarrollan las vistas necesarias para hacer un CRUD de un evento dentro de un lugar de interés. Al igual que en las vistas desarrolladas en el fichero explicado en el apartado anterior, se comprueba que el administrador tiene una sesión iniciada en el sistema. También se comprueba que el administrador que intenta crear un evento en un lugar de interés sea el mismo que creó el lugar de interés.

7.2.3. *Templates*

En esta iteración se han implementado los siguientes *templates*:

- En el fichero *inicioAdmin.html* se implementa la página principal de la aplicación de cara al administrador. En ella únicamente se muestra un listado con los lugares de interés creados por él.
- En los ficheros *crearEvento.html* y *crearLugarInteres.html* se implementan las páginas que muestran los formularios *EventoForm* y *LugarInteresForm* respectivamente.
- En los ficheros *evento.html* y *lugarInteres.html* se implementan las páginas que muestran la información de los eventos y los lugares de interés, respectivamente.

CAPÍTULO 8

Cuarta iteración

8.1. Requisitos implementados

Debido a que no es posible verificar que el usuario registrado como administrador de lugares de interés sea realmente alguien que tenga relación con dichos lugares de interés, se ha decidido eliminar el rol de administrador. De este modo, todos los usuarios podrán crear y editar cualquier lugar de interés. La idea es que haya la mayor información posible sobre los lugares de interés, y se puedan corregir los errores. Para futuras versiones se propone la creación de un usuario con rol moderador que será el encargado de confirmar que la información de los lugares de interés es correcta.

Teniendo esto en cuenta, los requisitos funcionales desarrollados en esta iteración han sido los siguientes:

- Creación de lugares de interés.
- Visualización de lugares de interés.
- Modificación de lugares de interés.
- Búsqueda de lugares de interés.
- Valoración de rutas.
- Borrado de valoraciones de rutas.
- Valoración de lugares de interés.

- Borrado de valoraciones de lugares de interés.
- Añadido de lugares de interés a días.

8.2. Desarrollo

8.2.1. Modelos

Como se ha comentado en el apartado anterior, no es posible verificar que el administrador de lugares de interés sea una persona que realmente tiene una relación con dichos lugares de interés, por lo que se ha decidido eliminar este tipo de usuario, eliminando así este modelo.

Eliminado el administrador, no tiene sentido que los lugares de interés contengan eventos, pues los usuarios no tienen por qué saber si en un lugar se va a celebrar algo, por lo que también se ha decidido eliminar este modelo.

8.2.2. Vistas

En esta iteración se han modificado las vistas relativas a los lugares de interés y se han añadido vistas para las valoraciones.

Con respecto a las modificaciones en las vistas relativas a los lugares de interés se puede comprobar que el código que antes era común en las mismas ha sido reemplazado por el siguiente:

Código 8.1: Control de sesiones de la aplicación

```
if "usuario" in request.session:
        usuario = models.Usuario.objects.filter(alias =
            request.session["usuario"])[0]
        if usuario == None:
                form = forms.RegistroForm()
                return render(request, 'login.html', {'form': form})
        else:
        # Implementacion de la vista
else:
        form = forms.RegistroForm()
```

```
    return render(request, 'login.html', {'form': form})
```

Como se puede observar, en estas vistas no se comprueba que el usuario sea el creador del lugar de interés, sino que únicamente se comprueba que haya un usuario registrado en el sistema.

En el fichero *views_ruta.py* se han producido varias modificaciones:

- Se ha modificado la vista *busqueda(request)* para que se busquen lugares de interés cuyo título o localidad contengan la consulta.
- Se han añadido las vistas *addValoracion(request)* y *borrarValoracion(request, id_valoracion)* para añadir una valoración a una ruta, o borrar una valoración hecha en una ruta, respectivamente.
- Además, cuando se carga la página de una ruta, se carga también una lista con las valoraciones asociadas a la ruta.
- Se ha implementado una función para calcular la valoración media de una ruta.

En el fichero *views_lugar_interes.py* además de implementar las vistas para el añadido y borrado de valoraciones de los mismos (similares a las vistas de añadido y borrado de valoraciones de las rutas), cabe destacar la vista *buscar_lugares_interes(request)*. Contra todo pronóstico, esta vista se llama desde la página de información de un día, y se utiliza para buscar lugares de interés dada una localidad, y recargará la página del día mostrando un listado con los lugares de interés encontrados. La vista que añade dichos lugares de interés al día es *add_lugar_interes(request)*.

8.2.3. Formularios

En los formularios relativos al acceso y al registro se puede observar que se ha eliminado el campo que hacía referencia al administrador.

Se ha añadido una nueva clase al fichero, *EditarLugarInteresForm*. El propósito de esta clase de formulario es que el usuario no pueda modificar el nombre, la dirección ni la localidad del lugar de interés cuando entra en la página que muestra el formulario de edición del mismo.

Finalmente, se ha implementado una clase *ValoracionForm* que se utilizará para añadir

valoraciones tanto en la página de información de la ruta como en la del lugar de interés.

8.2.4. *Templates*

En esta iteración se han producido los siguientes cambios en los *templates*:

- En *ruta.html* y *lugarInteres.html* se han implementado los formularios de las valoraciones y debajo se muestran las valoraciones relativas a la ruta o al lugar de interés de los cuales se está consultando la información.
- En *dia.html* se ha añadido un formulario de búsqueda para que el creador del día pueda buscar y añadir lugares de interés al mismo. Estos lugares de interés se mostrarán debajo del formulario de búsqueda.

CAPÍTULO 9

Quinta iteración

9.1. Requisitos implementados

En esta iteración se han implementado únicamente dos requisitos funcionales. El peso de la misma ha recaído en el desarrollo de tareas asíncronas, validaciones de datos en los formularios e internacionalización. Los requisitos implementados han sido los siguientes:

- Borrado de lugares de interés en días.
- Recuperación de la contraseña.

También se ha modificado la creación de lugares de interés, ahora se comprueba que no existe un lugar de interés con el mismo nombre en la misma ciudad.

9.2. Desarrollo

9.2.1. Vistas

views_usuario.py

La vista para la recuperación de la contraseña (*recuperarPassword(request)*) se ha implementado en el fichero *views_usuario.py*, en ella, primero se comprueba que el usuario no haya iniciado sesión en el sistema. En caso de no haber iniciado sesión, se

comprueba que exista un usuario con el *email* introducido en el formulario de recuperación de la contraseña.

En caso de que exista un usuario con el *email* introducido, se generará una contraseña con caracteres alfanuméricos de longitud 10 y se asignará al usuario. Luego se procede al envío de la contraseña al *email* vinculado al usuario.

Para el envío del *email* se ha utilizado la clase de *Django EmailMultiAlternatives*. Que se importará de *django.core.mail*. Además, es necesario modificar el fichero *settings.py* y añadir la siguiente configuración:

Código 9.1: Configuración del *email* en el fichero *settings.py*

```
EMAIL_HOST = 'smtp.gmail.com' # Servidor del e-mail, en este caso es Gmail
EMAIL_HOST_PASSWORD = 'password'
EMAIL_HOST_USER = 'email@gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
DEFAULT_FROM_EMAIL = EMAIL_HOST_USER
```

views_lugar_interes.html

Se ha implementado la vista *removeLugarInteresEnDia(request, id_lugarInteres)* que permitirá al creador de un día quitar lugares de interés previamente añadidos.

También se ha modificado la vista *crearLugarInteres(request)* de modo que se valide que no haya ya un lugar de interés con el mismo nombre y la misma localización creado en el sistema.

Además, se ha añadido un método que calculará la valoración media de un lugar de interés.

views_ruta.py y *views_dia.py*

Se han implementado métodos para calcular el precio tanto de una ruta como para calcular el precio de un día.

El precio del día será la suma de los precios de todos los lugares de interés del mismo y el precio de la ruta será la suma del precio de sus días.

9.2.2. *Templates*

Como se ha comentado anteriormente, el peso de esta iteración ha recaído en el desarrollo de operaciones asíncronas y el uso de la API de *Google Maps*, por lo que se hará hincapié en los mismos. Al implementarse en diferentes archivos *.html*, se comentará lo desarrollado, sin dar importancia al archivo en el que se desarrolla.

Tareas asíncronas

Antes de comentar el desarrollo, se hace necesario explicar qué es una tarea asíncrona. A grandes rasgos, en el ámbito de la ingeniería web las tareas asíncronas son aquellas que se ejecutan en segundo plano y evitan que la página tenga que estar refrescándose.

Para implementar estas tareas se ha utilizado *AJAX* y *JQuery*, aunque existen otras tecnologías como pueden ser *Celery* o *Parsley*. ¿Por qué no se ha utilizado *Celery* o *Parsley*? Pues bien, *Celery* está pensado para la ejecución de tareas intensivas que bloquearían la aplicación en caso de no ejecutarse en segundo plano. *Parsley*, por otro lado, se utiliza para las validaciones de campos en formularios en tiempo real, pero se ha quedado obsoleto.

Teniendo esto en cuenta, se procede a explicar las tareas asíncronas desarrolladas en esta iteración:

- **Borrado de días directamente desde la tabla que se muestra en la página de la ruta.** Esta tarea se invoca en el atributo *onclick* dentro de la etiqueta $<a>$ del borrado del día. Como se puede comprobar se ha eliminado el atributo *href* de la misma, esto es debido a que la llamada a la vista que se hace dentro de este atributo tiene más prioridad que la llamada a la tarea en *JQuery*, además de que no hay ningún interés en acceder a una nueva página.

  Se puede observar (listado **??**) que lo primero que hace la función es pedir confirmación sobre el borrado. En caso de aceptarlo, construye la URL para el mismo y se la pasa al método *ajax()* de *JQuery*. El método *ajax()* llamará a la URL cons-

truida anteriormente y, en caso de éxito en la operación, modificará la página actual, en caso de fallo, lanzará un mensaje de error.

Código 9.2: Borrado de días asíncrono

```
<td><a onclick="borrarDia({{dia.id}})"><span class="glyphicon
    glyphicon-trash" aria-hidden="true"></span>{% trans "Borrar" %}</a>
<script type="text/javascript">
        function borrarDia(id_dia){
                var confirmacion = confirm('Borrar el dia?');
                if (confirmacion == true){
                        var $url = "/"+id_dia+"/borrarDia/";
                        $.ajax({
                                url: $url,
                                success: function(data){
                                        $('body').html(data);
                                },
                                error: function(){
                                        alert("Error");
                                }
                        });
                }
        }
</script>
\label{borrado_dias_asincrono}
```

- **Añadido de valoraciones a rutas.** (listado **??**) Para realizar esta operación es necesario que la etiqueta *<form>* tenga un único atributo, este será el atributo *id*. La tarea se realizará creando un objeto con el formulario en *JQuery* y llamando desde el objeto al método *submit*. Dentro del método *submit* se llama al método *ajax()*, en el que se serializan los datos para su tratamiento, se indica si el formulario es de tipo *GET* o *POST* y la URL de la vista en la que se tratará el formulario, en este caso, */addValoracion/*.

Código 9.3: Añadido de valoraciones asíncrono

```
<form id="form-valoracion-ruta">{% csrf_token %}
        <input type="hidden" name="id_ruta" value={{ruta.id}} />
        <table>
                {{ form.as_table }}
        </table>
```

```
        <input type="submit" value="Valorar"><br/>
</form>
<script type="text/javascript">
        $('#form-valoracion-ruta').submit(function(event){
                event.preventDefault();
                $.ajax({
                        data: $(this).serialize(),
                        type: 'POST',
                        url: '/addValoracion/',
                        success: function(data){
                                $('body').html(data);
                        },
                        error: function(){
                                alert("Error");
                        }
                })
        });
</script>
\label{add_valoraciones}
```

- Del resto de tareas asíncronas implementadas no se va a comentar el código, pues esto supondría repetir los dos puntos anteriores. Sí que se procede a enumerarlas:
  - Seguimiento de rutas.
  - Cancelación del seguimiento de rutas.
  - Borrado de valoraciones de rutas.
  - Búsqueda de lugares de interés en la página de días.
  - Añadido de lugares de interés a días.
  - Borrado de lugares de interés en días.
  - Seguimiento de usuarios.
  - Cancelación del seguimiento de usuarios.
  - Añadido de valoraciones a lugares de interés.
  - Borrado de valoraciones de lugares de interés.

API de *Google Maps*

Para mejorar la experiencia del usuario, se muestran mapas indicando dónde se encuentran los lugares de interés de las rutas. Para ello se ha hecho uso de la API de *Google Maps*, que ofrece diferentes funcionalidades. En este caso se utilizará la geolocalización dada una dirección.

Dependiendo de si se desea mostrar un único marcador (como es el caso de la página de información de los lugares de interés) o múltiples marcadores (como es el caso de la página de información de día o ruta), la implementación de la función *initMap()* (función encargada de mostrar el mapa con los marcadores) será diferente.

En el caso de querer mostrar un único marcador, la función *initMap()* creará un nuevo objeto de tipo *Map* en el que se indicará el estilo del mapa y otro de tipo *Geocoder*, al que se le indicará la dirección en la que se desea poner el marcador. También se llamará a una función que será la encargada de localizar el marcador en la dirección indicada y centrará el mapa en función del marcador.

En el caso de querer mostrar más marcadores, el desarrollo se complica. En la función *initMap()* únicamente se inicializan las variables globales. También se implementa una función *geocodeAddress(address, next)* que geolocaliza la dirección y llama a la función *createMarker(add, lat, lng)*, que será la encargada de añadir el marcador al mapa.

Internacionalización

La internacionalización se ha llevado a cabo tanto en los *templates* como en el fichero *forms.py*, aunque de maneras diferentes.

En *forms.py* se utiliza la función *_()* para indicar las cadenas que son traducibles, mientras que en los *templates* es necesario incluir { *% load i18n %*} antes de la etiqueta <*html*>. En los *templates* las cadenas que son traducibles se indican de la siguiente manera:
{ *% trans "Cadena" %*}

La información sobre cómo hacer las cadenas traducibles ha sido extraída de la do-

cumentación de *Django*, donde no viene mucha información sobre cómo traducir las cadenas. Debido a la falta de tiempo no se ha podido profundizar en cómo se traducirían las mismas.

CAPÍTULO 10

Sexta iteración

10.1. Modificaciones

Este iteración iteración ha consistido en arreglar fallos menores, hacer la interfaz más intuitiva y añadir un *timeline* a la página principal de la aplicación.

La implementación del *timeline* corresponde a la implementación del requisito *Monitorización de acciones*.

10.2. Desarrollo

10.2.1. Modelos

Para esta iteración se ha modificado el archivo *models.py*. En él se ha implementado una case *Log* que almacenará los cambios que hace un usuario en una ruta.

También se han quitado los campos que hacen referencia a imágenes de los modelos.

10.2.2. Vistas

En la vista *inicio(request)* en el archivo *views_usuario.py* se cargan todos los objetos de la clase *Log* cuyo usuario esté en la lista de seguidos del usuario que tiene iniciada la sesión, o cuya ruta esté en la lista de rutas seguidas del usuario que tiene iniciada la sesión.

En las vistas *crearRuta(request)* y *editarRuta(request, id_ruta)* en el fichero *views_ruta.py* y en las vistas *anadirDia(request, id_ruta)*, *borrarDia(request, id_dia)* y *editarDia(request, id_dia)* en el fichero *views_dia.py* se observa un fragmento de código común:

Código 10.1: *Log* de operaciones

```
log = models.Log(usuario=usuario,
                                ruta=ruta,
                                fecha=timezone.now(),
                                log='accion realizada por el usuario')
log.save()
```

En él se guardan los cambios que ha realizado el usuario para posteriormente mostrarlos en el *Timeline*.

En el campo *log* se almacena la acción que realizada, ya sea creación o modificación.

10.2.3. *Templates*

En esta iteración los cambios realizados en los ficheros *.html* son principalmente estéticos:

- Los enlaces que antes había en el *Navbar* ahora están contenidos dentro de un *Dropdown* que se encuentra en el *Navbar*.
- En el *Dropdown* se han añadido enlaces a la página de creación de ruta, lugar de interés, perfil de usuario, edición del perfil y *log out*.
- El *Navbar* pasa a estar fijo en la parte superior de la página, de manera que el usuario siempre lo tiene a la vista.
- En la página de información de la ruta se ha añadido un enlace al perfil del creador de la misma.

- La página principal ahora muestra un panel con las rutas creadas por el usuario y las rutas que sigue y también el *Timeline* comentado anteriormente.
- Finalmente, se han mejorado los botones *Volver*, estos botones antes redireccionaban a la página principal de la aplicación y ahora redireccionan a la página que se ha visitado anteriormente.

CAPÍTULO 11

Conclusiones

11.1. Opinión sobre la unión de *Django* y *Amazon Web Services*

Para poner en contexto a futuros lectores de este proyecto, mi intención a la hora de desarrollar esta aplicación era utilizar *Java Server Faces*, pero debido a la poca información que encontré en *Internet* de aplicaciones *Web* desarrolladas en *Java Server Faces* y desplegadas en *Amazon Web Services*, decidí aventurarme a utilizar *Django* (a pesar de que no tenía conocimientos de *Python*) para el mismo pues, tras varios días investigando, me dio la sensación de que había una gran cantidad de información respecto al tema y no perdería mucho tiempo investigando sobre cómo desplegar la aplicación.

Pues bien, a favor de *Django* se puede decir que es una tecnología fácil de aprender una vez se tiene experiencia en el mundo de la programación (a penas tardé 8 horas en aprender *Python* y *Django*); las conexiones con bases de datos locales son mucho más sencillas que las de *Java*, pues solo hay que modificar unas líneas de código; finalmente, pero no menos importante, la existencia de ficheros como *forms.py* ahorra mucho trabajo a la hora de programar el *front-end* de la aplicación, pues ahorra a los programadores el tener que hacer formularios a mano.

A pesar de esto, *Django* presenta también inconvenientes, sobre todo a la hora de desplegar una aplicación en un servidor. Como se ha comentado en el párrafo anterior, *Django* facilita al programador las conexiones con las bases de datos locales, pero, por el contrario, conectar *Django* a una base de datos remota requiere de mucho tiempo y paciencia.

11.2. Análisis de resultados

En el anteproyecto se propuso el desarrollo de una aplicación *Web* utilizando el *cloud* de Amazon. Si bien se ha desarrollado la aplicación web al completo, no me ha sido posible desplegarla en *Amazon Web Services*, pues la documentación disponible en *Internet* era escasa o estaba desfasada.

También se propuso la entrega de un manual de usuario, este se puede encontrar en el *Anexo C*.

Sería interesante hablar también de las estimaciones que se hicieron en el anteproyecto. Si bien se hizo la estimación del cuadro 11.1, la realidad se aleja bastante de la misma, quedando el tiempo empleado como se refleja en el cuadro 11.2.

Fase	Horas
Análisis de requisitos	10
Análisis de requisitos	51
Diseño	51
Implementación	52
Pruebas	51
Integración	51
Memoria	30

Cuadro 11.1: Estimaciones hechas en el anteproyecto

El estudio inicial se prolongó debido a las horas investigando primero cómo alojar la aplicación web implementada en *Django* en *Amazon Web Services* y luego las horas investigando cómo hacer las migraciones a una base de datos remota.

Fase	Horas
Estudio inicial	51
Análisis de requisitos	14
Diseño	3
Implementación	104
Pruebas	26
Integración	8
Memoria	90

Cuadro 11.2: Horas empleadas en el proyecto

Al ser una idea completamente mía, las fases de análisis de requisitos y diseño se han acortado notablemente.

Debido a que no tenía conocimientos de *Django*, la fase de implementación se ha alargado hasta durar el doble de horas de lo previsto.

CAPÍTULO 12

Bibliografía

Bibliografía

Adding an amazon rds db instance to your python application environment. (s.f.). Descargado de http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-rds.html (recuperado el 25 de junio de 2017) (No citado)

Amazon web services. (s.f.). Descargado de https://es.wikipedia.org/wiki/Amazon_Web_Services (recuperado el 5 de junio de 2017) (Citado en la página 5.)

Api de google maps. (s.f.). Descargado de https://developers.google.com/maps/documentation/javascript/ (recuperado el 25 de junio de 2017) (No citado)

Bootstrap. (s.f.). Descargado de http://getbootstrap.com/ (recuperado el 25 de junio de 2017) (Citado en la página 6.)

Documentación de django. (s.f.). Descargado de https://www.djangoproject.com/ (recuperado el 25 de junio de 2017) (Citado en la página 2.)

What is iam? (s.f.). Descargado de http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html (recuperado el 5 de junio de 2017) (Citado en la página 6.)

Bibliografía

APÉNDICE A

Casos de uso

Nombre: registro
Actor primario: usuario
Precondición: el usuario no debe estar registrado en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario tendrá una cuenta registrada en el sistema.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Crear una cuenta”.
2. El sistema muestra un formulario con los campos necesarios para completar un registro.
3. El usuario cumplimenta los campos del formulario.
4. El usuario selecciona la opción “Enviar”.
5. El sistema comprueba la validez de los datos introducidos por el usuario.
6. El sistema crea el perfil de usuario.
7. El sistema muestra la página principal de la aplicación.

Extensiones:

3.a. El usuario no ha completado alguno de los campos obligatorios.

3.a.1. El sistema reenvía al usuario el formulario indicando los campos que deben ser completados.

3.a.2. El usuario completa los campos restantes.

3.a.3. Sigue por el paso 4 del escenario principal.

3.b. El usuario ha introducido un alias que ya existe.

3.b.1. El sistema reenvía al usuario el formulario indicando que ya existe un usuario con ese alias.

3.b.2. El usuario cambia el alias.

3.b.3. Sigue por el paso 4 del escenario principal.

3.c. Los datos introducidos en los campos relativos a la contraseña no coinciden.

3.c.1. El sistema reenvía al usuario el formulario indicando que los campos de la contraseña no coinciden.

3.c.2. El usuario corrige el error.

3.c.3. Sigue por el paso 4 del escenario principal.

3.d. Los datos introducidos en los campos relativos al *e-mail* no coinciden.

3.d.1 El sistema reenvía al usuario el formulario indicando que los campos del *e-mail* no coinciden.

3.d.2. El usuario corrige el error.

3.d.3. Sigue por el paso 4 del escenario principal.

3.e. El usuario ha introducido una contraseña demasiado corta.

3.e.1. El sistema reenvía el formulario al usuario indicando que la contraseña es demasiado corta.

3.e.2. El usuario corrige el error.

3.e.3. Sigue por el paso 4 del escenario principal.

3.f. El usuario ha introducido un *e-mail* sin arroba.

3.f.1. El sistema reenvía al usuario el formulario indicando que el *e-mail* debe llevar una arroba.

3.f.2. El usuario corrige el error.

3.f.3. Sigue por el paso 4 del escenario principal.

3.g. El usuario introduce un caracter no alfanumérico en el alias.

3.g.1. El sistema reenvía al usuario el formulario indicando que el alias debe ser alfanumérico.

3.g.2. El usuario corrige el error.

3.g.3. Sigue por el paso 4 del escenario principal.

3.h. El usuario ha introducido un *e-mail* que ha está siendo utilizado en el sistema.

3.h.1. El sistema reenvía al usuario el formulario indicando que el *e-mail* ya está siendo utilizado.

3.h.2. El usuario corrige el error.

3.h.3. Sigue por el paso 4 del escenario principal.

Nombre: acceso
Actor primario: usuario
Precondición: el usuario debe tener una cuenta registrada en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá acceder al sistema.
Escenario principal (camino feliz):

1. El usuario introduce sus datos en el formulario de la pantalla de inicio de la aplicación.
2. El usuario selecciona “Acceder”.
3. El sistema verifica los datos del usuario.
4. El sistema confirma el éxito del acceso.
5. El sistema muestra la pantalla principal de la aplicación.

Extensiones:

1.a. El usuario introduce datos incorrectos.

1.a.1. El sistema notifica al usuario de que se ha producido un error.

1.b.1. Sigue por el paso 1 del escenario principal.

Nombre: creación de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber accedido al mismo.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario crea una ruta con éxito.

Escenario principal (camino feliz):

1. El usuario selecciona la opción “Crear una nueva ruta”.
2. El sistema muestra un formulario con los datos necesarios para crear una ruta.
3. El usuario cumplimenta los datos que estime necesarios.
4. El usuario selecciona la opción “Crear”.
5. El sistema valida los datos recibidos a través del formulario.
6. El sistema muestra la página principal de la aplicación.

Extensiones:

3.a. El usuario no cumplimenta todos los campos obligatorios.

3.a.1. El sistema reenvía al usuario el formulario indicando los campos que deben ser completados.

3.a.2. El usuario cumplimenta los datos restantes.

3.a.3. Sigue por el paso 4 del escenario principal.

*.a. El usuario no cumplimenta los campos y cancela la creación[1].

*.a.1. El sistema cierra el formulario de creación de la ruta, cancelando la operación y devolviendo al usuario a la página anterior.

Nombre: visualización de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber al menos una ruta creada en el mismo.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá visualizar la página de información de una ruta.
Escenario principal (camino feliz):

1. El usuario selecciona una ruta existente en el sistema.
2. El sistema muestra una página con la información de la ruta.

Extensiones:

Nombre: modificación de rutas
Actor primario: usuario

[1] Esta extensión puede darse en cualquier paso del escenario principal.

Precondición: el usuario debe estar registrado en el sistema y debe tener creada al menos una ruta.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: El usuario modificará exitosamente la ruta deseada.
Escenario principal (camino feliz):

1. El usuario selecciona una ruta de entre las creadas por él.
2. El sistema muestra una página con la información de la ruta y las opciones de la misma.
3. El usuario selecciona la opción “Modificar”.
4. El sistema muestra un formulario con los campos modificables de la ruta.
5. El usuario modifica los campos que estime necesarios.
6. El usuario selecciona la opción “Guardar”.
7. El sistema valida los cambios.
8. El sistema guarda los cambios.
9. El sistema muestra la página principal de la aplicación.

Extensiones:

*.a. El usuario cancela la acción[2].

*.a.1. El sistema cierra el formulario de modificación de la ruta, cancelando la operación y devolviendo al usuario a la página anterior.

5.a. El usuario no cumplimenta todos los campos obligatorios.

5.a.1. El sistema reenvía el formulario al usuario indicando los campo que deben ser completados.

5.a.2. El usuario modifica los campos no válidos.

5.a.3. Sigue por el paso 6 del escenario de éxito.

Nombre: borrado de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe tener creada al menos una ruta.
Garantía mínima: en caso de error, el sistema notificará al usuario.

[2]Esta extensión puede darse en cualquier paso del escenario principal.

Garantía de éxito: el usuario borrará una ruta que haya creado previamente.
Escenario principal (camino feliz):

1. El usuario selecciona una ruta de las que ha creado.
2. El sistema muestra una página con la información de la ruta y las opciones de la misma.
3. El usuario selecciona la opción “Borrar”.
4. El sistema pide confirmación.
5. El usuario confirma la acción.
6. El sistema elimina la ruta.
7. El sistema muestra la página principal de la aplicación.

Extensiones:

5.a. El usuario cancela la acción.

5.a.1. El sistema muestra la página con la información de la ruta y las opciones de la misma.

Nombre: creación de días
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe tener creada al menos una ruta.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario creará un día asociado a una ruta.
Escenario principal (camino feliz):

1. El usuario selecciona una ruta de entre las creadas por él.
2. El sistema muestra una página con la información de la ruta y las opciones de la misma.
3. El usuario selecciona la opción “Añadir día”.
4. El sistema muestra un formulario con los campos necesarios para la creación de un día.
5. El usuario cumplimenta los campos que estime necesarios.
6. El usuario selecciona la opción “Guardar”.
7. El sistema valida los campos.

8. El sistema crea un día nuevo asociado a una ruta.
9. El sistema muestra la página de la ruta asociada al día.

Extensiones:

*.a. El usuario cancela la operación[3].

*.a.1. El sistema cierra el formulario de creación del día, cancelando la operación y devolviendo al usuario a la página anterior.

5.a. El usuario no completa todos los campos obligatorios.

5.a.1. El sistema reenvía el formulario indicando los campos que deben ser completados.

5.a.2. El usuario completa los campos obligatorios.

5.a.3. Sigue por el paso 6 del escenario principal.

Nombre: visualización de días
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber creada al menos una ruta con un día asociado.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá ver la página con los datos del día seleccionado
Escenario principal (camino feliz):

1. El usuario selecciona un día existente en el sistema.
2. El sistema muestra una página con la información del día.

Extensiones:

Nombre: modificación de días
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe tener creada al menos una ruta y debe tener creado al menos un día dentro de dicha ruta.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá modificar los datos correspondientes a un día previamente creado por él
Escenario principal (camino feliz):

1. El usuario selecciona una ruta creada por él.

[3]Esta extensión puede darse en cualquier paso del escenario principal.

2. El usuario muestra la pantalla de la ruta con las opciones de la misma.
3. El usuario selecciona un día dentro de la ruta.
4. El sistema muestra una página con la información del día y las opciones del mismo.
5. El usuario selecciona la opción “Modificar”.
6. El sistema muestra un formulario con los campos modificables del día.
7. El usuario modifica los campos que estime oportunos.
8. El usuario selecciona la opción “Guardar”.
9. El sistema valida los cambios en los campos.
10. El sistema guarda los cambios.
11. El sistema muestra la página de la ruta asociada al día.

Extensiones:

*.a. El usuario cancela la operación[4].

*.a.1. El sistema cierra el formulario de modificación del día, cancelando la operación y devolviendo al usuario a la página anterior.

7.a. Usuario no completa todos los campos obligatorios.

7.a.1. El sistema reenvía el formulario indicando los campos que deben ser completados.

7.a.2. El usuario completa los campos obligatorios.

7.a.3. Sigue por el paso 8 del escenario principal.

Nombre: borrado de días
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe tener creada al menos una ruta y debe tener creado al menos un día dentro de la ruta.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá borrar un día dentro de una ruta creada por él.
Escenario principal (camino feliz):

1. El usuario selecciona una ruta creada por él.

[4]Esta extensión puede darse en cualquier paso del escenario principal a partir del paso 6.

2. El sistema muestra una página con la información de la ruta y las opciones de la misma.
3. El usuario selecciona un día dentro de la ruta.
4. El sistema muestra una página con la información del día y las opciones del mismo.
5. El usuario selecciona la opción “Borrar”.
6. El sistema pide confirmación.
7. El usuario confirma la acción.
8. El sistema borra el día.
9. El sistema muestra una página con la información de la ruta y las opciones de la misma.

Extensiones:

7.a. El usuario cancela la acción.

7.a.1. El sistema no borra el día,

7.a.2. El sistema muestra una página con la información de la ruta y las opciones de la misma.

Nombre: salida del sistema
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber accedido al mismo.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá cerrar sesión en el sistema.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Cerrar sesión”.
2. El sistema borra la sesión del usuario.
3. El sistema muestra la página de inicio de la aplicación.

Extensiones:

Nombre: búsqueda de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al

mismo y debe haber al menos una ruta creada en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el sistema mostrará un listado de coincidencias.
Escenario principal (camino feliz):

1. El usuario introduce un parámetro de búsqueda.
2. El sistema comprueba si existe alguna coincidencia.
3. El sistema muestra una página con las coincidencias.

Extensiones:

1.a. El usuario introduce un parámetro vacío.

1.a.1. El sistema no hace nada.

1.b. El usuario introduce un parámetro que no tiene ninguna coincidencia.

1.b.1. El sistema muestra en la página de búsqueda un mensaje indicando que no se han encontrado coincidencias.

Nombre: seguimiento de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos una ruta creada en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario añade la ruta a su lista de seguidas.
Escenario principal (camino feliz):

1. El usuario selecciona la ruta que desea seguir.
2. El sistema muestra la página de la ruta.
3. El usuario selecciona la opción “Seguir”.
4. El sistema comprueba que el usuario no sigue ya esa ruta.
5. El sistema añade la ruta a la lista de rutas seguidas del usuario.

Extensiones:

3.a. El usuario ya sigue la ruta seleccionada.

3.a.1. El sistema notifica al usuario de que ya sigue la ruta seleccionada.

3.a.2. El sistema cancela la acción.

3.b. El usuario es el creador de la ruta seleccionada.

3.b.1. El sistema notifica al usuario de que no puede seguir una ruta creada por él.

3.b.2. El sistema cancela la acción.

Nombre: cancelación del seguimiento de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe seguir al menos una ruta.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario elimina la ruta de su lista de seguidas.
Escenario principal (camino feliz):

1. El usuario selecciona la ruta que desea dejar de seguir.
2. El sistema muestra la página de la ruta.
3. El usuario selecciona la opción “Dejar de seguir”.
4. El sistema comprueba que el usuario sigue la ruta.
5. El sistema elimina la ruta de la lista de rutas seguidas del usuario.

Extensiones:

3.a. El usuario no sigue la ruta seleccionada.

3.a.1. El sistema notifica al usuario de que no sigue la ruta seleccionada.

3.a.2. El sistema cancela la acción.

3.b. El usuario es el creador de la ruta seleccionada.

3.b.1. El sistema notifica al usuario de que no puede dejar de seguir una ruta creada por él.

3.b.2. El sistema cancela la acción.

Nombre: búsqueda de usuarios
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos otro usuario registrado en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el sistema mostrará un listado de coincidencias.
Escenario principal (camino feliz):

1. El usuario introduce un parámetro de búsqueda.
2. El sistema comprueba si existe alguna coincidencia.
3. El sistema muestra una página con las coincidencias.

Extensiones:

1.a. El usuario introduce un parámetro vacío.

1.a.1. El sistema no hace nada.

1.b. El usuario introduce un parámetro que no tiene ninguna coincidencia.

1.b.1. El sistema muestra en la página de búsqueda un mensaje indicando que no se han encontrado coincidencias.

Nombre: seguimiento de usuarios
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos otro usuario registrado en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario añade al otro usuario a su lista de seguidos.
Escenario principal (camino feliz):

1. El usuario selecciona al usuario que desea seguir.
2. El sistema muestra la página del usuario.
3. El usuario selecciona la opción “Seguir”.
4. El sistema comprueba que el usuario no sigue ya a ese usuario.
5. El sistema añade al otro usuario a la lista de usuarios seguidos del usuario.

Extensiones:

3.a. El usuario ya sigue al usuario seleccionado.

3.a.1. El sistema notifica al usuario de que ya sigue al usuario seleccionado.

3.a.2. El sistema cancela la acción.

3.b. El usuario se sigue a sí mismo.

3.b.1. El sistema notifica al usuario de que no puede seguirse a sí mismo.

3.b.2. El sistema cancela la acción.

Nombre: cancelación del seguimiento de usuarios
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe seguir al menos a un usuario.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario elimina al otro usuario de su lista de seguidos.
Escenario principal (camino feliz):

1. El usuario selecciona al usuario que desea dejar de seguir.
2. El sistema muestra la página del otro usuario.
3. El usuario selecciona la opción “Dejar de seguir”.
4. El sistema comprueba que el usuario sigue al otro usuario.
5. El sistema elimina al otro usuario de la lista de usuarios seguidos del usuario.

Extensiones:

3.a. El usuario no sigue al usuario seleccionado.

3.a.1. El sistema notifica al usuario de que no sigue al usuario seleccionado.

3.a.2. El sistema cancela la acción.

3.b. El usuario se deja de seguirse a sí mismo.

3.b.1. El sistema notifica al usuario de que no puede dejar de seguirse a sí mismo.

3.b.2. El sistema cancela la acción.

Nombre: modificación del perfil
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber accedido al mismo.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario modifica su perfil con éxito.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Mi perfil”.
2. El sistema muestra la página del perfil de usuario.
3. El usuario selecciona la opción “Modificar perfil”.

4. El sistema muestra en una página un formulario con los campos modificables del perfil.
5. El usuario modifica los campos que estime necesarios.
6. El usuario selecciona la opción “Guardar.
7. El sistema valida los cambios.
8. El sistema guarda los cambios.
9. El sistema muestra la página del perfil de usuario.

Extensiones:

*.a. El usuario cancela la acción[5].

*.a.1. El sistema cierra el formulario de modificación del perfil, cancelando la operación y devolviendo al usuario a la página anterior.

5.a. El usuario no completa todos los campos obligatorios.

5.a.1. El sistema reenvía el formulario indicando los campos que deben ser completados.

5.a.2. El usuario completa los campos obligatorios.

5.a.3. Sigue por el paso 6 del escenario principal.

5.b. El usuario introduce un alias que ya existe.

5.b.1. El sistema reenvía al usuario el formulario indicando que ya existe un usuario con ese alias.

5.b.2. El usuario cambia el alias.

5.b.3. Sigue por el paso 6 del escenario principal.

5.c. Los datos introducidos en el campo relativo a la antigua contraseña no son válidos.

5.c.1. El sistema reenvía al usuario el formulario indicando que la contraseña no es correcta.

5.c.2. El usuario introduce una contraseña correcta.

5.c.3. Sigue por el paso 6 del escenario principal.

5.d. Los datos introducidos en los campos relativos a la nueva contraseña no coinciden.

[5]Esta extensión puede darse en cualquier paso del escenario principal.

5.d.1. El sistema reenvía al usuario el formulario indicando que los campos de la nueva contraseña no coinciden.

5.d.2. El usuario corrige el error.

5.d.3. Sigue por el paso 6 del escenario principal.

5.e. Los datos introducidos en los campos relativos al *e-mail* no coinciden.

5.e.1. El sistema reenvía al usuario el formulario indicando que los campos del *e-mail* no coinciden.

5.e.2. El usuario corrige el error.

5.e.3. Sigue por el paso 6 del escenario principal.

5.f. El usuario introduce un *e-mail* que ya está siendo utilizado.

5.f.1. El sistema reenvía al usuario el formulario indicando que el *e-mail* ya está siendo utilizado.

5.f.2. El usuario corrige el error.

5.f.3. Sigue por el paso 6 del escenario principal.

5.g. El usuario introduce una contraseña demasiado corta.

5.g.1. El sistema reenvía el formulario al usuario indicando que la contraseña es demasiado corta.

5.g.2. El usuario corrige el error.

5.g.3. Sigue por el paso 6 del escenario principal.

5.h. El usuario ha introducido un *e-mail* sin arroba.

5.h.1. El sistema reenvía al usuario el formulario indicando que el *e-mail* debe llevar una arroba.

5.h.2. El usuario corrige el error.

5.h.3. Sigue por el paso 6 del escenario principal.

5.i. El usuario introduce un caracter no alfanumérico en el alias.

5.i.1. El sistema reenvía al usuario el formulario indicando que el alias debe ser alfanumérico.

5.i.2. El usuario corrige el error.

5.i.3. Sigue por el paso 6 del escenario principal.

Nombre: borrado del perfil
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber accedido al mismo.
Garantía mínima: en caso de fallo, el sistema notificará al usuario.
Garantía de éxito: el usuario borrará su perfil.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Mi perfil”.
2. El sistema muestra la página del perfil del usuario.
3. El usuario selecciona la opción “Eliminar perfil”.
4. El sistema pide confirmación al usuario.
5. El usuario confirma la acción.
6. El sistema elimina el usuario de la base de datos.
7. El sistema muestra la página de inicio de la aplicación.

Extensiones:

5.a. El usuario cancela la acción.

5.a.1. El sistema cancela la acción.

5.a.2. El sistema muestra la página del perfil del usuario.

Nombre: creación de lugares de interés
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema y debe haber accedido al mismo.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario crea con éxito un lugar de interés
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Crear lugar de interés”.
2. El sistema muestra una página con el formulario con los datos del lugar de interés.
3. El usuario cumplimenta los datos que estime necesarios.
4. El usuario selecciona la opción “Crear”.

5. El sistema crea el nuevo lugar de interés.
6. El sistema muestra la página de inicio de la aplicación.

Extensiones:

*.a. El usuario cancela la operación[6]

*.a.1. El sistema cierra el formulario de creación del lugar de interés, cancelando la operación y devolviendo al usuario a la página anterior.

3.a. El usuario no cumplimenta los datos obligatorios.

3.a.1. El sistema reenvía el formulario al usuario indicando los campos que deben ser completados.

3.a.2. El usuario cumplimenta los datos restantes.

3.a.3. Sigue por el paso 4 del escenario principal.

3.b. El usuario introduce datos de un lugar de interés que ya existe.

3.b.1. El sistema reenvía al usuario el formulario indicando el error.

3.b.2. El usuario corrige los datos erróneos.

3.b.3. Sigue por el paso 4 del escenario principal.

Nombre: modificación de lugares de interés
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos un lugar de interés creado en él.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario modificará exitosamente un lugar de interés.
Escenario principal (camino feliz):

1. El usuario selecciona un lugar de interés.
2. El sistema muestra la página de información del lugar de interés.
3. El usuario selecciona la opción “Modificar”.
4. El sistema muestra una página con el formulario con los campos modificables del lugar de interés.
5. El usuario modifica los campos que estime necesarios.
6. El usuario selecciona la opción “Guardar”.

[6]Esta extensión puede darse en cualquier paso del escenario principal.

7. El sistema valida los cambios.
8. El sistema guarda los cambios.
9. El sistema muestra la página principal de la aplicación.

Extensiones:

*.a. El usuario cancela la acción[7].

*.a.1. El sistema cierra el formulario de modificación del lugar de interés, cancelando la operación y devolviendo al usuario a la página anterior.

5.a. El usuario no completa todos los campos obligatorios.

5.a.1. El sistema reenvía al usuario el formulario indicando los campos que debe completar.

5.a.2. El usuario corrige los errores.

5.a.3. Sigue por el paso 6 del escenario principal.

Nombre: visualización de lugares de interés
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos un lugar de interés creado.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario será capaz de ver la información de un determinado lugar de interés.
Escenario principal (camino feliz):

1. El usuario selecciona un lugar de interés.
2. El sistema muestra la página del lugar de interés seleccionado.

Extensiones:

1.a. El lugar de interés no existe.

1.a.1. El sistema muestra una página indicando el error.

Nombre: valoración de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos una ruta creada.

[7] Esta extensión puede darse en cualquier paso del escenario principal

Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario será capaz de valorar una ruta.
Escenario principal (camino feliz):

1. El usuario selecciona la ruta que desea comentar.
2. El sistema muestra la página de la ruta.
3. El usuario completa el formulario de la valoración.
4. El usuario selecciona “Publicar”.
5. El sistema añade la valoración a la ruta.
6. El sistema muestra la página de la ruta con la valoración.

Extensiones:

3.a. el usuario no completa alguno de los campos obligatorios.

3.a.1. El sistema reenvía al usuario el formulario indicando los campos que son obligatorios.

3.a.2. El usuario completa los campos obligatorios.

3.a.3. sigue por el paso 4 del escenario principal.

Nombre: borrado de valoraciones de rutas
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, haber accedido al mismo y tener creada alguna valoración en una ruta previamente seleccionada.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá borrar una valoración previamente hecha en una ruta.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Eliminar valoración”.
2. El sistema pide confirmación.
3. El usuario confirma la acción.
4. El sistema elimina la valoración.
5. El sistema muestra la página de la ruta sin la valoración del usuario.

Extensiones:

3.a. El usuario cancela la acción.

3.a.1. El sistema cancela la acción.

3.a.2. El sistema muestra la página de la ruta.

Nombre: valoración de lugares de interés
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos un lugar de interés creado.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario será capaz de valorar un lugar de interés.
Escenario principal (camino feliz):

1. El usuario selecciona el lugar de interés que desea comentar.
2. El sistema muestra la página del lugar de interés.
3. El usuario completa el formulario de la valoración.
4. El usuario selecciona “Publicar”.
5. El sistema añade la valoración al lugar de interés.
6. El sistema muestra la página del lugar de interés con la valoración.

Extensiones:

3.a. el usuario no completa alguno de los campos obligatorios.

3.a.1. El sistema reenvía al usuario el formulario indicando los campos que son obligatorios.

3.a.2. El usuario completa los campos obligatorios.

3.a.3. sigue por el paso 4 del escenario principal.

Nombre: borrado de valoraciones de lugares de interés
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, haber accedido al mismo y tener creada alguna valoración en un lugar de interés previamente seleccionado.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario eliminará una valoración previamente hecha en un lugar de interés.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Eliminar valoración”.

2. El sistema pide confirmación.
3. El usuario confirma la acción.
4. El sistema elimina la valoración.
5. El sistema muestra la página del lugar de interés sin la valoración del usuario.

Extensiones:

3.a. El usuario cancela la acción.

3.a.1. El sistema cancela la acción.

3.a.2. El sistema muestra la página del lugar de interés.

Nombre: búsqueda de lugares de interés
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe haber al menos un lugar de interés creado en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el sistema mostrará un listado de coincidencias.
Escenario principal (camino feliz):

1. El usuario introduce un parámetro de búsqueda.
2. El sistema comprueba si existe alguna coincidencia.
3. El sistema muestra una página con las coincidencias.

Extensiones:

1.a. El usuario introduce un parámetro vacío.

1.a.1. El sistema no hace nada.

1.b. El usuario introduce un parámetro que no tiene ninguna coincidencia.

1.b.1. El sistema muestra en la página de búsqueda un mensaje indicando que no se han encontrado coincidencias.

Nombre: añadido de lugares de interés a días
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe tener al menos un día creado dentro de una ruta. **Garantía mínima:** en caso de error, el sistema notificará al usuario. **Garantía de éxito:** el usuario podrá

añadir uno o más lugares de interés a un día. **Escenario principal (camino feliz):**

1. El usuario selecciona una ruta.
2. El sistema muestra la página de la ruta.
3. El usuario selecciona un día.
4. El sistema muestra la página del día.
5. El usuario introduce el nombre de una ciudad en el buscador de lugares de interés.
6. El sistema muestra un listado con los lugares de interés que hay en la ciudad que no se han añadido al día.
7. El usuario selecciona uno o varios lugares de interés del listado.
8. El usuario selecciona “Añadir”.
9. El sistema añade los lugares de interés al día.
10. El sistema muestra la página del día con los nuevos lugares de interés.

Extensiones:

5.a. El usuario no introduce ninguna ciudad en el buscador de lugares de interés.

5.a.1. El sistema no hace nada.

Nombre: borrado de lugares de interés en días
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema, debe haber accedido al mismo y debe tener al menos una ruta creada que contenga días que incluyan lugares de interés.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario podrá eliminar un lugar de interés de la lista de lugares de interés asociada a un día
Escenario principal (camino feliz):

1. El usuario selecciona una ruta.
2. El sistema muestra la página de la ruta.
3. El usuario selecciona un día.
4. El sistema muestra la página del día.
5. El usuario selecciona “Eliminar lugar de interés”.

6. El sistema pide confirmación.
7. El usuario confirma la acción.
8. El sistema quita el lugar de interés de la lista de lugares de interés asociada al día
9. El sistema muestra la página del día sin el lugar de interés.

Extensiones:

7.a. El usuario cancela la acción.

7.a.1. El sistema cancela la acción.

7.a.2. El sistema muestra la página del día con el lugar de interés.

Nombre: recuperación de la contraseña
Actor primario: usuario
Precondición: el usuario debe estar registrado en el sistema.
Garantía mínima: en caso de error, el sistema notificará al usuario.
Garantía de éxito: el usuario recibirá un *e-mail* con la nueva contraseña.
Escenario principal (camino feliz):

1. El usuario selecciona la opción “Recuperar contraseña”.
2. El sistema muestra una página con un formulario con los campos necesarios para recuperar la contraseña.
3. El usuario cumplimenta el formulario.
4. El usuario selecciona “Enviar”.
5. El sistema valida los datos.
6. El sistema envía un *e-mail* al usuario con su nueva contraseña.
7. El sistema muestra la página de inicio de la aplicación.

Extensiones:

3.a. El usuario no ha completado los campos necesarios.

3.a.1. El sistema reenvía al usuario el formulario indicando los datos que deben ser completados.

3.a.2. El usuario completa los campos obligatorios.

3.a.3. Sigue por el paso 4 del escenario principal.

3.b. El usuario introduce un *e-mail* que no está vinculado a ningún usuario del sistema.

3.b.1. El sistema reenvía al usuario el formulario indicando que no existe ningún usuario con ese *e-mail*.

3.b.2. El usuario corrige el error.

3.b.3. Sigue por el paso 4 del escenario principal.

APÉNDICE B

Casos de prueba

Nombre: Registrar una cuenta de usuario con éxito.
Objetivo: Este caso de prueba verifica el comportamiento del sistema en el caso de uso *Registro* en su escenario principal.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema crea el nuevo usuario.

04 El sistema muestra la página de inicio, en la que habrá un enlace con el texto "¿Todavía no has creado ninguna ruta? Crea tu primera ruta aquí".

Valores de prueba:

usuarioForm.alias = 'prueba1'

usuarioForm.password = '12345678'

usuarioForm.confirmacionPassword = '12345678'

usuarioForm.email = 'prueba@socialroute.com'

usuarioForm.confirmacionEmail = 'prueba@socialroute.com'

Resultados observables: En la página de administración se puede comprobar que

se ha insertado el usuario en la base de datos.

Nombre: Registro de una cuenta de usuario ya registrada.
Objetivo: Este caso de prueba verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.b.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que ya existe un usuario con el alias introducido.

04 El sistema muestra por pantalla un mensaje con el siguiente texto: “Ya existe un usuario con ese nombre de usuario”.

Valores de prueba:

usuarioForm.alias = ’prueba1’

usuarioForm.password = ’12345678’

usuarioForm.confirmacionPassword = ’12345678’

usuarioForm.email = ’prueba@socialroute.com’

usuarioForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página de administracion se puede comprobar que no se ha insertado el usuario en la base de datos.
Notas:

- Es necesario que se haya creado previamente un usuario cuyo alias sea “prueba1”.
- Se considera que una cuenta de usuario ha sido ya registrada cuando ya existe un usuario con el alias indicado.

Nombre: Registro de una cuenta de cuyas contraseñas no coinciden.
Objetivo: Este caso de prueba verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.c.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva

cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que los campos relativos a la contraseña no coinciden.

04 El sistema muestra por pantalla un mensaje con el siguiente texto: “Las contraseñas no coinciden”.

Valores de prueba:

usuarioForm.alias = ’prueba’

usuarioForm.password = ’12345678’

usuarioForm.confirmacionPassword = ’87654321’

usuarioForm.email = ’prueba@socialroute.com’

usuarioForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página de administración se puede comprobar que no se ha insertado el usuario en la base de datos.

Nombre: Registro de una cuenta cuyos *e-mails* no coinciden.
Objetivo: Este caso de prueba verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.d.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que los campos relativos al *e-mail* no coinciden

04 El sistema muestra por pantalla un mensaje con el siguiente texto: “Los emails no coinciden”.

Valores de prueba:

usuarioForm.alias = ’prueba’

usuarioForm.password = ’12345678’

usuarioForm.confirmacionPassword = ’12345678’

usuarioForm.email = 'prueba@socialroute.com'

usuarioForm.confirmacionEmail = 'prueba@socialroute.es'

Resultados observables: En la página de administración se puede comprobar que no se ha insertado el nuevo usuario en la base de datos.

Nombre: Registro de una cuenta sin completar los campos obligatorios
Objetivo: Este caso de prueba verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.a.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que hay campos incompletos.

04 El sistema muestra de nuevo el formulario con los campos obligatorios marcados en rojo.

Valores de prueba:

usuarioForm.alias = ""

usuarioForm.password = ""

usuarioForm.confirmacionPassword = ""

usuarioForm.email = ''

usuarioForm.confirmacionEmail = ""

Resultados observables: En la página de administración se puede comprobar que no se ha insertado el nuevo usuario en la base de datos.

Nombre: Registro de una cuenta de usuario con contraseña de menos de 8 caracteres.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.e.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que la longitud de la contraseña es menor de 8 caracteres.

04 El sistema muestra por pantalla un mensaje con el siguiente texto: “Este campo debe tener al menos 8 caracteres”

Valores de prueba:

usuarioForm.alias = ’prueba’

usuarioForm.password = ’123’

usuarioForm.confirmacionPassword = ’123’

usuarioForm.email = ’prueba@socialroute.com’

usuarioForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página de administración se puede comprobar que no se ha insertado el nuevo usuario en la base de datos.

Nombre: Registro de un usuario introduciendo un *e-mail* sin arroba.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.g.
Acciones:

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que el *e-mail* no contiene el caracter ’@’.

04 El sistema muestra por pantalla un mensaje con el siguiente texto: “Introduzca un *e-mail* válido”

Valores de prueba:

usuarioForm.alias = ’prueba’

usuarioForm.password = ’12345678’

usuarioForm.confirmacionPassword = ’12345678’

usuarioForm.email = ’pruebasocialroute.com’

usuarioForm.confirmacionEmail = 'prueba@socialroute.com'

Resultados observables:

Nombre: Registro de un usuario introduciendo un alias que no es alfanumérico.
Objetivo: En este caso de prueba el verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.g.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que el campo relativo al alias contiene caracteres que no son alfanuméricos.

04 El sistema reenvía el formulario y muestra por pantalla el siguiente mensaje: "Solo se permiten caracteres alfanuméricos".

Valores de prueba:

usuarioForm.alias = 'prueba–'

usuarioForm.password = '12345678'

usuarioForm.confirmacionPassword = '12345678'

usuarioForm.email = 'prueba@socialroute.com'

usuarioForm.confirmacionEmail = 'prueba@socialroute.com'

Resultados observables: En la página de administración se puede comprobar que no se ha insertado el nuevo usuario en la base de datos.

Nombre: Registro de un usuario introduciendo un *e-mail* que ya está siendo utilizado.
Objetivo: En este caso de prueba el verifica el comportamiento del sistema en el caso de uso *Registro* cuando se produce la extensión 3.h.
Acciones: 00 -> 01 (usuarioForm) -> 02 -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de crear una nueva cuenta.

01 El sistema muestra el formulario para el registro de usuarios.

02 La prueba introduce los datos relativos a la nueva cuenta de usuario y los envía.

03 El sistema detecta que el campo relativo al alias contiene caracteres que no son alfanuméricos.

04 El sistema reenvía el formulario y muestra por pantalla el siguiente mensaje: “El *e-mail* que ha introducido ya está en uso”.

Valores de prueba:

usuarioForm.alias = ’prueba2’

usuarioForm.password = ’1234567890’

usuarioForm.confirmacionPassword = ’1234567890’

usuarioForm.email = ’prueba@socialroute.com’

usuarioForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página de administración se puede comprobar que no se ha insertado el nuevo usuario en la base de datos.

Nombre: Acceso correcto al sistema.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Acceso* en su escenario principal.
Acciones: 00 (registroForm) -> 01 -> 02

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación, cuyo título es “Tus rutas”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: Como resultado, se puede comprobar que se accede correctamente a la aplicación.

Nombre: Acceso al sistema introduciendo datos erróneos.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Acceso* cuando se produce la extensión 1.a.
Acciones: 00 (registroForm) -> 01 -> 02

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema detecta que los datos introducidos son incorrectos.

02 El sistema reenvía el formulario y muestra un mensaje con el siguiente texto: “Usuario o contraseña no válido(s)”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’123123132’

Resultados observables: Como resultado, se puede comprobar que no se puede acceder al sistema.

Nombre: Creación de una ruta correctamente.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de rutas* en su escenario principal.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04(rutaForm) -> 05 -> 06 -> 07

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona la opción de crear una nueva ruta.

04 El sistema muestra una página con un formulario para crear una nueva ruta.

05 La prueba introduce los datos de la ruta y los envía.

06 El sistema crea una nueva ruta.

07 El sistema muestra la página de inicio en la que habrá un enlace a la ruta cuyo nombre sea el título de la misma.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

rutaForm.titulo = ’Ruta de prueba’

rutaForm.descripcion = ’Descripción’

Resultados observables: En la página de administración de puede comprobar que se ha insertado una nueva ruta.

Nombre: Creación de una ruta con datos incompletos.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de rutas* cuando se produce la extensión 3.a.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04(rutaForm) -> 05 -> 06 -> 07

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona la opción de crear una nueva ruta.

04 El sistema muestra una página con un formulario para crear una nueva ruta.

05 La prueba introduce los datos de la ruta y los envía.

06 El sistema detecta que hay campos obligatorios incompletos.

07 El sistema reenvía el formulario marcando en rojo los campos incompletos que necesitan ser completados.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

rutaForm.titulo = ''

rutaForm.descripcion = ''

Resultados observables: En la página de administración de puede comprobar que se ha insertado una nueva ruta.

Nombre: Cancelación de la creación de una ruta.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de rutas* cuando se produce la extensión *.a.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona la opción de crear una nueva ruta.

04 El sistema muestra una página con un formulario para crear una nueva ruta.

05 La prueba pulsa el enlace “Volver”.

06 El sistema muestra la página de inicio.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: En la página de administración de puede comprobar que se ha insertado una nueva ruta.

Los casos de prueba relativos a la edición de rutas serán muy similares a los casos de prueba relativos a la creación de rutas, por lo que no van a ser especificados.

Nombre: Borrado de una ruta correctamente.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de rutas* en su escenario principal.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona “Borrar”.

06 El sistema muestra un mensaje de alerta pidiendo confirmación.

07 La prueba confirma la acción.

08 El sistema borra la ruta.

09 El sistema muestra la página de inicio en la que no aparecerá el enlace a la ruta.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En la página de administración de puede comprobar que se ha eliminado la ruta.

Nombre: Cancelación del borrado de una ruta.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de rutas* cuando se produce la extensión 5.a.
Acciones:00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona "Borrar".

06 El sistema muestra un mensaje de alerta pidiendo confirmación.

07 La prueba cancela la acción.

08 El sistema muestra la página de la ruta.

Resultados observables: En la página de administración se puede comprobar que no se ha eliminado la ruta.

Nombre: Creación de un día correctamente.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de días* en su escenario principal.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 (diaForm) -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla prin-

cipal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona “Añadir día”.

06 El sistema muestra una página con un formulario para crear un nuevo día.

07 La prueba introduce los datos relativos al nuevo día y los envía.

08 El sistema crea un nuevo día.

09 El sistema muestra el nuevo día en la página de la ruta asociada.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

diaForm.titulo = ’Día de prueba’

diaForm.titulo = ’Descripción’

Resultados observables: En la página de administración se puede comprobar que se ha insertado un nuevo día en la base de datos.

Nombre: Creación de un día con datos incompletos.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de días* cuando se produce la extensión 7.a.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 (diaForm) -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona “Añadir día”.

06 El sistema muestra una página con un formulario para crear un nuevo día.

07 La prueba introduce los datos relativos al nuevo día y los envía.

08 El sistema detecta que hay campos obligatorios vacíos.

09 El sistema reenvía el formulario marcando en rojo los campos incompletos que necesitan ser completados.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

diaForm.titulo = ''

diaForm.titulo = ''

Resultados observables: En la página de administración se puede comprobar que no se ha insertado un nuevo día en la base de datos.

Nombre: Cancelación de la creación de un día.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de días* cuando se produce la extensión *.a.
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona "Añadir día".

06 El sistema muestra una página con un formulario para crear un nuevo día.

07 La prueba selecciona "Volver".

08 El sistema muestra la página anterior, que en este caso será la página de ruta.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En la página de administración se puede comprobar que

no se ha insertado un nuevo día en la base de datos.

Los casos de prueba relativos a la edición de días serán muy similares a los casos de prueba relativos a la creación de días, por lo que no van a ser especificados.

Nombre: Borrado de un día correctamente.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de días* en su escenario principal
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10 -> 11

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la página principal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona uno de los días de entre los que se muestran en la página de la ruta.

06 El sistema muestra la página de información del día.

07 La prueba selecciona “Borrar”.

08 El sistema pide confirmación.

09 La prueba confirma la acción.

10 El sistema borra la ruta.

11 El sistema muestra la página de la ruta en la que no aparecerá el día.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: En la página de administración se puede comprobar que se ha eliminado el día.

Nombre: Cancelación del borrado de un día.

Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de días* en su escenario principal
Acciones: 00 (registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10 -> 11

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la página principal.

04 El sistema muestra la página de información de la ruta.

05 La prueba selecciona uno de los días de entre los que se muestran en la página de la ruta.

06 El sistema muestra la página de información del día.

07 La prueba selecciona “Borrar”.

08 El sistema pide confirmación.

09 La prueba cancela la acción.

10 El sistema muestra la página de la ruta en la que aparecerá el día.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: En la página de administración se puede comprobar que no se ha eliminado el día.

Nombre: Salida del sistema.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de prueba *Salida del sistema* en sus escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Cerrar sesión”.

04 El sistema vacía la variable de sesión que almacenaba los datos del usuario.

05 El sistema muestra la página de inicio de la aplicación.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: Se puede comprobar que ya no hay una sesión iniciada en el sistema intentando acceder a cualquier página de la aplicación y comprobar que esta redirecciona a la página de inicio.

Nombre: Búsqueda de rutas exitosa.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de rutas* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

busquedaForm.valor = ’Sevilla’

Resultados observables: En la página en la que se muestran los resultados de la búsqueda se puede observar que hay al menos una ruta cuyo título, descripción o nombre de creador contengan la palabra ’Sevilla’.

Nombre: Búsqueda de rutas con parámetro de búsqueda vacío.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso

de uso *Búsqueda de rutas* cuando se produce la extensión 1.a.
Acciones:00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba envía el formulario de búsqueda vacío.

04 El sistema permanece en la página principal de la aplicación.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

busquedaForm.valor = ''

Resultados observables: El sistema no realiza ninguna búsqueda si los parámetros de la misma están vacíos.

Nombre: Búsqueda de rutas sin coincidencias.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de rutas* cuando se produce la extensión 1.b.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en la página de búsqueda un mensaje indicando que no se han encontrado coincidencias.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

busquedaForm.valor = 'asdfa'

Resultados observables: En la página de búsqueda no se muestra ninguna ruta.

Nombre: Seguimiento de rutas correcto.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Seguimiento de rutas* en su escenario principal
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

05 La prueba selecciona una ruta.

06 El sistema muestra la página de la ruta.

07 La prueba selecciona la opción “Seguir ruta”.

08 El sistema añade la ruta seleccionada a la lista de rutas seguidas por el usuario.

09 El sistema muestra la página de la ruta.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

busquedaForm.valor = ’a’

Resultados observables: En el perfil del usuario se puede comprobar que la ruta se ha añadido a la lista de rutas seguidas.

Nombre: Seguimiento de una ruta anteriormente seguida.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Seguimiento de rutas* cuando se produce la extensión 3.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para seguir una ruta de una ruta que ya sigue.

04 El sistema redirecciona a la página de la ruta.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En el perfil del usuario se puede comprobar que la ruta no se ha añadido a la lista de rutas seguidas.
Notas: Este caso de prueba únicamente se puede dar si se accede al seguimiento de la ruta por medio de una URL.

Nombre: Seguimiento de una ruta creada por el usuario.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso e uso *Seguimiento de rutas* cuando se produce la extensión 3.b.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para seguir una ruta de una ruta creada por él.

04 El sistema redirecciona a la página de la ruta.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En el perfil del usuario se puede comprobar que la ruta no se ha añadido a la lista de rutas seguidas.
Notas: Este caso de prueba únicamente se puede dar si se accede al seguimiento de la ruta por medio de una URL.

Nombre: Dejar de seguir una ruta correctamente.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso

de uso *Cancelación del seguimiento de rutas* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

05 La prueba selecciona una ruta.

06 El sistema muestra la página de la ruta.

07 La prueba selecciona la opción “Dejar de seguir ruta”.

08 El sistema elimina la ruta seleccionada de la lista de rutas seguidas por el usuario.

09 El sistema muestra la página de la ruta.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

busquedaForm.valor = ’a’

Resultados observables: En el perfil de usuario se puede comprobar que se ha eliminado la ruta de la lista de rutas seguidas.

Nombre: Dejar de seguir una ruta que no está en la lista de seguidas.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso e uso *Cancelación del seguimiento de rutas* cuando se produce la extensión 3.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para seguir una ruta de una ruta que no ha seguido

previamente.

04 El sistema redirecciona a la página de la ruta.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En el perfil del usuario se puede comprobar que la ruta no estaba en la lista de rutas seguidas antes de hacer los tests y tampoco está después de ejecutarlos.
Notas: Este caso de prueba únicamente se puede dar si se accede a la cancelación seguimiento de la ruta por medio de una URL.

Nombre: Dejar de seguir una ruta creada por el usuario.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso e uso *Cancelación del seguimiento de rutas* cuando se produce la extensión 3.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para seguir una ruta de una ruta creada por el usuario que tiene iniciada sesión en el sistema.

04 El sistema redirecciona a la página de la ruta.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En el perfil del usuario se puede comprobar que la ruta no estaba en la lista de rutas seguidas antes de hacer los tests y tampoco está después de ejecutarlos.
Notas: Este caso de prueba únicamente se puede dar si se accede a la cancelación del seguimiento de la ruta por medio de una URL.

Nombre: Búsqueda de usuarios exitosa.

Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de usuarios* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

busquedaForm.valor = 'sbalmes'

Resultados observables: En la página en la que se muestran los resultados de la búsqueda se puede observar que hay al menos un usuario cuyo alias contiene la la cadena 'sbalmes'.

Nombre: Búsqueda de usuarios con parámetro de búsqueda vacío.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de usuarios* cuando se produce la extensión 1.a.
Acciones:00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba envía el formulario de búsqueda vacío.

04 El sistema permanece en la página principal de la aplicación.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

busquedaForm.valor = ''

Resultados observables: El sistema no realiza ninguna búsqueda si los parámetros de la misma están vacíos.

Nombre: Búsqueda de usuarios sin coincidencias.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de usuarios* cuando se produce la extensión 1.b.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en la página de búsqueda un mensaje indicando que no se han encontrado coincidencias.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

busquedaForm.valor = 'asdfa'

Resultados observables: En la página de búsqueda no se muestra ningún usuario.

Nombre: Seguimiento de usuarios correcto.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Seguimiento de usuarios* en su escenario principal
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

05 La prueba selecciona un usuario.

06 El sistema muestra la página del usuario.

07 La prueba selecciona la opción “Seguir usuario”.

08 El sistema añade al usuario seleccionada a la lista de usuarios seguidos por el usuario.

09 El sistema muestra la página del usuario.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

busquedaForm.valor = ’a’

Resultados observables: En el perfil del usuario se puede comprobar que el usuario seguido se ha añadido a la lista de usuarios seguidos.

Nombre: Seguimiento de un usuario anteriormente seguido.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Seguimiento de usuarios* cuando se produce la extensión 3.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para seguir un usuario de un usuario que ya sigue.

04 El sistema redirecciona a la página del usuario.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: En el perfil del usuario se puede comprobar que el usuario no se ha añadido a la lista de usuarios seguidos.
Notas: Este caso de prueba únicamente se puede dar si se accede al seguimiento del usuario por medio de una URL.

Nombre: Seguimiento del propio usuario.

Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso e uso *Seguimiento de usuarios* cuando se produce la extensión 3.b.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para seguirse a sí mismo.

04 El sistema redirecciona a la página del usuario.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En el perfil del usuario se puede comprobar que no se ha añadido a sí mismo a la lista de usuarios seguidos.
Notas: Este caso de prueba únicamente se puede dar si se accede al seguimiento de la ruta por medio de una URL.

Nombre: Dejar de seguir un usuario correctamente.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Cancelación del seguimiento de usuarios* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

05 La prueba selecciona un usuario.

06 El sistema muestra la página del usuario.

07 La prueba selecciona la opción “Dejar de seguir usuario”.

08 El sistema elimina al usuario seleccionado de la lista de usuarios seguidos por el usuario.

09 El sistema muestra la página del usuario.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

busquedaForm.valor = 'a'

Resultados observables: En el perfil de usuario se puede comprobar que se ha eliminado al usuario de la lista de usuarios seguidos.

Nombre: Dejar de seguir a un usuario que no está en la lista de seguidos.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso e uso *Cancelación del seguimiento de usuarios* cuando se produce la extensión 3.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba carga la URL para dejar de seguir a un usuario de un usuario al que no ha seguido previamente.

04 El sistema redirecciona a la página del usuario.

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

Resultados observables: En el perfil del usuario se puede comprobar que el otro usuario no estaba en la lista de usuarios seguidos antes de hacer los tests y tampoco está después de ejecutarlos.
Notas: Este caso de prueba únicamente se puede dar si se accede a la cancelación del seguimiento del usuario por medio de una URL.

Nombre: Modificación del perfil con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso

de uso *Modificación del perfil* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06 -> 07

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba introduce los datos relativos al perfil de usuario que desee modificar y los envía.

06 El sistema valida los datos y los guarda.

07 El muestra la página del perfil de usuario.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba11’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’123456789’

editarPerfilForm.confirmacionPassword = ’123456789’

editarPerfilForm.email = ’prueba11@socialroute.com’

editarPerfilForm.confirmacionEmail = ’prueba11@socialroute.com’

Resultados observables: En la página del perfil del usuario se puede comprobar que se han modificado el nombre de usuario y el *e-mail*.

Nombre: Cancelación de la modificación del perfil.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión *.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba selecciona “Volver”.

06 El sistema muestra la página anterior, que en este caso será la página de inicio.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: en la página del perfil de usuario se puede comprobar que no ha habido ninguna modificación en los datos del mismo.

Nombre: Modificación del perfil sin completar los campos obligatorios.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba envía el formulario de modificación del perfil vacío.

06 El sistema reenvía el formulario indicando los campos que deben ser completados.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ”

editarPerfilForm.password = ”

editarPerfilForm.nuevaPassword = ”

editarPerfilForm.confirmacionPassword = ”

editarPerfilForm.email = ”

editarPerfilForm.confirmacionEmail = ”

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil introduciendo un alias que ya existe.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.b.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “Ya existe un usuario con ese alias, introduzca otro”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’rarrebola’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’12345678’

editarPerfilForm.confirmacionPassword = ’12345678’

editarPerfilForm.email = ’prueba@socialroute.com’

editarPerfilForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página del perfil del usuario se puede comprobar que

no se ha producido ningún cambio.

Nombre: Modificación del perfil introduciendo una contraseña actual no válida.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.c.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “La contraseña introducida no es válida”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba1’

editarPerfilForm.password = ’asdfasdfasdfadfad’

editarPerfilForm.nuevaPassword = ’12345678’

editarPerfilForm.confirmacionPassword = ’12345678’

editarPerfilForm.email = ’prueba@socialroute.com’

editarPerfilForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil cuyas nuevas contraseñas no coinciden.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.d.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “Las contraseñas no coinciden”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba1’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’1111111111’

editarPerfilForm.confirmacionPassword = ’4444444444’

editarPerfilForm.email = ’prueba@socialroute.com’

editarPerfilForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil cuyas nuevos *e-mails* no coinciden.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.e.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “Los emails no coinciden”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba1’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’12345678’

editarPerfilForm.confirmacionPassword = ’12345678’

editarPerfilForm.email = ’prueba@socialroute.es’

editarPerfilForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil cuando el *e-mail* indicado ya está siendo utilizado.

Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.f.

Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “El email que desea usar ya está siendo usado, introduzca otro”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba1’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’12345678’

editarPerfilForm.confirmacionPassword = ’12345678’

editarPerfilForm.email = ’rocarrpas@uma.es’

editarPerfilForm.confirmacionEmail = ’rocarrpas@uma.es’

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil introduciendo una contraseña demasiado corta.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.g.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “Este campo debe tener al menos 8 caracteres”.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba1’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’1’

editarPerfilForm.confirmacionPassword = ’1’

editarPerfilForm.email = 'prueba@socialroute.com'

editarPerfilForm.confirmacionEmail = 'prueba@socialroute.com'

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil introduciendo un *e-mail* sin arroba.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.h.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona "Editar perfil".

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: "Introduzca un e-mail válido".

Valores de prueba:

registroForm.alias = 'prueba1'

registroForm.password = '12345678'

editarPerfilForm.alias = 'prueba1'

editarPerfilForm.password = '12345678'

editarPerfilForm.nuevaPassword = '12345678'

editarPerfilForm.confirmacionPassword = '12345678'

editarPerfilForm.email = 'pruebasocialroute.com'

editarPerfilForm.confirmacionEmail = 'pruebasocialroute.com'

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación del perfil introduciendo un caracter no alfanumérico en el alias.

Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación del perfil* cuando se produce la extensión 5.i.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(editarPerfilForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Editar perfil”.

04 El sistema muestra la página de edición del perfil.

05 La prueba completa todos los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje: “Solo se permiten caracteres alfanuméricos”.

Valores de prueba:

registroForm.alias = ’prueba1?’

registroForm.password = ’12345678’

editarPerfilForm.alias = ’prueba1’

editarPerfilForm.password = ’12345678’

editarPerfilForm.nuevaPassword = ’12345678’

editarPerfilForm.confirmacionPassword = ’12345678’

editarPerfilForm.email = ’prueba@socialroute.com’

editarPerfilForm.confirmacionEmail = ’prueba@socialroute.com’

Resultados observables: En la página del perfil del usuario se puede comprobar que no se ha producido ningún cambio.

Nombre: Borrado del perfil con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado del perfil* en su escenario principal
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08 -> 09

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Mi perfil”.

04 El sistema muestra la página del perfil de usuario.

05 La prueba selecciona “Eliminar perfil”.

06 El sistema pide confirmación.

07 La prueba confirma la acción.

08 El sistema elimina la cuenta de usuario.

09 El sistema carga la página de inicio de la aplicación.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: Se puede comprobar que no es posible iniciar sesión con el usuario cuya cuenta ha sido eliminada.

Nombre: Cancelación del borrado de perfil.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado del perfil* cuando se produce la extensión 5.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Mi perfil”.

04 El sistema muestra la página del perfil de usuario.

05 La prueba selecciona “Eliminar perfil”.

06 El sistema pide confirmación.

07 La prueba cancela la acción.

08 El sistema muestra la página del perfil de usuario.

Valores de prueba:

registroForm.alias = ’prueba1’

registroForm.password = ’12345678’

Resultados observables: Se puede comprobar que se puede iniciar sesión sin problema, pues la cuenta de usuario no ha sido eliminada.

Nombre: Creación de lugares de interés con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de lugares de interés* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(crearLugarInteresForm) -> 06 -> 07

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Crear lugar de interés”.

04 El sistema carga una página con el formulario de creación de lugares de interés.

05 La prueba completa los campos del formulario y los envía.

06 El sistema valida los campos y crea un nuevo lugar de interés.

07 El sistema carga la página principal de la aplicación.

Valores de prueba:

registroForm.alias = ’rarrebola’

registroForm.password = ’incendiosdenieve’

crearLugarInteresForm.nombre = ’Castillo de Gibralfaro’

crearLugarInteresForm.direccion = ’Castillo de Gibralfaro’

crearLugarInteresForm.localidad = ’Málaga’

crearLugarInteresForm.descripcion = ’El castillo de Gibralfaro o alcázar de Gibralfaro es una fortificación situada en la ciudad española de Málaga.’

crearLugarInteresForm.horario = ’Todos los días de 9:00 a 20:00’

crearLugarInteresForm.precio = ’5.00’

Resultados observables: Se puede observar que existe una página para el lugar de interés creado.

Nombre: Cancelación de la creación de un lugar de interés.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de lugares de interés* cuando se produce la extensión *.a.
Acciones:

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Crear lugar de interés”.

04 El sistema carga una página con el formulario de creación de lugares de interés.

05 La prueba selecciona “Volver”.

06 El sistema carga la página anterior, que en este caso será la página principal de la aplicación.

Valores de prueba:

registroForm.alias = ’rarrebola’

registroForm.password = ’incendiosdenieve’

Resultados observables: Se puede comprobar que no se ha creado ningún lugar de interés con nombre vacío.

Nombre: Creación de lugares de interés sin completar los campos obligatorios.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de lugares de interés* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(crearLugarInteresForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona “Crear lugar de interés”.

04 El sistema carga una página con el formulario de creación de lugares de interés.

05 La envía el formulario vacío.

06 El sistema reenvía el formulario indicando los campos que son obligatorios.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

crearLugarInteresForm.nombre = ''

crearLugarInteresForm.direccion = ''

crearLugarInteresForm.localidad = ''

crearLugarInteresForm.descripcion = ''

crearLugarInteresForm.horario = ''

crearLugarInteresForm.precio = ''

Resultados observables: Se puede comprobar que no se ha creado ningún lugar de interés con el nombre vacío.

Nombre: Creación de lugares de interés ya existentes.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Creación de lugares de interés* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05(crearLugarInteresForm) -> 06

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona "Crear lugar de interés".

04 El sistema carga una página con el formulario de creación de lugares de interés.

05 La prueba completa los campos del formulario y los envía.

06 El sistema reenvía el formulario con el mensaje "El lugar de interés ya existe".

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

crearLugarInteresForm.nombre = 'Castillo de Gibralfaro'

crearLugarInteresForm.direccion = 'Castillo de Gibralfaro'

crearLugarInteresForm.localidad = 'Málaga'

crearLugarInteresForm.descripcion = 'El castillo de Gibralfaro o alcázar de Gibralfaro es una fortificación situada en la ciudad española de Málaga.'

crearLugarInteresForm.horario = 'Todos los días de 9:00 a 20:00'

crearLugarInteresForm.precio = '5.00'

Resultados observables: Se puede comprobar que no se ha creado el lugar de interés por duplicado y que tampoco se han modificado sus campos.

Nombre: Modificación de lugares de interés con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación de lugares de interés* en su escenario principal.
Acciones: 00(registroForm) -> 01 ->02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09(editarLugarInteresForm) -> 10 -> 11

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona un lugar de interés.

06 El sistema muestra la página del lugar de interés.

07 La prueba selecciona la opción “Editar”.

08 El sistema carga una página con el formulario de modificación de lugares de interés.

09 La prueba completa los campos del formulario y los envía.

10 El sistema valida los cambios y los guarda.

11 El sistema carga la página principal de la aplicación.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'Castillo de Gibralfaro'

editarLugarInteresForm.descripcion = 'Descripción'

editarLugarInteresForm.horario = 'Horario'

editarLugarInteresForm.precio = '2'

Resultados observables: En la página del lugar de interés se puede observar que los datos relativos a la descripción, horario y precio.

Nombre: Cancelación de la modificación de lugares de interés
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación de lugares de interés* cuando se produce la extensión *.a.
Acciones: 00(registroForm) -> 01 ->02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona un lugar de interés.

06 El sistema muestra la página del lugar de interés.

07 La prueba selecciona la opción “Editar”.

08 El sistema carga una página con el formulario de modificación de lugares de interés.

09 La prueba selecciona “Volver”.

10 El sistema carga la página anterior.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'Castillo de Gibralfaro'

Resultados observables: En la página del lugar de interés se puede comprobar que no se ha producido ningún cambio.

Nombre: Modificación de lugares de interés sin completar campos obligatorios.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Modificación de lugares de interés* en su escenario principal.
Acciones: 00(registroForm) -> 01 ->02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09(editarLugarInteresForm) -> 10

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona un lugar de interés.

06 El sistema muestra la página del lugar de interés.

07 La prueba selecciona la opción “Editar”.

08 El sistema carga una página con el formulario de modificación de lugares de interés.

09 La prueba envía el formulario vacío.

10 El sistema reenvía el formulario indicando los campos que son obligatorios..

Valores de prueba:

registroForm.alias = ’rarrebola’

registroForm.password = ’incendiosdenieve’

busquedaForm.valor = ’Castillo de Gibralfaro’

editarLugarInteresForm.descripcion = ”

editarLugarInteresForm.horario = ”

editarLugarInteresForm.precio = ”

Resultados observables: En la página del lugar de interés se puede comprobar que no se ha producido ningún cambio.

Nombre: Valoración de rutas con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Valoración de rutas* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07(valoracionForm) -> 08

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona una ruta.

06 El sistema muestra la página de información de la ruta.

07 La prueba completa el formulario de valoración y lo envía.

08 El sistema muestra la página de información de la ruta con el nuevo comentario.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'Flandes'

valoracionForm.puntuacion = '9'

valoracionForm.comentario = 'Comentario de la ruta'

Resultados observables: En la página de información de la ruta se puede observar el nuevo comentario y se puede comprobar que la puntuación de la misma ha cambiado.

Nombre: Valoración de rutas sin completar los campos obligatorios.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema *Valoración de rutas* cuando se produce la extensión 3.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07(valoracionForm) -> 08

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona una ruta.

06 El sistema muestra la página de información de la ruta.

07 La prueba completa el formulario de valoración y lo envía.

08 El sistema muestra reenvía el formulario indicando que se deben completar los campos obligatorios.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'Flandes'

valoracionForm.puntuacion = ''

valoracionForm.comentario = ''

Resultados observables: En la página de información de la ruta se puede ver que no se ha añadido ningún comentario y que la valoración de la misma no ha cambiado.

Nombre: Borrado de valoraciones de rutas correcto.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de valoraciones de rutas* en su escenario principal
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona una ruta.

06 El sistema muestra la página de información de la ruta.

07 La prueba selecciona la opción “Eliminar comentario”.

08 El sistema pide confirmación.

09 La prueba confirma la acción.

10 El sistema muestra la página de información de la ruta sin la valoración.

Valores de prueba:

registroForm.alias = ’rarrebola’

registroForm.password = ’incendiosdenieve’

busquedaForm.valor = ’Flandes’

Resultados observables: En la página de información de la ruta se puede observar que se ha eliminado el comentario y que la puntuación de la ruta ha cambiado.

Nombre: Cancelación del borrado de valoraciones de rutas.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de valoraciones de rutas* en su escenario principal
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce un parámetro de búsqueda y lo envía.

04 El sistema muestra un listado de coincidencias.

05 La prueba selecciona una ruta.

06 El sistema muestra la página de información de la ruta.

07 La prueba selecciona la opción “Eliminar comentario”.

08 El sistema pide confirmación.

09 La prueba cancela la acción.

10 El sistema muestra la página de información de la ruta con la valoración.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'Flandes'

Resultados observables: En la página de información de la ruta se puede observar el comentario y que la puntuación de la ruta no ha cambiado.

Los casos de prueba relativos a las valoraciones de lugares de interés (ya sea creación o borrado) serán muy similares a los casos de prueba relativos a las valoraciones de rutas (creación y borrado), por lo que no van a ser especificados.

Nombre: Búsqueda de lugares de interés exitosa.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de lugares de interés* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en una página el listado de coincidencias.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'Manneken Pis'

Resultados observables: En la página en la que se muestran los resultados de la búsqueda se puede observar que hay al menos un lugar de interés que contiene la cadena 'Manneken Pis'.

Nombre: Búsqueda de lugares de interés con parámetro de búsqueda vacío.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de lugares de interés* cuando se produce la extensión 1.a.
Acciones:00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba envía el formulario de búsqueda vacío.

04 El sistema permanece en la página principal de la aplicación.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = ''

Resultados observables: El sistema no realiza ninguna búsqueda si los parámetros de la misma están vacíos.

Nombre: Búsqueda de lugares de interés sin coincidencias.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Búsqueda de lugares de interés* cuando se produce la extensión 1.b.
Acciones: 00(registroForm) -> 01 -> 02 -> 03(busquedaForm) -> 04

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba introduce el parámetro de búsqueda y lo envía.

04 El sistema muestra en la página de búsqueda un mensaje indicando que no se han encontrado coincidencias.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

busquedaForm.valor = 'asdfa'

Resultados observables: En la página de búsqueda no se muestra ningún lugar de interés.

Nombre: Añadido de lugares de interés a días con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Añadido de lugares de interés a días* en su escenario principal.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07(buscarLugarInteresForm) -> 08 -> 09 -> 10 -> 11

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de la ruta.

05 La prueba selecciona un día.

06 El sistema muestra la página del día.

07 La prueba completa el formulario de búsqueda de lugares de interés y lo envía.

08 El sistema muestra un listado de lugares de interés.

09 La prueba selecciona uno de los lugares de interés y selecciona “Añadir”.

10 El sistema añade el lugar de interés al día.

11 El sistema muestra la página del día.

Valores de prueba:

registroForm.alias = ’rarrebola’

registroForm.password = ’incendiosdenieve’

buscarLugarInteresForm.localidad = ’Amberes’

Resultados observables: En la página del día se puede comprobar que se ha añadido el lugar de interés a la lista de lugares de interés. También se puede comprobar que el lugar de interés se ha añadido al mapa.

Nombre: Añadido de lugares de interés a días con la búsqueda vacía.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Añadido de lugares de interés a días* cuando se produce la extensión 5.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07(buscarLuga-

rInteresForm) -> 08

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de la ruta.

05 La prueba selecciona un día.

06 El sistema muestra la página del día.

07 La prueba envía el formulario de búsqueda de lugares de interés vacío.

08 El sistema no hace nada.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

buscarLugarInteresForm.localidad = ''

Resultados observables: No se ha añadido ningún lugar de interés al día.

Nombre: Borrado de lugares de interés en días con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de lugares de interés en días* en su escenario principal
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10 -> 11

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de la ruta.

05 La prueba selecciona un día.

06 El sistema muestra la página del día.

07 La prueba selecciona “Eliminar lugar de interés”.

08 El sistema pide confirmación.

09 La prueba confirma la acción.

10 El sistema elimina el lugar de interés del día.

11 El sistema muestra la página del día sin el lugar de interés.

Valores de prueba:

registroForm.alias = ’rarrebola’

registroForm.password = ’incendiosdenieve’

Resultados observables: En la página del día se puede observar que se ha eliminado un lugar de interés. En el mapa tampoco se mostrará el lugar de interés.

Nombre: Cancelación del borrado de lugares de interés en días.
Objetivo: En este caso de prueba se verifica el comportamiento del sistema en el caso de uso *Borrado de lugares de interés en días* en su extensión 7.a.
Acciones: 00(registroForm) -> 01 -> 02 -> 03 -> 04 -> 05 -> 06 -> 07 -> 08 -> 09 -> 10

00 La prueba carga la página de inicio, introduce los datos de una cuenta de usuario y los envía.

01 El sistema verifica los datos de la cuenta de usuario.

02 El sistema muestra la página principal de la aplicación.

03 La prueba selecciona una ruta de entre las que se muestran en la pantalla principal.

04 El sistema muestra la página de la ruta.

05 La prueba selecciona un día.

06 El sistema muestra la página del día.

07 La prueba selecciona “Eliminar lugar de interés”.

08 El sistema pide confirmación.

09 La prueba confirma la acción.

11 El sistema muestra la página del día con el lugar de interés.

Valores de prueba:

registroForm.alias = 'rarrebola'

registroForm.password = 'incendiosdenieve'

Resultados observables: En la página del día se puede observar que el lugar de interés no se ha eliminado y que permanece en el mapa.

Nombre: Recuperación de la contraseña con éxito.
Objetivo: En este caso de prueba se verifica el comportamiento el sistema en el caso de uso *Recuperación de la contraseña* en su escenario principal.
Acciones: 00 -> 01 -> 02(recuperarContraseñaForm) -> 03 -> 04

00 La prueba carga la página de inicio y selecciona la opción de recuperar la contraseña.

01 El sistema muestra el formulario para la recuperación de la contraseña.

02 La prueba introduce los datos para la recuperación de la contraseña y los envía.

03 El sistema genera una nueva contraseña para la cuenta indicada.

04 El sistema redirige al usuario a la página de inicio de la aplicación.

Valores de prueba:

recuperarContraseñaForm.email = 'rocarrpas@uma.es'

Resultados observables: En caso de intentar iniciar sesión con la antigua contraseña, el sistema lanzará un mensaje de error.

Nombre: Recuperación de la contraseña con los campos obligatorios vacíos.
Objetivo: En este caso de prueba se verifica el comportamiento el sistema en el caso de uso *Recuperación de la contraseña* cuando se produce la extensión 3.a.
Acciones: 00 -> 01 -> 02(recuperarContraseñaForm) -> 03

00 La prueba carga la página de inicio y selecciona la opción de recuperar la contraseña.

01 El sistema muestra el formulario para la recuperación de la contraseña.

02 La prueba envía el formulario de recuperación de la contraseña vacío.

03 El sistema reenvía el formulario indicando los campos que son obligatorios.

Valores de prueba:

recuperarContraseñaForm.email = “”

Resultados observables:

Nombre: Recuperación de la contraseña introduciendo un *e-mail* que no se ha registrado en el sistema.
Objetivo: En este caso de prueba se verifica el comportamiento el sistema en el caso de uso *Recuperación de la contraseña* cuando se produce la extensión 3.b.
Acciones: 00 -> 01 -> 02(recuperarContraseñaForm) -> 03

00 La prueba carga la página de inicio y selecciona la opción de recuperar la contraseña.

01 El sistema muestra el formulario para la recuperación de la contraseña.

02 La prueba introduce los datos para la recuperación de la contraseña y los envía.

03 El sistema reenvía el formulario con el mensaje “No existen usuarios con el *email* indicado”.

Valores de prueba:

recuperarContraseñaForm.email = ’rocioarrebola95@uma.es’

Resultados observables:

APÉNDICE C

Manual de usuario

C.1. Introducción

Social Route es una aplicación web ideada para que los usuarios puedan compartir sus experiencias a la hora de viajar, pudiendo las rutas turísticas que han hecho de modo que el resto de usuarios puedan suscribirse a las mismas y seguirlas.

El objetivo principal de *Social Route* es ahorrar tiempo a sus usuarios, que ya no tendrán que perder días planeando sus viajes.

C.2. Instalación

C.2.1. Requisitos del sistema

Al no estar alojada en la *web*, para poder ejecutar *Social Route* será necesario tener el directorio con el proyecto en el equipo que desee ejecutar la aplicación, además, se pide tener instalado:

- *Python* 3.4
- *Django* 1.10.6

C.2.2. Instalación

La aplicación no necesita una instalación como tal, sino que será suficiente con ejecutar el siguiente comando en la consola del equipo:

```
\socialroute > python manage.py runserver
```

Hay que tener en cuenta que este comando debe ejecutarse dentro del directorio que contiene el proyecto, es decir, el directorio *socialroute*.

Una vez ejecutado este comando bastará con abrir el navegador y acceder a la dirección `http://127.0.0.1:8000/`.

C.3. Funcionalidades

El usuario podrá realizar las siguientes acciones.

C.3.1. Crear cuenta

Para crear una cuenta, será necesario hacer *click* en el enlace *¿No tienes cuenta? Regístrate* C.1.

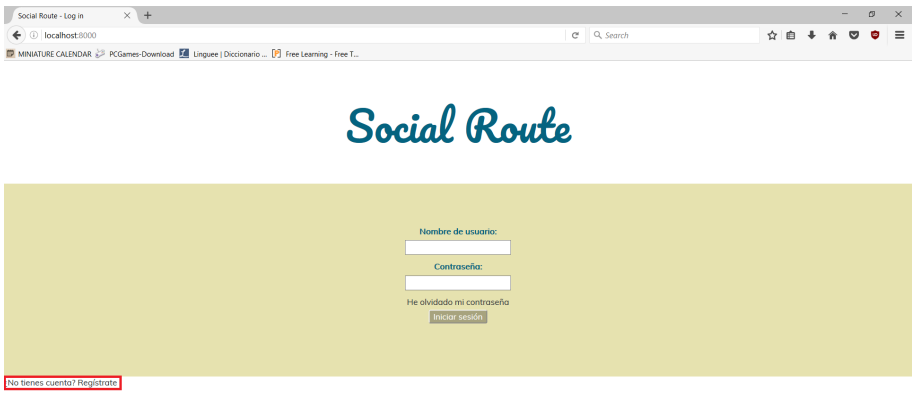

Figura C.1: Registro de usuario. Paso 1. Página de inicio.

Este enlace redirigirá a una página con el formulario de registro del usuario. Es necesario completar todos los campos del formulario. Una vez completados, es necesario pulsar *Registrarse* C.2.

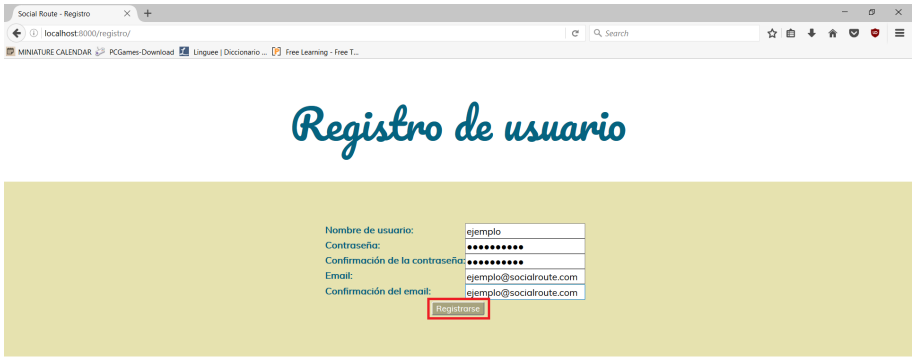

Figura C.2: Registro de usuario. Paso 2. Página de registro.

C.3.2. Recuperar la contraseña

Para recuperar la contraseña será necesario hacer *click* en el enlace *He olvidado mi contraseña*C.3.

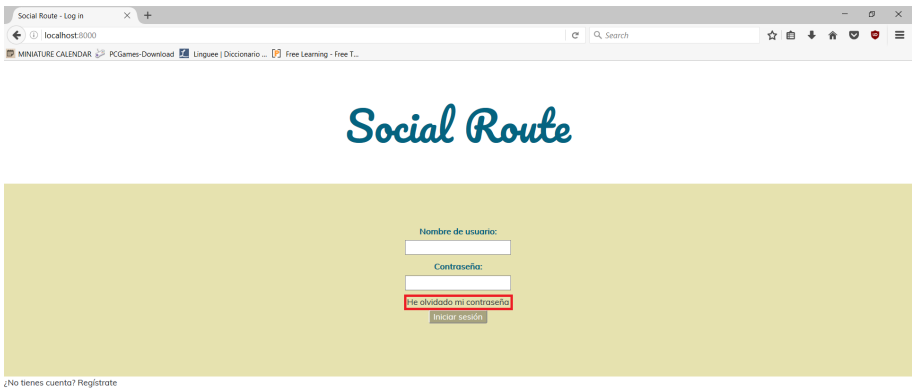

Figura C.3: Recuperar la contraseña. Paso 1. Página de inicio.

Este enlace redirigirá a la página de recuperación de la contraseña, en la que será necesario indicar el **e-mail** asociado a la cuenta de usuario y enviarlo. Esto enviará la nueva contraseña al *e-mail* indicado C.4.

Figura C.4: Recuperar la contraseña. Paso 2. Página de recuperación de la contraseña.

C.3.3. Iniciar sesión

Para el inicio de sesión bastará con completar el formulario de la página de inicio y seleccionar *Iniciar sesión* C.5.

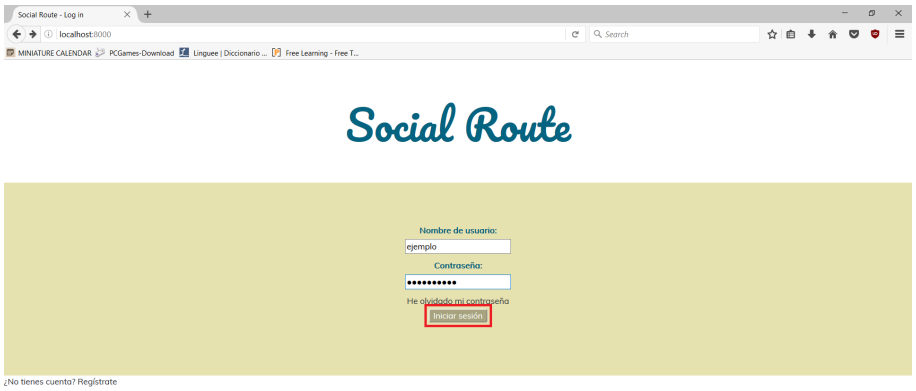

Figura C.5: Iniciar sesión. Paso 1. Página principal.

C.3.4. Cerrar sesión

Para cerrar sesión será suficiente con seleccionar *Cerrar sesión* en el *Dropdown* de la barra de navegación en cualquier página de la aplicación C.6.

Figura C.6: Cerrar sesión.

C.3.5. Crear una ruta

Existen dos maneras de crear una ruta:

1. Seleccionando *Crear una nueva ruta* en la página principal C.7.
2. Seleccionando *Crear ruta* en el *Dropdown* de la barra de navegación en cualquier página de la aplicación C.8.

Sea cual sea la opción elegida, se abrirá la página de creación de la ruta, en la que se mostrará un formulario que se completará según se estime conveniente y se pulsará *Guardar* C.9.

C.3.6. Editar una ruta

Para editar una ruta se seleccionará la opción *Editar* en la página de la ruta C.10.

Tras seleccionar *Editar*, se abrirá la página de edición de la ruta, que será similar a la página de creación de la misma. En ella se hacen los cambios pertinentes y, una vez

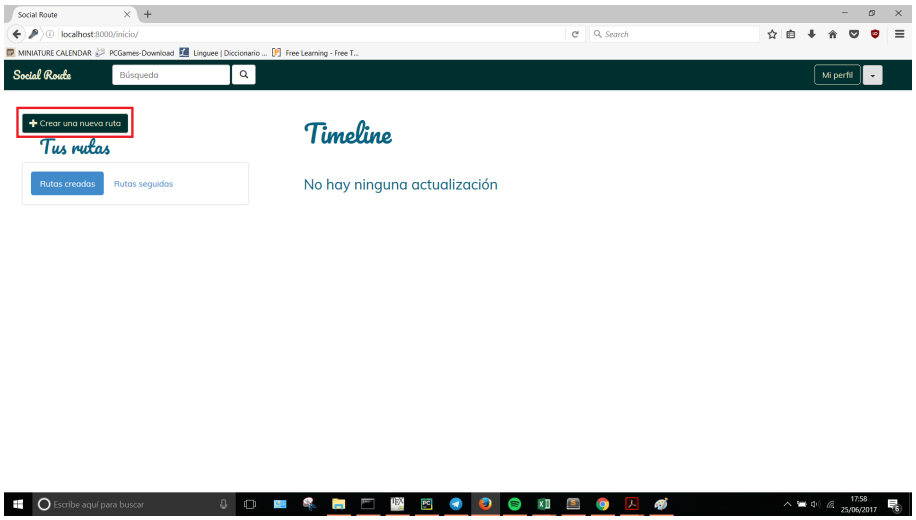

Figura C.7: Crear ruta. Paso 1.a. Pantalla principal

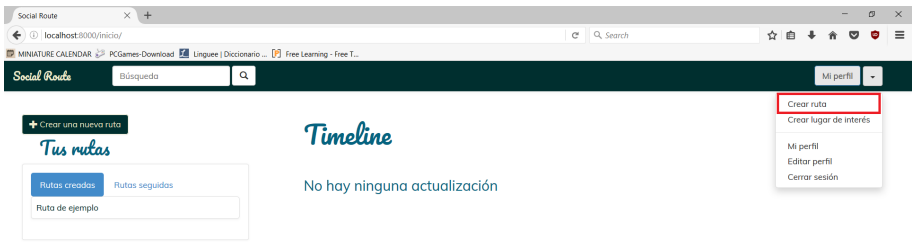

Figura C.8: Crear ruta. Paso 1.b. Pantalla principal

hechos, se selecciona *Guardar* C.11.

C.3.7. Borrar una ruta

Para borrar una ruta se seleccionará la opción *Borrar* en la página de la ruta C.12.

Tras seleccionar la opción borrar el sistema pedirá que se confirme la acción C.13.

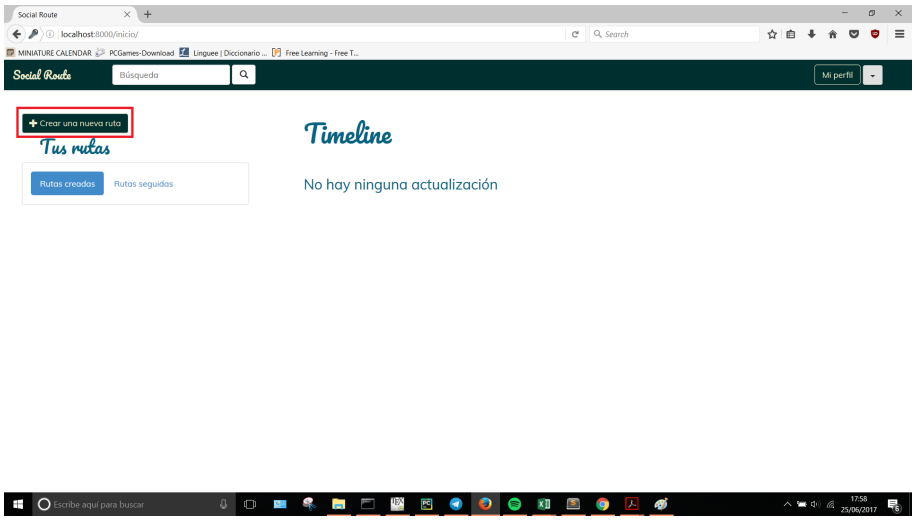

Figura C.9: Crear ruta. Paso 2. Formulario de creación de la ruta.

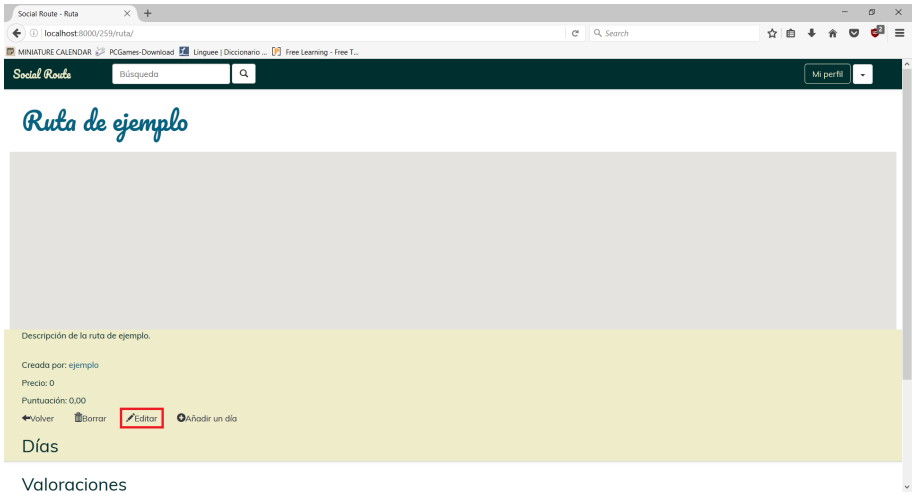

Figura C.10: Editar ruta. Paso 1. Página de ruta

C.3.8. Crear un día

Para crear un día se seleccionará la opción *Añadir día* en la página de la ruta C.14.

Tras seleccionar *Añadir día* se abrirá la página de creación del día. En ella se cumplimentan los campos y se selecciona *Guardar* C.15.

C.3. Funcionalidades

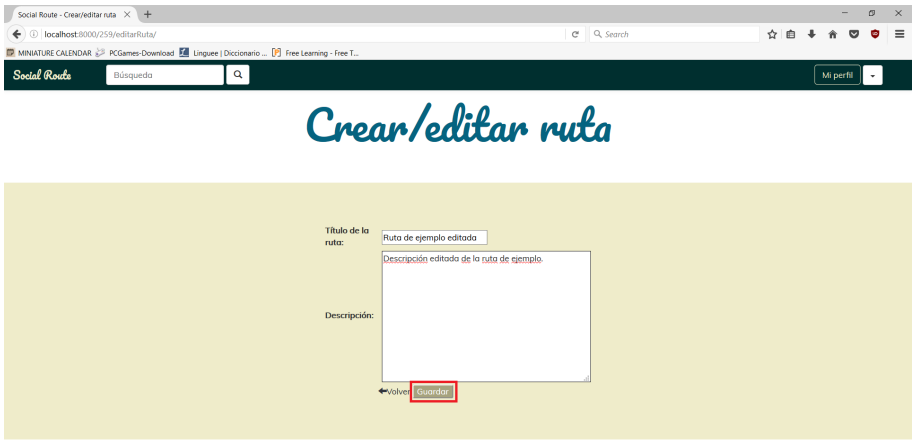

Figura C.11: Editar ruta. Paso 2. Página de edición de ruta

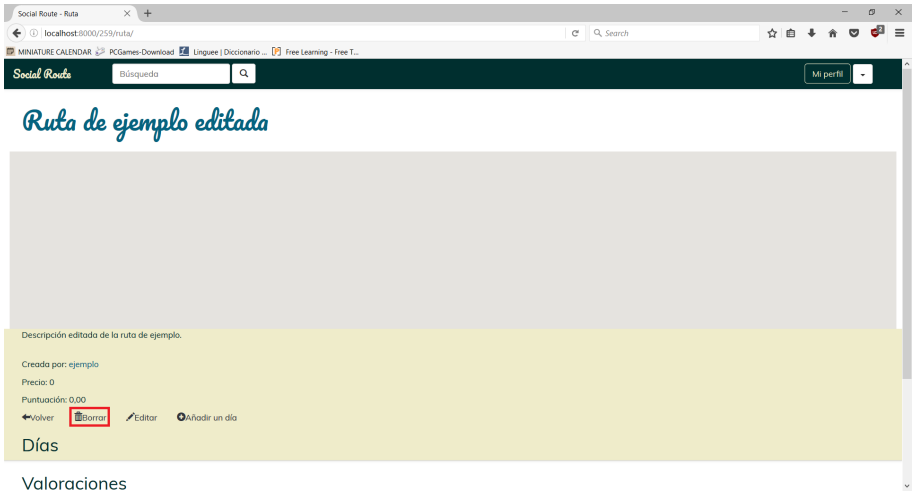

Figura C.12: Borrar ruta. Paso 1. Página de ruta

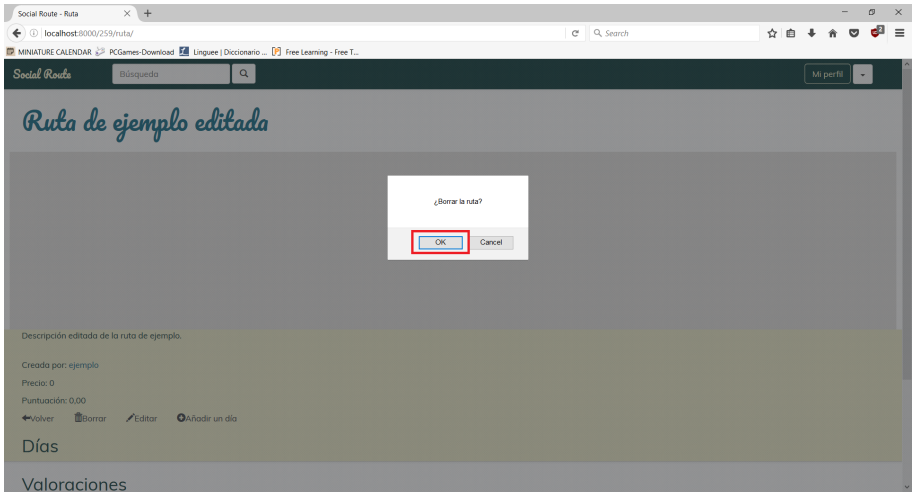

Figura C.13: Borrar ruta. Paso 2. Confirmación de la acción

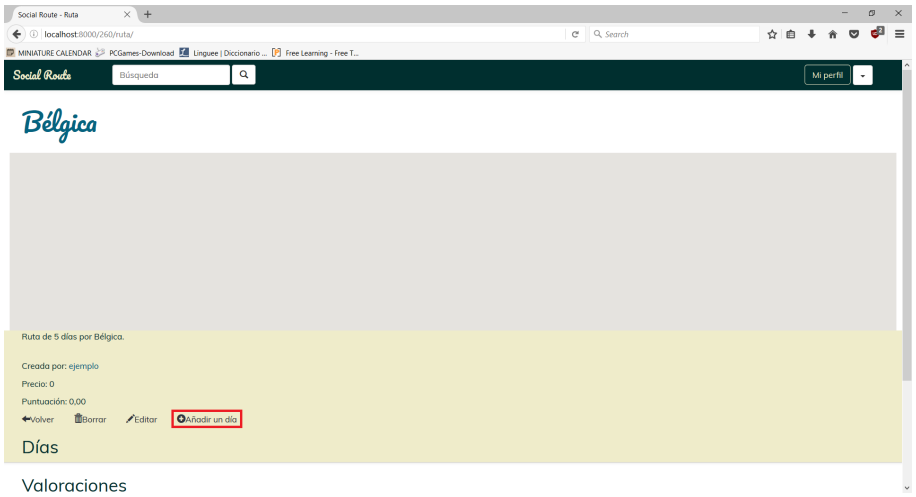

Figura C.14: Crear día. Paso 1. Página de ruta

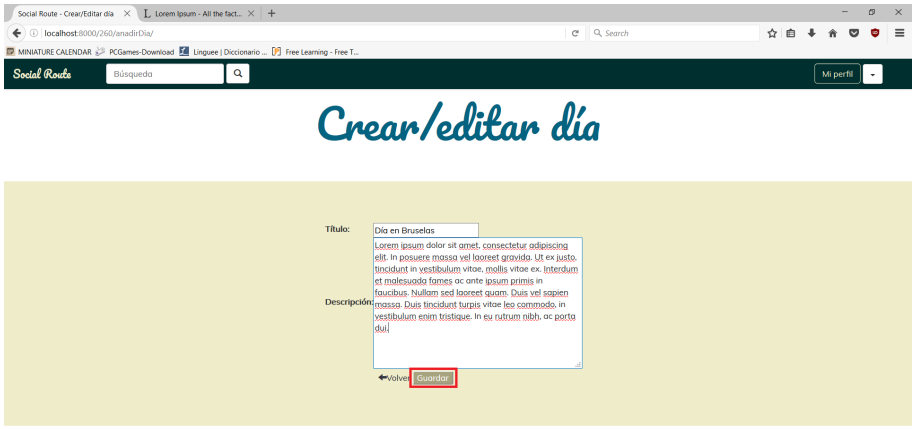

Figura C.15: Crear día. Paso 2. Página de creación del día

C.3.9. Editar un día

Para editar un día se seleccionará la opción *Editar* en la página de información del día C.16.

Tras seleccionar *Editar* se abrirá la página de edición del día. En ella se modifican los campos que se estimen necesarios y se selecciona *Guardar* C.17.

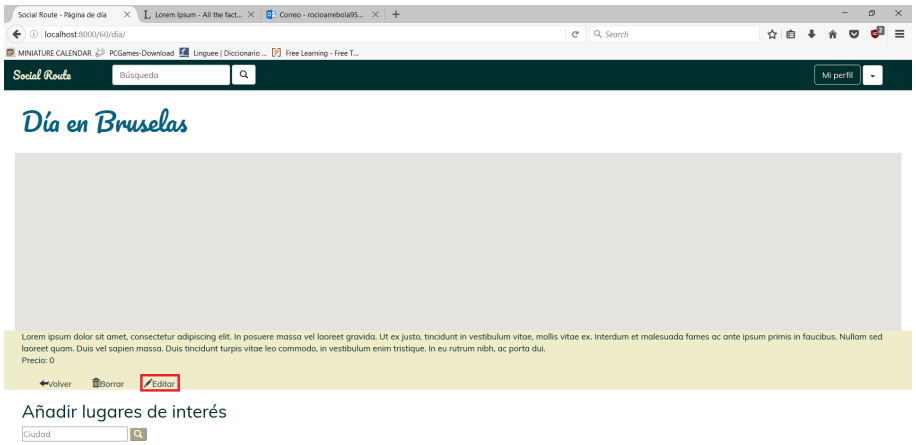

Figura C.16: Editar día. Paso 1. Página de día

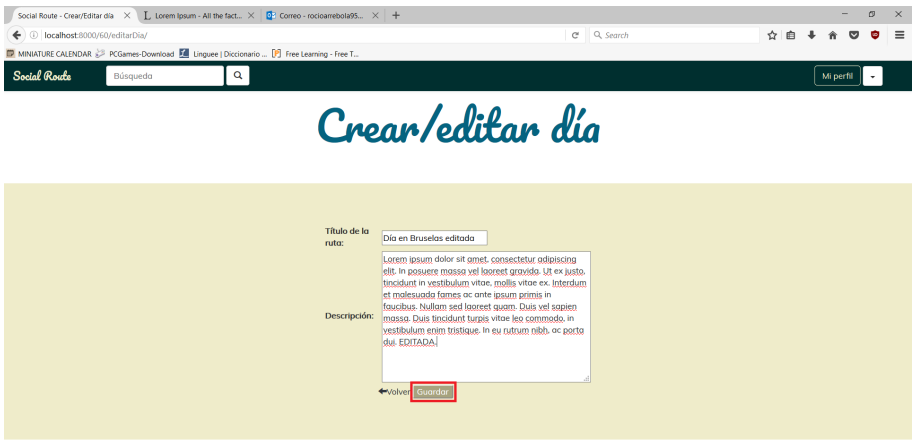

Figura C.17: Editar día. Paso 2. Página de edición del día

C.3.10. Añadir un lugar de interés a un día

Para añadir un lugar de interés de un día se introducirá el nombre de una ciudad en el campo de búsqueda que hay en la página del día y se seleccionará el botón con el símbolo de la lupa C.18.

La página cargará un listado con los lugares de interés de la ciudad buscada. Se seleccionarán aquellos que se quieran añadir y se pulsará *Añadir* C.19.

Los lugares de interés añadidos se mostrarán en la página del día C.20.

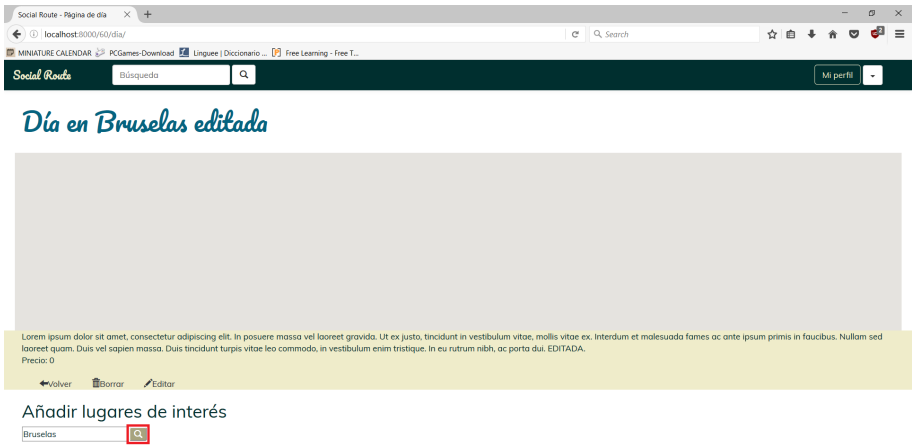

Figura C.18: Añadir lugar de interés a día. Paso 1. Página de día

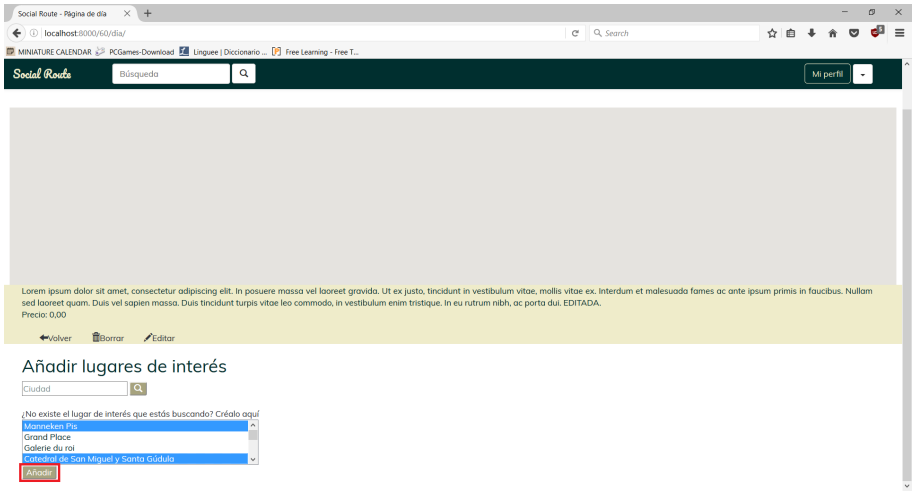

Figura C.19: Añadir lugar de interés a día. Paso 2. Página de día

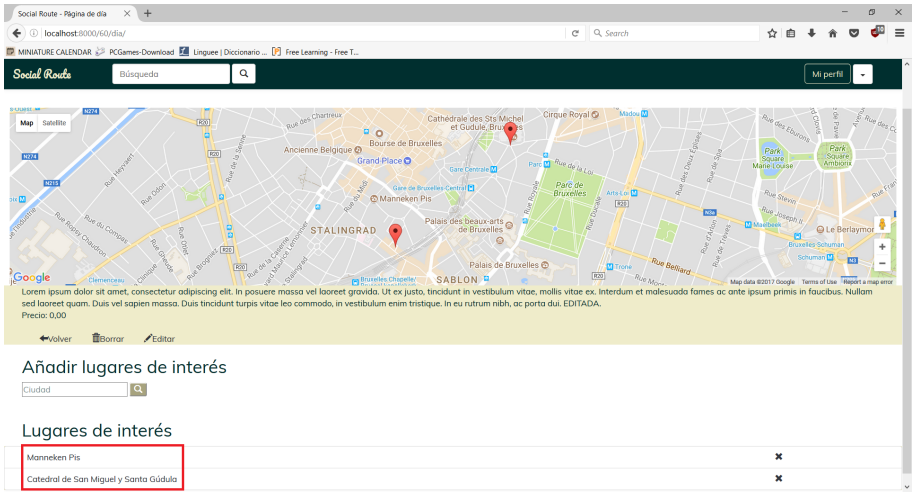

Figura C.20: Añadir lugar de interés a día. Paso 3. Página de día

C.3.11. Quitar un lugar de interés de un día

Para quitar un lugar de interés de un día habrá que pulsar el botón con el símbolo *X* que hay al lado del lugar de interés que se desea quitar C.21.

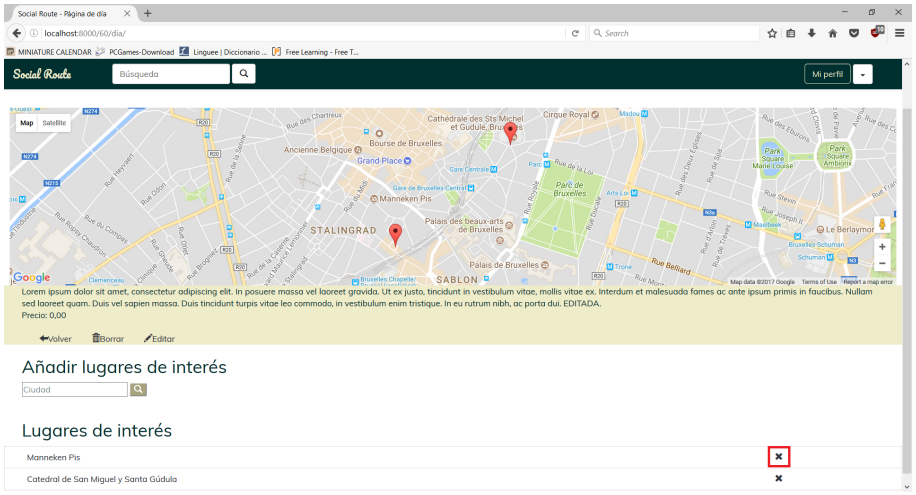

Figura C.21: Quitar lugar de interés de día. Paso 1. Página de día

Tras pulsar el botón con el símbolo *X*, es necesario confirmar la acción C.22.

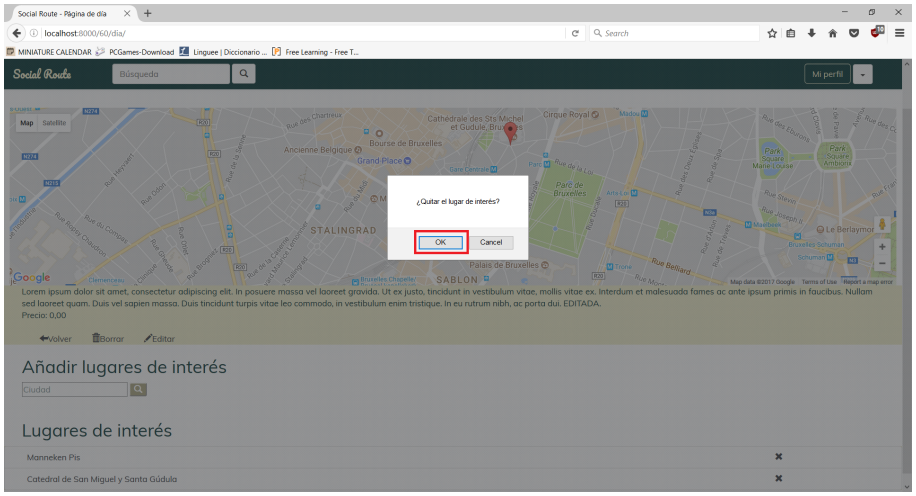

Figura C.22: Quitar lugar de interés de día. Paso 2. Página de día

C.3.12. Borrar un día

Para eliminar un día habrá que seleccionar *Borrar* en la página del día C.23.

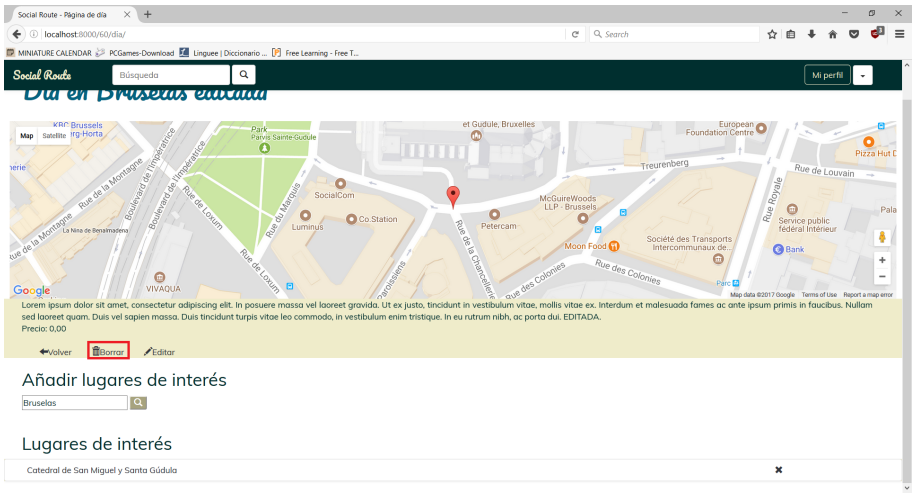

Figura C.23: Borrar día. Paso 1. Página de día

Tras seleccionar *Borrar* será necesario confirmar la acción C.24.

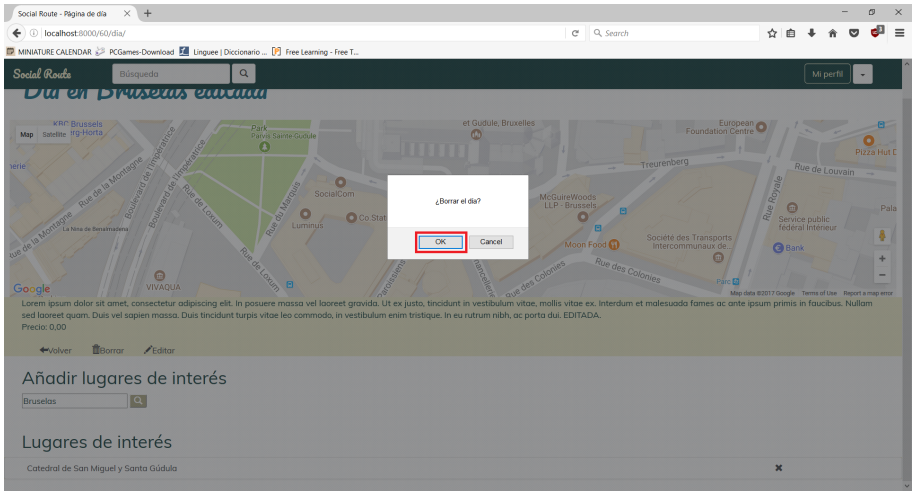

Figura C.24: Borrar día. Paso 2. Página de día

C.3.13. Crear un lugar de interés

Es posible crear un lugar de interés desde cualquier página de la aplicación, para ello se seleccionará *Crear ruta* en el *Dropdown* de la barra de navegación C.25.

Tras seleccionar *Crear lugar de interés* se abrirá la página de creación del lugar de interés. En ella se cumplimentarán los campos y se seleccionará *Guardar* C.26.

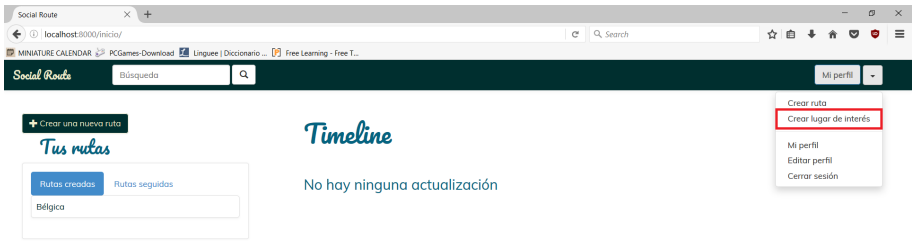

Figura C.25: Crear lugar de interés. Paso 1. Página principal

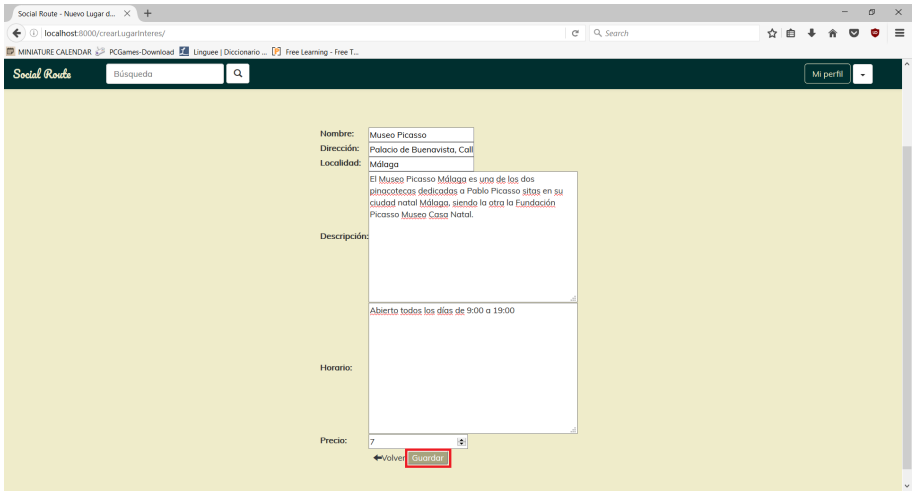

Figura C.26: Crear lugar de interés. Paso 2. Página de creación del lugar de interés

C.3.14. Editar un lugar de interés

Para editar un lugar de interés se seleccionará la opción *Editar* en la página del lugar de interés C.27.

Tras seleccionar *Editar* se abrirá la página de edición del lugar de interés. En ella se modificarán los campos que se estimen necesarios y se seleccionará *Guardar* **??**.

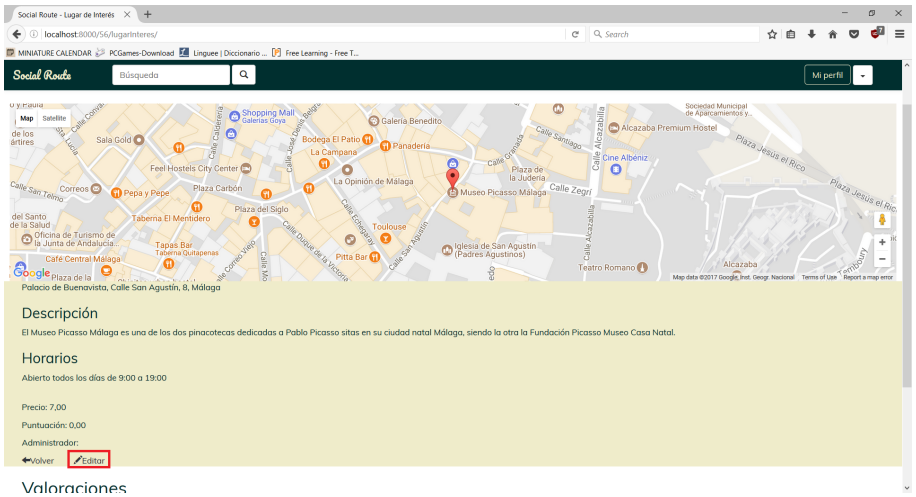

Figura C.27: Editar lugar de interés. Paso 1. Página de lugar de interés

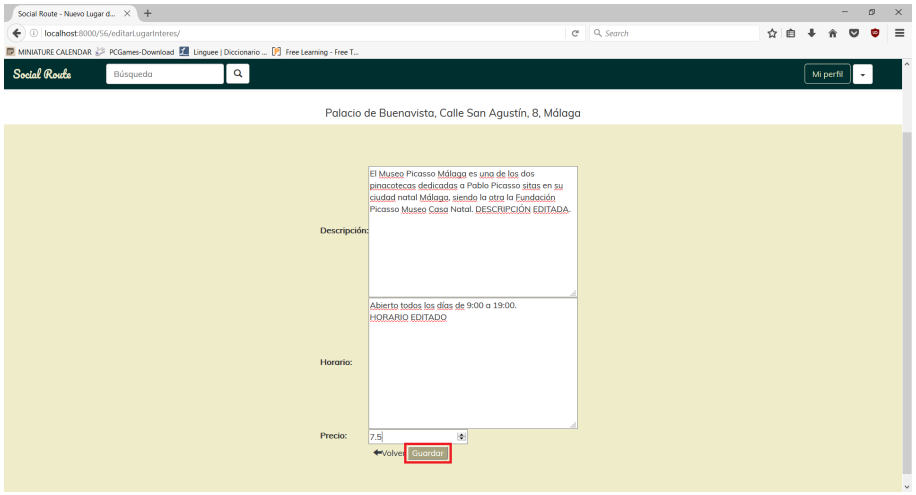

Figura C.28: Editar lugar de interés. Paso 2. Página de edición del lugar de interés

C.3.15. Búsqueda

Para realizar búsquedas bastará con introducir algún parámetro de búsqueda en el formulario que hay en la barra de navegación y pulsar el botón con el símbolo de la lupa C.29.

C.3.16. Seguir una ruta

Para seguir una ruta bastará con pulsar el botón *Seguir* en la página de la ruta C.30.

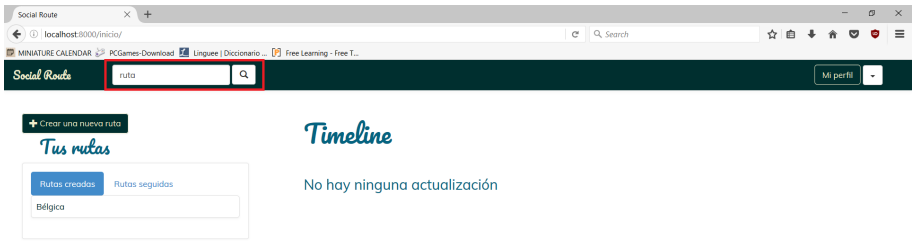

Figura C.29: Búsqueda. Página principal

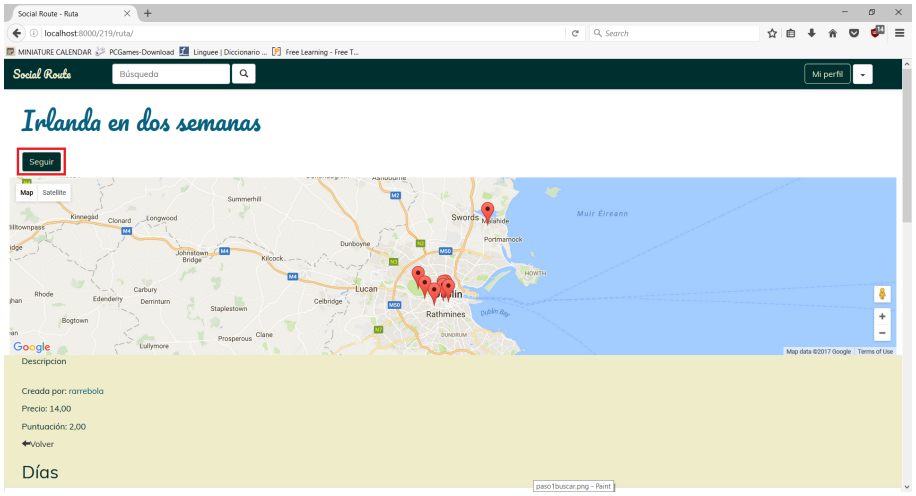

Figura C.30: Seguir una ruta. Página de ruta

C.3.17. Dejar de seguir una ruta

Para dejar de seguir una ruta bastará con pulsar el botón *Dejar de seguir* en la página de la ruta C.31.

C.3.18. Seguir a un usuario

Para seguir a un usuario bastará con pulsar el botón *Seguir* en la página del usuario C.32.

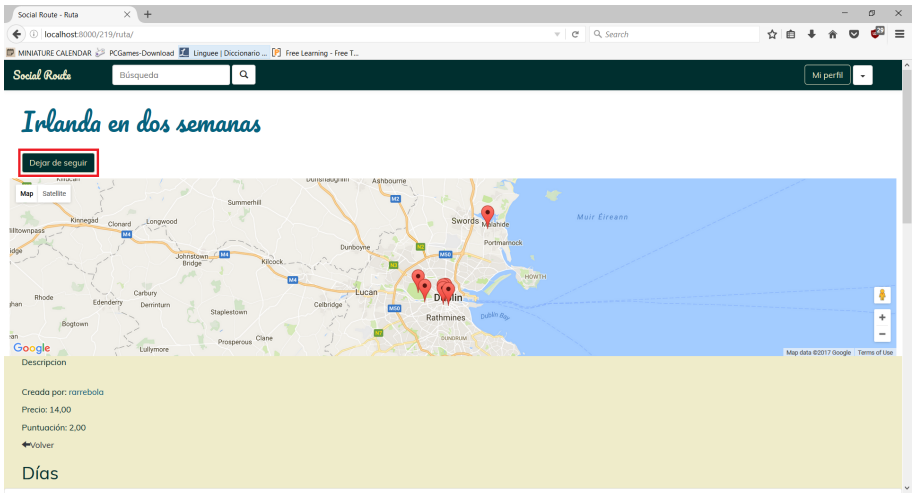

Figura C.31: Dejar de seguir una ruta. Página de ruta

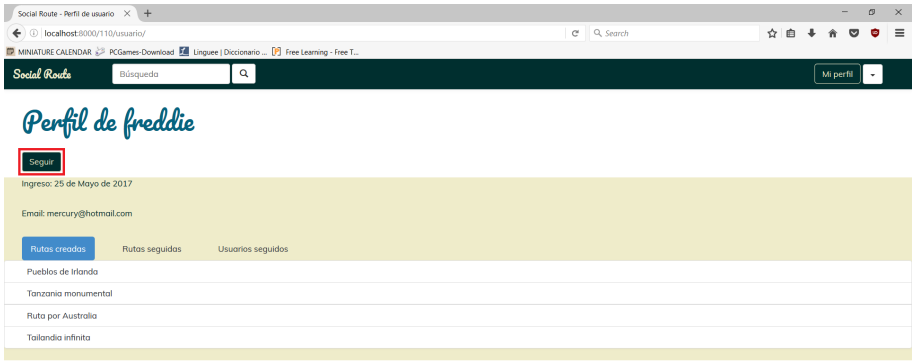

Figura C.32: Seguir a un usuario. Página de usuario

C.3.19. Dejar de seguir un usuario

Para dejar de seguir a un usuario bastará con pulsar el botón *Dejar de seguir* en la página del usuario C.33.

C.3.20. Valorar una ruta

Para valorar una ruta bastará con completar el formulario que hay en la página de la misma y seleccionar *Valorar* C.34.

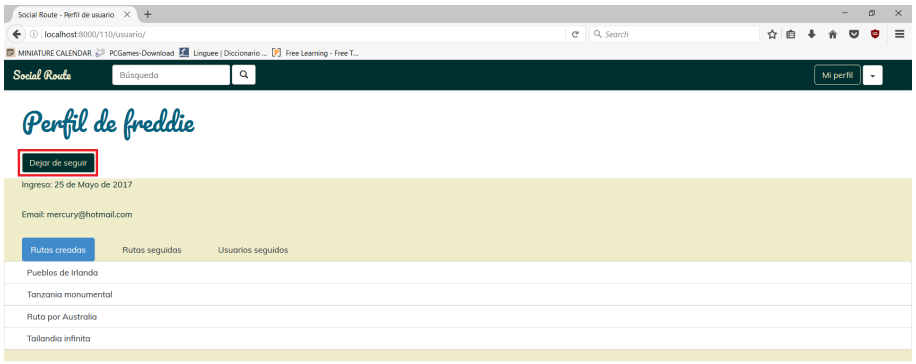

Figura C.33: Dejar de seguir a un usuario. Página de usuario

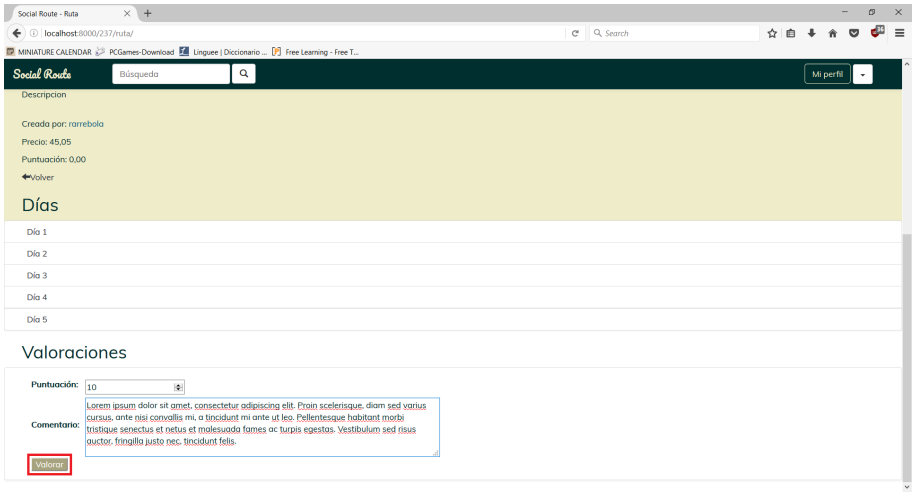

Figura C.34: Valorar una ruta. Página de ruta

C.3.21. Borrar la valoración de una ruta

Para borrar la valoración de una ruta habrá que seleccionar *Borrar*, justo debajo de la valoración hecha C.35.

Tras seleccionar *Borrar* será necesario confirmar la acción C.36.

C.3.22. Valorar un lugar de interés

Para valorar un lugar de interés bastará con completar el formulario que hay en la página del mismo y seleccionar *Valorar* C.37.

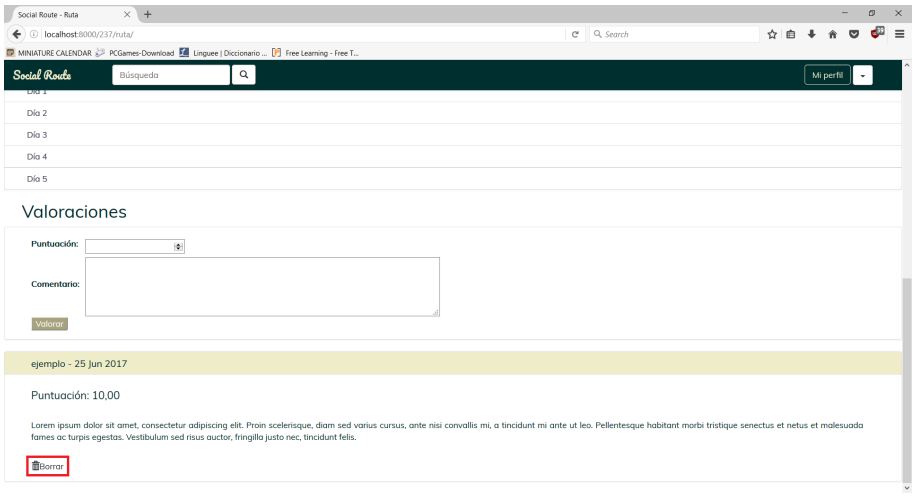

Figura C.35: Borrar la valoración de una ruta. Paso 1. Página de ruta

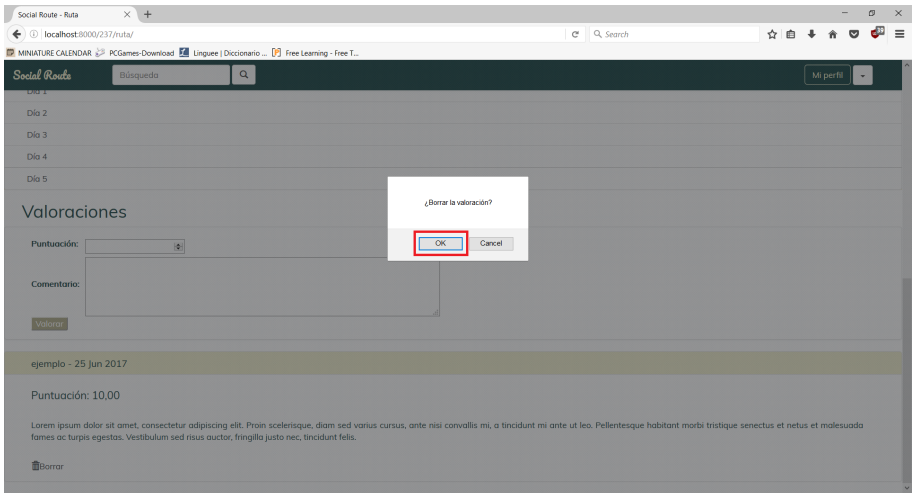

Figura C.36: Borrar la valoración de una ruta. Paso 2. Página de ruta

C.3.23. Borrar la valoración de un lugar de interés

Para borrar la valoración de un lugar de interés habrá que seleccionar *Borrar*, justo debajo de la valoración hecha C.38.

Tras seleccionar *Borrar* será necesario confirmar la acción C.39.

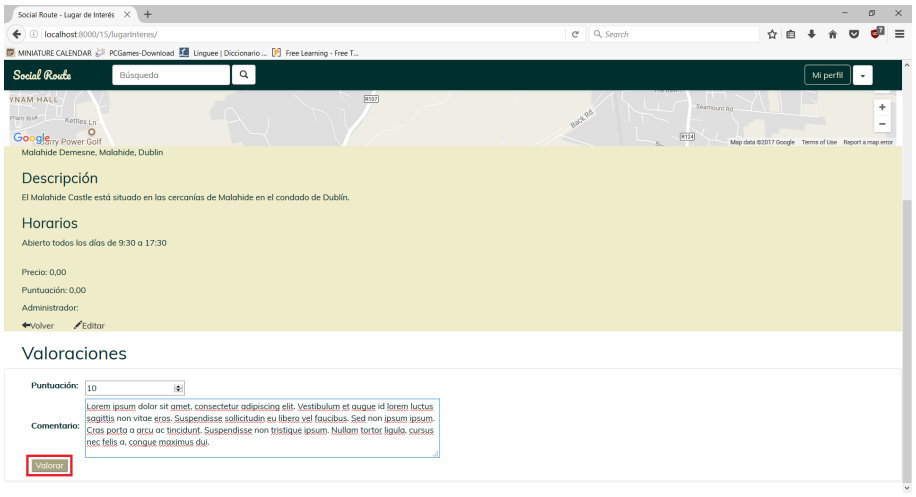

Figura C.37: Valorar una ruta. Página de lugar de interés

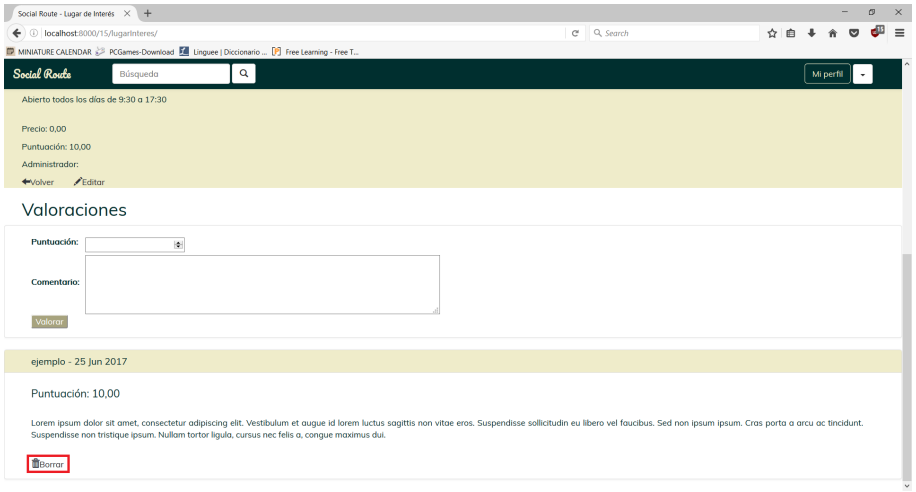

Figura C.38: Borrar la valoración de un lugar de interés. Paso 1. Página de lugar de interés

C.3.24. Editar perfil de usuario

Existen dos maneras posibles de editar el perfil:

1. Seleccionando *Editar perfil* en la página del perfil de usuario C.40.
2. Seleccionando *Editar perfil* en el *Dropdown* de la barra de navegación en cualquier página de la aplicación C.41.

Sea cual sea la opción elegida, se abrirá la página de edición de perfil, en la que se mostrará un formulario que se modificará según se estime conveniente y se pulsará *Guardar* C.42.

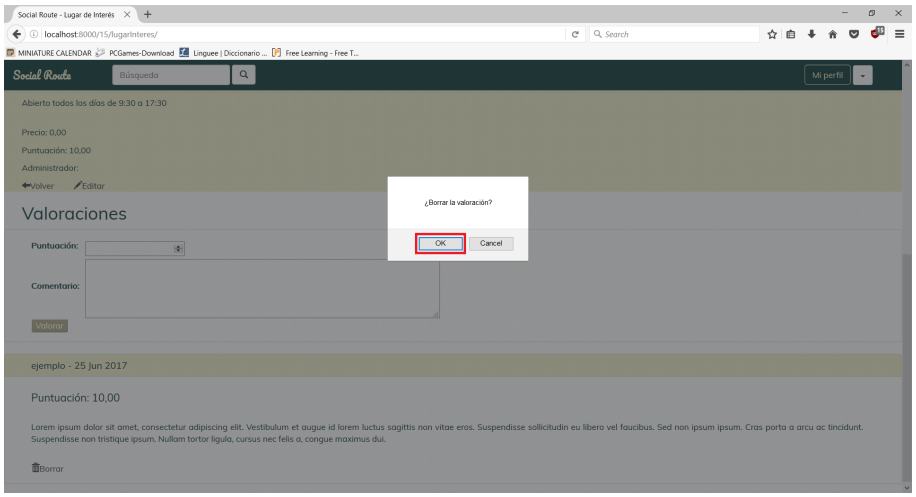

Figura C.39: Borrar la valoración de un lugar de interés. Paso 2. Página de lugar de interés

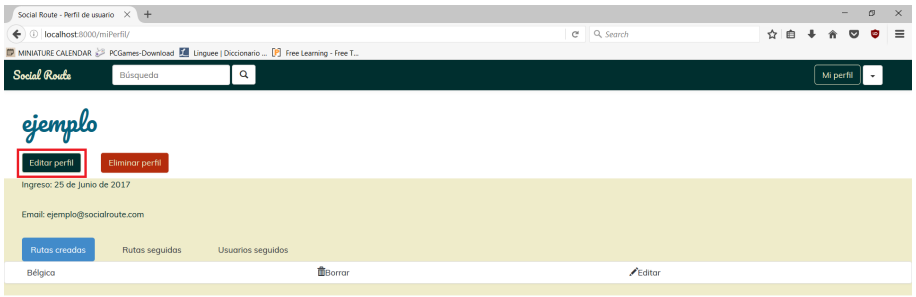

Figura C.40: Editar perfil de usuario. Paso 1.a. Página del perfil de usuario

C.3.25. Eliminar perfil de usuario

Para eliminar el perfil de usuario habrá que seleccionar *Eliminar perfil* en la página del perfil de usuario C.43.

Tras seleccionar *Eliminar perfil* será necesario confirmar la acción C.44.

C.3. Funcionalidades

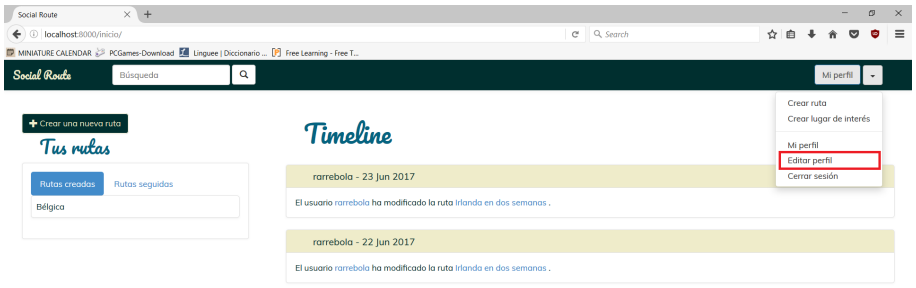

Figura C.41: Editar perfil de usuario. Paso 1.b. Página principal

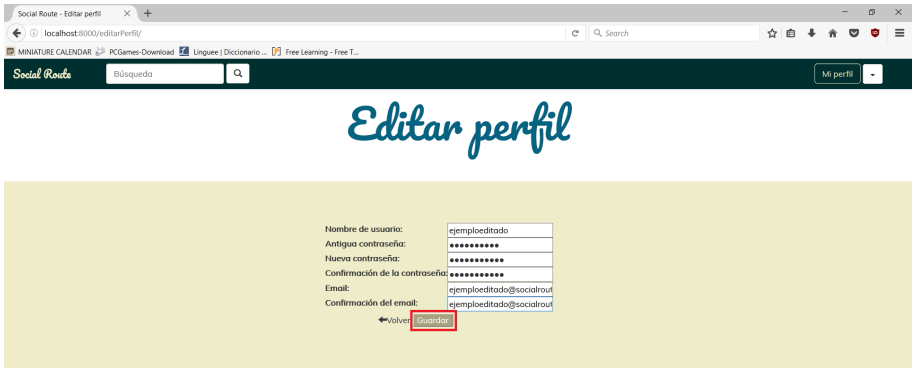

Figura C.42: Editar perfil de usuario. Paso 2. Página de edición del perfil de usuario

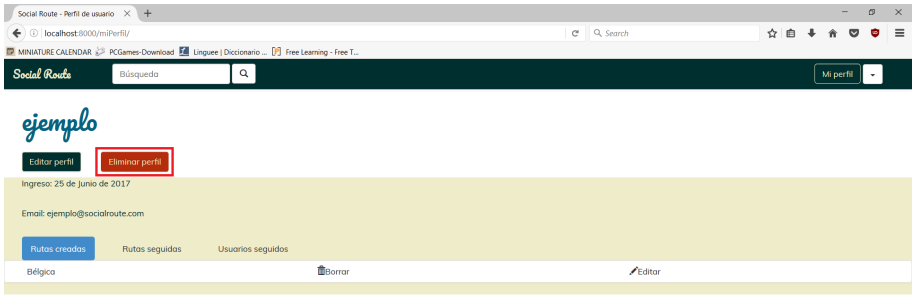

Figura C.43: Eliminar perfil de usuario. Paso 1. Página de perfil de usuario

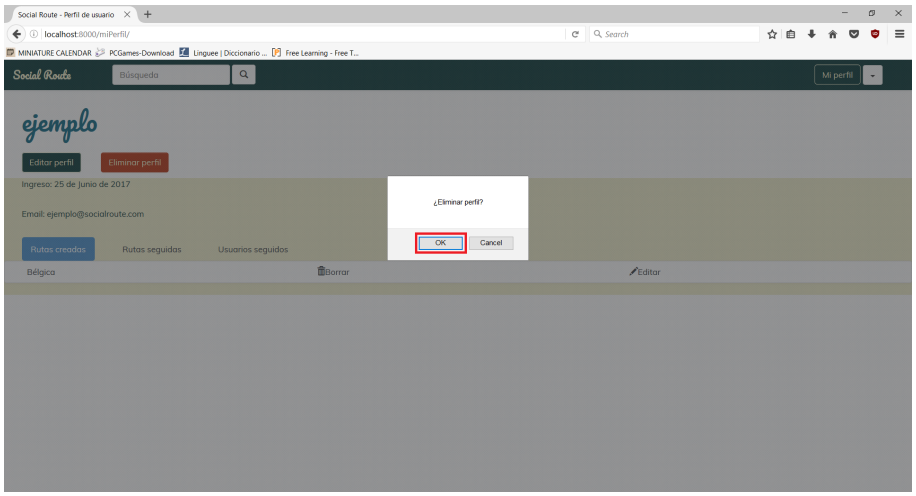

Figura C.44: Eliminar perfil de usuario. Paso 2. Página de perfil de usuario