

**TFG del Grado en Ingeniería
Informática**

**Aplicación de ayuda para
viajeros
Documentación Técnica**

Presentado por Andrés Rodríguez Rosales
en Universidad de Burgos — 7 de julio
de 2021

Tutor: Jesús Manuel Maudes Raedo

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	25
Apéndice B Especificación de Requisitos	29
B.1. Introducción	29
B.2. Objetivos generales	29
B.3. Catalogo de requisitos	29
B.4. Especificación de requisitos	30
Apéndice C Especificación de diseño	41
C.1. Introducción	41
C.2. Diseño de datos	41
C.3. Diseño procedimental	45
C.4. Diseño arquitectónico	47
Apéndice D Documentación técnica de programación	49
D.1. Introducción	49
D.2. Estructura de directorios	49
D.3. Manual del programador	54

D.4. Compilación, instalación y ejecución del proyecto	59
D.5. Pruebas del sistema	61
Apéndice E Documentación de usuario	63
E.1. Introducción	63
E.2. Requisitos de usuarios	63
E.3. Manual del usuario	63
Bibliografía	73

Índice de figuras

A.1. Issues del sprint 0.	3
A.2. Issues del sprint 1.	5
A.3. Issues del sprint 1.	6
A.4. Issues del sprint 1.	7
A.5. Issues del sprint 1.	8
A.6. Issues del sprint 1.	9
A.7. Issues del sprint 1.	10
A.8. Issues del sprint 1.	10
A.9. Issues del sprint 2.	12
A.10. Issues del sprint 2.	13
A.11. Issues del sprint 2.	14
A.12. Issues del sprint 2.	15
A.13. Issues del sprint 2.	16
A.14. Issues del sprint 2.	17
A.15. Issues del sprint 2.	18
A.16. Issues del sprint 3.	20
A.17. Issues del sprint 3.	21
A.18. Issues del sprint 3.	22
A.19. Issues del sprint 3.	23
A.20. Issues del sprint 4.	24
A.21. Issues del sprint 4.	24
A.22. Requirements encontrados por pipreqs.	28
 B.1. Diagrama de casos de uso	 40
 C.1. Diagrama E-R de las tablas utilizadas.	 43
C.2. Diagrama relacional de las tablas utilizadas.	44
C.3. Diagrama de secuencias para la búsqueda de viajes.	46

C.4. Diagrama de casos de uso para la búsqueda de viajes.	47
C.5. Diagrama de despliegue.	48
D.1. Árbol de directorios del proyecto (I).	51
D.2. Árbol de directorios del proyecto (II).	52
D.3. Árbol de directorios del proyecto (III).	53
D.4. Listado de la carpeta project.	54
D.5. Listado de la carpeta scripts.	55
D.6. Creando superusuario en Django.	56
D.7. Creando superusuario en Django.	56
D.8. Creando superusuario en Django.	58
D.9. Ejecución de shell en Django.	59
D.10.Ejemplo de consultas vía shell en Django.	59
D.11.Ejemplo de consultas vía shell en Django.	60
E.1. Pantalla inicial de la aplicación.	64
E.2. Botones agregados al mapa.	64
E.3. Boton Origin Location seleccionado.	65
E.4. Origen seleccionado en el mapa.	65
E.5. Origen seleccionado en el mapa.	65
E.6. Origen seleccionado en el mapa.	66
E.7. Campo de precio máximo.	67
E.8. Campo de tipo de ordenación.	67
E.9. Campo de tipo de ordenación.	68
E.10.Muestra de viajes sin ordenación ni filtros.	69
E.11.Muestra de viajes con ordenación ascendente por precio.	69
E.12.Muestra de viajes con ordenación descendente por precio.	70
E.13.Establezco el precio máximo como 10€.	71
E.14.Establezco el precio máximo como 10€.	71
E.15.Muestra del asistente de Moovit.	72

Índice de tablas

A.1. Costes relacionados con Recursos Humanos	25
A.2. Gasto relacionado con material hardware	26
A.3. Gasto relacionado con material software mensualmente	26
A.4. Gasto global del proyecto	27
A.5. Licencia de cada dependencia del proyecto	28

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En esta sección se tratará acerca de aspectos como pueden ser la planificación de los tiempos que se ha llevado a cabo, la viabilidad económica y la viabilidad legal del proyecto.

A.2. Planificación temporal

Para el desarrollo de la planificación temporal del proyecto nos hemos basado en la metodología ágil **Scrum** [7]. Esta metodología se basa en realizar entregas parciales del software, e ir adaptando su desarrollo en función de las opiniones que se vayan recibiendo por parte del cliente.

El desarrollo del proyecto se ha dividido en sprints o iteraciones. Cada iteración contiene un conjunto de issues o cuestiones abordadas.

A continuación mostraremos el conjunto de sprints, con sus fechas aproximadas y una explicación de los temas solucionados o abordados.

Sprint 0

Duración: 21-4-2021 al 5-5-2021 Tareas realizadas:

- Aprendizaje del framework Django.
- Creación de tablas para gestionar nodos y peticiones.

- Extracción de la información de distintas fuentes de datos.
- Importación de datos de nodos.

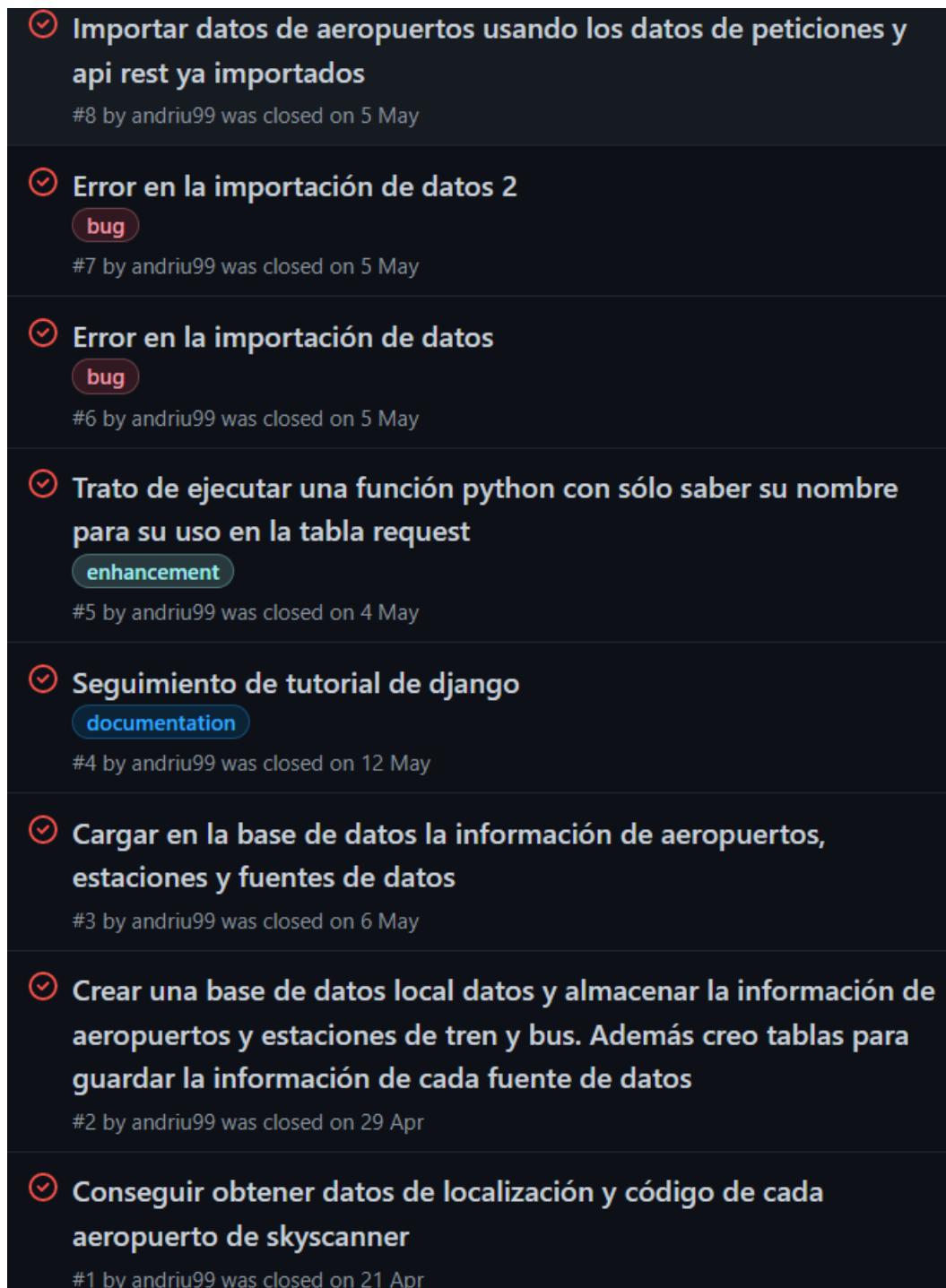


Figura A.1: Issues del sprint 0.

Sprint 1

Duración: 6-5-2021 al 20-5-2021

Tareas realizadas:

- Continuo desarrollando el modelo de tablas de la aplicación. Se creara la tabla para almacenar los viajes.
- Exploración de la API de Google Maps.
- Extracción de la información de distintas fuentes de datos.
- Creación de una interfaz muy básica.

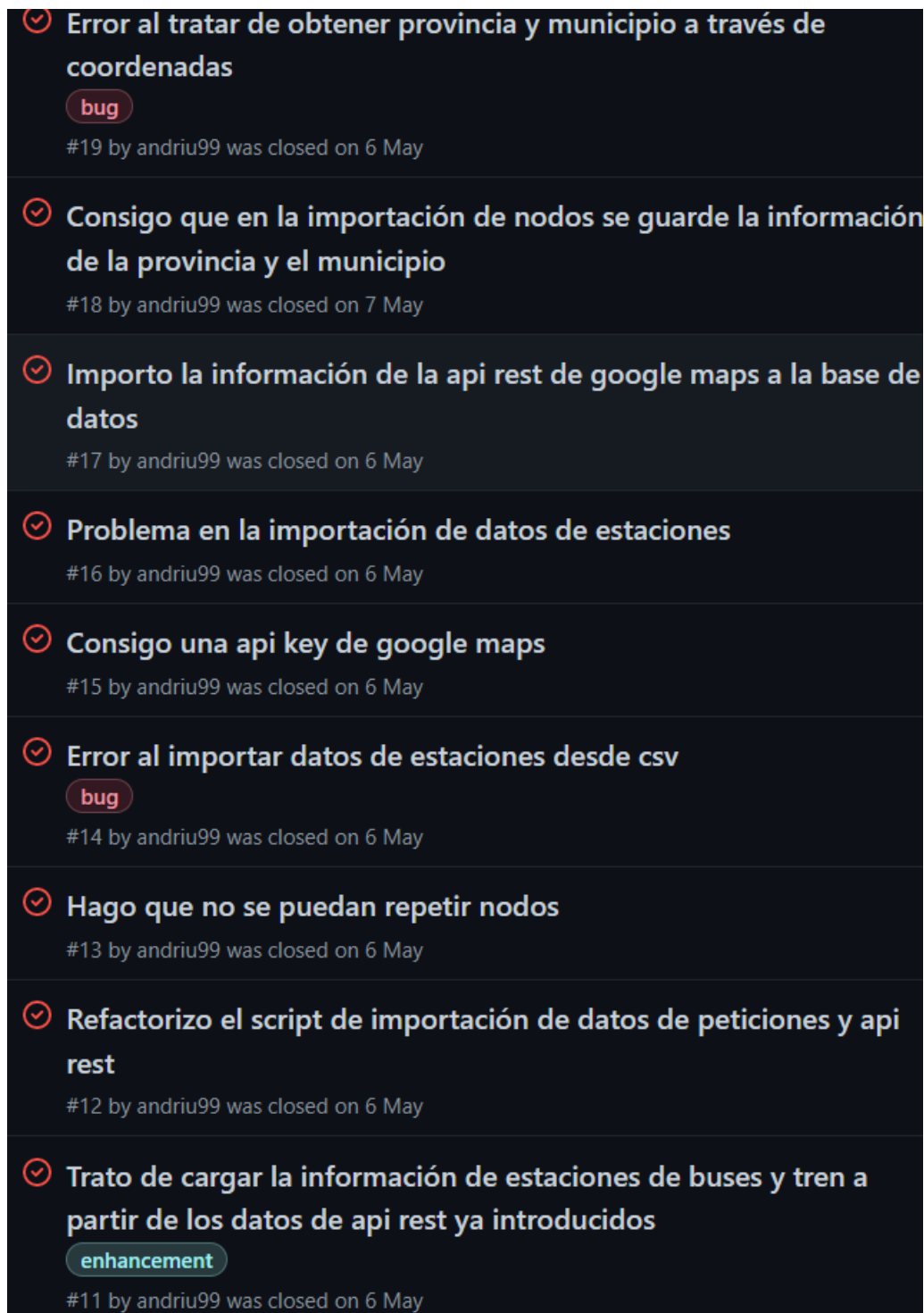


Figura A.2: Issues del sprint 1.

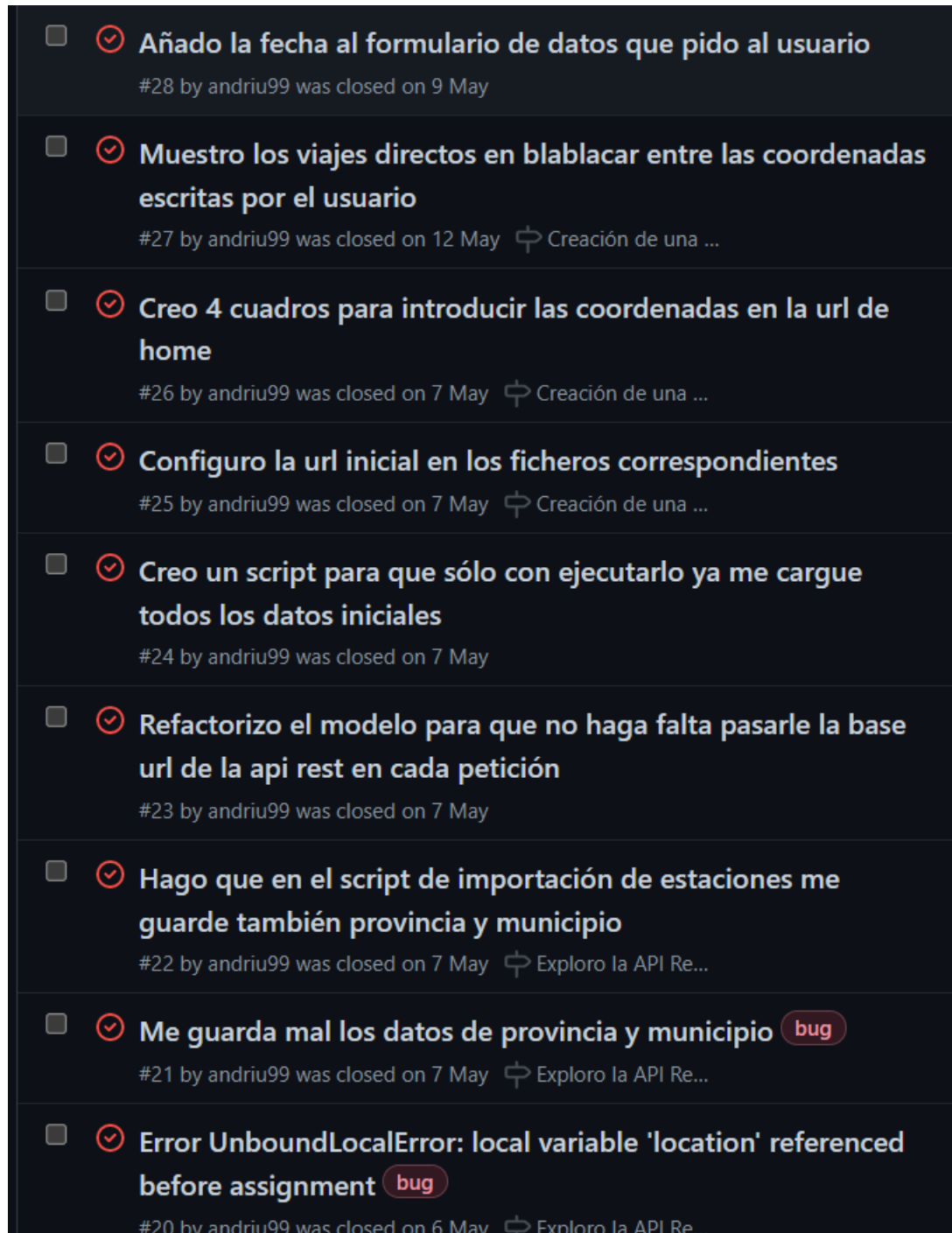


Figura A.3: Issues del sprint 1.



Figura A.4: Issues del sprint 1.

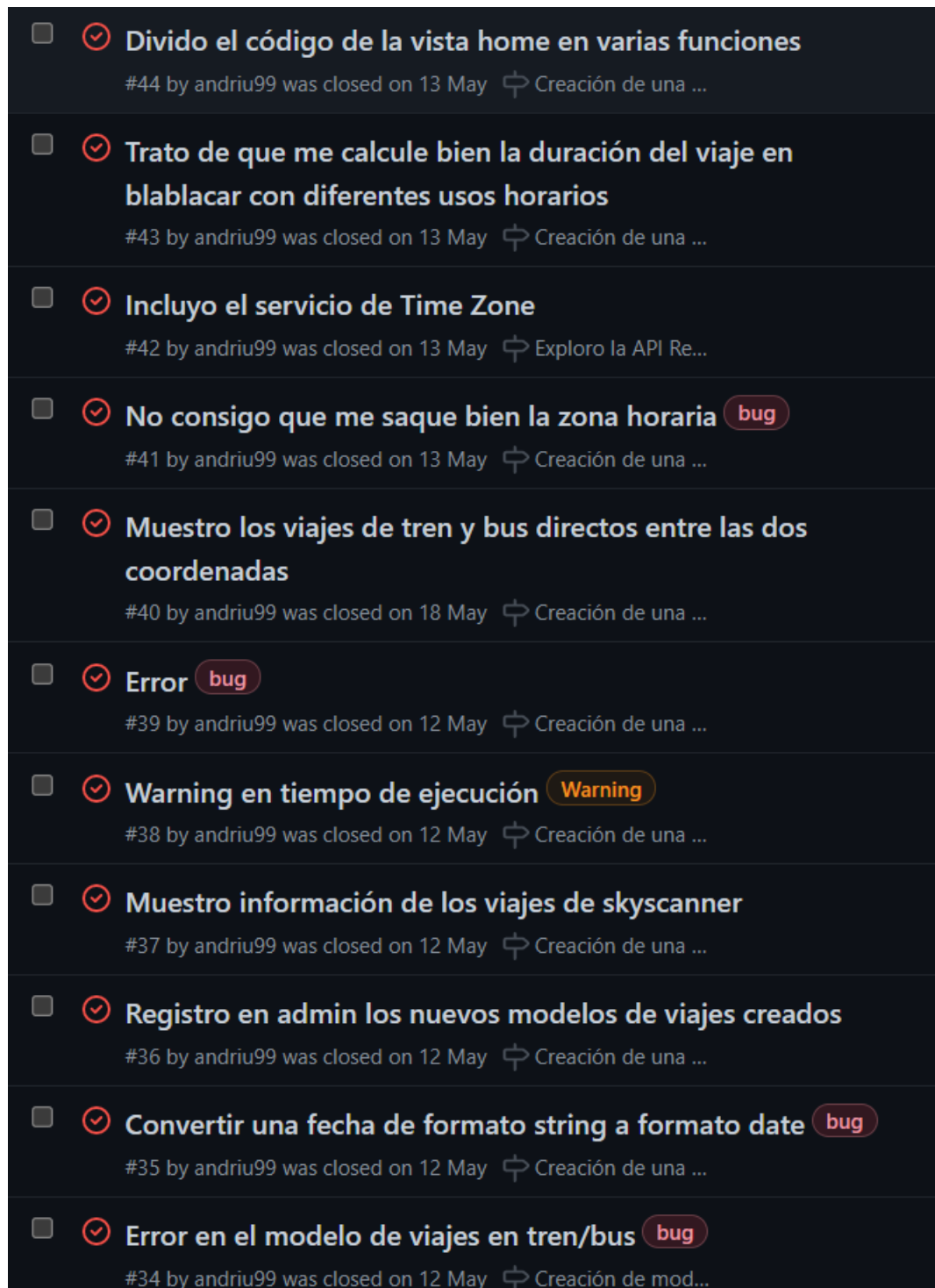


Figura A.5: Issues del sprint 1.



Figura A.6: Issues del sprint 1.

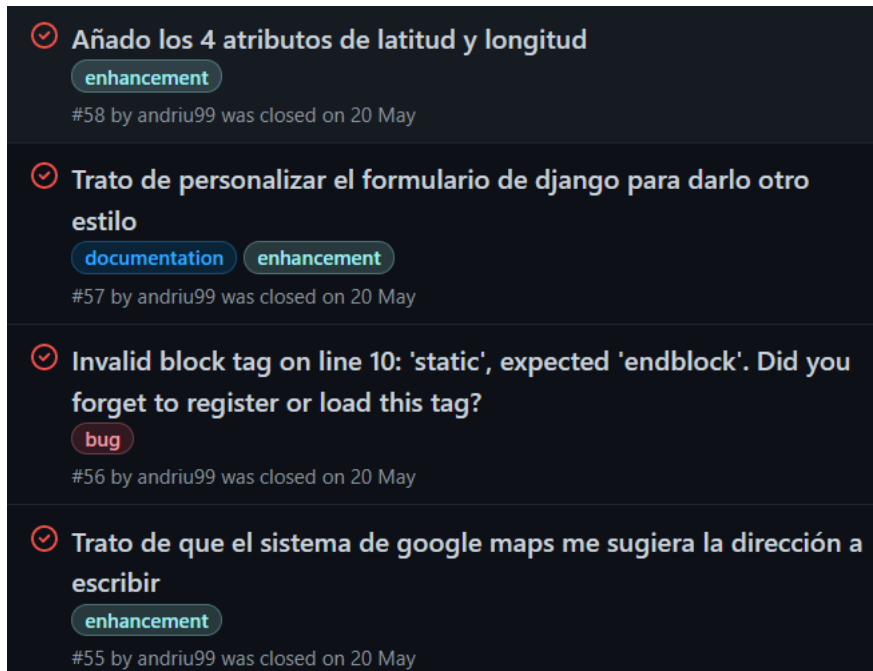


Figura A.7: Issues del sprint 1.

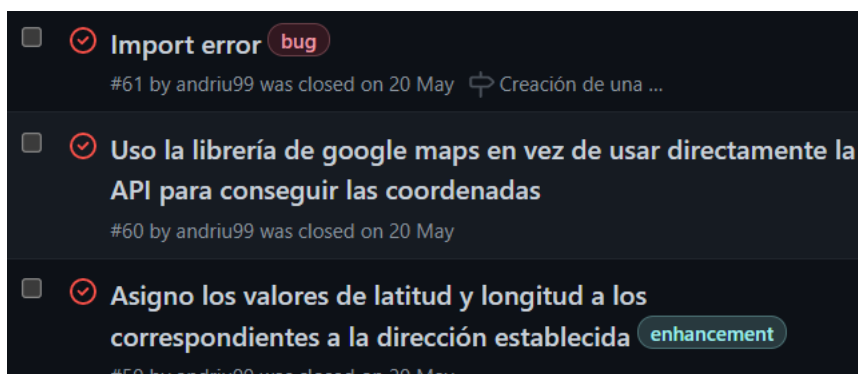


Figura A.8: Issues del sprint 1.

Sprint 2

Duración: 21-5-2021 al 5-6-2021 Tareas realizadas:

- Creación de un script que actualice los viajes en bus y en tren, y que borre los viajes anteriores a la fecha actual.
- Mejoras estéticas.
- Mejoras en el algoritmo de búsqueda de viajes.
- Investigación acerca de como se pueden plantear los viajes con transbordo.

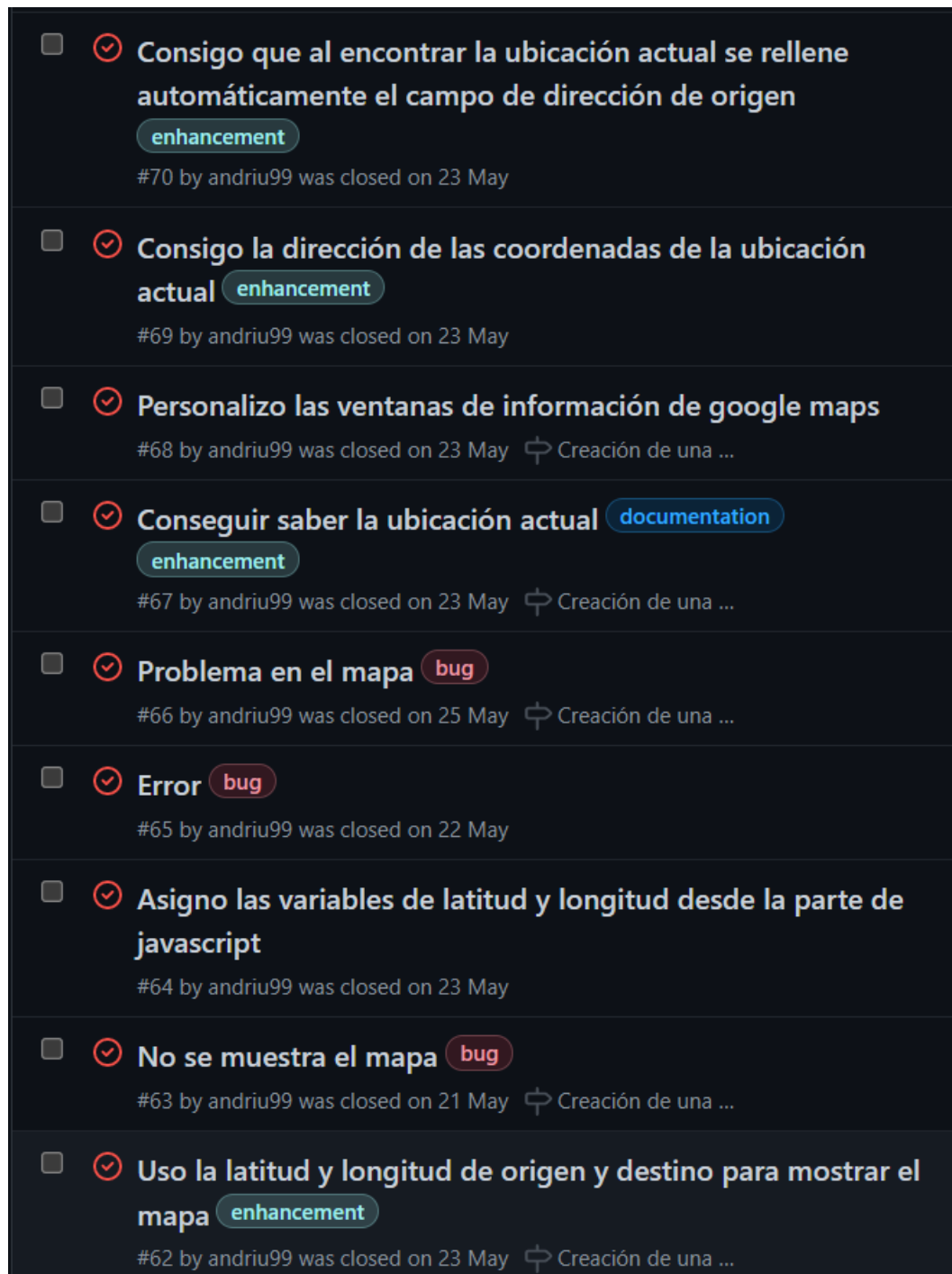


Figura A.9: Issues del sprint 2.

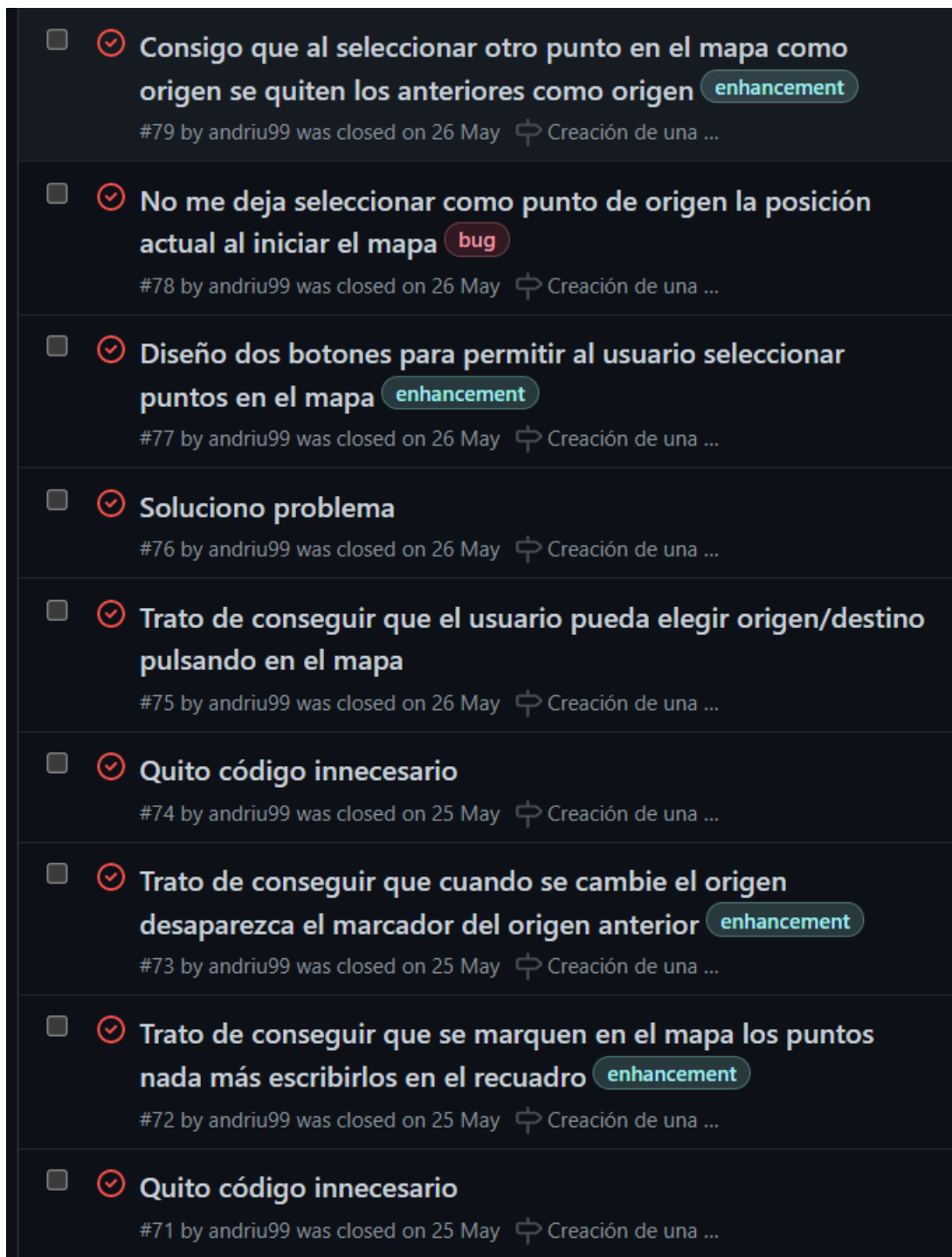


Figura A.10: Issues del sprint 2.

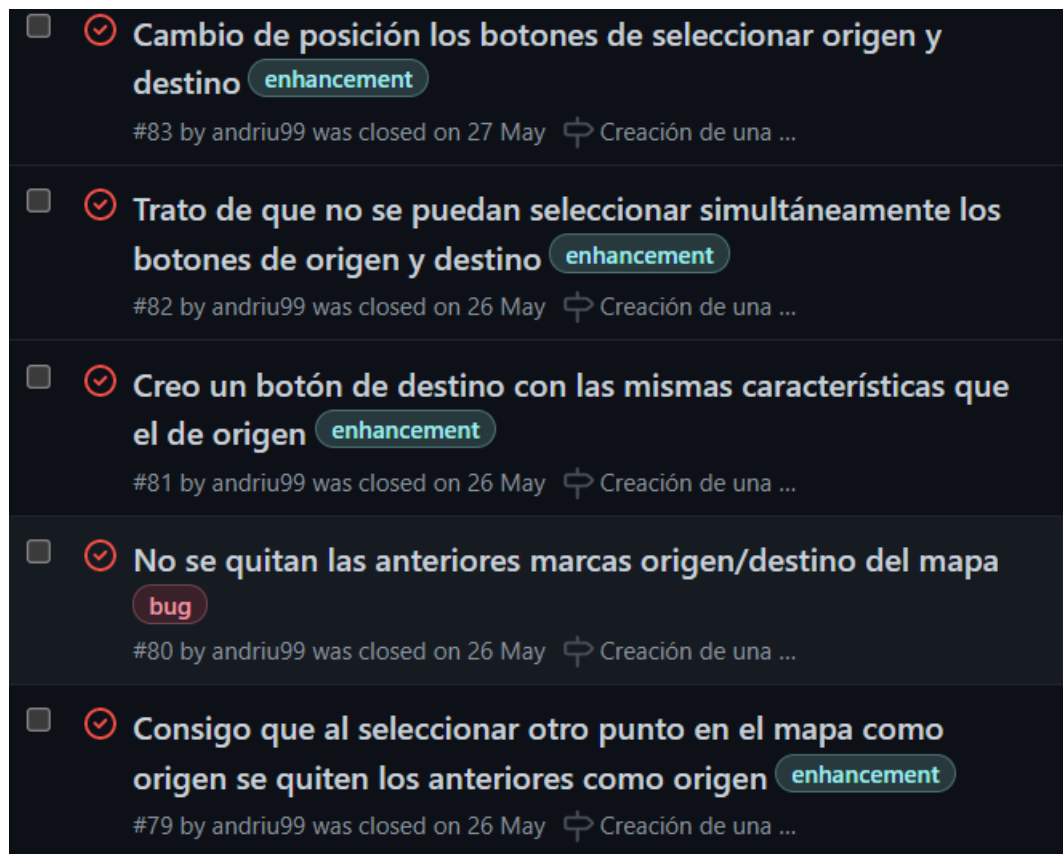


Figura A.11: Issues del sprint 2.

8 issues are listed on the board, all marked as closed. The issues are:

- ☐ **Creo un queryset de django con los viajes que me interesan y los recibo en la vista origen**
#92 by andriu99 was closed on 28 May 🔗 Creación de un si...
- ☐ **Cuando haya algún viaje entre el municipio de origen y destino no busco en trainline y muestro esos viajes**
enhancement
#91 by andriu99 was closed on 28 May 🔗 Creación de un si...
- ☐ **Error** **bug**
#90 by andriu99 was closed on 27 May 🔗 Creación de un si...
- ☐ **Error** **bug**
#89 by andriu99 was closed on 27 May 🔗 Creación de un si...
- ☐ **Encuentro el municipio según las coordenadas de origen y destino**
#88 by andriu99 was closed on 27 May 🔗 Creación de un si...
- ☐ **Busco la forma de filtrar en la base de datos para conseguir los viajes que incluyan un determinado municipio**
#87 by andriu99 was closed on 27 May 🔗 Creación de un si...
- ☐ **Bug: algunas estaciones tienen la localización como unknown en cuyo caso debemos buscar por provincia** **bug**
#86 by andriu99 was closed on 27 May 🔗 Creación de un si...
- ☐ **Mejoro el bucle de búsqueda de bus y tren**
#85 by andriu99 was closed on 27 May 🔗 Creación de un si...
- ☐ **Mejora sobre los botones de origen y destino**
#84 by andriu99 was closed on 29 May 🔗 Creación de una ...

Figura A.12: Issues del sprint 2.

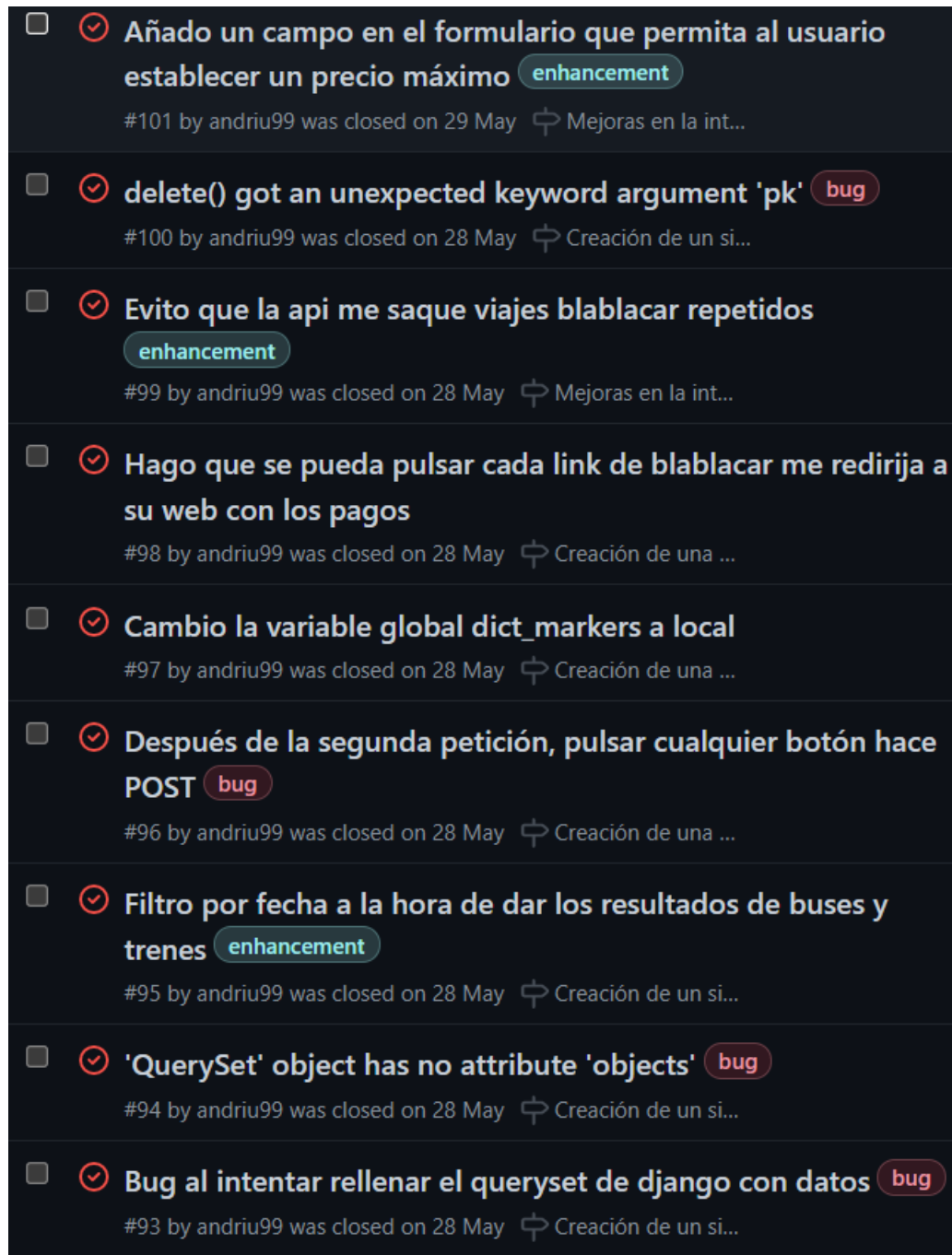


Figura A.13: Issues del sprint 2.

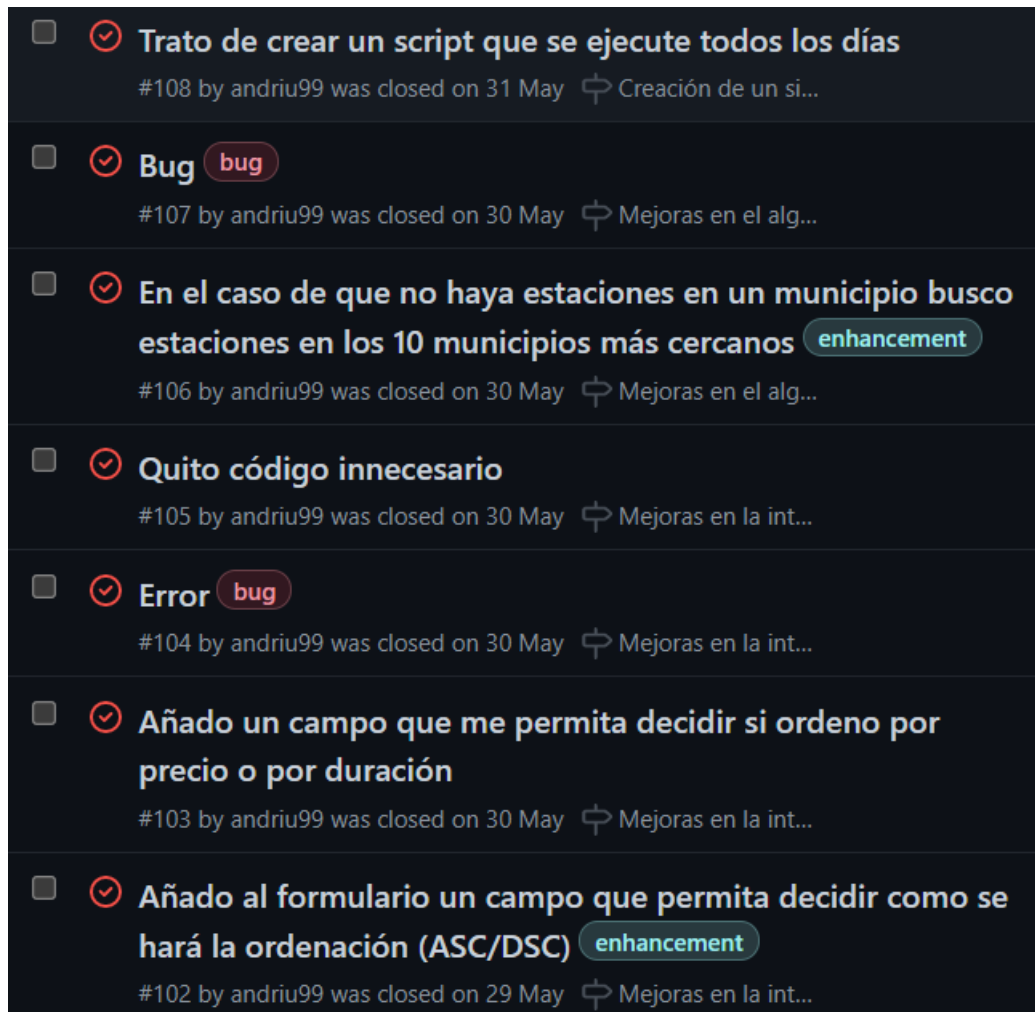


Figura A.14: Issues del sprint 2.

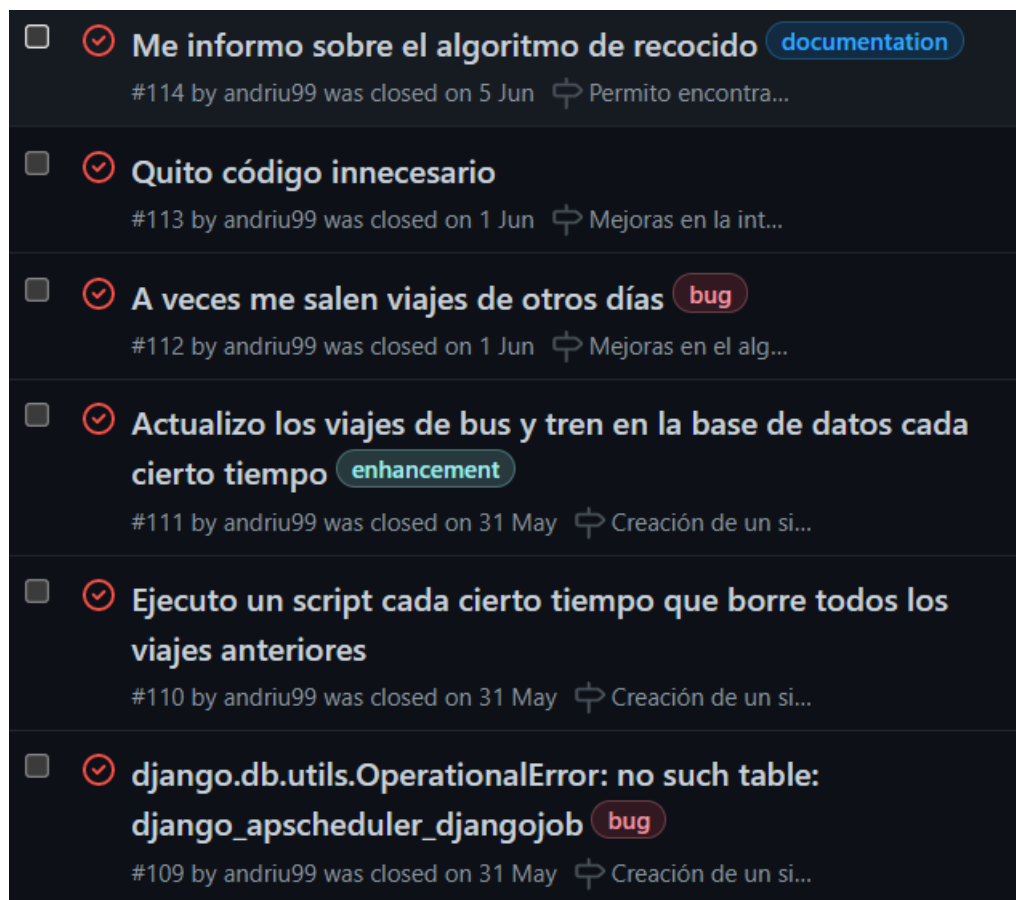


Figura A.15: Issues del sprint 2.

Sprint 3

Duración: 6-5-2021 al 22-6-2021

- Implemento el algoritmo de Dijkstra para los viajes con transbordos.
- Mejoras estéticas de situación de elementos en la pantalla.



Figura A.16: Issues del sprint 3.

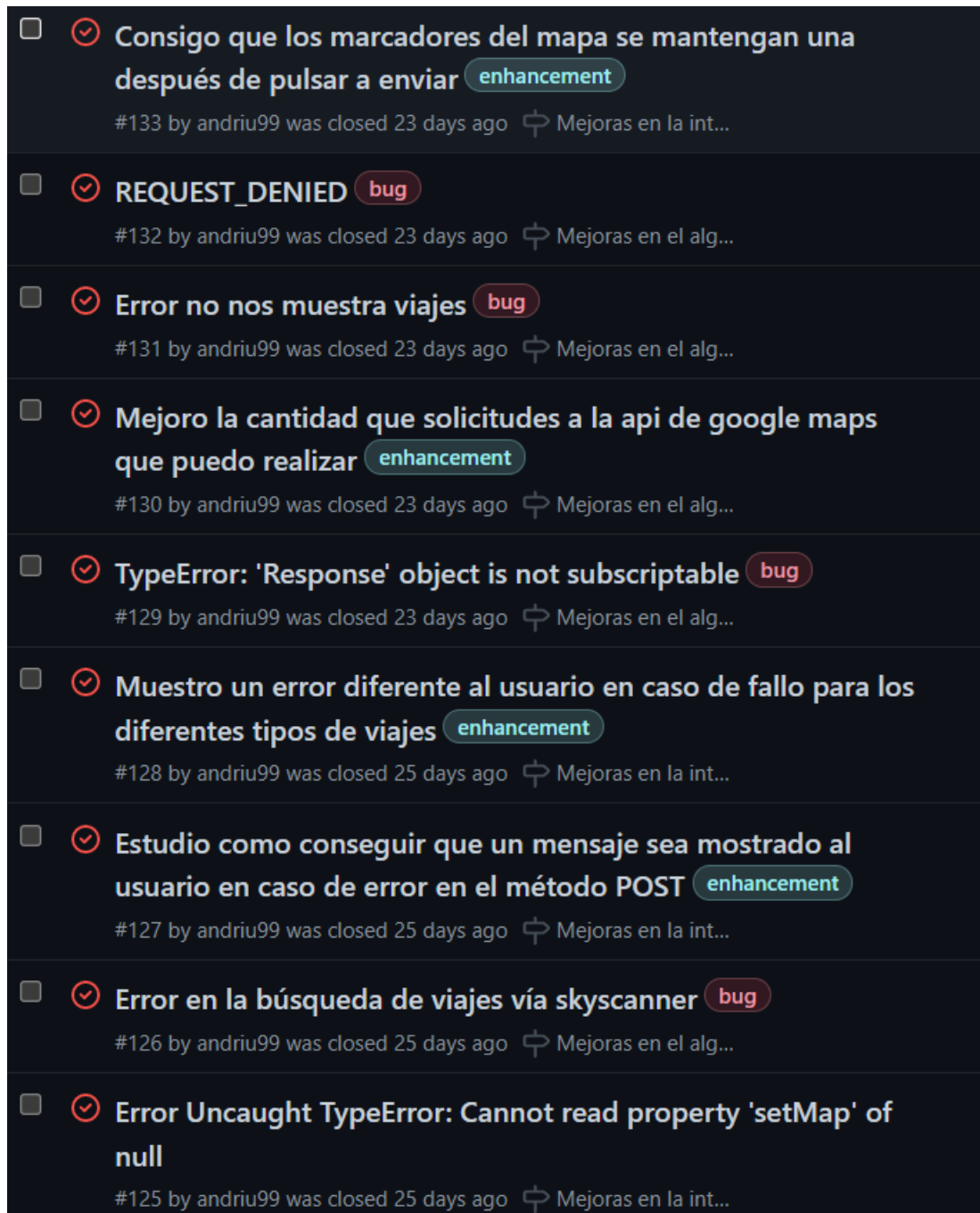


Figura A.17: Issues del sprint 3.

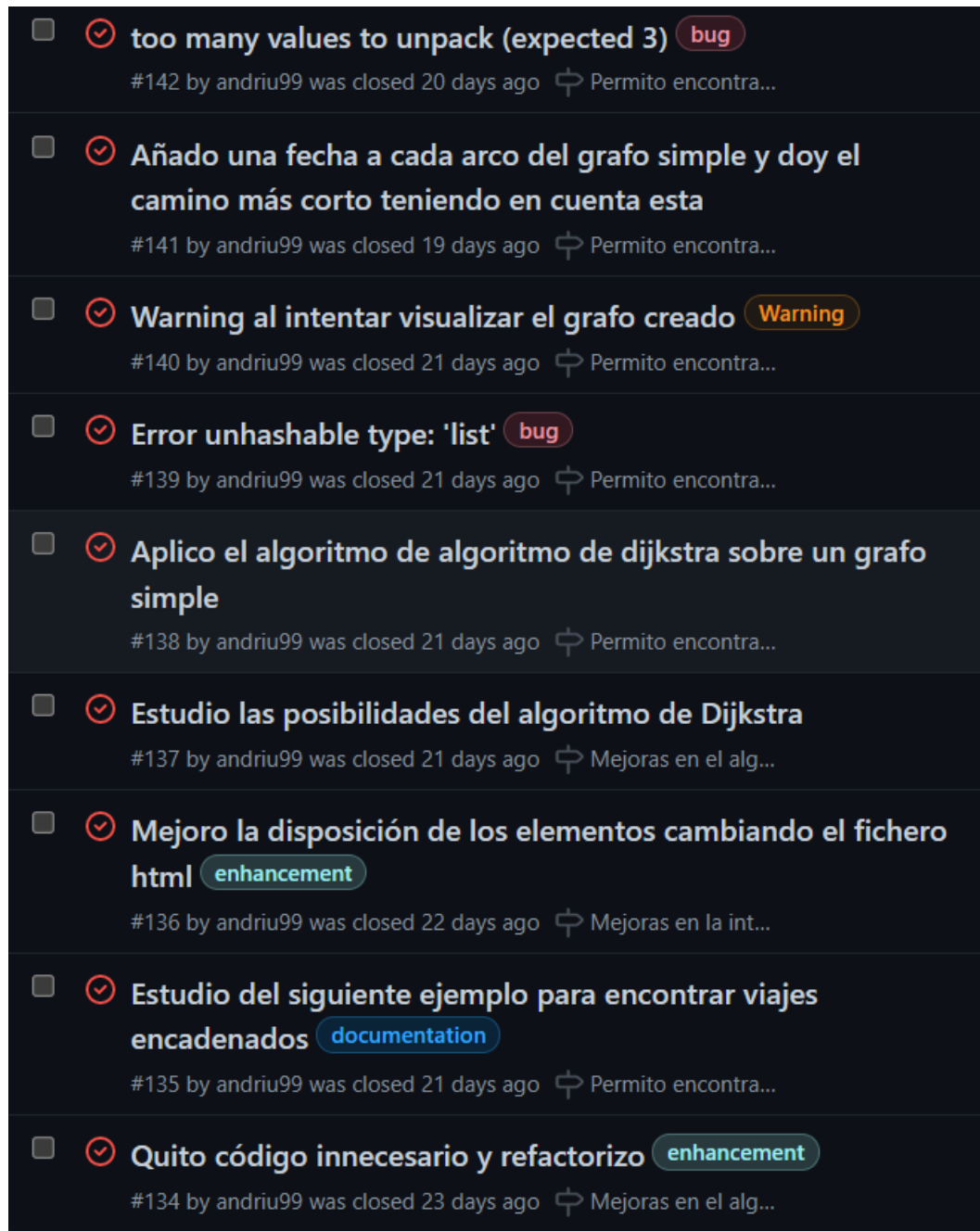


Figura A.18: Issues del sprint 3.

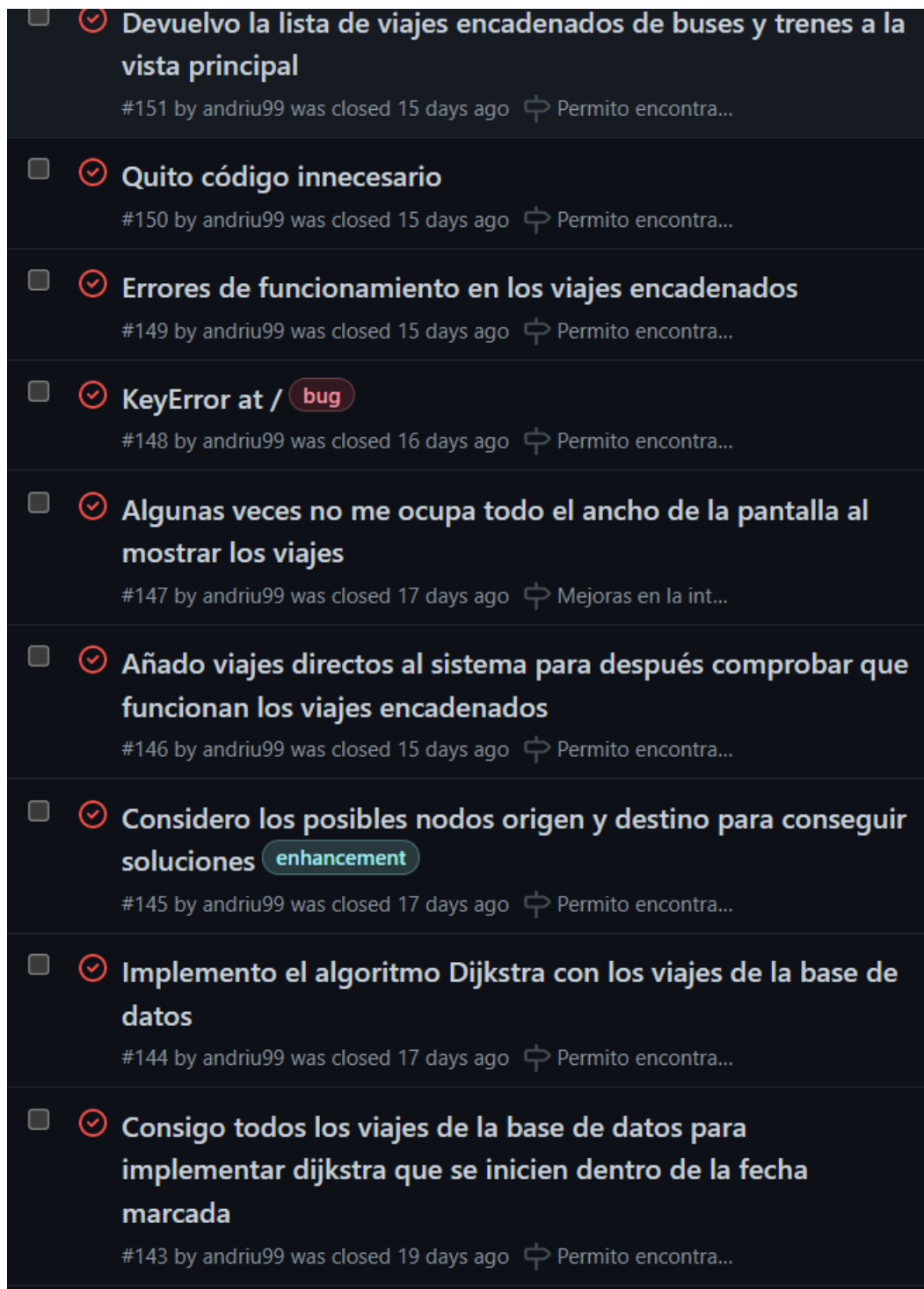


Figura A.19: Issues del sprint 3.

Sprint 4

Duración: 23-6-2021 al 8-7-2021 Tareas realizadas:

- Refactorizaciones.
- Documentar el proyecto.

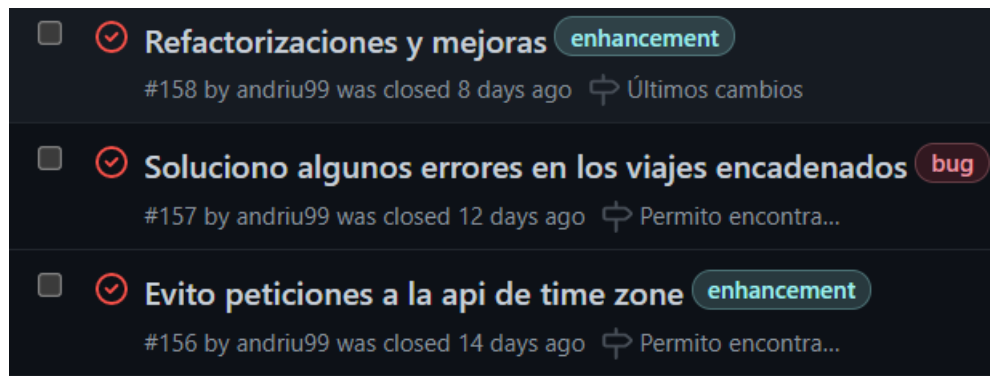


Figura A.20: Issues del sprint 4.

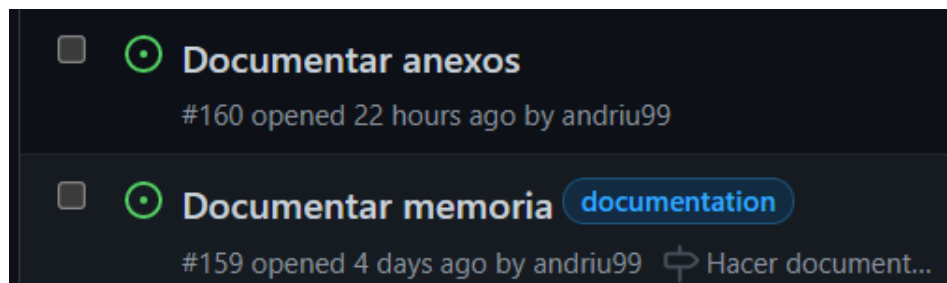


Figura A.21: Issues del sprint 4.

A.3. Estudio de viabilidad

En este apartado se trata de analizar si el proyecto es viable económicamente y legalmente.

Viabilidad económica

Se van a analizar los costos y posibles ingresos del proyecto.

Es importante destacar que se hará desde el punto de vista del empresario que financia el proyecto.

Coste de Recursos Humanos

Habrán tres trabajadores en la empresa. Abajo detallo en una tabla el coste de cada concepto:

Concepto	Cantidad mensual
Salario Bruto	1 610 €
Contingencias comunes (23.6 %)	379.96 €
Desempleo contrato general (5,50 %)	88.5 €
Contrato de duración determinada a tiempo completo (6.7 %)	107.87 €
Fogasa (0.2 %)	32.2 €
Formación Profesional (0.6 %)	96.6 €
Total	2315.13 €

Tabla A.1: Costes relacionados con Recursos Humanos

En total los empleados estarán contratado un total de 3 mensualidades, lo que conlleva un gasto de **6 945.39 €/empleado** (contando pagas extra).

En total gastaríamos **27 000 €** en este apartado.

Costes de hardware

En este apartado se desglosarán los costes de material hardware. Se considerará una amortización en 5 años y un uso de 3 meses.

Concepto	Coste	Coste amortizado
Ordenador Pórtatil	750 €	37.5 €

Tabla A.2: Gasto relacionado con material hardware

El gasto total en costes de hardware será **37.5 €**.

Costes de software

La mayoría de los servicios son gratuitos (API de Trainline, Servicios de Moovit, API Skyscanner y API de Blablacar), a excepción de la API REST de Google Maps que tiene un límite de llamadas gratuitas [4], este equivale a 200 \$ mensuales. Intentaremos hacer una estimación aunque su costo dependerá del uso que se haga de la aplicación. Nos basaremos en nuestra propia experiencia usando la aplicación y en los datos de uso que nos provee Google Cloud de nuestro uso [3] para estimar qué servicios serán más y menos usados.

A continuación se mostrará una tabla con nuestra estimaciones. Cabe resaltar que los precios totales pueden variar por la fluctuación de la moneda, ya que los precios en [4] son en dólares.

Servicio de Google Maps	Número de solicitudes	Precio total
API Maps JavaScript	5000	21 €
Places API	2000	4.75 €
Geocoding API	75 000	314 €
Distance Matrix API	20 000	83€

Tabla A.3: Gasto relacionado con material software mensualmente

Como hemos podido ver el gasto total mensual asciende a 422 €, no obstante como tenemos un crédito de 200 \$, se nos queda entorno a **240 €** mensuales. Un total de **720 €** de gasto total en este apartado en los 3 meses de duración del proyecto.

Cabe destacar que podríamos obtener descuentos comunicandonos con el servicio de ventas de google, no obstante a priori no podemos saber la cuantía de estos.

Costes totales

Concepto	Cantidad
Costes relacionados con Recursos Humanos	6 945.39€
Gasto relacionado con material hardware	37.5 €
Costes de software	720 €

Tabla A.4: Gasto global del proyecto

Los costes totales ascienden a 7 702.89 €.

Posibles vías para obtener ingresos

En este apartado simplemente enumeramos una serie de posibles fuentes de ingreso:

- Incluir mecanismos de visualización de publicidad en la aplicación.
- Podríamos incluir algunos servicios (servicio de búsqueda de viajes con transbordo u otros) como de pago. Deberíamos en ese caso diferenciar entre usuarios corrientes de la aplicación y usuarios que pagan una cuota determinada.

Viabilidad legal

En primer lugar listaremos las dependencias del proyecto con sus respectivas licencias. Hemos utilizado pipreqs [6] para obtener las librerías utilizadas con su versión correspondiente, que se han guardado en un fichero como vemos a continuación:

```
APScheduler==3.7.0
Django==3.2.3
googlemaps==4.4.5
numpy==1.20.1
timezonefinder==5.2.0
requests==2.25.1
pytz==2021.1
```

Figura A.22: Requirements encontrados por pipreqs.

Después hemos procesado este fichero a través del analizador de licencias VersionEye [10].

Dependencia	Versión	Licencia
Django	3.23	BSD-3-Clause y BSD
numpy	1.20.1	BSD
APScheduler	3.7.0	MIT
googlemaps	4.4.5	Apache-2.0
pytz	2021.1	MIT
requests	2.25.1	Apache-2.0
timezonefinder	5.2.0	MIT

Tabla A.5: Licencia de cada dependencia del proyecto

De las licencias del proyecto la más re restrictiva es la licencia Apache-2.0. Esta es compatible con la licencia GLP versión 3 [2], por tanto en este proyecto elegiremos GLP versión 3 como nuestra licencia.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado se expondrán los requisitos de la aplicación detalladamente, a parte de los objetivos generales y los casos de uso.

B.2. Objetivos generales

Los objetivos generales también expuestos en la memoria del proyecto son los siguientes:

- Desarrollar una aplicación que permita realizar búsquedas utilizando diferentes fuentes de datos externas.
- La aplicación será capaz de buscar viajes en diferentes medios de transporte.
- Identificar fuentes de datos externas que sean útiles para la resolución de los problemas abordados por el proyecto.
- El sistema será capaz de sugerir viajes con trasbordos al usuario.
- La aplicación desarrollada sera fácil de usar.
- Reducir al máximo la dependencia de cambios en las fuentes de datos externas.

B.3. Catalogo de requisitos

En esta sección se enumerarán tanto los requisitos funcionales como los no funcionales.

Requisitos funcionales

- **RF-1:** El usuario será capaz de seleccionar puntos de origen y destino, bien escribiendo su dirección o señalando los puntos en el mapa.
- **RF-2:** El sistema mostrará al usuario los viajes directos disponibles.
- **RF-3:** El usuario será capaz de indicar un precio máximo a la hora de mostrar viajes directos.
- **RF-4:** El usuario será capaz de ordenar los viajes de cada tipo de viaje (Blablacar, Skyscanner, bus y tren), ascendentemente o descendentemente, según su precio o duración.
- **RF-5:** El sistema guardará los viajes en bus y en tren. Será capaz de actualizar su contenido una vez al día.
- **RF-6:** El sistema será capaz de sugerir al usuario viajes con transbordos.
- **RF-7:** La aplicación mostrará al usuario un mapa de cómo llegar al punto de partida o al de destino final, partiendo de la ubicación seleccionada.

Requisitos no funcionales

- **RNF-1:** La aplicación será fácilmente utilizable.
- **RNF-2:** No se alterarán los datos de los Nodos (aeropuertos y estaciones) durante la búsqueda de viajes.
- **RNF-3:** El sistema será fácilmente actualizable tras un cambio en la forma de acceso a cualquier fuente de datos.

B.4. Especificación de requisitos

Se mostrarán las plantillas de casos de uso correspondientes.

CU-1	Establecer origen y destino
<hr/>	
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-1
Descripción	Se establecen las coordenadas de origen y de destino.
Precondición	El usuario será capaz de seleccionar en el mapa o de escribir la dirección deseada en un cuadro de texto.
Acciones	<ol style="list-style-type: none">1. El usuario establece un punto de origen y uno de destino señalando en el mapa. Otra opción es escribiendo una dirección, en cuyo caso se traducirá esa dirección a coordenadas.2. Guardamos los datos de latitud y longitud de origen y destino en nuestro formulario.
Postcondición	Los datos de localización están disponibles correctamente.
Excepciones	<ul style="list-style-type: none">■ Se podría dar alguna debida a cambios en la API de Google Maps.
Importancia	Alta

CU-1 Establecer origen y destino.

CU-2	Mostrar los viajes directos en Blablacar
<hr/>	
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-2
Descripción	Se muestran al usuario los viajes directos en Blablacar.
Precondición	Tenemos guardada la información de localización de origen y destino, y el usuario ha elegido un día en el calendario.
Acciones	<ol style="list-style-type: none">1. Hacemos una petición a la API de Blablacar con las coodenadas de origen y destino correspondientes y la fecha introducida por el usuario.2. Guardamos los datos que nos proporciona Blablacar en nuestra base de datos.3. En el caso de haber encontrado viajes: Utilizamos esos datos para mostrarlos al usuario.
Postcondición	Ninguna.
Excepciones	<ul style="list-style-type: none">■ Se podría dar alguna debida a cambios en la API de Blablacar.
Importancia	Media

CU-2 Mostrar los viajes directos en Blablacar.

CU-3		Mostrar los viajes directos en Skyscanner
<hr/>		
Versión	1.0	
Autor	Andrés Rodríguez Rosales	
Requisitos	RF-2	
Descripción	Se muestran al usuario los viajes directos en avión.	
Precondición	Tenemos guardada la información de localización de origen y destino, y el usuario ha elegido un día.	
Acciones	<ol style="list-style-type: none">1. Decidimos los nodos (aeropuertos) que consideramos como origen y los que consideramos como destino. Para ello realizamos estos pasos que valdrán igual para el destino:<ul style="list-style-type: none">■ Consultamos si hay algún aeropuerto en el municipio de origen. Si lo hay consideramos los nodos de ese municipio.■ Si no hay, consideramos los nodos de la provincia.2. Hacemos una petición a la API de Skyscanner con cada par de nodos origen y destino en la fecha marcada.3. Si hubiera viajes, serán mostrados al usuario.	
Postcondición	Ninguna.	
Excepciones	<ul style="list-style-type: none">■ Se podría dar alguna debida a cambios en la API de Skyscanner o por hacer más peticiones de lo que permite la cuota.	
Importancia	Media	

CU-3 Mostrar los viajes directos en Skyscanner.

CU-4	Mostrar los viajes directos en bus y tren
<hr/>	
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-2
Descripción	Se muestran al usuario los viajes directos en bus y en tren.
Precondición	Tenemos guardada la información de localización de origen y destino, y el usuario ha elegido un día en el calendario.
Acciones	<ol style="list-style-type: none">1. En primer lugar, decidimos los nodos (estaciones) que consideramos como origen y los que consideramos como destino. Para ello realizamos estos pasos que valdrán igual para el destino:<ul style="list-style-type: none">■ Consultamos si hay alguna estación en el municipio de origen. Si lo hay consideramos todos los nodos de ese municipio.■ Si no hay, consideramos los nodos de la provincia. De todos ellos sólo tendremos en cuenta los 10 más cercanos, es decir, los 10 cuyo tiempo para llegar desde el municipio de origen sea mínimo.2. Hacemos una petición a la API de Trainline con cada par de nodos origen y destino en la fecha marcada.3. Si hubiera viajes, serán mostrados al usuario separando viajes en bus de viajes en tren.
Postcondición	Ninguna.

Excepciones

- Se podría dar alguna debida a cambios en la API de Trainline o por hacer más peticiones de lo que permite la cuota.

Importancia Media

CU-4 Mostrar los viajes directos en bus y tren.

CU-5 Ordenación y filtrado de los datos	
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-3, RF-4
Descripción	Se muestran al usuario los viajes en avión.
Precondición	Tenemos guardada la información de viajes que vamos a mostrar en la base de datos.
Acciones	<ol style="list-style-type: none">1. Si el usuario ha decidido un precio máximo, los datos de viajes con precio mayor a ese no serán mostrados.2. Si ha elegido ordenación ascendente, ordenaremos los diferentes tipos de viajes de esa forma. Lo mismo sucedería si ha elegido ordenación descendente
Postcondición	Ninguna.
Excepciones	<ul style="list-style-type: none">■ Ninguna.
Importancia	Media

CU-5 Ordenación y filtrado de los datos.

CU-6	Actualización de datos de viajes en bus y en tren
<hr/>	
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-5
Descripción	Los viajes en bus y en tren son guardados permanentemente en la base de datos, y utilizados más tarde si fuera necesario.
Precondición	Tenemos viajes en bus y en tren guardados en la base de datos.
Acciones	<ol style="list-style-type: none">1. Se ejecutará cada 24 horas un script en segundo plano que hará lo siguiente:<ul style="list-style-type: none">■ Borrará cada viaje cuya fecha de inicio sea anterior a la actual.■ El resto de viajes, serán utilizados. Se ejecutará una consulta a la API de Trainline entre cada par de nodos encontrados en las fechas de las que se tenga registro, y se guardará esa información en la base de datos. Posteriormente los viajes utilizados para buscar nuevos viajes se borrarán.
Postcondición	Ninguna.
Excepciones	<ul style="list-style-type: none">■ Las que pueda devolver la API de Trainline.
Importancia	Media

CU-6 Actualización de datos de viajes en bus y en tren.

CU-7	Búsqueda de viajes con transbordos
<hr/>	
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-6
Descripción	El sistema permite encontrar viajes con transbordos en bus y en tren.
Precondición	Tenemos viajes en bus y en tren guardados en la base de datos.
Acciones	<ol style="list-style-type: none">1. Se ejecutará el algoritmo un algoritmo de camino mínimo entre los nodos que tengan viajes guardados en nuestra Base de datos en el día indicado por el usuario.2. Cada vez que se decida que una solución es mejor según su precio, se evaluará también si es posible en cuestión de fechas en caso de no serlo no será considerada como una solución mejor.
Postcondición	Ninguna.
Excepciones	<ul style="list-style-type: none">■ Ninguna.
Importancia	Media

CU-7 Búsqueda de viajes con transbordos.

CU-8	Mostrar un mapa de transporte público
Versión	1.0
Autor	Andrés Rodríguez Rosales
Requisitos	RF-7
Descripción	Se mostrarán las opciones de cómo llegar al punto de comienzo del viaje o al punto de finalización de este.
Precondición	Ninguna.
Acciones	<ol style="list-style-type: none">1. Según el usuario haya hecho click en la estación/aeropuerto origen o destino, se mostrará el mapa de cómo llegar hasta ella desde el punto de origen/destino.
Postcondición	Ninguna.
Excepciones	<ul style="list-style-type: none">■ Ninguna.
Importancia	Media

CU-8 Mostrar un mapa de transporte público.

A continuación mostramos el diagrama de casos de uso:

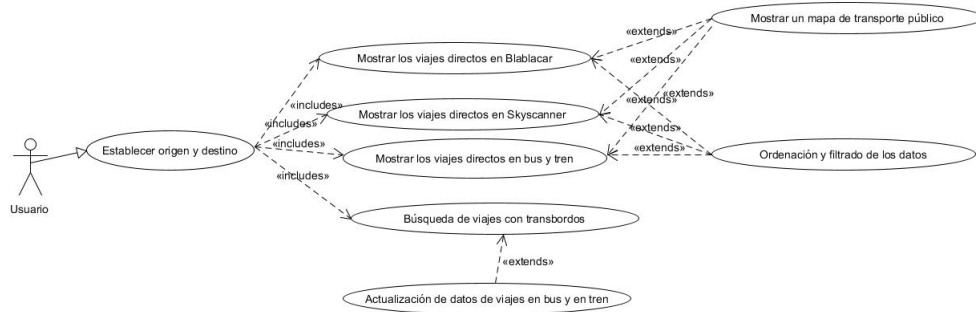


Figura B.1: Diagrama de casos de uso

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se hablará acerca de la forma en la que guardamos los datos en nuestra aplicación.

C.2. Diseño de datos

Aquí podemos ver un diagrama de clases de cómo están relacionados los datos en la base de datos.

Podemos apreciar las siguientes clases:

- RESTApi
Guarda la información de las API-REST usadas.
- Request
Guarda la información necesaria para realizar cada petición.
- Node
Guarda la información de cada nodo.
- Trip
Guarda la información de cada viaje.
- blablaTrip
Guarda la información específica de los viajes en Blablacar.

- `skyscannerTrip`
Guarda la información específica de los viajes en avión.
- `busOrTrainTrip`
Guarda la información específica de los viajes en bus o en tren.

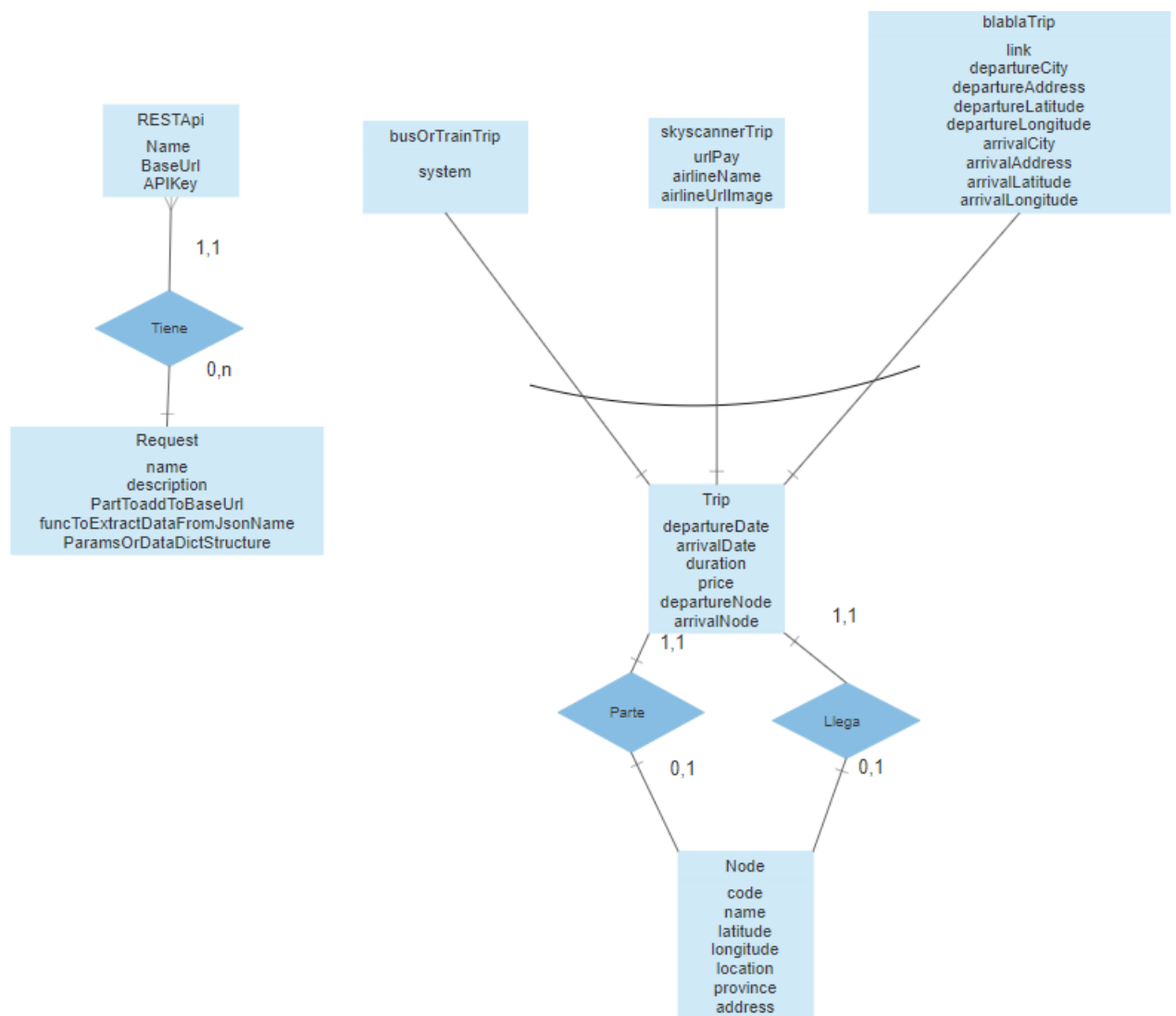


Figura C.1: Diagrama E-R de las tablas utilizadas.

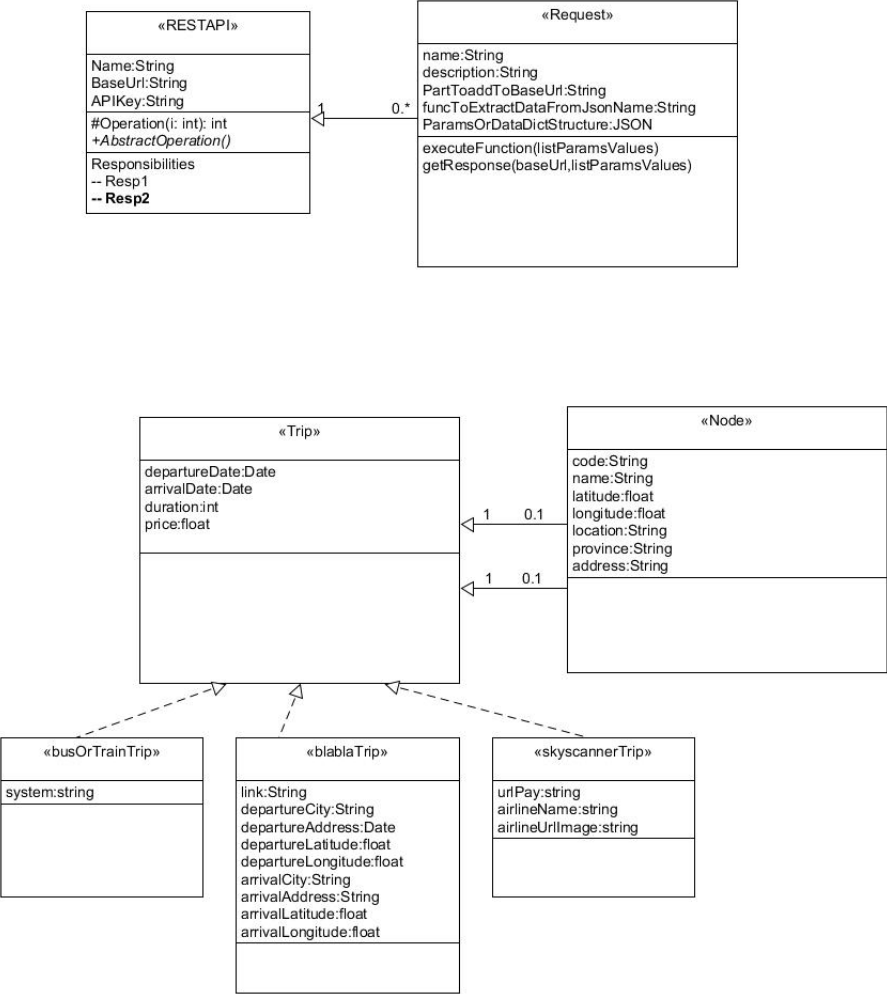


Figura C.2: Diagrama relacional de las tablas utilizadas.

C.3. Diseño procedimental

A continuación mostraremos el diagrama de secuencias que nos indica la forma en la cual la aplicación funciona.

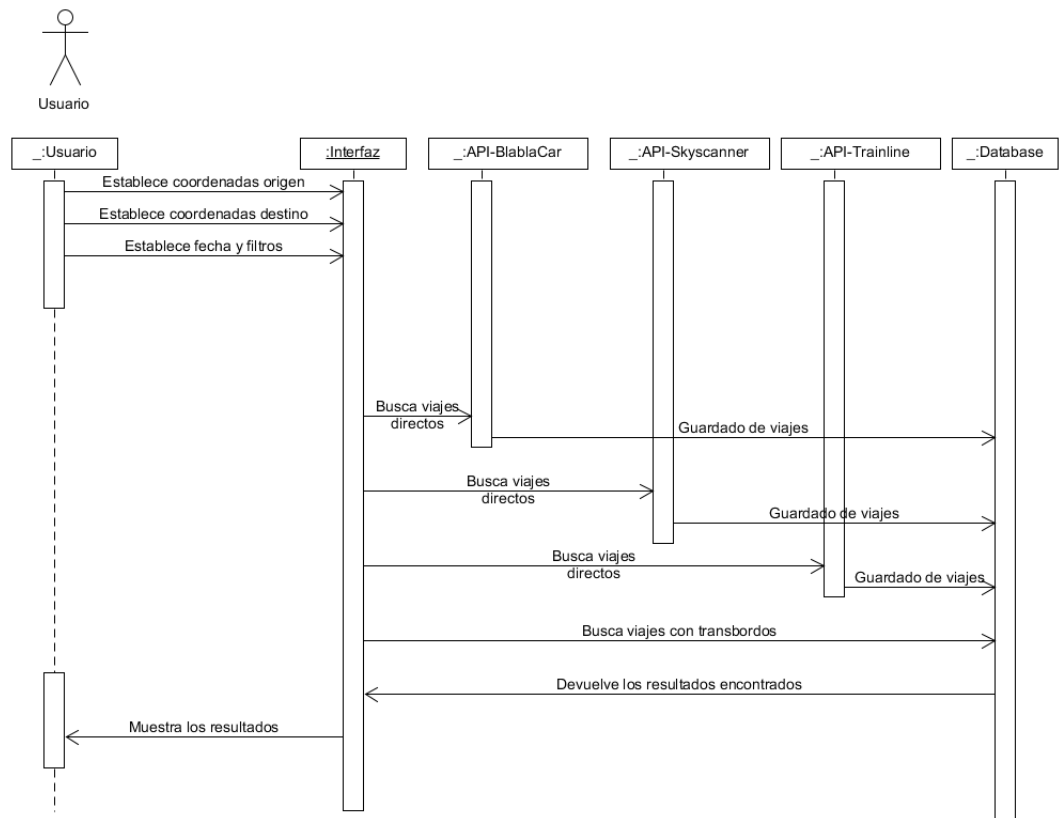


Figura C.3: Diagrama de secuencias para la búsqueda de viajes.

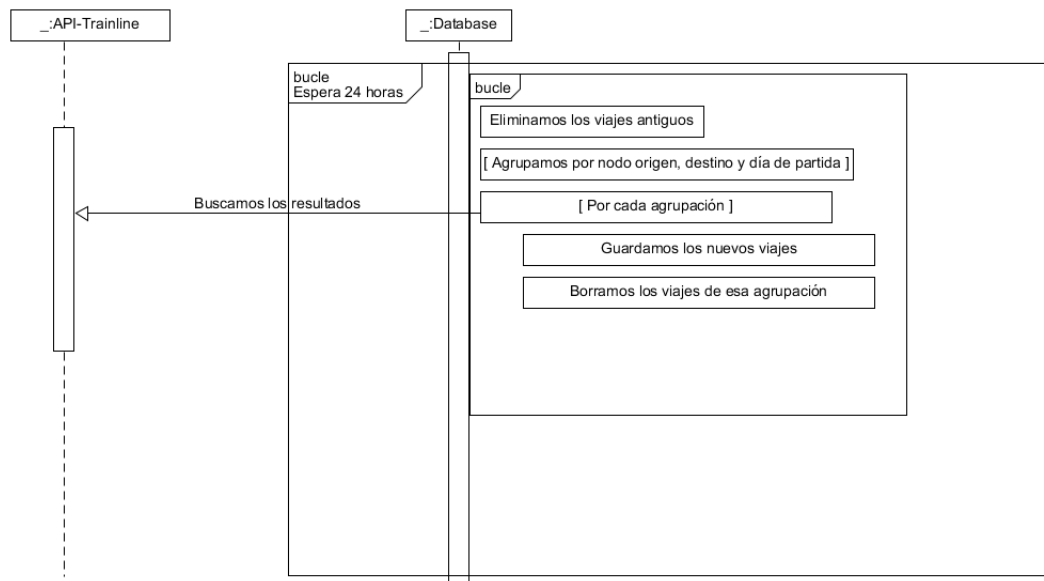


Figura C.4: Diagrama de casos de uso para la búsqueda de viajes.

C.4. Diseño arquitectónico

El framework utilizado (Django [1]) utiliza el patrón de arquitectura de software conocido como Modelo-Vista-Controlador. Básicamente trata de separar los datos del resto del sistema ,es decir, separa el guardado de la información de la interacción con el usuario [8].

Según [5] se basa en tres elementos interconectados:

■ Controlador

Recibe la solicitud de la API o del usuario. Maneja el flujo de solicitudes. Es importante recalcar que nunca modifica o gestiona la lógica de los datos. Podemos decir que hace de intermediario entre el modelo y la vista.

■ Modelo

Se encarga de interactuar con la base de datos para borrar, añadir o modificar elementos. Recibe y envía datos directamente al controlador.

■ Vista

Maneja la presentación de los datos al usuario. Podemos destacar que para ello recibe datos del controlador y que se renderiza de manera dinámica.

En concreto en Django utiliza diferentes archivos para el implementar MVC [9]. El fichero **models.py** contiene una descripción de cada tabla de la base de datos como un fichero Python. El fichero **views.py** contiene la lógica de la página y las vistas disponibles. También tenemos ficheros que nos indican que vista es llamada en concreto según la URL (**urls.py**).

Además del diseño arquitectónico de Django, tenemos un diseño arquitectónico propio de llamadas a las APIs. A continuación, podemos ver un diagrama de despliegue de cómo accedemos a ellas:

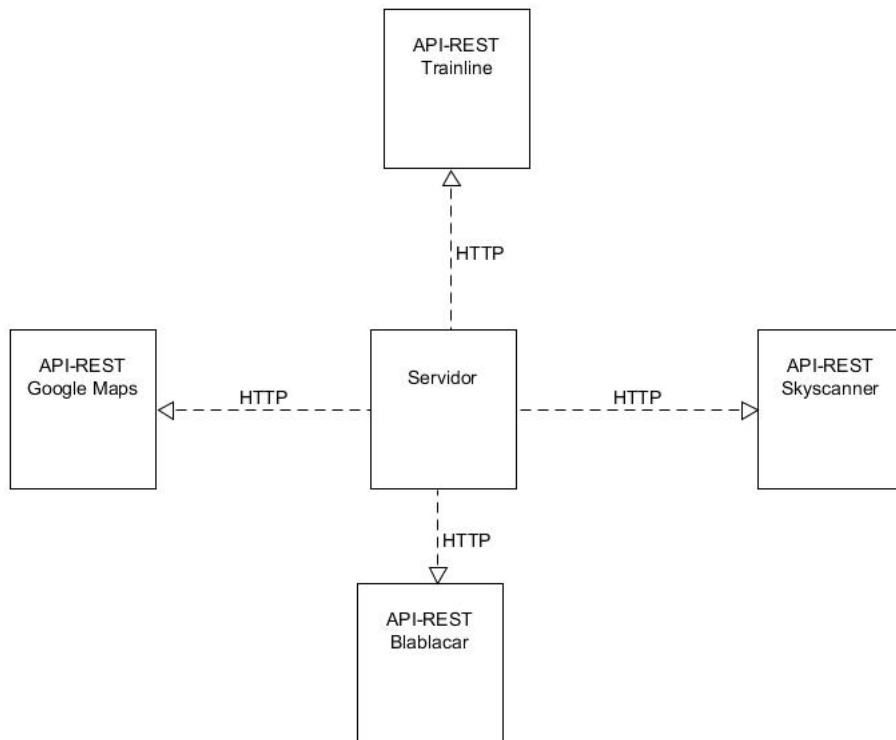


Figura C.5: Diagrama de despliegue.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado se va a hablar acerca de:

- Estructura de directorios de la aplicación
- Manual del programador
- Forma en la cuál debemos instalar el proyecto

D.2. Estructura de directorios

A continuación, hablaremos acerca de la estructura de directorios del proyecto. Vamos a explicar cada carpeta dentro de la carpeta `project`, ya que fuera sólo nos es útil el fichero `requirements` que nos servirá para instalar las librerías necesarias.

- **Elementos que no están dentro de ningún subdirectorio**

Podemos ver la base de datos `db.sqlite3`. También vemos `manage.py`, este nos es útil para tareas en las que nos ayuda Django. Nos proporciona las herramientas necesarias para probar la aplicación, así como una consola desde la que podremos manipular la base de datos.

- **Elementos dentro de la carpeta scripts**

Son scripts que nos permiten inicializar la base de datos, con los datos de estaciones y aeropuertos. Además tenemos el fichero `stationsData.txt`, que nos provee de datos de estaciones de buses y trenes.

- **Elementos dentro del subdirectorio project**

Destaca el fichero de configuración **`settings.py`**, y la parte donde definimos las URLs generales de nuestro proyecto **`urls.py`**.

- **Elementos dentro del subdirectorio app**

- **Ficheros sin carpeta**

Podemos destacar `admin.py` (nos sirve para incluir en el panel de administración de bases de datos las tablas que deseemos), `apps.py` (definimos los scripts en segundo plano), `forms.py` (formularios Django), `urls.py` (urls concretas de nuestra aplicación), `views.py` (definimos qué hacen nuestras vistas).

- **`funtionsRequest`**

Define para cada API-REST utilizada un fichero, en el cual establecemos la forma en la que extraemos información del JSON devuelto de cada petición.

- **`migrations`**

Es donde el framework Django guarda las migraciones hechas a la base de datos.

- **`otherFunctions`**

Funciones de nodos y fechas. Nos ayudan en el desarrollo de algoritmos de búsqueda de viajes.

- **`scheduler`**

Desde aquí definimos lo que hará nuestro script en segundo plano.

- **`static`**

Ficheros Javascript que nos sirven para establecer un mapa en la aplicación, y ficheros CSS que nos sirven para dar estilo a la misma.

- **`templates`**

Definen en HTML la estructura de las diferentes páginas de nuestra aplicación.

- **`viewFunctions`**

Define funciones que ayudarán a la vista principal a realizar sus labores.

```
C:.\n  memoria-tfg.zip\n  requirements.txt\n  tree.txt\n\nproject\n  db.sqlite3\n  manage.py\n\n  app\n    admin.py\n    apps.py\n    forms.py\n    models.py\n    tests.py\n    urls.py\n    views.py\n\n  funtionsRequest\n    airportsRequests.py\n    blablacarRequests.py\n    bustrainRequests.py\n    googleMapsRequests.py\n    __init__.py
```

Figura D.1: Árbol de directorios del proyecto (I).

```
migrations
    0001_initial.py
    __init__.py

    __pycache__
        0001_initial.cpython-39.pyc
        __init__.cpython-39.pyc

otherFunctions
    dateFunctions.py
    nodesFunctions.py

    __pycache__
        dateFunctions.cpython-39.pyc
        nodesFunctions.cpython-39.pyc

scheduler
    scheduler.py
    __init__.py

    __pycache__
        scheduler.cpython-39.pyc
        __init__.cpython-39.pyc

static
    google_maps.js
    moovit_script.js
    style.css

templates
    app
        home.html
        layout.html
        new.html

viewFunctions
    homeviewFunctions.py

    __pycache__
        homeviewFunctions.cpython-39.pyc
```

Figura D.2: Árbol de directorios del proyecto (II).

```
viewFunctions
    homeviewFunctions.py

__pycache__
    homeviewFunctions.cpython-39.pyc

__pycache__
    admin.cpython-39.pyc
    apps.cpython-39.pyc
    forms.cpython-39.pyc
    models.cpython-39.pyc
    urls.cpython-39.pyc
    views.cpython-39.pyc

project
    asgi.py
    settings.py
    urls.py
    wsgi.py
    __init__.py

__pycache__
    settings.cpython-39.pyc
    urls.cpython-39.pyc
    wsgi.cpython-39.pyc
    __init__.cpython-39.pyc

scripts
    loadAirportsData.py
    loadAllData.py
    loadRequestsInfo.py
    loadStationsData.py
    stationsData.txt
    __init__.py
```

Figura D.3: Árbol de directorios del proyecto (III).

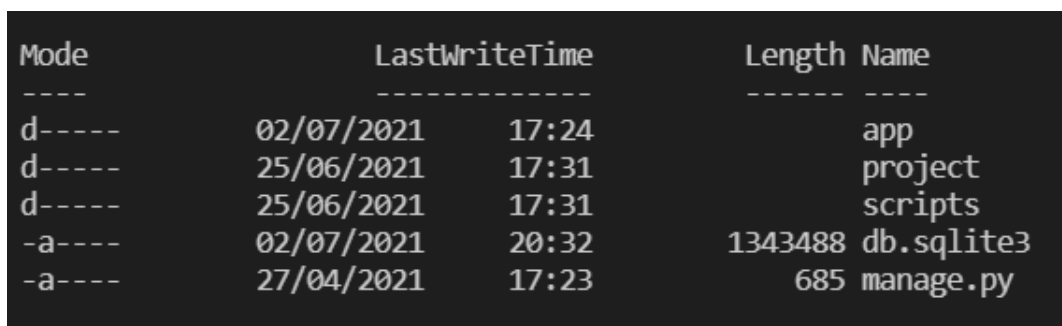
D.3. Manual del programador

En esta sección se van a explicar aquellas cuestiones que es conveniente que un nuevo desarrollador del proyecto conozca.

Ejecutar scripts de inicialización de datos

Lo primero de todo, debemos estar en nuestra consola de comandos en la carpeta project, la cual si listamos podemos ver lo siguiente:

- Carpeta app
- Carpeta project
- Carpeta scripts
- Base de datos
- Fichero manage.py



Mode	LastWriteTime		Length	Name
----	-----		-----	----
d-----	02/07/2021	17:24		app
d-----	25/06/2021	17:31		project
d-----	25/06/2021	17:31		scripts
-a----	02/07/2021	20:32	1343488	db.sqlite3
-a----	27/04/2021	17:23	685	manage.py

Figura D.4: Listado de la carpeta project.

A continuación nos fijamos en el contenido de la carpeta scripts:

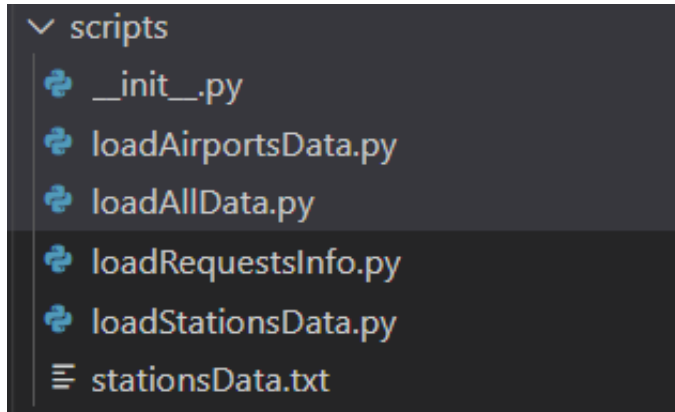


Figura D.5: Listado de la carpeta scripts.

Todos los ficheros, a excepción de `__init__.py` son scripts ejecutables. Cada uno carga un tipo de datos iniciales (peticiones, aeropuertos y estaciones). No obstante, el fichero `loadAllData` ejecuta los otros 3 scripts ejecutables simplificando así este proceso.

Para ejecutar un script basta con ejecutar el comando:

```
python manage.py runscript script_name
```

Crear usuario para el panel de administración de Django

Primero debemos estar en la carpeta `project`, tal como hablabamos en [D.3.](#)

Para acceder al panel de administración de Django debemos crear un superusuario, para ello ejecutamos lo siguiente:

```
PS C:\Users\ANDRES\Desktop\GlobalcodigoTfg\project> python manage.py createsuperuser
Username: example
Email address: hola@example.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]:
```

Figura D.6: Creando superusuario en Django.

Para la creación del superusuario se ha usado el comando:

```
python manage.py createsuperuser
```

Ejecutar la aplicación

Ejecutamos el siguiente comando:

```
python manage.py runserver
```

Para probar el funcionamiento de un script en segundo plano se aconseja añadir al comando anterior `–reload`, para no ejecutarlo dos veces seguidas a la vez.

```
PS C:\Users\ANDRES\Desktop\GlobalcodigoTfg\project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 03, 2021 - 18:29:19
Django version 3.2.3, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figura D.7: Creando superusuario en Django.

Acceder al panel de administración de bases de datos de Django

Primero, se ha de ejecutar la aplicación como veíamos en la subsección anterior (D.3).

Después, abrimos un navegador y escribimos la URL:

`http://127.0.0.1:8000/admin/`

Posteriormente ponemos nuestro usuario y nuestra contraseña, y podremos ver el panel:

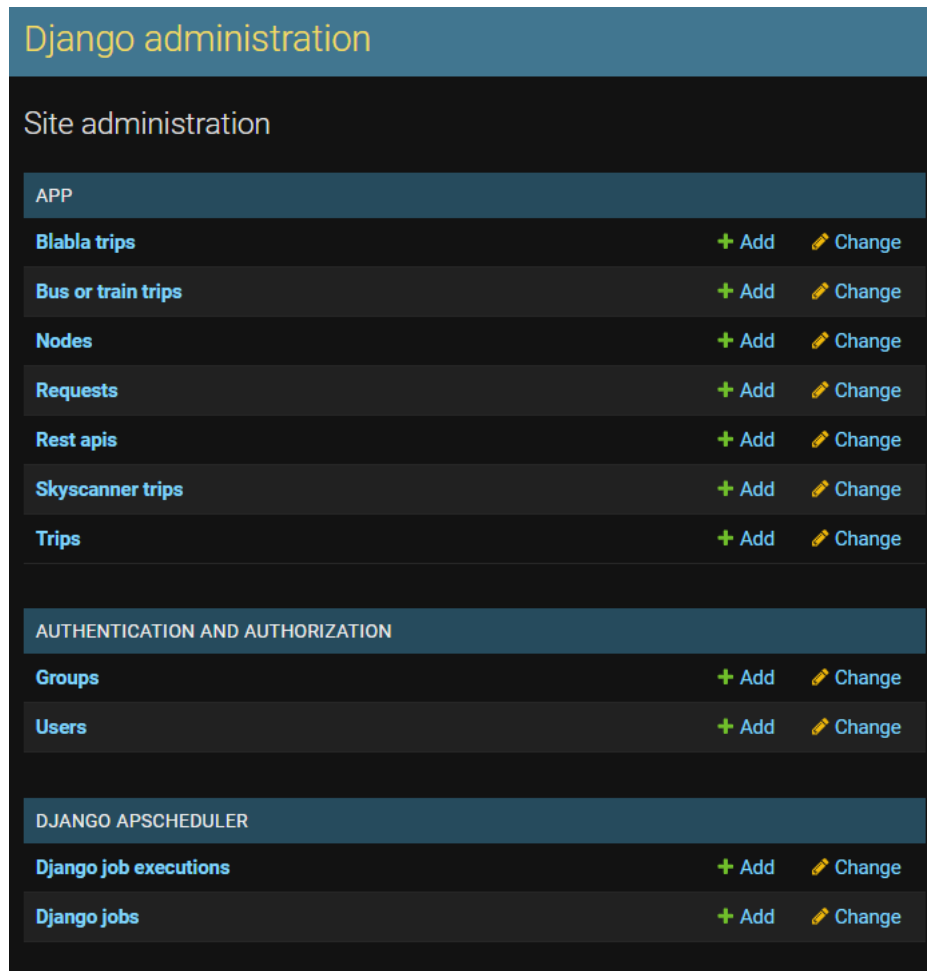


Figura D.8: Creando superusuario en Django.

Desde aquí podremos fácilmente borrar y añadir viajes, nodos, API-REST y peticiones.

Consola de comandos de Django

Nos sirve para hacer operaciones sobre la base de datos como consultar, borrar y añadir información.

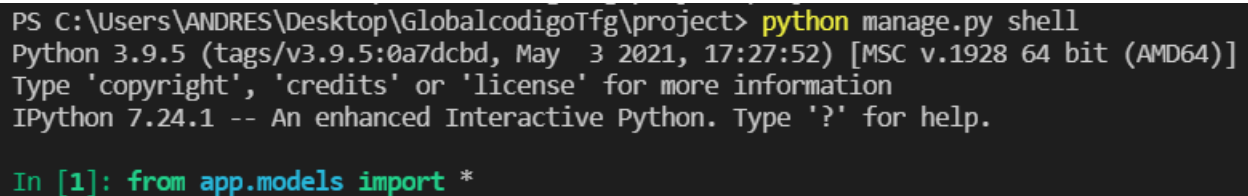
Debemos ejecutar el comando:

```
python manage.py shell
```

Después, debemos importar nuestros modelos (tablas):

```
from app.models import *
```

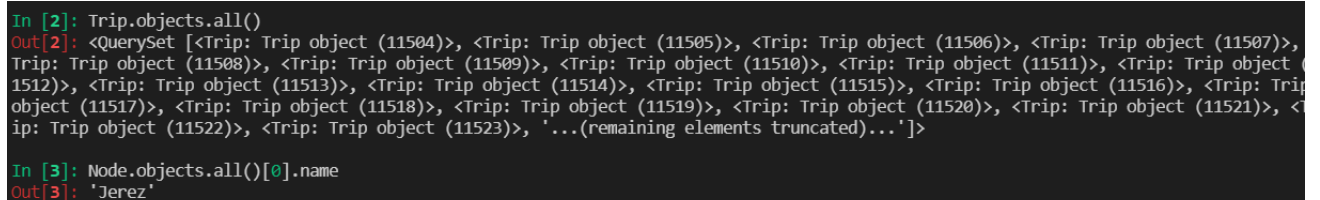
A continuación vemos algunos ejemplos de su uso:



```
PS C:\Users\ANDRES\Desktop\GlobalcodigoTfg\project> python manage.py shell
Python 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.24.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from app.models import *
```

Figura D.9: Ejecución de shell en Django.



```
In [2]: Trip.objects.all()
Out[2]: <QuerySet [
<Trip: Trip object (11504)>, <Trip: Trip object (11505)>, <Trip: Trip object (11506)>, <Trip: Trip object (11507)>,
<Trip: Trip object (11508)>, <Trip: Trip object (11509)>, <Trip: Trip object (11510)>, <Trip: Trip object (11511)>, <Trip: Trip object (11512)>,
<Trip: Trip object (11513)>, <Trip: Trip object (11514)>, <Trip: Trip object (11515)>, <Trip: Trip object (11516)>, <Trip: Trip object (11517)>,
<Trip: Trip object (11518)>, <Trip: Trip object (11519)>, <Trip: Trip object (11520)>, <Trip: Trip object (11521)>, <Trip: Trip object (11522)>,
<Trip: Trip object (11523)>, '...(remaining elements truncated)...']>

In [3]: Node.objects.all()[0].name
Out[3]: 'Jerez'
```

Figura D.10: Ejemplo de consultas vía shell en Django.

D.4. Compilación, instalación y ejecución del proyecto

En este apartado se va a hablar acerca de todo lo necesario para que un desarrollador nuevo pruebe la aplicación.

En primer lugar debemos de tener las siguientes librerías instaladas (a parte de Python 3.9.5):

- APScheduler (versión 3.7.0)
- Django (versión 3.2.3)
- googlemaps (versión 4.4.5)
- numpy (versión 1.20.1)

- timezonefinder (versión 5.2.0)
- requests (versión 2.25.1)
- pytz (versión 2021.1)

Para facilitar esto, podemos utilizar un fichero llamado requirements.txt fuera de project.

Nos situamos donde está el fichero requirements.txt:

```
PS C:\Users\ANDRES\Desktop\GlobalcodigoTfg> dir

Directorio: C:\Users\ANDRES\Desktop\GlobalcodigoTfg

Mode                LastWriteTime         Length Name
----                -
d-----          03/07/2021    17:12             project
-a-----          02/07/2021    17:53      207888 memoria-tfg.zip
-a-----          30/06/2021     1:56         124 requirements.txt
-a-----          03/07/2021    17:12         6022 tree.txt
```

Figura D.11: Ejemplo de consultas vía shell en Django.

Allí ejecutamos:

pip install -r requirements.txt

Evidentemente, al ser una aplicación web también se requerirá de conexión a internet.

D.5. Pruebas del sistema

En nuestro caso, solamente hemos realizado pruebas manuales. A continuación vamos a enumerar aquellas que hemos realizado:

- **Pruebas del mapa interactivo**

Hemos probado correctamente la selección de diferentes ubicaciones en el mapa interactivo de la aplicación. También hemos probado que cuando elegimos un nuevo origen (o destino), quitamos el marcador anterior del mapa.

- **Pruebas de búsqueda de viajes directos en tren**

Se ha probado la búsqueda de viajes directos entre ciudades (Burgos-Madrid, Burgos-Barcelona, Madrid-Barcelona, etc), así como la búsqueda de viajes desde una ubicación cuyo municipio asociado no tenía un municipio asociado.

- **Pruebas de búsqueda de viajes directos en bus**

Se han hecho las mismas pruebas que para los viajes directos en trenes.

- **Pruebas de búsqueda de viajes directos en avión**

Se han probado viajes entre Madrid y Barcelona en avión.

- **Pruebas de búsqueda de viajes directos en Blablacar**

Se han probado búsquedas entre Burgos y otras ciudades como Logroño o Madrid.

- **Pruebas de viajes con transbordos**

Hemos realizado búsquedas desde la ciudad de Burgos a la ciudad de Madrid. Posteriormente, hemos buscado viajes desde la ciudad de Madrid a ciudades como Córdoba o Albacete. Para probar que los viajes con transbordo se realizan correctamente, buscamos después viajes entre Burgos y Albacete o Córdoba.

- **Pruebas de adición de API-REST**

Se ha probado la adición de nuevas API-REST, tanto desde un script externo como desde el panel de administración de Django.

- **Pruebas de borrado de API-REST**

Se ha probado que el borrado de una API-REST supone el borrado de las peticiones asociadas a ella. Se ha realizado desde el panel de administración de Django.

- **Pruebas de modificación de API-REST**

Se ha probado la modificación de la API-Key de una API-REST. Se ha realizado desde el panel de administración de Django.

- **Pruebas de adición de Request**

Se ha probado la adición de nuevas peticiones, tanto desde un script externo como desde el panel de administración de Django.

- **Pruebas de borrado de API-REST**

Se ha probado que el borrado de una Request no supone el borrado de la API-REST ni imposibilita la realización de otras peticiones. Se ha realizado desde el panel de administración de Django.

- **Pruebas de ejecución del script en segundo plano.**

Se ha probado a ejecutar el script cada 5 minutos, en vez de cada 24 horas, para así probarlo con mayor celeridad. Se ha comprobado que se borran los viajes anteriores a la fecha actual, y que se añadían los nuevos viajes

Apéndice E

Documentación de usuario

E.1. Introducción

En este apéndice se hablará acerca del uso básico de la aplicación. Además, se explicarán los requisitos que los usuarios deben cumplir.

E.2. Requisitos de usuarios

Los requisitos que todo usuario debe cumplir son:

- Disponer de conexión a Internet.
- Disponer de un navegador web compatible con HTML5 [\[11\]](#).

E.3. Manual del usuario

Esta sección va a tratar acerca de cómo el usuario debe interactuar con la aplicación.

Elección de origen y destino

La elección del punto de origen y de destino se realiza de forma similar.

A continuación podemos observar la pantalla inicial de la aplicación:

Para marcar el origen pulsando en el mapa seguimos los siguientes pasos:

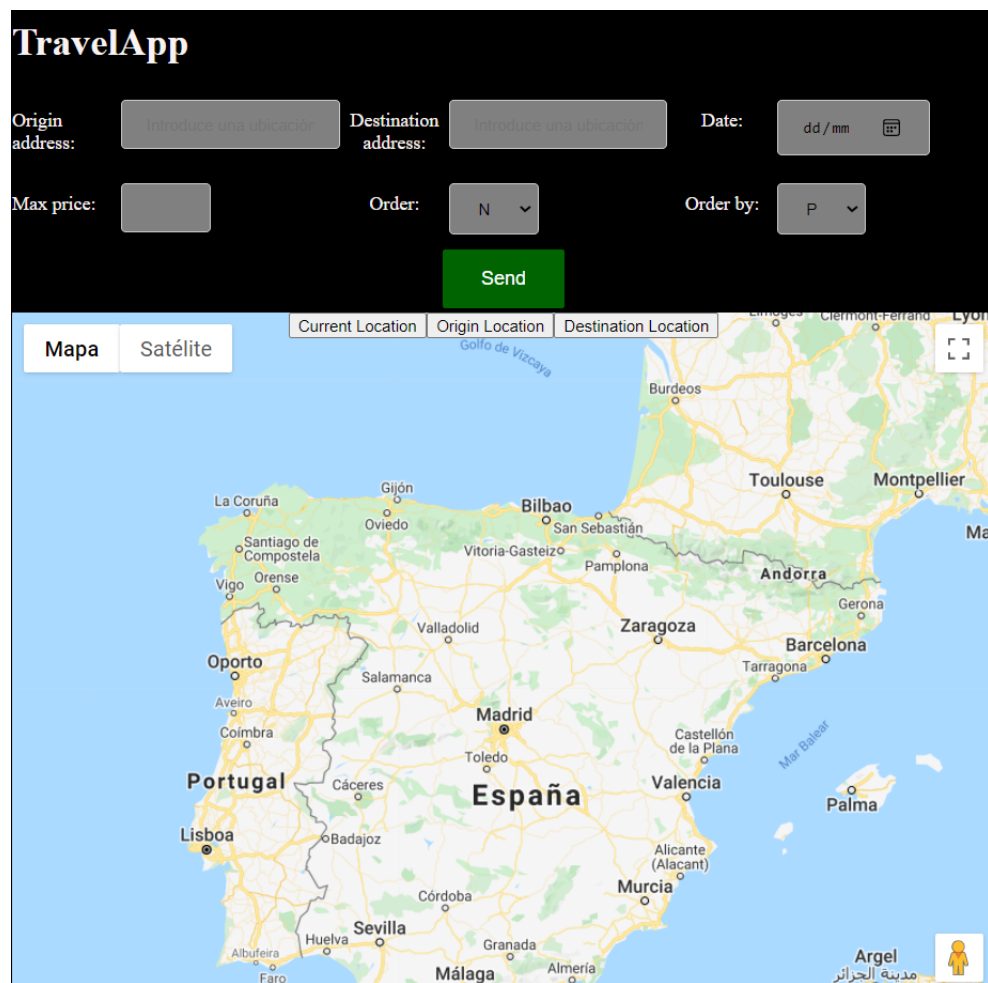


Figura E.1: Pantalla inicial de la aplicación.

- Marcamos el botón Origin Location

En la siguiente imagen podemos ver los botones que hemos añadido al mapa:

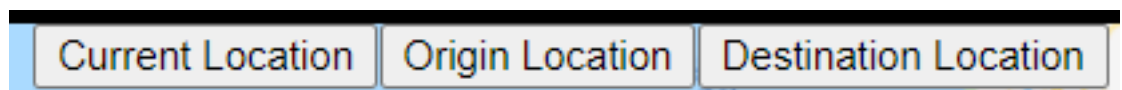


Figura E.2: Botones agregados al mapa.

A continuación, marcamos el botón Origin Location:

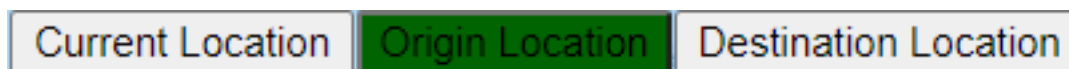


Figura E.3: Boton Origin Location seleccionado.

- **Seleccionamos el punto que queremos en el mapa**

Para esta acción, sólo debemos hacer click en el mapa. A continuación vemos un ejemplo de esto:

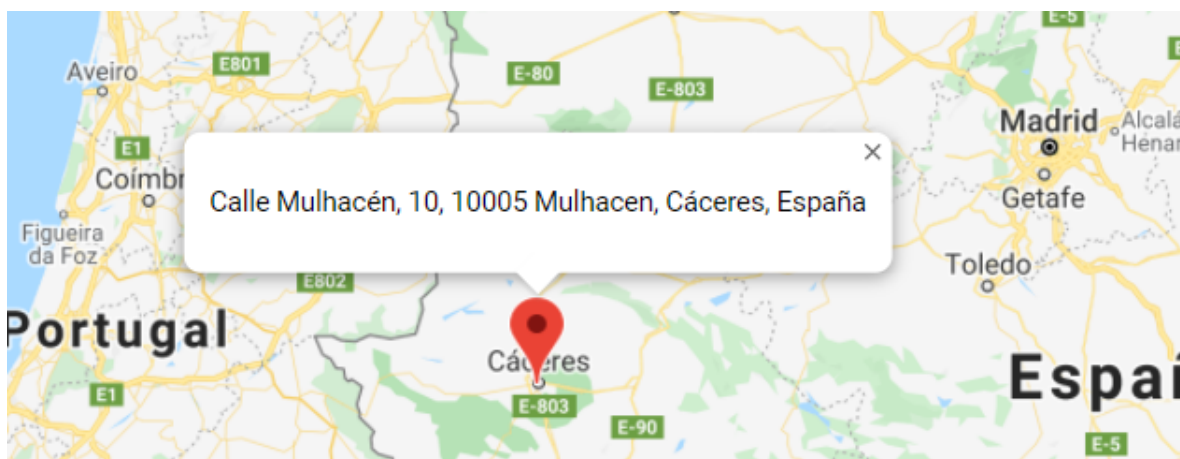


Figura E.4: Origen seleccionado en el mapa.

Como hemos observado en [E.6](#) vemos la dirección en el marcador. Aparte de esto, en la casilla de Origin address (que veíamos en [E.1](#)) también se ha colocado automáticamente esta dirección:

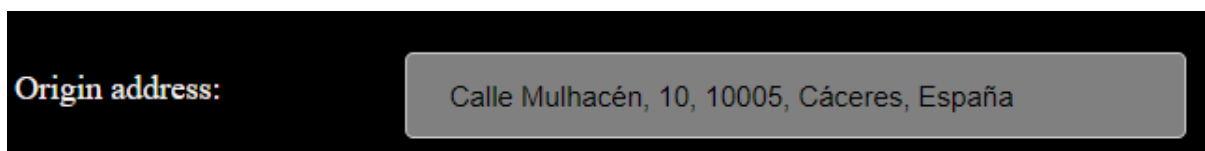


Figura E.5: Origen seleccionado en el mapa.

- **Desmarcamos el botón Origin Location**

Al desmarcarlo, al pulsar en el mapa ya no se nos establecerá el origen. Se quitará el color verde. La estética de los botones volverá a ser igual que en E.2.

Una vez desmarcado, podremos marcar el botón Destination Location (no nos dejará tener dos botones marcados a la vez) y seleccionar el destino de la misma forma.

Tenemos otra forma de seleccionar el origen que no tenemos para el destino. Esta se hace a través de la ubicación que nos da nuestro navegador. En E.2, pudimos ver la existencia de un botón con el texto Current Location. Marcándolo nos establecé un marcador en el mapa con nuestra ubicación actual, y rellena la calle de origen como nuestra ubicación.

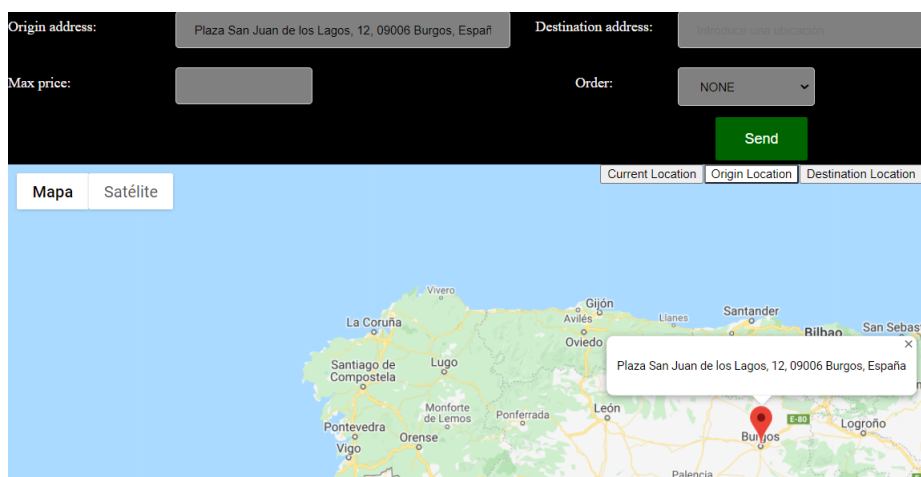


Figura E.6: Origen seleccionado en el mapa.

Ahora, simplemente pulsando el botón Send encontraremos los viajes. En caso de que para un tipo de viajes no nos encuentre viajes, no mostrará el título de su sección (por ejemplo Listado de viajes en tren).

Filtros para los viajes

Podremos filtrar los viajes dentro de cada tipo (Blablacar, Bus, Tren y Avión) para ello utilizamos los campos de filtrado, los cuales podemos ver a continuación.

- Campo de precio máximo

Podemos establecer un precio máximo. La aplicación no mostrará viajes con un precio superior.

A screenshot of a user interface element. On the left, the text 'Max price:' is displayed in a yellow font. To its right is a rectangular, light gray input field with rounded corners and a thin black border.

Figura E.7: Campo de precio máximo.

En caso de dejarse vacío, simplemente no se aplicará el criterio de descartar viajes por precio.

■ Campo de la forma de ordenación

Nos dice si debemos ordenar ascendentemente, descendentemente, o no ordenar los viajes.

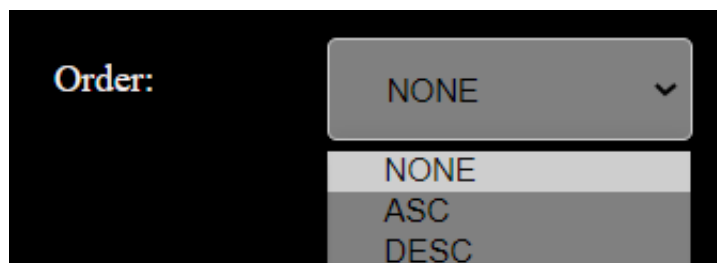
A screenshot of a user interface element. On the left, the text 'Order:' is displayed in a yellow font. To its right is a dropdown menu. The menu is currently open, showing a list of options: 'NONE', 'ASC', and 'DESC'. The 'NONE' option is highlighted with a light gray background. The dropdown menu has a dark gray background and a light gray border.

Figura E.8: Campo de tipo de ordenación.

De la imagen anterior podemos deducir tres posibilidades:

- NONE
No se aplica ordenación.
- ASC
Se aplica ordenación ascendente.
- DESC
Se aplica ordenación descendente.

■ Campo de la forma de ordenación

Nos indica si ordenamos por precio o por duración del viaje.

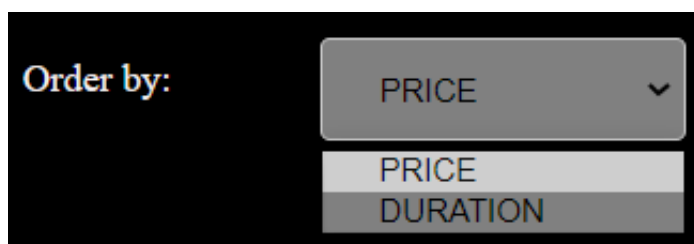


Figura E.9: Campo de tipo de ordenación.

A continuación vemos un ejemplo, sin ordenación ni precio máximo, de un viaje entre Coruña y Santiago de Compostela en tren.

Lista de viajes en tren				
Hora Salida	Hora Llegada	Precio	Nombre de la Estación de Salida	Nombre de la estación de llegada
06:46	07:25	6.3	Santiago de Compostela	A Coruña
07:42	08:13	7.6	Santiago de Compostela	A Coruña
08:35	09:03	7.6	Santiago de Compostela	A Coruña
09:42	10:10	7.6	Santiago de Compostela	A Coruña
11:22	11:58	6.3	Santiago de Compostela	A Coruña
11:52	12:20	7.6	Santiago de Compostela	A Coruña
12:25	12:53	7.6	Santiago de Compostela	A Coruña
14:05	14:47	6.3	Santiago de Compostela	A Coruña
14:11	14:40	16.0	Santiago de Compostela	A Coruña
14:11	14:40	14.4	Santiago de Compostela	A Coruña
14:11	14:40	21.3	Santiago de Compostela	A Coruña
14:44	15:12	7.6	Santiago de Compostela	A Coruña
16:15	16:43	7.6	Santiago de Compostela	A Coruña
16:36	17:13	6.3	Santiago de Compostela	A Coruña

Figura E.10: Muestra de viajes sin ordenación ni filtros.

Vemos un ejemplo aplicando una ordenación ascendente por precio, y otra descendente:

Hora Salida	Hora Llegada	Precio	Nombre de la Estación de Salida	Nombre de la estación de llegada
06:46	07:25	6.3	Santiago de Compostela	A Coruña
11:22	11:58	6.3	Santiago de Compostela	A Coruña
14:05	14:47	6.3	Santiago de Compostela	A Coruña
16:36	17:13	6.3	Santiago de Compostela	A Coruña
07:42	08:13	7.6	Santiago de Compostela	A Coruña
08:35	09:03	7.6	Santiago de Compostela	A Coruña
09:42	10:10	7.6	Santiago de Compostela	A Coruña
11:52	12:20	7.6	Santiago de Compostela	A Coruña
12:25	12:53	7.6	Santiago de Compostela	A Coruña
14:44	15:12	7.6	Santiago de Compostela	A Coruña
16:15	16:43	7.6	Santiago de Compostela	A Coruña
14:11	14:40	14.4	Santiago de Compostela	A Coruña
14:11	14:40	16.0	Santiago de Compostela	A Coruña
14:11	14:40	21.3	Santiago de Compostela	A Coruña

Figura E.11: Muestra de viajes con ordenación ascendente por precio.

Hora Salida	Hora Llegada	Precio	Nombre de la Estación de Salida	Nombre de la estación de llegada
14:11	14:40	21.3	Santiago de Compostela	A Coruña
14:11	14:40	16.0	Santiago de Compostela	A Coruña
14:11	14:40	14.4	Santiago de Compostela	A Coruña
07:42	08:13	7.6	Santiago de Compostela	A Coruña
08:35	09:03	7.6	Santiago de Compostela	A Coruña
09:42	10:10	7.6	Santiago de Compostela	A Coruña
11:52	12:20	7.6	Santiago de Compostela	A Coruña
12:25	12:53	7.6	Santiago de Compostela	A Coruña
14:44	15:12	7.6	Santiago de Compostela	A Coruña
16:15	16:43	7.6	Santiago de Compostela	A Coruña
06:46	07:25	6.3	Santiago de Compostela	A Coruña
11:22	11:58	6.3	Santiago de Compostela	A Coruña
14:05	14:47	6.3	Santiago de Compostela	A Coruña
16:36	17:13	6.3	Santiago de Compostela	A Coruña

Figura E.12: Muestra de viajes con ordenación descendente por precio.

Para el caso de E.15 vamos a establecer el precio máximo como 10:

A dark-themed user interface element. On the left, the text 'Max price:' is displayed in a light blue font. To its right is a light gray rectangular input field with rounded corners, containing the number '10' in a dark gray font.

Figura E.13: Establezco el precio máximo como 10€.

A continuación vemos el resultado:

Lista de viajes en tren				
Hora Salida	Hora Llegada	Precio	Nombre de la Estación de Salida	Nombre de la estación de Llegada
07:42	08:13	7.6	Santiago de Compostela	A Coruña
08:35	09:03	7.6	Santiago de Compostela	A Coruña
09:42	10:10	7.6	Santiago de Compostela	A Coruña
11:52	12:20	7.6	Santiago de Compostela	A Coruña
12:25	12:53	7.6	Santiago de Compostela	A Coruña
14:44	15:12	7.6	Santiago de Compostela	A Coruña
16:15	16:43	7.6	Santiago de Compostela	A Coruña
06:46	07:25	6.3	Santiago de Compostela	A Coruña
11:22	11:58	6.3	Santiago de Compostela	A Coruña
14:05	14:47	6.3	Santiago de Compostela	A Coruña
16:36	17:13	6.3	Santiago de Compostela	A Coruña

Figura E.14: Establezco el precio máximo como 10€.

Hemos visto como los viajes de precio igual a 14.4, 21.3 y 16 € ya no aparecen.

De igual forma podríamos repetir el proceso pero ordenando por el campo duración.

Ayuda de moovit

Para acceder a la ayuda de Moovit para llegar hasta un sitio, sólo tendremos que pulsar sobre el nombre de la estación origen o destino en los viajes directos o en la hora de llegada y partida en los viajes en Blablacar. Nuestro navegador nos abrirá una nueva pestaña en la que podremos utilizar el asistente de moovit. Automáticamente, se propondrá una calle como origen y otra como destino (si no se da forma de llegar, borrar dirección de origen o de llegada hasta ver sugerencias de Moovit).

A continuación, vemos un ejemplo:

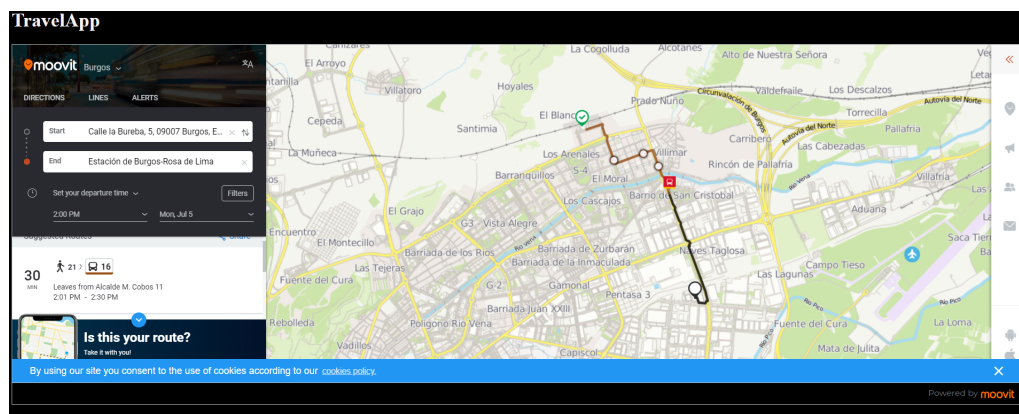


Figura E.15: Muestra del asistente de Moovit.

Bibliografía

- [1] Django. django. <https://www.djangoproject.com/>, 2021. [Online; Accedido 24-Junio-2021].
- [2] gnu.org. A quick guide to gplv3. <https://www.gnu.org/licenses/quick-guide-gplv3.html>, 2021. [Online; Accedido 28-Junio-2021].
- [3] Google. Google cloud. <https://cloud.google.com/>, 2021. [Online; Accedido 28-Junio-2021].
- [4] Google. Precios que se ajustan a tus necesidades. <https://cloud.google.com/maps-platform/pricing?hl=es>, 2021. [Online; Accedido 28-Junio-2021].
- [5] Ma-no.org. El concepto de modelo-vista-controlador (mvc) explicado. <https://www.ma-no.org/es/programacion/el-concepto-de-modelo-vista-controlador-mvc-explicado>, 2021. [Online; Accedido 24-Junio-2021].
- [6] pipreqs. Project description. <https://pypi.org/project/pipreqs/>, 2021. [Online; Accedido 28-Junio-2021].
- [7] proyectosagiles.org. Qué es scrum. <https://proyectosagiles.org/que-es-scrum/>, 2021. [Online; Accedido 28-Junio-2021].
- [8] Ph.D. Steve Burbeck. How to use model-view-controller (mvc). <https://web.archive.org/web/20120429161935/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>, 2021. [Online; Accedido 24-Junio-2021].

- [9] Uniwebsidad.com. El patrón de diseño mvc. <https://uniwebsidad.com/libros/django-1-0/capitulo-1/el-patron-de-diseno-mvc>, 2021. [Online; Accedido 24-Junio-2021].
- [10] VersionEye. Open source license compliance security. <https://www.versioneye.com/>, 2021. [Online; Accedido 28-Junio-2021].
- [11] Wikipedia. Html5. <https://es.wikipedia.org/wiki/HTML5>, 2021. [Online; Accedido 28-Junio-2021].