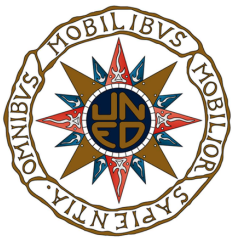


UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería Informática

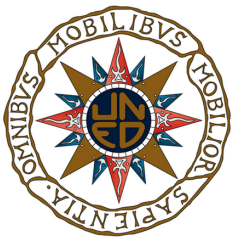
APLICACIÓN PARA LA GESTIÓN DE VIAJES EN VEHÍCULOS COMPARTIDOS

ANTONIO MANUEL NAVAS TORRES

Dirigido por: ALFONSO URQUIA MORALEDA

Codirigido por: CARLA MARTÍN VILLALBA

Curso: 2022/23 (convocatoria de septiembre)



APLICACIÓN PARA LA GESTIÓN DE VIAJES EN VEHÍCULOS COMPARTIDOS

Proyecto de Fin de Grado en Ingeniería Informática
de modalidad específica

Realizado por: ANTONIO MANUEL NAVAS TORRES

Dirigido por: ALFONSO URQUIA MORALEDA

Codirigido por: CARLA MARTIN VILLALBA

Fecha de lectura y defensa: SEPTIEMBRE DE 2023

Resumen

Este documento muestra el proceso de desarrollo y construcción de una aplicación para la gestión de viajes en vehículos compartidos como herramienta de apoyo a la toma de decisión. El nombre elegido para la aplicación desarrollada es Seats2Share, en referencia al concepto de compartir asientos.

Esta herramienta está diseñada como aplicación cliente-servidor para ser usada por múltiples usuarios. Aquellos que estén registrados y tengan acceso al sistema podrán establecer sus rutas a realizar y tendrán acceso a otros usuarios cerca de su área para solicitar viajar en el vehículo si las condiciones entre ellos son acorde a sus intereses.

Se ha realizado un diseño en una arquitectura de tres capas, siendo el componente principal la capa intermedia que es una API REST diseñada para realizar todas las operaciones que la aplicación precisa para cumplir con los requisitos funcionales. Actuando como intermediario entre la capa de datos y la capa de presentación. Esta API se encarga además de generar un grafo con los usuarios más próximos al usuario activo cuando éste define sus puntos de origen y destino.

Los usuarios tienen acceso en tiempo real sobre el estado de las reservas de sus viajes así como la posibilidad de coordinar los desplazamientos mediante mensajes a través de la propia aplicación. Pudiendo valorar el servicio recibido una vez finalizado.

La aplicación descrita en este documento está lista para ser usada y plenamente funcional en sus requisitos previstos para su elaboración.

Abstract

This document shows the process for developing and building of an application to manage trips in shared vehicles, as a decision support system. This application has been named Seats2Share as a reference to the concept about sharing seats on vehicles.

This tool has been designed as a client-server application to be used for different users. Those registered users that have access to the system, will be able to set their routes and will have access to the nearest users in the area to whom ask for travelling in their vehicles, if the travel conditions are according to their interest.

The design of this application is a three levels architecture, being the most important the middle layer that is an API REST designed to operate all the requests that this tool need to accomplish its main requirements. This middle layer works between data layer and presentation layer and one of its main task is to generate the graph of all nearest users to the active user when is defining the origin and destination points.

Users have access in real time to all active bookings and their state, also they can coordinate the travels with other users by messaging throughout the application. Users are able to rate others users experience when the travels has been finished.

The application described on this document is ready to be used and it is functional in all its functional requirements that were planned to accomplish.

Palabras clave

Economía

Colaborativa

Movilidad

Sostenible

Compartir

Vehículo

Ruta

Nodo

Grafo

Reserva

Keywords

Sharing

Economy

Sustainable

Mobility

To Share

Vehicle

Route

Node

Graph

Booking

Índice general

Índice de figuras

Índice de tablas

Índice de códigos fuente

Capítulo 1

Introducción, objetivos y estructura

1.1. Introducción

La movilidad en las grandes ciudades es todo un reto en la actualidad, especialmente la movilidad en superficie , que es una de las principales causantes de la emisión de gases en la atmósfera mermando la calidad de vida de las ciudades. Uno de los factores a minimizar es la contaminación asociada al uso del vehículo privado en las grandes ciudades, fomentando el uso de vehículos eléctricos o de bajas emisiones y limitando el acceso a los vehículos que no cumplan con los requisitos ambientales [1].

Incentivar vehículos más sostenibles con el medio ambiente no soluciona otro de los grandes retos de las ciudades: la congestión del tráfico. Por lo que es preciso reducir el número de vehículos en circulación, siendo una de las soluciones el fomentar el uso de vehículos compartidos, evitando así los desplazamientos con un único pasajero dentro de los vehículos, reduciendo el número de vehículos privados en circulación y de esa forma reduciendo el tráfico en superficie.

La economía colaborativa es un concepto reciente y amplio en su definición, que ayudado por el auge de Internet a nivel global, facilita que los interesados en dicha actividad económica, puedan compartir recursos, medios o servicios. Una de las variantes de la economía colaborativa estaría ligada a la movilidad sostenible, que *no se ciñe exclusivamente a la dimensión económica, sino también atañe lo ambiental y social* [2].

Seats2Share, nombre que le hemos dado a la aplicación, es una aplicación web orientada a facilitar el contacto entre usuarios que realizan recorridos similares. Seats2Share actúa como una herramienta de sistema de soporte de la decisión, Decision Support System (DSS), orientada a la economía colaborativa y más en particular a la movilidad sostenible.

Como herramienta DSS, con las rutas aportadas por el conjunto de usuarios, la aplicación muestra al usuario registrado las personas con rutas coincidentes y que están disponibles, para así decidir si acepta compartir o no su vehículo. De esta forma, si los usuarios aceptan compartir sus vehículos con otros usuarios, se podría reducir el número de vehículos privados en circulación; con los beneficios que eso conllevaría en las grandes ciudades. Por otro lado, el tiempo empleado en los desplazamientos en transporte público por parte de los usuarios de esta aplicación podría ser reducido, al verse beneficiados de un viaje más personalizado.

1.2. Objetivos

El objetivo principal de este proyecto es poner en contacto a usuarios dispuestos a realizar trayectos en vehículos compartidos, ya sea como conductor del vehículo o como pasajero, mediante reservas de viajes.

Los usuarios al registrarse deberán indicar cuál será su rol:

- Conductor.
- Pasajero.

En función del rol elegido podrán tomar las decisiones siguientes:

- Ambos:
  - Ajustar sus horas de salida y llegada.
  - Indicar tiempos máximos de espera y distancia máxima aceptada para desplazarse fuera del punto de salida o llegada.
  - Establecer sus puntos de salida y llegada.
  - Ver en un mapa todos los usuarios activos, ya sean pasajeros o conductores.
  - Consultar y actuar en el panel de viajes sobre las reservas en curso.
  - Modificar datos de su perfil.
  - Usar el sistema de mensajes breves para coordinar los desplazamientos.
- Si es conductor:
  - Introducir datos de su vehículo y plazas disponibles para su reserva.
  - Aceptar o rechazar las reservas realizadas por los pasajeros.
- Si es pasajero:
  - Consultar los conductores más cercanos a su punto de salida.
  - Realizar reservas a los conductores seleccionados.

La aplicación web irá mostrando mediante mensajes a los usuarios, los distintos eventos que se van produciendo; para que de esa forma el usuario pueda tomar la decisión que más se ajuste a sus intereses.

Al ser una aplicación orientada al uso en grandes ciudades, está realizada de forma que pueda ser usada en cualquier ciudad. Si bien se ha decidido mostrar el mapa de inicio y global en su punto central en Madrid, a efectos prácticos, cuando el usuario ha establecido sus puntos de salida y llegada el mapa de su perfil mostrará ambos puntos centrados, independientemente de que sea en Madrid o no.

Se enumeran las tareas a realizar para completar el desarrollo de la aplicación:

1. Establecer los requisitos del proyecto.
2. Definir las fuentes de datos de los mapas a usar durante el proyecto.
3. Se creará una API REST basada en Django REST Framework (DRF) para así generar las comunicaciones entre la aplicación cliente Frontend y las Bases de datos (BBDD).
4. El Frontend será basado en Vue 3, ya que ofrece múltiples ventajas al ser progresivo y modular.
5. Las BBDD serán por un lado PostgreSQL para los datos de los usuarios y por otro MongoDB para los datos de rutas, trayectos, reservas ...
6. Se procederá a integrar todos los módulos.
7. Se desplegarán las bases de datos en servidores externos (MongoDB Atlas y Render).
8. Se confeccionarán pruebas de usuarios para verificar los requisitos.
9. Se utilizará el diseño adaptativo o Responsive Design para su uso en cualquier dispositivo.
10. Se redactará la documentación del proyecto.
11. Despliegue del proyecto como aplicación web con los requisitos iniciales.

1.3. Estructura

Esta memoria está organizada en siete capítulos y tres anexos:

- **Capítulo 1 Introducción, objetivos y estructura:** Expone un marco actual sobre la problemática ambiental en las grandes ciudades en cuanto a la movilidad sostenible, ofreciendo una posible solución en la elaboración de este proyecto. Se marcan unos objetivos a realizar y se enumera la estructura de la documentación del proyecto.
- **Capítulo 2 Revisión del estado del arte:** En este capítulo se muestran las técnicas empleadas para la elaboración del grafo con las rutas para ser mostradas en el mapa a los usuarios y así éstos puedan tomar las decisiones más adecuadas.
- **Capítulo 3 Análisis:** A lo largo de este capítulo veremos varios diagramas en los que se analiza el comportamiento y funcionalidades del proyecto. El análisis es uno de los puntos más importantes en el desarrollo de un proyecto y aquí mostraremos parte de ese estudio.
- **Capítulo 4 Diseño:** En este capítulo se estudiará el diseño aplicado en el proyecto con las distintas tecnologías que se emplean y la justificación de su elección. Una correcta integración de la estructura en tres niveles nos facilita el desarrollo del proyecto.

- **Capítulo 5 Implementación y pruebas:** En este capítulo se describirán los lenguajes y herramientas software empleadas para el desarrollo del proyecto, así como la implementación de todos los elementos que forman la aplicación. Finalizará el capítulo con una batería de pruebas que permitan afirmar que el proyecto cumple con la funcionalidad prevista en las fases de análisis y diseño.
- **Capítulo 6 Planificación del proyecto:** En este capítulo se describirán las distintas fases que forman parte de la planificación del proyecto, así como las iteraciones que forman parte del mismo. Se mostrarán los diagramas de Gantt para comprobar la desviación con respecto a la planificación inicial prevista.
- **Capítulo 7 Conclusiones y trabajos futuros:** En este capítulo se hará un repaso de todo el desarrollo del proyecto y los trabajos que aún quedarían por hacer para una mejora y actualización del mismo.
- **Anexo A Manual de despliegue:** En este anexo se describirán los pasos a seguir para desplegar el proyecto tal y como aparece en este documento.
- **Anexo B Manual de usuario:** En este anexo se incluyen las instrucciones de uso de la aplicación a nivel usuario.
- **Anexo C Código fuente:** En este anexo se incluye todo el código fuente de los archivos usados en el proyecto.

Capítulo 2

Revisión del estado del arte

2.1. Introducción

En este capítulo se verán las herramientas y algoritmos disponibles para poder visualizar los recorridos entre puntos en un mapa integrado en la aplicación web. Comenzamos el capítulo con las distintas herramientas para visualizar mapas en aplicaciones web así como la información que podemos obtener de ellas. Continuamos con la búsqueda de recorridos examinando los algoritmos que se emplean para ello en los mapas de este tipo.

Se examinará cómo aplicar la selección de los nodos en función de la distancia y finalizará el capítulo con una breve descripción de algunas aplicaciones ya existentes similares a la del proyecto.

2.2. Mapas y nodos

Los mapas digitalizados han supuesto un avance inimaginable en el acceso gratuito a datos cartográficos. Gracias a plataformas como OpenStreetMaps [3] o Google Maps [4], tenemos a nuestro alcance una cantidad de datos cartográficos que van desde carreteras, edificios, accidentes geográficos hasta incluso dónde hay escaleras o antenas. Este tipo de información va en aumento, lo cual nos permite poder obtener mapas de casi cualquier tipo imaginable.

Una aplicación como la que estamos planteando en este proyecto, necesita este tipo de información cartográfica. Para ello, hay que estudiar cuáles son las fuentes de datos, cómo se obtienen, su licencia de uso o limitaciones. La primera pregunta que nos surge es qué información cartográfica es la más apropiada para una aplicación de este tipo. Las plataformas indicadas anteriormente disponen de una API que nos permite el acceso a datos cartográficos a nivel mundial. La documentación en ellas es bastante extensa y lo más importante: el acceso a dicha información es gratuita. Millones de personas cada día consultan datos cartográficos sobre dichas plataformas mediante aplicaciones que usan dichas APIs.

La principal información cartográfica que necesitamos son las coordenadas de nuestros puntos de origen y destino, si nuestra intención es mostrar en un mapa dichos puntos. Para ello, cualquiera de esas plataformas nos muestran en su documentación cómo obtener las

coordenadas. Nos centraremos en OpenStreetMaps principalmente porque además de ser de código abierto, es un mapa editable que permite a los usuarios añadir datos geográficos o información cartográfica [5]. El acceso a la información cartográfica no es tema baladí, ya que ese tipo de información es mucho más importante de lo que en principio uno podría pensar [6].

El modelo de datos de OpenStreetMaps está formado por tres elementos básicos que nos ofrecen la información que necesitamos [7].

- Nodos: Es el elemento base, representa un punto específico y contiene sus coordenadas y un identificador.
- Vías: Es una lista ordenada de nodos, con un identificador, que permite dibujar ríos, carreteras, calles, ... Cualquier objeto que sea lineal. Pueden indicar también el sentido.
- Relaciones: Es un elemento con una etiqueta tipo y además un grupo de otros miembros que pueden ser una lista ordenada de nodos, vías u otras relaciones. Por ejemplo: puentes, cruces de carreteras, líneas de transporte, etc.

En la Figura 2.1 podemos ver cómo se muestra el Puente de Segovia, en Madrid, usando la capa de mapa estándar de OpenStreetMaps. En la Figura 2.2 podemos ver esa misma vista del Puente de Segovia, mostrando el modelo de datos tanto de nodos, vías y relaciones. Se observan en la Figura 2.2 las vías y los nodos aislados, así como las relaciones en forma de áreas. Estos puntos se muestran visualmente de esta forma, pero cada uno de ellos tiene un identificador, con sus correspondientes coordenadas geográficas. Obtener esos puntos es el objetivo a conseguir y para ello tenemos varias herramientas. Una de ellas es Nominatim [8]. Esta herramienta nos permite realizar consultas y obtener el identificador del nodo y sus coordenadas haciendo uso de su API.

Haciendo la consulta siguiente:

https://nominatim.openstreetmap.org/ui/search.html?q=puente+de+segovia%2C+madrid

Se obtiene como resultado la Figura 2.3

Por lo que una opción viable sería hacer peticiones HTTP a dicha dirección URL. En las peticiones habría que indicar como parámetros la calle, número y ciudad. Obteniendo así las coordenadas del punto. Lamentablemente, según la documentación de Nominatim, este tipo de consulta a la API será abandonada en un futuro cercano. Además, hay una limitación de una petición por segundo, lo cual influiría en un uso masivo de la aplicación. Por lo que hay que tener en cuenta estas limitaciones a la hora de emplear esta opción. Descargar Nominatim es otra alternativa y así podemos evitar el límite de peticiones por segundo, pero la instalación requiere mucho almacenamiento. Como referencia la descarga del planeta requiere 128GB de memoria RAM y al menos 1TB de espacio en disco [9]. Eso en un servidor son unos costes muy elevados.

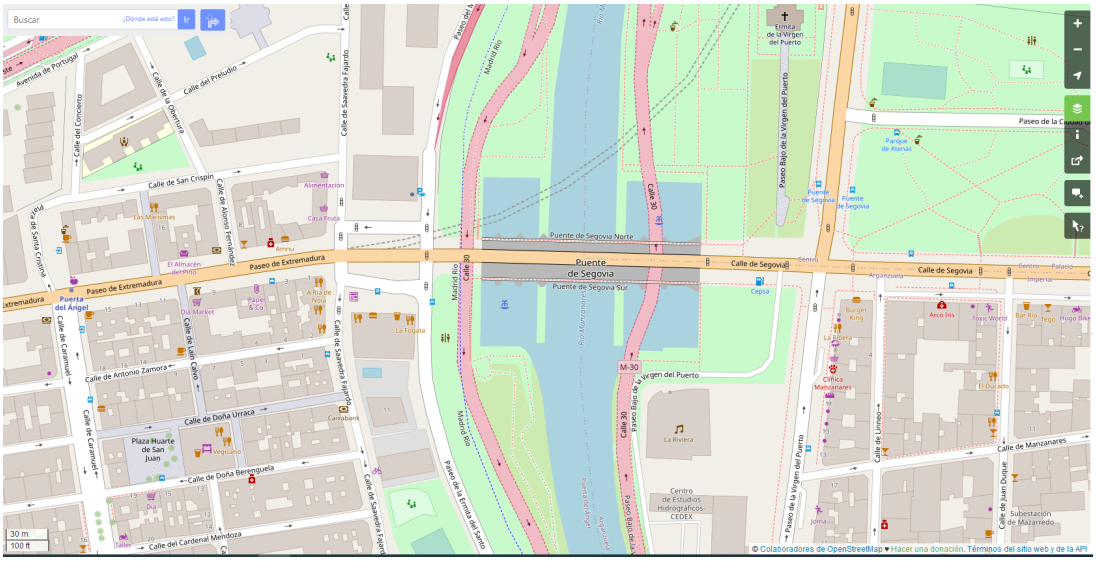


Figura 2.1: Capa estándar en OpenStreetMaps.

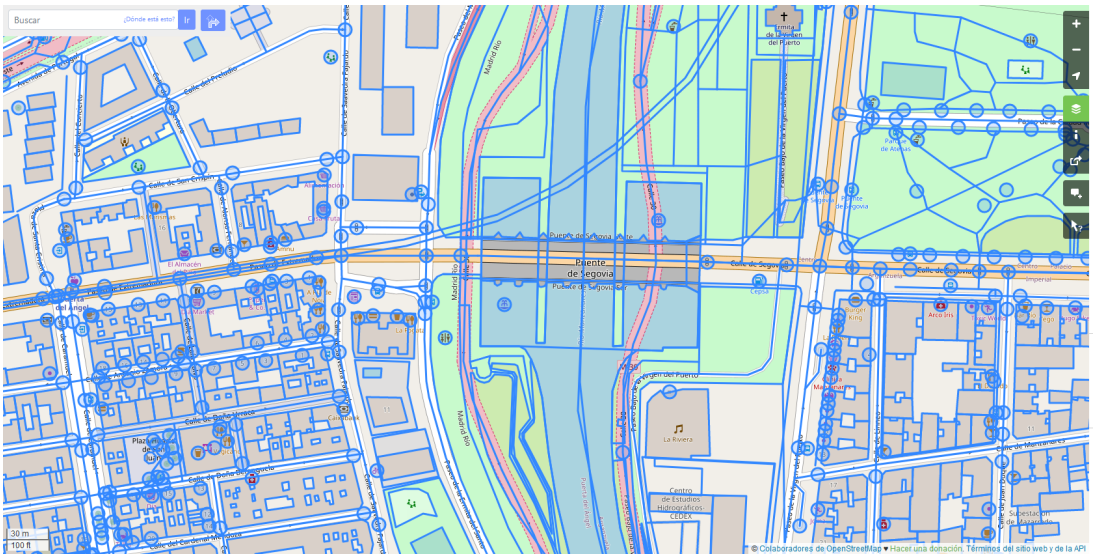


Figura 2.2: Capa modelo de datos en OpenStreetMaps.

Puente de Segovia link to this page

Name	**Puente de Segovia** (name)
Type	man_made:bridge
Last Updated	2023-09-01T17:29:30.765463+00:00
Search Rank	30
Address Rank	30 (house / building)
Importance	0.31831815181825923
Coverage	Polygon
Centre Point (lat,lon)	40.413965950000005,-3.722856554592399
OSM	way 380275213
Place Id	283959667 (on this server)
Wikipedia Calculated	es:Puente_de_Segovia
Computed Postcode	28005 (how?)
Address Tags	
Extra Tags	**1** (layer) **attraction** (tourism) **Q2838423** (wikidata) **es:Puente de Segovia** (wikipedia) **yes** (wheelchair) **arch** (bridge:structure) **Q2838423** (name:etymology:wikidata)

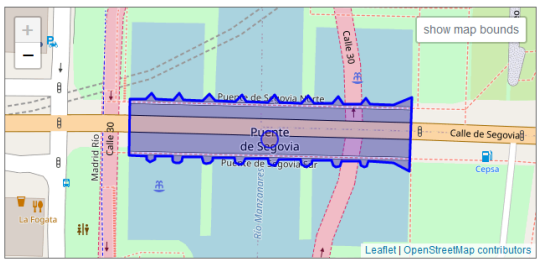


Figura 2.3: Datos obtenidos en consulta en Nominatim.

Otra forma de obtener las coordenadas es mediante el uso de un mapa integrado en el Frontend. Usando por ejemplo la librería leaflet-routing-machine [10] podemos tener el mapa en nuestra aplicación web y obtener las coordenadas del centro del mapa o incluso en el lugar en el que hagamos *click* con el ratón por ejemplo. Esta librería es compatible con varios frameworks, aunque lleva varios años sin actualizarse, cubre las necesidades básicas para una aplicación web. Concretamente para obtener las coordenadas de los puntos de origen y destino es mucho más rápido y visual que mediante Nominatim.

Leaflet-routing-machine se apoya en Project OSRM [11] para obtener la información cartográfica, así como el cálculo de rutas. En este caso OSRM permite hasta 512 peticiones en una única conexión, en intervalos de 5 segundos, por lo que la mejora frente a Nominatim es bastante superior. El proyecto OSRM es el más usado para la obtención de datos cartográficos a través de mapas integrados en las aplicaciones webs que usan OpenStreetMaps como fuente de mapas.

Finalmente existe otra herramienta, muy similar a *leaflet-routing-machine* que también se apoya en OSRM. Se trata de Mapbox [12] y aunque tiene un servicio de pago para usos más específicos en cuanto a los mapas; tiene una versión gratuita con las mismas funcionalidades que *leaflet-routing-machine* y además está totalmente actualizado y funcional con todos los Frameworks.

Se puede deducir que hay varias opciones de obtener las coordenadas de un punto cuando uno quiere establecer un origen y un destino. Las herramientas vistas no son las únicas, pero sí las más usadas. En ellas el obtener la información que necesitamos es accesible. El funcionamiento en Google Maps, el más usado en la actualidad, es muy similar al de OpenStreetMaps en cuanto a las funciones a realizar, pero a partir de cierto número de peticiones y accesos, hay que pagar por continuar con esos servicios [13]. Sabiendo cómo obtener los nodos de origen y destino, el siguiente paso es establecer las rutas o recorridos más idóneos.

2.3. Búsqueda de rutas

Salvo Nominatim, las herramientas anteriores permiten obtener recorridos entre puntos. Los recorridos calculados son en base al medio de transporte elegido, ya sea en vehículo, bici, transporte público o andando, por citar algunos ejemplos. Para ello si seguimos con el modelo de datos de OpenStreetMaps, al estar formado por nodos, vías y relaciones, las relaciones entre estos elementos nos muestran un grafo ponderado y dirigido. Por lo que un algoritmo de búsqueda de caminos más cortos es el más idóneo en este caso.

El algoritmo más usado en estas situaciones es el algoritmo Dijkstra, que nos permite obtener el camino más corto dado un punto de origen, con respecto al resto de nodos del grafo. Para ello una solución es tener un grafo con todos los nodos de un área en concreto y utilizar dicho algoritmo. Se pueden descargar parcialmente datos desde OpenStreetMap y con la información que obtenemos vía Nominatim crear el grafo, pero eso limitaría las búsquedas a la zona descargada, limitando el área de uso de la aplicación. Por lo que la opción más práctica es recurrir a algunas de las herramientas que nos ofrecen esa posibilidad a nivel global, como OSRM.

Open Source Routing Machine (OSRM) es la herramienta principal para obtener rutas usando OpenStreetMaps. Para calcular los recorridos utiliza en concreto dos algoritmos de búsqueda de recorridos:

- Multi-Level Dijkstra (MLD).
- Contraction Hierarchies (CH).

OSRM recomienda utilizar por defecto el MLD como algoritmo de búsqueda de caminos más cortos si las distancias no son muy grandes. Este algoritmo es una variación del algoritmo Dijkstra, su funcionamiento está basado en el uso de las particiones multinivel ya realizadas sobre el grafo y las celdas superpuestas precalculadas para acelerar heurísticamente el cálculo [14]. Estas particiones multinivel lo que hacen es dividir el grafo en subgrafos o particiones más pequeñas que luego serán procesadas para asignarles un nivel al nodo en función de la distancia con respecto al origen. Así se establece una jerarquía de niveles creando una estructura multinivel. Los nodos dentro de cada partición se asignan a niveles según sus distancias desde el nodo fuente dentro de esa partición. Esta asignación garantiza que los nodos más cercanos al origen se procesen antes que los nodos más alejados.

En cada nivel de jerarquía se crea un grafo de celdas, dichos grafos representan cómo están conectadas cada una de las particiones, siendo una celda la representación de una partición y las aristas la conectividad entre esas particiones, garantizando así la conectividad del grafo. Cuando las particiones multinivel están completas, el MLD puede ser usado mejorando así el coste computacional con respecto al clásico algoritmo Dijkstra al no tener que examinar todo el grafo, sino que se calcula sobre cada una de las particiones consultando en los distintos niveles y grafos de celdas. Veamos el algoritmo multinivel Dijkstra en pseudocódigo traducido del expuesto en [14].

Entrada: Grafo $G = (V, E, \text{len})$, fuente y objetivo s, t, datos para len, celda mapeada en celdaNivel
Salida: Distancias d[v], árbol de camino más corto que contiene los nodos superpuestos dados por padre[·]
paraTodos $v \in \boldsymbol{V}$ **hacer**
    d[v] $\leftarrow \infty$
    padre[v] $\leftarrow$ null
Q.INSERTAR(s, 0)
d[s] $\leftarrow$ 0
mientras Q no esté vacío hacer
    u $\leftarrow$ Q.BORRARMINIMO()
    nivel $\leftarrow$ nivel(u)
    celda $\leftarrow$ celdaNivel(u)
    paraTodos (u,v) $\in$ Enivel/celda, (u, v) $\in$ EnivelSup/celda hacer
        si d[u] + len(u, v) $<$ d[v] entonces
            d[v] $\in$ d[u] + len(u,v)
            padre[v] $\in$ u
            Q.ACTUALIZAR(v, d[v])

Veamos el comportamiento del algoritmo:

- Se inicializan las estructuras de datos, todas las distancias se establecen como infinito salvo el nodo fuente que se establece a 0.

- El algoritmo comienza por el nivel más bajo, Nivel 0, y procede nivel por nivel. En cada nivel se exploran nodos y sus vecinos en ese nivel.

- En cada nivel se extrae el nodo de menor distancia. Se relajan los nodos vecinos del mismo nivel, para ello se calcula la distancia tentativa desde el nodo fuente a cada nodo vecino a través del nodo actual. Si es menor que la distancia registrada anteriormente, se actualiza la distancia.

- Se relajan ahora los nodos vecinos en niveles superiores, para ello se comprueba si relajar los vecinos del nodo actual conduce a caminos más cortos hacia nodos en niveles superiores. Actualizando la distancia en caso afirmativo.

- Finaliza el algoritmo cuando llega al nodo objetivo o se han procesado todos los niveles.

Este algoritmo lo que hace es centrarse en nodos más cercanos al origen antes de expandirse a nodos más lejanos, lo que nos permite un ahorro muy sustancial en grafos de gran tamaño. Aunque hay que tener en cuenta que depende de una buena asignación de niveles para que sea óptimo.

La Figura 2.4 muestra un grafo de búsqueda multinivel ilustrado empleando el MLD entre el punto S y el punto T. Las celdas más claras son de niveles más cercanos y cuanto más oscuras de niveles más altos. El recorrido en rojo sería el más corto de acuerdo a este grafo. Es una representación ilustrativa similar al grafo mostrado en [14].

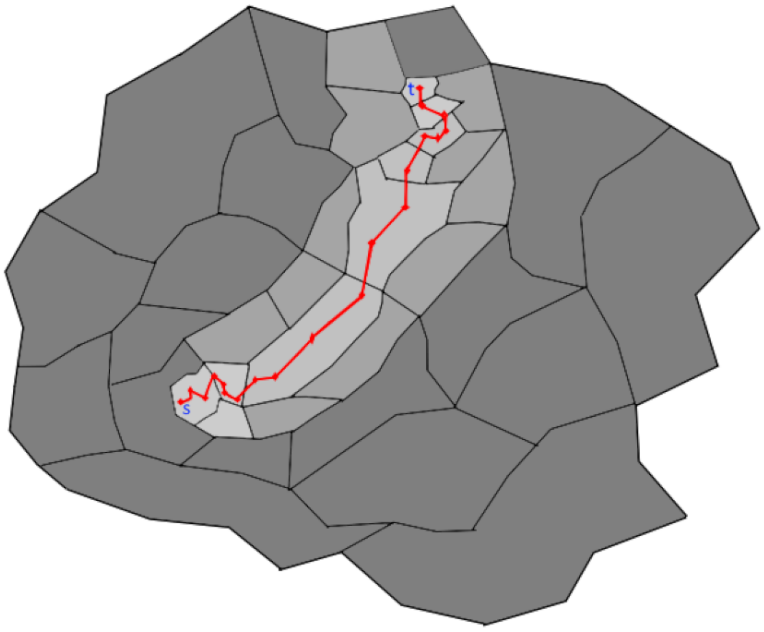


Figura 2.4: Grafo de búsqueda Multinivel Dijkstra.

El otro algoritmo que se utiliza en OSRM es el Contraction Hierarchies o Jerarquías de contracción. El cual es más recomendado para buscar rutas en grafos más grandes. En nuestra aplicación web no habría que considerar el usar este algoritmo al estar enfocado a distancias más cortas. El algoritmo de jerarquías de contracción consiste en preprocesar un grafo contrayendo de forma selectiva nodos y aristas, dicha selección de nodos puede ser por cualquier criterio: número de aristas o importancia en la red, en el caso de carreteras si son autopistas, vías rápidas, etc. De esta forma se va creando una jerarquía en la que se va representado de forma más abstracta el grafo, reduciendo así la complejidad de los algoritmos de búsqueda. Este preprocesamiento permite que luego se puedan realizar consultas de caminos más cortos de forma óptima, ya que el algoritmo comienza la búsqueda siguiendo la jerarquía que se haya creado. En realidad el coste computacional de este algoritmo está en crear el preprocesamiento. Al igual que ocurría con el MLD, el determinar los niveles o en este caso la jerarquía, hará que el coste de obtener el camino más corto sea óptimo o casi óptimo si se ha hecho un buen preprocesamiento.

Se observa que Mapbox también utiliza MLD como algoritmo para obtener los recorridos [15], ya que se apoya en OSRM, por lo que de esta forma el algoritmo para la búsqueda de rutas será el MLD de cara a la aplicación web. Para ello Mapbox nos ofrece la siguiente dirección para recibir un objeto Route que contendrá la información que necesitamos para obtener recorridos.

https://api.mapbox.com/directions/v5/profile/coordinates

Siendo *profile* un *String* que nos indica uno de los siguientes cuatro tipos:

- mapbox/driving-traffic: Para obtener recorridos teniendo en cuenta el estado del tráfico, de acuerdo a datos actuales e históricos.
- mapbox/driving: Para obtener recorridos sin tener en consideración el estado del tráfico. Se tienen en cuenta los recorridos más rápidos.
- mapbox/walking: Devuelve el camino más corto para ir a pie.
- mapbox/cycling: Devuelve el camino más corto y seguro para ir en bicicleta, evitando autopistas o carreteras muy transitadas.

En *coordinates* tendríamos una lista formada por entre 2 y 25 pares de objetos de coordenadas *longitude,latitude* separados por punto y coma, que nos indicarían el orden en el que se visitarían.

El objeto Route que recibiríamos al realizar esta petición, tendría varias propiedades, siendo estas las más relevantes:

- duration: Tiempo estimado en segundos para recorrer todos los *waypoints*.
- distance: Distancia recorrida en recorrer los *waypoints*.
- waypoints: Array con todos puntos y sus coordenadas ordenados en el orden que son visitados.

2.4. Selección de los nodos más adecuados

Las fuentes de información de mapas digitales así como el método de búsqueda de rutas están claros y definidos, por lo que el siguiente paso es establecer cuáles serán los nodos más adecuados para mostrar al usuario aquellos conductores o pasajeros disponibles en su área. La forma más idónea sería crear un grafo dirigido y ponderado en el que tendríamos las distancias entre cada uno de los puntos de origen de los usuarios. En dicho grafo, cada nodo sería un usuario y las aristas la distancia a los otros usuarios. Habrá que calcular por tanto dicha distancia.

Al recibir un objeto Route como el descrito en la Sección 2.3, tenemos la información de distancia y tiempo, pero para ello se ha realizado una consulta a la API de Mapbox. Para generar el grafo habría que hacer un número considerable de peticiones y por ende, un incremento de los costes y tiempo de respuesta. Cuando las distancias a comparar entre usuarios son pequeñas, se puede utilizar un método de cálculo de distancias entre dos puntos geográficos, al que podríamos añadirle un factor de corrección si quisiéramos compensar el margen de error que se produce; ya que una línea recta siempre será menos distancia que ir de un punto a otro por las diversas calles. El objetivo de esta aplicación web es ver a los usuarios más cercanos. Por lo que la distancia en línea recta entre ambos usuarios pueda ser ser usada como una referencia a tener en cuenta; asumiendo un margen de error si estamos hablando de distancias de hasta 2 o 3 km; evitando el tener que realizar múltiples peticiones a través de la API de Mapbox y ahorrando tiempo y costes.

La Figura 2.5 nos ilustra el objetivo a conseguir. El problema a resolver es mostrar únicamente aquellos nodos que nos interesan, es decir, aquellos que cumplen los criterios de distancia entre ambos usuarios. Para ello habrá que hacer una discriminación en función de la distancia y aceptar o rechazar nodos en el grafo en función de ésta.

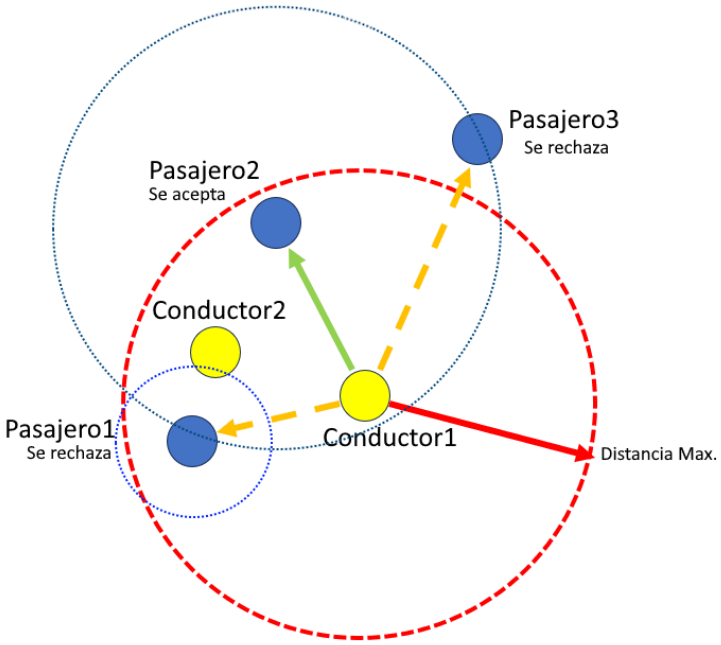


Figura 2.5: Nodos en función de la distancia.

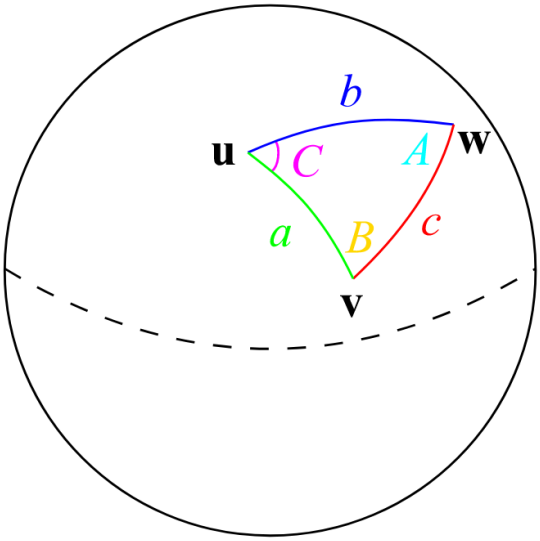


Figura 2.6: Triángulo esférico ley del semiverseno.

Una forma de resolverlo sería con las herramientas vistas en la Sección 2.3 al calcular los recorridos punto a punto y obteniendo así sus distancias. Pero eso ya se ha visto que supondría un elevado número de peticiones para generar el grafo y supondría un aumento en el tiempo de respuesta e incremento de costes. Otra alternativa sería el uso de la distancia del semiverseno. La fórmula del semiverseno es una ecuación que nos permite calcular la distancia entre dos puntos, sabiendo sus coordenadas geográficas, aplicando la ley de los semiversenos.

La ley de los semiversenos establece que en una esfera unidad, existe un *triángulo esférico* en la superficie de esa esfera, con tres puntos u, v y w conectados mediante tres círculos máximos. Se forman tres arcos: arco a de u a v, arco b de u a w y arco c de w a v. Como se muestra en la Figura 2.6 [16].

La aplicación de la fórmula del semiverseno es un caso especial de la ley de los semiversenos que considera al punto u como el Polo Norte, así los puntos v y w son las coordenadas sobre las que queremos calcular su distancia. Se muestra a continuación la fórmula del semiverseno.

$$d = 2r \cdot arcsin \cdot \left(\sqrt{sin^2 \cdot \left(\frac{\phi_2 - \phi_1}{2} \right) + cos\left(\phi_1\right) \cdot cos\left(\phi_2\right) \cdot sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right) \tag{2.1}$$

Donde d es la distancia que queremos obtener, r es el radio de la Tierra, ϕ_1 y ϕ_2 se corresponden a la latitud de los puntos y λ_1 y λ_2 a la longitud de los puntos, todos expresados en radianes.
Se puede implementar dicha fórmula con sus conversiones correspondientes sin ningún problema o bien usar una librería espefícia para ello como por ejemplo la que existe para Python Haversine [17]. Esta aproximación tiene un margen de error asumible en distancias pequeñas como las que se van a gestionar en la aplicación web y consideramos que es la solución más idónea buscando el equilibrio tiempo de respuesta y costes. La Figura 2.7 nos muestra una ilustración de cómo sería el grafo una vez se han examinado los nodos de la Figura 2.5.

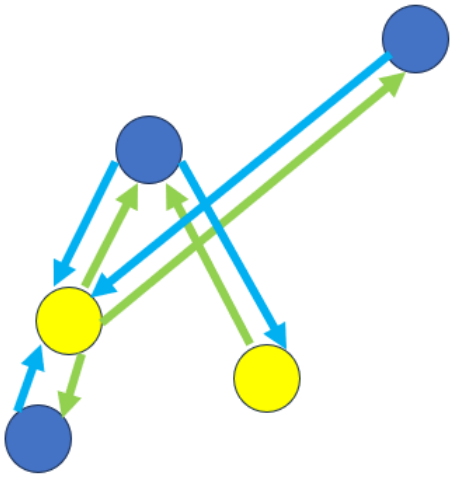


Figura 2.7: Nodos examinados.

2.5. Aplicaciones existentes

En las secciones anteriores se mostró cómo resolver los problemas tanto de representación de los mapas en la aplicación web, los recorridos así como el criterio para seleccionar los nodos más adecuados, en esta sección veremos algunas aplicaciones existentes similares a la aplicación web de este proyecto. Los conceptos de economía colaborativa y movilidad sostenible están aplicándose ya en muchos lugares en forma de aplicaciones de vehículos compartidos, *carsharing* o *carpooling*.

Repsol Wible [18]: Aplicación móvil donde el usuario puede alquilar por minutos, horas o días una serie de vehículos híbridos, disponibles en Madrid. En dicha aplicación se alquila el vehículo y uno tiene que devolverlo en los parkings disponibles para ello. Las ventajas que ofrecen son principalmente que al ser vehículos híbridos no tienen restricciones en el acceso a los centros de ciudades, así como no tener que preocuparse de mantenimiento, seguros, etc. Es una evolución del alquiler tradicional de vehículos. En la página web de la aplicación se puede ver un mapa con los puntos de recogida o devolución de los vehículos, así como los puntos donde repostar.

Hopp Carpool [19]: Esta aplicación permite poner en contacto pasajeros y conductores para compartir sus viajes, mediante un pago por km por parte del pasajero. Ofrecen viajes gratis si pertenecen a grupos o empresas que hayan suscrito acuerdos y los conductores reciben recompensas por ello. Esta aplicación muestra en un mapa las rutas disponibles y permite coordinar los puntos de recogida entre pasajeros y conductores mediante mensajes entre ellos. Es una aplicación muy similar a la descrita en el proyecto.

Europe Carpooling [20]: Esta aplicación web indica rutas previstas por parte de conductores entre distintos puntos y su oferta de plazas así como su precio. Está más enfocado a un uso en largas distancias entre ciudades más que a un uso dentro de la propia ciudad. Tiene un buscador donde se indica el origen, destino y fecha. Devolviendo un listado de conductores disponibles, si los hay.

Blablacar [21]: Una de las pioneras en este tipo de aplicaciones. Esta aplicación permite buscar por punto de origen, destino y fecha y nos salen los conductores disponibles. Hay más filtros para personalizar la búsqueda. En el listado que nos muestran están las condiciones de los conductores, como fecha, tiempo de viaje, precio y limitaciones durante el viaje. Está más orientada a viajes entre ciudades como alternativa al transporte en ferrocarril, avión o autobús.

2.6. Conclusiones

En este capítulo hemos visto las diferentes herramientas disponibles para representar mapas y recorridos en un aplicación web, integrando así su uso y permitiendo obtener datos cartográficos de forma que puedan ser utilizados para su gestión posterior. Obtener los recorridos entre puntos requiere del uso de algoritmos de búsqueda de caminos más cortos como los que se han descrito en este capítulo.

También se ha examinado otra alternativa al cálculo de distancias entre puntos mediante la distancia del sermiverseno para así utilizarla de forma más eficiente en la elaboración del grafo correspondiente a los usuarios más cercanos. Finaliza el capítulo con una muestra de aplicaciones similares a la propuesta con una pequeña descripción de sus funcionalidades.

Capítulo 3

Análisis

3.1. Introducción

Teniendo en cuenta los planteamientos del capítulo anterior, en este capítulo vamos analizar el resultado de la toma de requisitos para poder tener un prototipo funcional, así como los diagramas en Unified Modeling Language (UML) [22], lenguaje de modelado unificado, empleados para su descripción. Es trivial tener en cuenta que un mal diseño puede resultar en un producto deficiente o que ni siquiera realice las funciones solicitadas por el cliente.

A lo largo del desarrollo de este proyecto, se ha ido refinando este análisis para llegar al resultado final expuesto en este capítulo. En las secciones siguientes se mostrará, además de la toma de requisitos funcionales y prácticos, diagramas UML correspondientes a dicho análisis.

3.2. Requisitos

A la hora de diseñar cualquier software o aplicación, la toma de requisitos es uno de los puntos cruciales. Es en esta toma donde el cliente le indica al desarrollador qué es exactamente lo que quiere, sus expectativas o qué acciones debería poder realizar con la aplicación. Para que el desarrollador pueda tener clara esa toma, ha de realizar unas buenas técnicas de extracción de la información para poder concluir en una toma efectiva de requisitos [23].

Tenemos dos tipos de requisitos, los funcionales y los no funcionales.

3.2.1. Requisitos funcionales

Los requisitos funcionales son aquellos que describen las funciones que, en este caso, el proyecto es capaz de realizar.

RF1: Gestión de usuarios.

1.1 El sistema debe permitir que un usuario se registre.

1.2 El sistema debe permitir que un usuario pueda acceder al sistema mediante autenticación.

1.3 El sistema debe permitir que un usuario pueda elegir qué rol ocupará.

1.3.1 Conductor.

1.3.2 Pasajero.

1.4 El sistema debe permitir que un usuario pueda modificar sus datos de autenticación.

RF2: Gestión del perfil.

2.1 Un usuario puede definir sus condiciones, puntos de recogida y limitaciones en cuanto a ser elegible por otro usuario.

2.2 El sistema debe permitir al usuario cambiar dichas características de su perfil.

2.3 El sistema debe permitir al usuario ver a otros usuarios cercanos a su posición en función de sus condiciones o limitaciones.

2.4 El sistema debe permitir al usuario tipo pasajero realizar reservas de plazas a usuarios tipo conductor.

2.5 El sistema debe permitir al usuario tipo conductor gestionar las reservas recibidas por el usuario tipo pasajero.

RF3: Gestión de rutas.

3.1 El sistema debe ser capaz de elaborar una colección de rutas con los datos aportados por los usuarios.

3.2 El sistema debe crear un grafo dirigido ponderado con los puntos de recogida de los usuarios.

3.3 El sistema debe mostrar a los usuarios los grafos correspondientes a su selección y mostrarlos en un mapa visible.

3.4 El sistema debe mostrar al usuario todos los tipos de usuarios activos en un mapa visible.

RF4: Gestión de reservas.

4.1 El sistema debe informar a los usuarios de las reservas disponibles.

4.2 El sistema debe permitir a los usuarios gestionar las reservas en curso.

4.2.1 Solicitar Reserva.

4.2.2 Aceptar Reserva.

4.2.3 Anular Reserva.

4.2.4 Rechazar Reserva.

4.2.5 Finalizar Reserva.

4.2.6 Reactivar Reserva.

4.2.7 Borrar Reserva.

4.2.8 Archivar Reserva.

RF5: Mensajería.

5.1 El sistema debería permitir a los usuarios comunicarse entre ellos.

5.2 El sistema debería permitir enviar mensajes a los usuarios.

5.3 El sistema debería permitir recibir mensajes a los usuarios.

RF6: Valoración.

6.1 El sistema debería permitir valorar a otros usuarios.

6.2 El sistema debería mostrar la valoración recibida a los usuarios de forma visible.

3.2.2. Requisitos no funcionales

Los requisitos no funcionales son aquellas características que afectan de una forma u otra al funcionamiento del sistema, en relación a varios factores, como el rendimiento, integridad, fiabilidad o seguridad [23].

RNF1: El sistema debe ofrecer una autenticación segura para el acceso al sistema.

RNF2: El sistema debe poder obtener los datos cartográficos de forma fiable y rápida.

RNF3: El sistema debe asegurar la integridad de los datos al almacenarlos en BBDD.

RNF4: La interfaz de usuario deber ser clara y con información suficiente para que el usuario pueda tomar las decisiones correctas.

RNF5: El usuario debe ser notificado de los eventos que se producen por parte del sistema.

RNF6: El número de acciones por parte del usuario debe ser las mínimas posibles.

3.2.3. Requisitos no contemplados

Seats2Share no es un proyecto finalizado, es un sistema que tiene aún muchas más funciones a incluir, pero en la toma de requisitos se ha decidido que los siguientes requisitos puedan ser incluidos en actualizaciones posteriores del producto.

RNC1: Autenticación por medio de medios más seguros y con posibilidad de autenticación por redes sociales, como Firebase: Firebase Auth [24].

RNC2: Interfaz de usuario para el administrador del sistema.

RNC3: Visualización de los grafos mediante Sigmajs [25] para su estudio por parte del administrador.

RNC4: Ofrecer al usuario variantes de las rutas con pasos intermedios.

RNC5: Migración a BBDD más específicas para grafos como Neo4j

RNC6: Personalización de la GUI en cuanto a idiomas a mostrar, color y visualización de distintos tipos de iconos en el mapa.

RNC7: Aplicación para dispositivos Android e iOS.

A medida que el producto se vaya usando irán surgiendo más funcionalidades no previstas.

3.3. Casos de uso

Los casos de uso son un tipo de diagramas UML [22], que nos permiten mostrar de forma muy visual las distintas acciones que un usuario puede realizar en un sistema, así como el resto de actores que participen en él [26].

Para una mayor comprensión del caso de uso de la Figura 3.1 vamos a elaborar los escenarios correspondientes siguiendo el modelo ofrecido por [27]. El formato de presentación de escenarios de casos de uso que se muestra en las siguientes páginas, me parece mucho más formal e intuitivo que en formato lista.

Ofrecen información relevante acerca de los requisitos funcionales y dependencias entre los distintos actores principales, secundarios así como otros casos de uso que intervienen. En la Tabla 3.1 mostramos el listado con los escenarios generados para cada uno de los casos de uso de la Figura 3.1

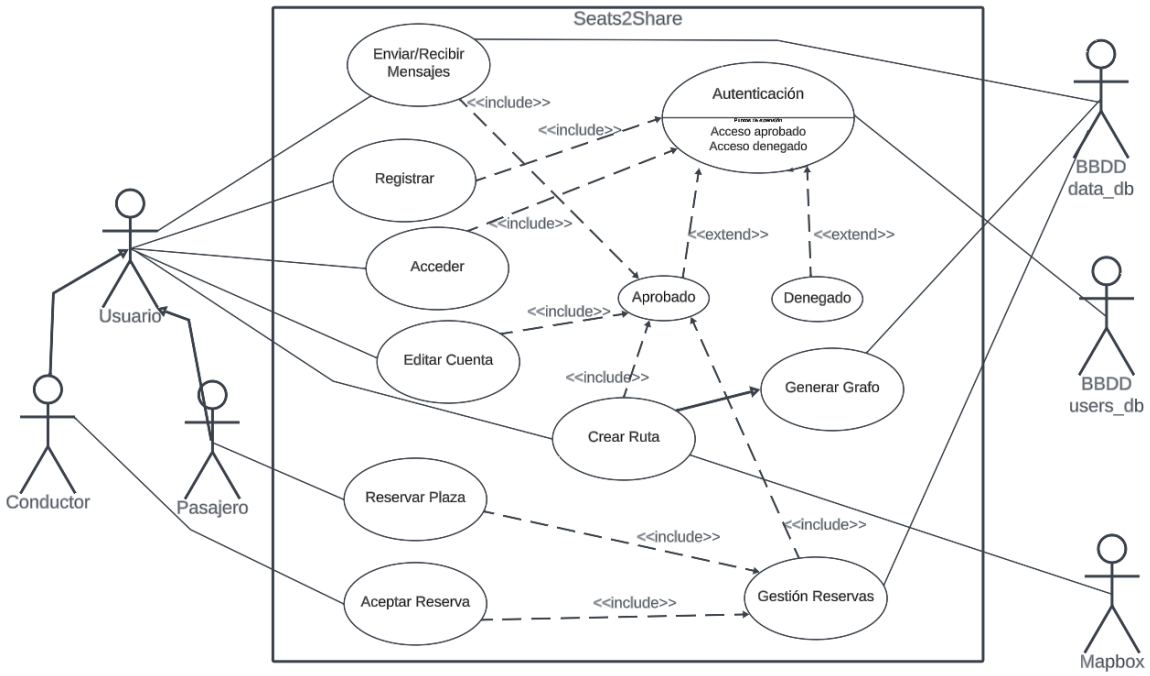


Figura 3.1: Caso de uso general.

Tabla 3.1: Escenarios de casos de uso.

—Caso de Uso—	————Acción————	—Tabla—
CU1	Registrar	3.2
CU2	Acceder	3.3
CU3	Editar Cuenta	3.4
CU4	Crear Ruta	3.5
CU5	Reservar Plaza	3.6
CU6	Aceptar Reserva	3.7
CU7	Enviar Mensajes	3.8
CU8	Recibir Mensajes	3.9

Tabla 3.2: Caso de uso: Registrar.

CU1:	Registrar.		
Actores:	Usuario, BBDD users_db.		
Dependencias:	RF1.1 Gestión de usuarios.		
Precondición:	No procede.		
Descripción:	El sistema deberá permitir al usuario registrarse y poder crear una cuenta para acceder al sistema.		
Secuencia normal:	**Paso**	**Acción**	
	1	El usuario intenta registrarse en el sistema.	
	2	El usuario ingresa nombre de usuario, correo y contraseña.	
	3	El sistema comprueba que los datos son válidos.	
	4	El sistema crea la cuenta de usuario y la almacena en la BBDD.	
	5	El sistema informa al usuario que se ha creado su cuenta.	
	6	El usuario tiene acceso al sistema.	
Postcondición:	El usuario tiene acceso al sistema y puede realizar operaciones.		
Excepciones:	**Paso**	**Acción**	
	3	Los datos no son válidos.	
		E1	El usuario introducido ya existe. El sistema informa al usuario del evento.
		E2	El correo introducido ya existe. El sistema informa al usuario del evento.
		E3	La contraseña no cumple las medidas de seguridad. El sistema informa al usuario del evento.
Comentarios:	No hay número máximo de intentos para registrarse.		

Tabla 3.3: Caso de uso: Acceder.

CU2:	Acceder.		
Actores:	Usuario, BBDD users_db.		
Dependencias:	RF1.2 Gestión de usuarios.		
Precondición:	RF1.1 El usuario debe tener una cuenta activa.		
Descripción:	El sistema deberá permitir al usuario acceder al sistema.		
Secuencia normal:	**Paso**	**Acción**	
	1	El usuario ingresa nombre de usuario y contraseña.	
	2	El sistema comprueba en la BBDD que los datos son válidos.	
	3	El sistema da acceso al usuario y le permite realizar operaciones.	
Postcondición:	El usuario tiene acceso al sistema y puede realizar operaciones.		
Excepciones:	**Paso**	**Acción**	
	2	Los datos no son válidos.	
		E1	El usuario introducido no existe. El sistema informa al usuario del evento.
		E2	La contraseña es incorrecta. El sistema informa al usuario del evento.
Comentarios:	No hay número máximo de intentos para acceder.		

Tabla 3.4: Caso de uso: Editar Cuenta.

CU3:	Editar Cuenta.	
Actores:	Usuario, BBDD users_db, BBDD data_db.	
Dependencias:	RF1.4 Gestión de usuarios RF2.2 Gestión del perfil.	
Precondición:	RF1.2 El usuario debe tener acceso al sistema.	
Descripción:	El sistema deberá permitir al usuario modificar sus datos de perfil.	
Secuencia normal:	**Paso**	**Acción**
	1	El usuario introduce los nuevos datos en su perfil.
	2	El sistema comprueba que los datos son válidos.
	3	El sistema lo almacena en la BBDD.
	4	El sistema informa al usuario del evento finalizado.
Postcondición:	Los datos del perfil del usuario han sido cambiados.	
Excepciones:	**Paso**	**Acción**
	2	Los datos no son válidos.
		E1 La nueva contraseña no cumple con las medidas de seguridad.
Comentarios:	No hay número máximo de intentos para cambiar datos.	

Tabla 3.5: Caso de uso: Crear Ruta.

CU4:	Crear Ruta.		
Actores:	Usuario, BBDD users_db, BBDD data_db, Mapbox.		
Dependencias:	RF3.1 Gestión de Rutas.		
Precondición:	RF1.2 El usuario debe tener acceso al sistema.		
Descripción:	El sistema deberá permitir al usuario crear una ruta.		
Secuencia normal:	**Paso**	**Acción**	
	1	El usuario establece el punto de origen.	
	2	El sistema consulta a Mapbox el punto de origen y lo dibuja.	
	3	El usuario marca el punto de destino.	
	4	El sistema consulta a Mapbox el punto de destino y lo dibuja.	
	5	El usuario guarda la ruta.	
	6	El sistema almacena la ruta en la BBDD.	
	7	El sistema informa al usuario del evento.	
Postcondición:	El usuario tiene una ruta creada y almacenada.		
Excepciones:	**Paso**	**Acción**	
	2	Las coordenadas no son válidas.	
		E1	El usuario marcó de forma incorrecta el punto de origen. No hubo desplazamiento en el mapa.
	3	Las coordenadas no son válidas.	
		E1	El usuario marcó de forma incorrecta el punto de destino. No hubo desplazamiento en el mapa.
Comentarios:	El usuario ha de realizar desplazamientos en el mapa antes de marcar puntos de origen o destino.		

Tabla 3.6: Caso de uso: Reservar Plaza.

CU5:	Reservar Plaza.		
Actores:	Pasajero, BBDD users_db, BBDD data_db.		
Dependencias:	RF4.2.1 Gestión de reservas.		
Precondición:	RF1.2 El usuario tiene acceso al sistema RF3.3.3 Gestión de rutas.		
Descripción:	El sistema deberá permitir al pasajero reservar una plaza a un conductor.		
Secuencia normal:	**Paso**	**Acción**	
	1	El pasajero tiene seleccionado a un conductor de los disponibles en su área.	
	2	El pasajero pulsa en reservar plaza.	
	3	El sistema consulta en la BBDD los datos de la ruta y del conductor.	
	4	El pasajero indica las condiciones de la reserva y envía la solicitud.	
	5	El sistema crea una reserva y la almacena en la BBDD.	
	6	El sistema informa al conductor que hay una reserva pendiente.	
	7	El sistema informa al pasajero que la reserva está enviada.	
Postcondición:	El pasajero tiene una reserva pendiente.		
Excepciones:	**Paso**	**Acción**	
	3	El conductor ha cambiado su oferta de plazas.	
		E1	Durante el transcurso de la reserva, el conductor cambió sus condiciones.
Comentarios:	No hay límite en el número de reservas que se pueden solicitar a un conductor.		

Tabla 3.7: Caso de uso: Aceptar Reserva.

CU6:	Aceptar Reserva.		
Actores:	Conductor, BBDD users_db, BBDD data_db.		
Dependencias:	RF4.2.2 Gestión de reservas.		
Precondición:	RF1.2 El usuario tiene acceso al sistema RF4.1 Gestion de reservas.		
Descripción:	El sistema deberá permitir al conductor aceptar una reserva pendiente.		
Secuencia normal:	**Paso**	**Acción**	
	1	El conductor consulta sus reservas pendientes.	
	2	Tras examinar las condiciones de la reserva decide aceptarla.	
	3	El sistema consulta en la BBDD la reserva y la modifica al estado aceptada.	
	4	El sistema informa al pasajero que su reserva ha sido aceptada.	
	5	En el panel de reservas del conductor se indica el nuevo estado.	
	6	El sistema informa al conductor el cambio de estado de su reserva.	
Postcondición:	El conductor tiene una reserva aceptada.		
Excepciones:	**Paso**	**Acción**	
	2	El pasajero ha anulado la reserva.	
		E1	Durante el transcurso de la reserva, el pasajero anuló la reserva.
Comentarios:	Sólo se puede aceptar una misma reserva una única vez. Existe la opción reactivar para reutilizar una misma reserva.		

Tabla 3.8: Caso de uso: Enviar Mensajes.

CU7:	Enviar Mensajes.		
Actores:	Usuario, BBDD users_db, BBDD data_db.		
Dependencias:	RF5.1 Mensajería RF5.2 Mensajería.		
Precondición:	RF1.2 El usuario tiene acceso al sistema.		
Descripción:	El sistema deberá permitir al usuario enviar mensajes a otros usuarios.		
Secuencia normal:	**Paso**	**Acción**	
	1	El usuario escribe el destinatario, asunto y cuerpo del mensaje.	
	2	El usuario pulsa enviar mensaje.	
	3	El sistema consulta el destinatario en la BBDD.	
	4	El sistema crea un mensaje y lo almacena en la BBDD.	
	5	El sistema indica al usuario que su mensaje ha sido enviado.	
	6	El sistema indica al receptor que ha recibido un mensaje.	
Postcondición:	El usuario ha enviado un mensaje.		
Excepciones:	**Paso**	**Acción**	
	3	El usuario no existe.	
		E1	El usuario indicó un nombre de usuario incorrecto.
	4	El mensaje no tiene asunto o cuerpo.	
		E1	El usuario no confeccionó correctamente el mensaje.
Comentarios:	Se pueden enviar mensajes ilimitados.		

Tabla 3.9: Caso de uso: Recibir Mensajes.

CU8:	Recibir Mensajes.	
Actores:	Usuario, BBDD users_db, BBDD data_db.	
Dependencias:	RF5.1 Mensajería RF5.3 Mensajería.	
Precondición:	RF1.2 El usuario tiene acceso al sistema RF5.2 Un usuario ha enviado un mensaje.	
Descripción:	El sistema deberá permitir al usuario recibir mensajes de otros usuarios.	
Secuencia normal:	**Paso**	**Acción**
	1	El usuario consulta el panel de mensajes.
	2	El sistema consulta en la BBDD si hay mensajes para el usuario.
	3	El sistema muestra en el panel de mensajes los mensajes recibidos.
	4	El usuario puede leer el mensaje recibido.
Postcondición:	El usuario ha recibido un mensaje.	
Excepciones:	No se producen.	
Comentarios:	Si no hay mensajes recibidos no se produce cambio alguno.	

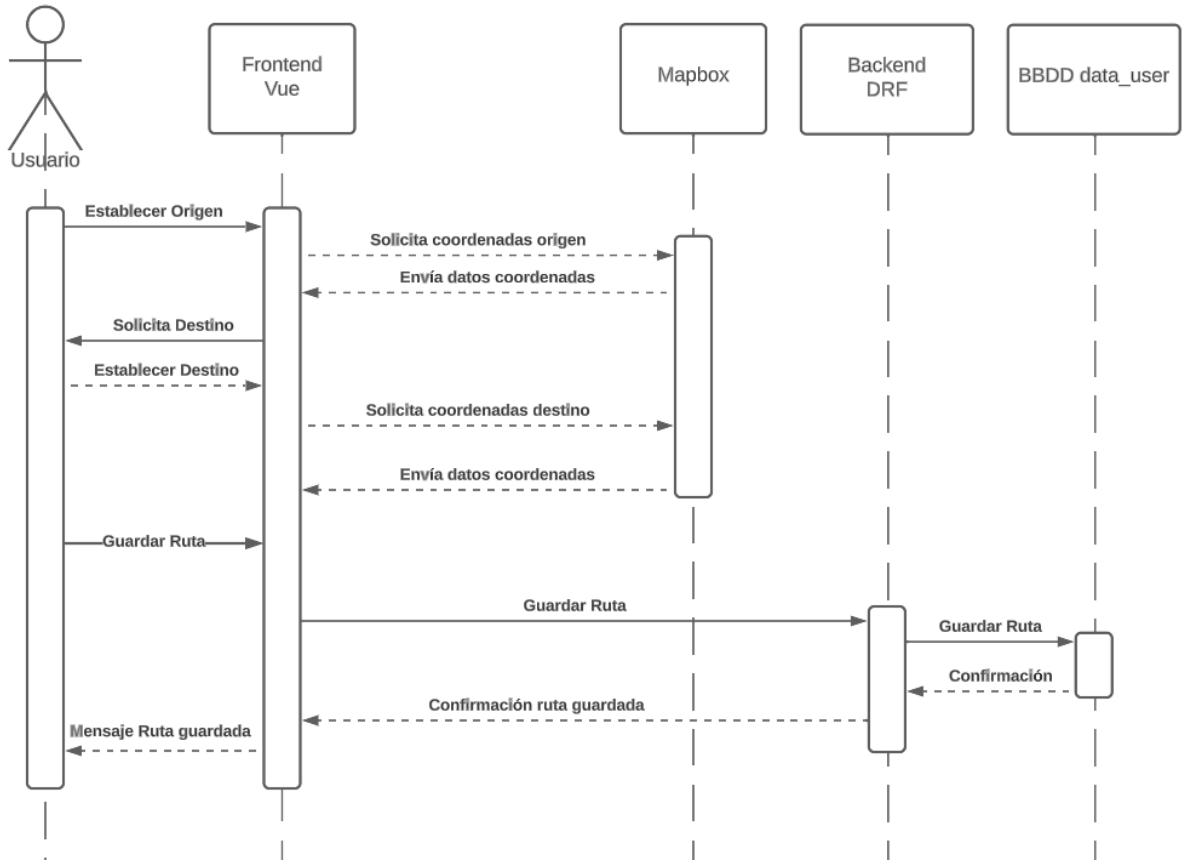


Figura 3.2: figure
Diagrama de secuencia: Crear Ruta.

3.4. Diagramas de secuencia

Los diagramas de secuencia son herramientas que nos muestran de forma gráfica los eventos que suceden por parte de todos los actores y sistemas en un caso de uso [22]. Se representan como un dibujo con líneas de tiempo y una sucesión de eventos ordenados de forma cronológica, con las actuaciones por parte de todos los actores y sistemas implicados.

En la Figura 3.2 tenemos el diagrama de secuencia correspondiente al **CU4** mostrado en la Tabla 3.5 y la Figura 3.3 representa el diagrama de secuencia correspondiente al caso de uso Generar Grafo, que no se incluyó en el listado anterior por no ser directamente activado por el usuario, sino que depende del éxito del **CU4** mostrado en la Tabla 3.5.

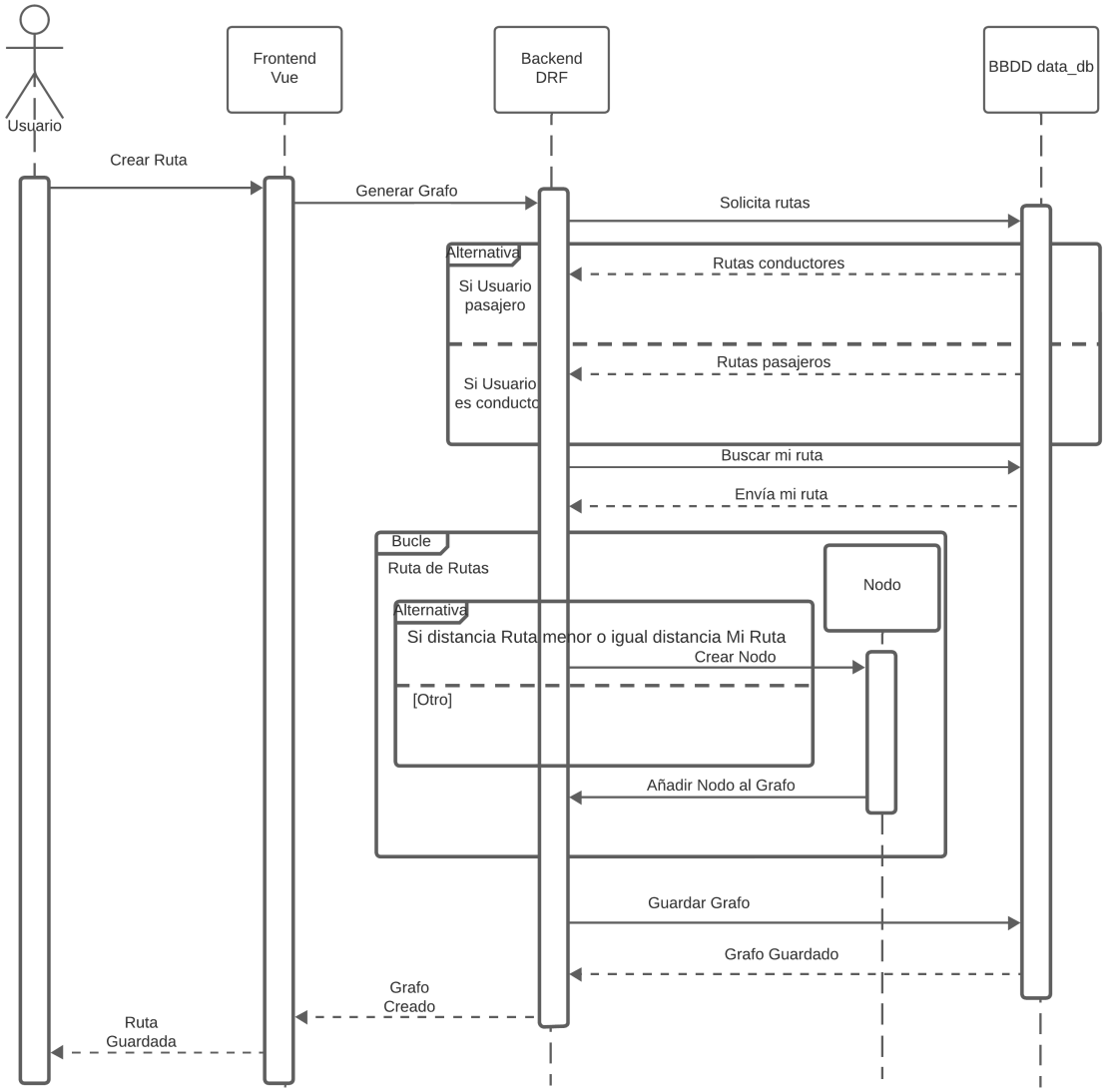


Figura 3.3: Diagrama de secuencia: Crear Grafo.

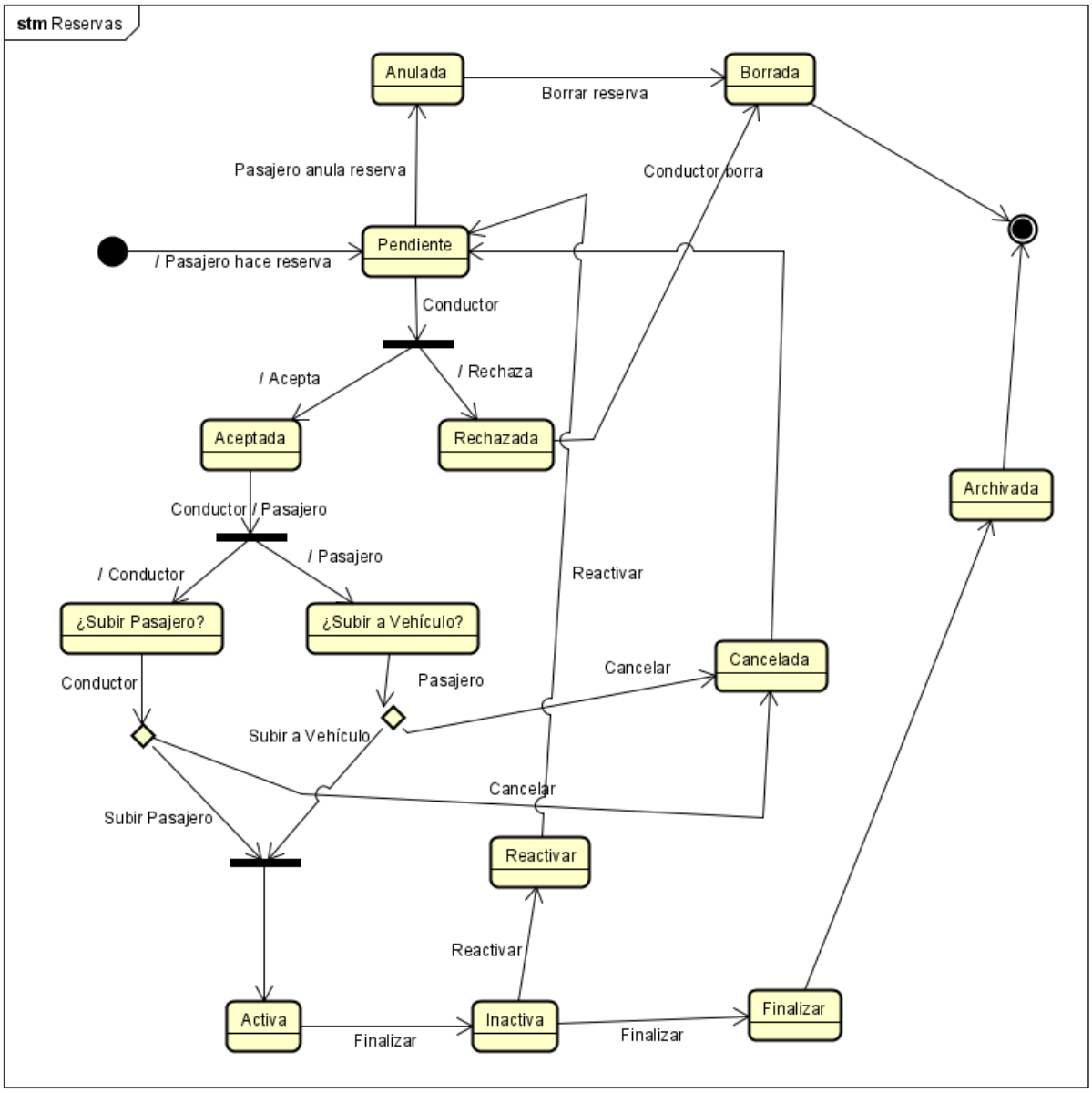


Figura 3.4: Diagrama de estados: Reserva.

3.5. Diagramas de estados

Los diagramas de estados nos muestran los eventos y distintos estados por los que un objeto puede comportarse en función de las acciones de los actores u otros elementos del sistema. Con los diagramas de estados podemos ver el ciclo de vida de un objeto [22].

En la Figura 3.1 uno de los casos de uso es Gestión de Reservas, el cual se va a describir en un diagrama de estados como muestra la Figura 3.4 para poder visualizar los distintos estados asociados a una reserva.

3.6. Conclusiones

Como se ha podido ver en este capítulo hemos mostrado varios análisis en los que se fundamenta el objetivo a cumplir en el desarrollo de este proyecto. La toma de requisitos es fundamental en caso de una errónea interpretación, por lo es uno de los principales aspectos a cuidar en análisis de un proyecto.

Los distintos casos de uso generales mostrados nos indican una idea global de las acciones que se pueden realizar en el proyecto Seats2Share. En los diagramas de secuencia hemos podido observar las acciones en cuanto a uno de los pilares del proyecto, las rutas y el grafo asociado a ellas. El diagrama de estados nos muestra el ciclo de vida de las reservas, en las que según la interacción de los usuarios en una misma reserva, ésta va tomando distintos estados hasta su finalización.

En conjunto, este es uno de los apartados en los que más tiempo hay que dedicar en el desarrollo de un proyecto. A lo largo de las distintas fases del proyecto, han sucedido varios cambios y refinamientos de los casos aquí analizados. La fase de análisis es fundamental para un correcto diseño del proyecto, cuestión que analizaremos en el próximo capítulo.

Capítulo 4

Diseño

4.1. Introducción

En este capítulo se hará un estudio de cómo se han diseñado cada uno de los componentes que forman el proyecto. Teniendo en cuenta que intervienen varios sistemas y tecnologías totalmente distintas. Se estudiará la elección de DRF frente a otros sistemas similares, así como el diseño para las distintas aplicaciones que forman el conjunto Backend.

Buscar una correcta funcionalidad y diseño ha sido más complicado de lo previsto, especialmente en el apartado Frontend. En un principio se tuvo en cuenta el estudio de dos variantes distintas de Frontend: Vue 3 y React. Integrar ambos Frontends con DRF fue uno de los principales problemas que más tiempo ha llevado resolver en el desarrollo del proyecto.

La integración con Vue 3 fue más sencilla que con React y ese fue el principal motivo por el que éste fue descartado. Finalizaremos con un estudio de los dos tipos de BBDD usados para el proyecto.

4.2. Estructura de la aplicación

El patrón de diseño Modelo Vista Controlador (MVC) es utilizado frecuentemente para *implementar interfaces de usuario, datos y lógica de control* [28]. Está más enfocado para el diseño de aplicaciones con Graphical User Interface (GUI), en las que el usuario interactúa con el entorno gráfico, activando elementos gráficos del mismo, los cuales se comunicarán con el controlador, que a su vez hará uso de la estructura lógica (modelo) y retornará una nueva vista.

En aplicaciones web, el patrón MVC tiene algunas variantes y para el proyecto se han utilizado:

- Model View Template (MVT). Este es el patrón de diseño que tenemos aplicado en DRF en la parte Backend. Tanto el modelo como la vista está definidos en sus correspondientes ficheros *models.py* y *views.py* para cada una de las aplicaciones. La parte *template* no es utilizada ya que eso está gestionado por parte del Frontend.

El controlador no está gestionado por el usuario, si no que forma parte del propio DRF que internamente realiza las operaciones sin control por nuestra parte [29].

- Model View ViewModel (MVVM). Este patrón de diseño es más específico para los Frameworks y en nuestro caso Vue 3 lo utiliza. Se basa en desacoplar la vista del modelo a través de un intermediario, el modelo de vista, que mediante enlaces de datos dobles interactúa con ellos, provocando el carácter reactivo de Vue 3 [30].

Por otro lado, la diversidad de tecnologías a emplear, así como su arquitectura cliente-servidor, hace que se tenga que diferenciar las responsabilidades de cada componente. En la Figura 4.1 se puede observar los diferentes componentes y las comunicaciones entre cada uno de ellos.

En la Figura 4.1 se observa que el diseño de la estructura es en tres niveles:

1. **Nivel de presentación:** Se corresponde con el Frontend.
2. **Nivel de aplicación:** Se corresponde con el Backend.
3. **Nivel de datos:** Formado por las BBDD.

Este diseño en tres niveles permite trabajar de forma separada cada uno de los niveles por un equipo de desarrollo, sin depender ni alterar el resto. La separación por niveles permite además más beneficios, como por ejemplo la escalabilidad en uno de los niveles sin afectar al resto, la mejora en cuanto a seguridad o incluso el añadir nuevas funcionalidades sin afectar al resto [31].

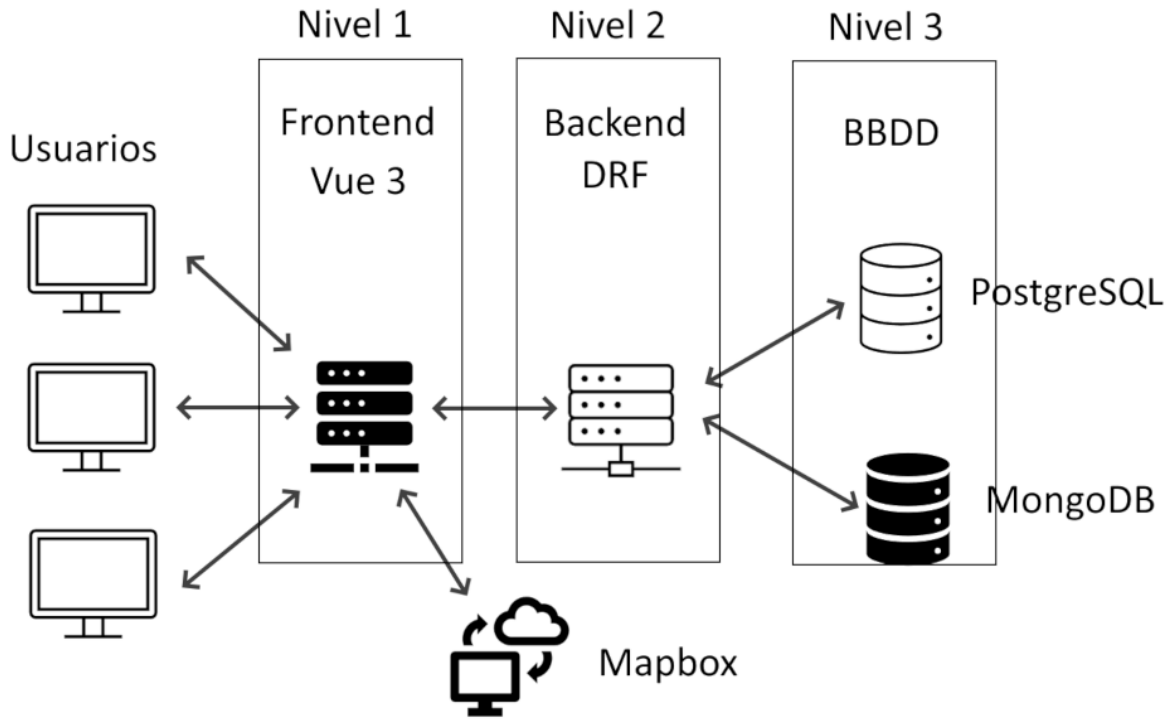


Figura 4.1: Esquema general de la aplicación.

Pero igualmente al trabajar en tres niveles, implica que las comunicaciones entre ellos han de estar correctamente definidas. Por eso se apostó como modelo de comunicación en el documento JSON. En una estructura típica cliente-servidor, el servidor cuando recibe las peticiones por parte del cliente, realiza las operaciones solicitadas y crea una respuesta como documento HTML y se lo envía al cliente [32].

En nuestro diseño al apostar por JSON en lugar de recibir un documento HTML recibimos un documento JSON y eso nos da la ventaja de no tener que recargar la página en el navegador, ya que una de las características del Frontend elegido es que es reactivo, por lo que los cambios se realizan sin recargar la página.

Las URL que comunican el Frontend con el Backend son invisibles al usuario, garantizando así una comunicación más segura. A este tipo de direcciones se les denomina Endpoint URL [33]. Las únicas URL que un usuario ve en su navegador son simples y meramente descriptivas, como por ejemplo:

https://seats2share.com/profile

En esa dirección de dominio ficticio, se accede al perfil del usuario pero todas las acciones que se realizan sobre el mismo no se ven reflejadas en el navegador, sino que son comunicaciones entre el Frontend y el Backend con las URL específicas para ello.

http://seats2share:8000/routes/nearest/

La dirección anterior con dominio ficticio es una petición POST HTTP que al ejecutarse en DRF devolverá en un documento JSON un Array con los nodos cercanos a mi posición. Esta dirección es invisible para el usuario y sólo es accesible entre el Frontend y el Backend.

4.3. Backend: API Django REST Framework

Pese a pertenecer al Nivel 2 o Nivel de aplicación, comenzamos con este componente porque es la base sobre la que se sustenta el proyecto Seats2Share. Un proyecto de este tipo necesita una serie de funcionalidades específicas que sólo se alcanzan con la construcción de una API REST. La comunicación entre distintas tecnologías se puede conseguir mediante uso de peticiones HTTP y sus verbos: GET, POST, PUT, PATCH, DELETE y Endpoint URL [34]. De esta forma la creación de una API es fundamental, garantizando así que cualquier dispositivo o tecnología pueda acceder a nuestra aplicación.

Existen muchos tipos de API REST y la elección de Django REST Framework (DRF) fue más por tener previamente conocimientos del lenguaje Python que de JavaScript, del cual hay también API REST. De hecho durante la fase de desarrollo inicial se trabajó con una API REST basada en NodeJS y se llegó a considerar su uso, ya que Vue 3 está basado en JavaScript y al haber el mismo entorno de desarrollo podría ser más beneficioso de cara a implementar funciones.

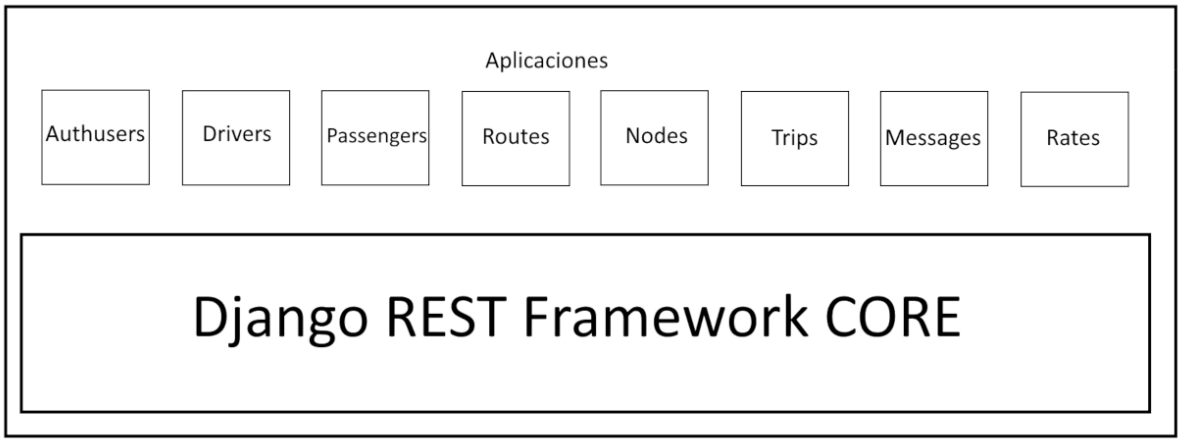


Figura 4.2: Esquema Django Rest Framework.

Finalmente DRF ofrecía un entorno más robusto y por eso fue el elegido para el proyecto. Django REST Framework [35] nos ofrece todo un conjunto de funcionalidades para poder crear nuestras aplicaciones y además puede usar distintas BBDD de forma sencilla y con apenas escribir un par de líneas de código. Para que DRF funcione necesita al menos crear una aplicación, que es la que se encargará de la parte *Modelo* y *Vista*. Recordemos que dentro del paradigma MVC el *Controlador* en este caso es algo interno de DRF y fuera de nuestro control. En el proyecto se han definido ocho aplicaciones distintas e independientes para que cada una de ellas se encargue de realizar las funciones y operaciones necesarias, la Figura 4.2 nos muestra el esquema.

- **Authusers**: En esta aplicación tenemos el modelo y vista de los usuarios.
- **Drivers**: En esta aplicación tenemos el modelo y vista de los conductores.
- **Passengers**: En esta aplicación tenemos el modelo y vista de los pasajeros.
- **Routes**: En esta aplicación tenemos el modelo y vista de las rutas.
- **Nodes**: En esta aplicación tenemos el modelo y vista de los nodos.
- **Trips**: En esta aplicación tenemos el modelo y vista de las reservas.
- **Messages**: En esta aplicación tenemos el modelo y vista de los mensajes.
- **Rates**: En esta aplicación tenemos el modelo y vista de las valoraciones.

Se pueden crear tantas aplicaciones como sea necesario, todas son independientes y por tanto el acoplamiento en el diseño es muy bajo. Todas las aplicaciones tienen la misma estructura de ficheros, veamos por ejemplo la estructura de la aplicación **Passengers**:

- **Carpeta** __pycache__.
- **Carpeta** migrations.
- **Fichero** __init__.py.

- **Fichero** admin.py.
- **Fichero** apps.py.
- **Fichero** models.py.
- **Fichero** passengers_db_router.py.
- **Fichero** serializers.py.
- **Fichero** urls.py.
- **Fichero** views.py.

A la hora de trabajar con DRF los ficheros que tenemos que codificar son principalmente *models.py*, *serializers.py*, *urls.py* y *views.py*. Los más importantes son *models.py* y *views.py* porque son precisamente los encargados de generar tanto el modelo como la vista de nuestra aplicación. Mostramos en el Código Fuente 4.1 el fichero *models.py* de la aplicación **Passengers**.

Código Fuente 4.1: Modelo Passengers.

```
from django.db import models
from django.utils import timezone
import uuid

class Passenger(models.Model):
    id = models.UUIDField(primary_key = True,
         default = uuid.uuid4,
         editable = False)
    passengerCreated = models.DateTimeField(default=timezone.now)
    userIDowner = models.CharField(max_length=100, null=False, unique=
   False)
    passengerStatus = models.BooleanField(default=True)
    passengerOrigin = models.JSONField(null=True)
    passengerDestination = models.JSONField(null=True)
    passengerStartTime = models.CharField(max_length=15, null=True)
    passengerEndTime = models.CharField(max_length=15, null=True)
    passengerWaitLimitStart = models.SmallIntegerField(null=True,
   default=4)
    passengerMaxDistanceAllowed = models.SmallIntegerField(null=True,
   default=2000)
    passengerConfigState = models.BooleanField(default=True)
    passengerSmoke=models.BooleanField(default=False)
    passengerCity=models.CharField(max_length=100, null=True)
    passengerPhone=models.CharField(max_length=50, null=True)
    passengerPremium=models.BooleanField(default=False)
    passengerVerified=models.BooleanField(default=False)
    objects = models.Manager()
```

El código expuesto en Código Fuente 4.1 nos sirve para crear el equivalente a una tabla en una BBDD y cada uno de los objetos que almacenaríamos en dicha BBDD se generan a través del fichero *serializers.py* del cual vemos su Código Fuente 4.2.

Código Fuente 4.2: Serializer Passengers.

```
from rest_framework import serializers
from .models import *

class PassengerSerializer(serializers.ModelSerializer):
    class Meta:
        model=Passenger
        fields= '__all__'

    def create(self, validated_data):

        return Passenger.objects.using("data_db").create(**
    validated_data)
```

Se puede observar que con muy pocas líneas de código los objetos se crean y pueden almacenar en la BBDD. De dicho proceso de conversión se encarga el propio DRF sin interacción por nuestra parte. Pero para acceder a dicho proceso necesitamos los Endpoint URL que hay que crear para cada aplicación. En el Código Fuente 4.3 podemos ver los Endpoint URL para dicha aplicación y las llamadas a las funciones de vistas correspondientes. Se pueden crear tantas Endpoint URL como uno desee siempre y cuando haya una vista para cada una de ellas.

Código Fuente 4.3: Urls Passengers.

```
from django.urls import path
from .views import *


urlpatterns = [

    path('passengers/', passengers_api_list_view, name='
    passengers_api_list'),
    path('adddpassenger/', add_passenger_api_view, name='
    add_passenger_api_view'),
    path('detail/',passenger_api_detail_view, name='
    passenger_api_detail_view'),
    path('update/', passenger_update_api_view, name='
    passenger_update_api_view')


]
```

Finalmente vemos el Código Fuente 4.4 donde tenemos el contenido del fichero *views.py* el cual contiene las vistas de la aplicación, que son las que nos permiten comunicarnos con la BBDD y obtener la respuesta deseada.

Código Fuente 4.4: Vistas Passengers.

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
from .models import Passenger
from .serializers import PassengerSerializer
from django.db.models.query_utils import Q
```

```
9
10 # obtenemos todos los pasajeros de la bbdd
11 @api_view(['GET'])
12 def passengers_api_list_view(request):
13     passengers = Passenger.objects.using("data_db").all()
14     serializer = PassengerSerializer(passengers, many=True)
15     return Response(serializer.data, status=status.HTTP_200_OK)
16
17 # creamos el pasajero que hemos recibido en request
18 @api_view(['POST'])
19 def add_passenger_api_view(request):
20     passenger_serializer = PassengerSerializer(data = request.data)
21     if passenger_serializer.is_valid():
22         passenger_serializer.create(passenger_serializer.data)
23         return Response("success", status=status.HTTP_200_OK)
24     return Response(passenger_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
25
26 # obtenemos el pasajero con username que buscamos
27 @api_view(['POST'])
28 def passenger_api_detail_view(request):
29     passengers = Passenger.objects.using("data_db").filter(userIDowner=
    request.data['userIDowner']).first()
30     serializer = PassengerSerializer(passengers)
31     return Response(serializer.data)
32
33 # modificamos los datos del pasajero con username indicado en request
34 @api_view(['PUT'])
35 def passenger_update_api_view(request):
36     passenger = Passenger.objects.using("data_db").filter(userIDowner=
    request.data['userIDowner']).first()
37     pax_serializer = PassengerSerializer(passenger, data=request.data)
38     if(pax_serializer.is_valid()):
39         pax_serializer.save()
40         return Response("success", status=status.HTTP_200_OK)
41     return Response(pax_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
```

Para el resto de aplicaciones la estructura y el código es muy similar. También hay que destacar que a diferencia de lo que uno podría esperar, no se emplea GET para obtener el detalle del pasajero, sino que es mediante una petición POST. El motivo es precisamente evitar incluir en la URL la consulta, además que desde el lado Frontend es más cómodo generar este tipo de peticiones mediante Axios, pero eso se verá en la siguiente sección.

El Código Fuente 4.5 es una función de las vistas definidas en la aplicación *nodes*, la cual nos permite crear el grafo que contiene todos los usuarios más cercanos de acuerdo a las distancias establecidas por ellos. Su secuencia de acciones es la siguiente:

- Si el usuario es pasajero se cargan los conductores, si es conductor se cargan los pasajeros desde la BBDD.
- Se obtiene la distancia máxima del usuario consultando en la BBDD su perfil.
- Se realiza un bucle entre todos los usuarios, ya sea conductores o pasajeros recibidos.
- Se realizan las conversiones necesarias, primero la distancia se convierte a metros

y las coordenadas se convierten a decimal ya que MongoDB utiliza un formato de almacenamiento de números en decimal distinto al que usa Python.

- Se calcula la distancia entre ellos con las coordenadas del usuario y las del candidato que muestra el bucle mediante la formula del semiverseno.
- Se crea un nodo temporal con todos los campos necesarios y se compara si la distancia obtenida mediante la fórmula del semiverseno está dentro de los límites.
- Si se cumplen las condiciones de la distancia se almacena el nodo en la BBDD
- Finaliza el bucle.

Código Fuente 4.5: Vista Crear Nodo.

```
# Esta vista crea el grafo de usuarios cercanos
@api_view(['POST'])
def add_node_api_view(request):

    if(request.data['nodeType']=="Pasajero"):

        drivers=Route.objects.using("data_db").filter(routeType="Conductor").all()
        print(drivers)
    else:
        drivers=Route.objects.using("data_db").filter(routeType="Pasajero").all()

    pax=Route.objects.using("data_db").filter(userIDowner=request.data['userIDowner']).first()
    for dis in drivers:
        if(request.data['nodeType']=="Pasajero"):
            dDistance = Driver.objects.using("data_db").filter(userIDowner=dis.userIDowner).first()
            maxDistance=dDistance.driverMaxDistanceAllowed*1000.00
        else:
            dDistance= Passenger.objects.using("data_db").filter(userIDowner=dis.userIDowner).first()
            maxDistance=dDistance.passengerMaxDistanceAllowed*1000.00

        paxPosition=(pax.routeOriginLat.to_decimal(),pax.routeOriginLng.to_decimal())
        driverPosition=(dis.routeOriginLat.to_decimal(),dis.routeOriginLng.to_decimal())
        realDistance=hs.haversine(paxPosition,driverPosition, unit=Unit.METERS)
        tempNode={
            "userIDowner":request.data['userIDowner'],
            "nodeUser":dis.userIDowner,
            "nodeOriginLat":dis.routeOriginLat.to_decimal(),
            "nodeOriginLng":dis.routeOriginLng.to_decimal(),
            "nodeDestLat":dis.routeDestLat.to_decimal(),
            "nodeDestLng":dis.routeDestLng.to_decimal(),
            "nodeDistance":realDistance,
            "nodeType":request.data['nodeType']
        }

```

```
35          if(realDistance<=maxDistance):
36               node_serializer = NodeSerializer(data = tempNode)
37               if node_serializer.is_valid():
38                  node_serializer.create(node_serializer.data)
39
40      return Response("success", status=status.HTTP_200_OK)
```

En resumen, la secuencia de una petición en DRF sería la siguiente:

1. Se accede a una Endpoint URL desde el Frontend.
2. Desde la URL se llama a la vista asociada.
3. En la vista asociada se llama al modelo y serializer de la aplicación.
4. Se hace la consulta a la BBDD correspondiente.
5. La BBDD responde y el serializer traduce la información a JSON.
6. Se le da una respuesta o promesa *Response* que recibirá el Frontend.

Es de destacar, que en el fichero de las vistas no aparece nada de SQL y esta es la ventaja del uso de una API REST, en la que se interactúa con una BBDD sin tener que escribir ningún tipo de sentencias SQL. El propio DRF en este caso se encarga de la traducción de las consultas con las BBDD de forma transparente, aunque con algunas limitaciones en el caso de MongoDB si hay que hacer búsquedas relacionales. Motivo por el que se tuvo que usar como BBDD PostgreSQL cuando esas búsquedas son necesarias.

Finalmente, la creación de una API nos permite que cualquier dispositivo pueda acceder a nuestras aplicaciones, así si por ejemplo nos planteamos realizar el **RNC7**: Creación de una aplicación Android o iOS, la API ya creada está lista para recibir las peticiones HTTP correspondientes, sin necesidad de tener que cambiar ni una línea de código en nuestra API.

4.4. Frontend: Vue 3

El Frontend es la parte del sistema que permite al usuario interactuar y visualizar los resultados de las peticiones. Pertenece al Nivel 1 por ser lo primero que se encuentra el usuario al interactuar con una aplicación. Para el proyecto se tuvo en cuenta varios Frontends, aunque finalmente había que decir sobre dos:

- Vue 3.
- React.

Ambos están basados en JavaScript y son muy usados en la actualidad por parte de diversos operadores. Por parte de React, Facebook está dando su apoyo y mantenimiento, existe una comunidad muy extensa de colaboradores, al ser una librería de código abierto y hay infinidad de componentes para usar. Por parte de Vue 3 no hay ningún gigante informático, pero también tiene su comunidad de colaboradores, es de código abierto y posee también muchos componentes para usar.

Al no disponer de conocimientos en JavaScript comencé con ambos para integrar el acceso a la API REST y con el que más cómodo me sentía y mejores resultados obtuve fue con Vue 3. Una de las características de Vue 3 es que es un Framework de una única página, es decir, sólo hay un único fichero HTML, sobre el que se van añadiendo el resto de componentes y el uso de un DOM virtual. Los componentes son pequeñas instancias de Vue 3 que pueden ser creadas y reutilizadas como elementos que se añaden a la aplicación.

En Vue 3 sólo existe una única aplicación, que por convenio se le llama *App* y se añade al HTML principal. Sobre esa *App* se irán añadiendo, reutilizando o eliminando cada uno de los componentes que forman el conjunto de la aplicación. No es necesario recargar la página, ya que Vue 3 la dibuja con los componentes que estén en ese momento. Esto permite crear una jerarquía de URL asociadas a ficheros, que al ser recorridos en el navegador, cargarán las vistas con sus correspondientes componentes, dando la sensación de navegar por distintas páginas sin necesidad de recarga alguna.

En el diseño de Seats2Share hay dos tipos de jerarquía:

- Sin estar autenticado. Acceso libre.
- Requiere de autenticación.

La página principal, el formulario de registro y el de login o acceso son las únicas vistas que no requieren estar autenticado en el sistema para ser visualizadas.

En cambio el resto de páginas sí que requiere estar autenticado:

- /profile Para ver el perfil del usuario.
- /config/user La vista en la que se puede cambiar la configuración.
- /livemap Esta vista muestra en un mapa todos los usuarios activos.
- /messages En esta vista vemos los mensajes recibidos y podemos enviar mensajes.
- /trips En esta vista vemos el panel de reservas.

La creación de este tipo de direcciones URL es muy sencilla y se hace en pocas líneas de código tal y como se puede ver en extracto de Código Fuente 4.6 que pertenece al fichero *router/index.js* que es donde se configuran las distintas URL a usar por parte del Frontend.

Código Fuente 4.6: Router index.js.

```
1  {
2    path: '/register',
3    name: 'registeru',
4    component: RegAccessView,
5    meta:{
6      needAuth:false
7    }
8  },
```

```
9   {
10    path: '/profile',
11    name: 'profileview',
12    component: ProfileUserView,
13    meta:{
14      needAuth:true
15    }
16  },
17  {
18    path:'/config/user',
19    name:'configuser',
20    component: ConfigUser,
21    meta:{
22      needAuth:true
23    }
24  },
25  {
26    path: '/livemap',
27    name: 'livemap',
28    component: LiveMapView,
29    meta:{
30      needAuth:true
31    }
32  },
```

Entramos ahora en el diseño en sí de los componentes que forman el Frontend del proyecto, la Figura 4.3 nos muestra la vista correspondiente al perfil de un usuario pasajero, justamente en el momento previo a reservar una plaza. En dicha Figura se pueden observar varios componentes delimitados por recuadros de diferente color. Así tenemos los siguientes componentes:

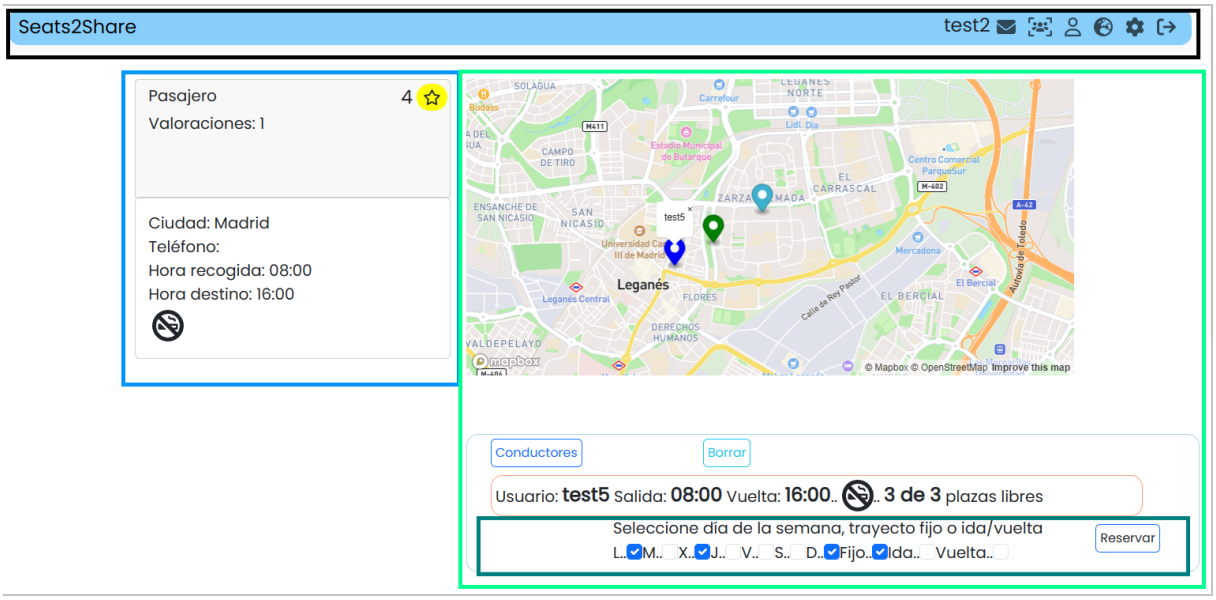


Figura 4.3: Componentes en página perfil.

- Rectángulo negro: *Header* o cabecera.
- Rectángulo azul: Panel lateral izquierdo, muestra información del pasajero, valoración y datos.

- Rectángulo verde: Este componente a su vez contiene otro. En la parte superior contiene por un lado el mapa que se obtiene a través de Mapbox y en la parte inferior los datos del conductor seleccionado.
- Rectángulo turquesa: Este componente forma parte del anterior y permite crear una reserva sobre el conductor seleccionado.

A la hora de confeccionar un componente Vue 3 hay dos formas de establecer el apartado *setup* del mismo. En el proyecto se ha apostado por el más recomendado por parte de Vue 3 que es el Composition API [36], por varios motivos, entre ellos la lógica es mucho más clara, está mejor estructurado y es el recomendado por parte de la documentación oficial [36].

Código Fuente 4.7: Componente MessagesComp.Vue.

```
<template>
    <div class="container">
        <MessagesRecComp/>
    </div>
</template>

<script setup>
import MessagesRecComp from './MessagesRecComp.vue';
</script>
```

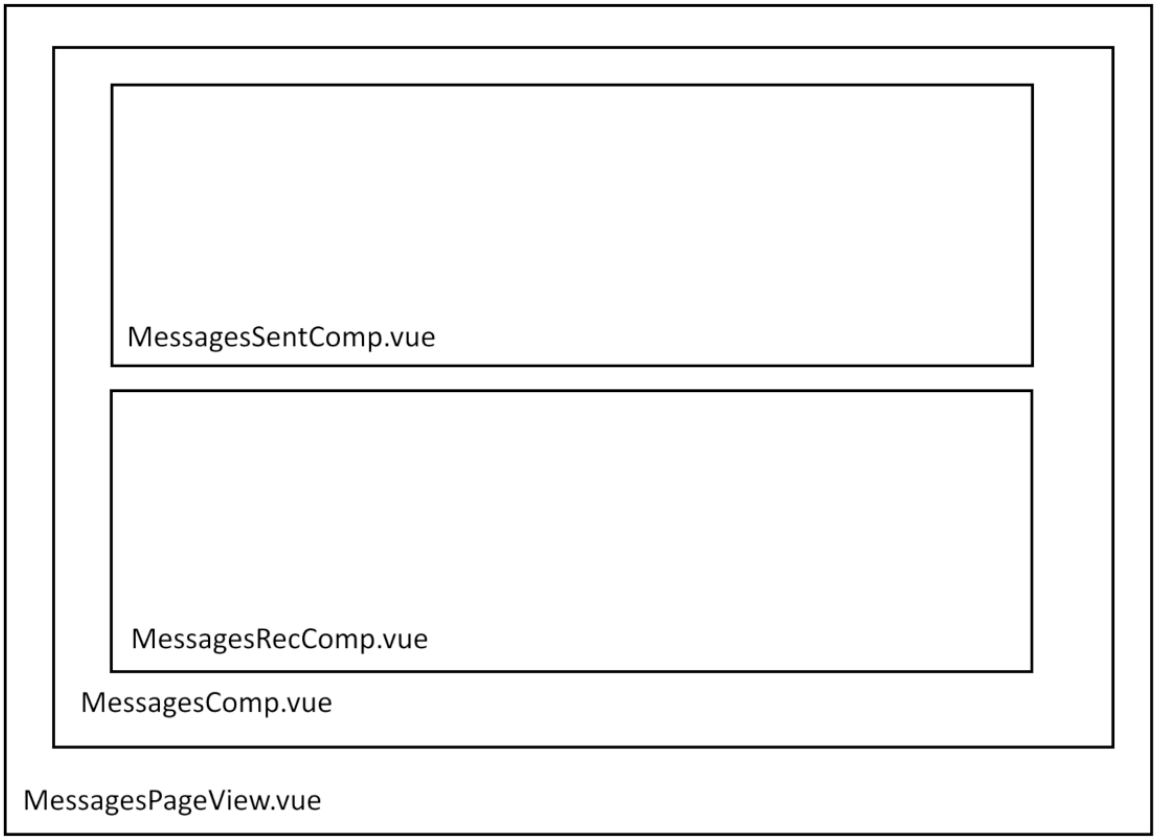


Figura 4.4: Componentes de MessagesPageView.vue.

En un componente tendremos los elementos que se muestran en el Código Fuente 4.7 perteneciente en este caso al componente que correspondería a los mensajes recibidos. Si vemos el Código Fuente 4.7 podemos observar que lo que hay entre las etiquetas HTML *template*, es un contenedor que a su vez tiene otro componente, en este caso *MessagesRecComp* que se importa desde el *Composition API* tal y como se ve.

Así la página que el usuario ve está formada por componentes, que a su vez cargan a otros componentes hasta el nivel más bajo. Esto hace que se pueda modificar sin problemas los componentes, desacoplando el código y fomentando la reutilización de componentes. En la Figura 4.4 podemos ver las capas que forman los distintos componentes que forman parte del componente *MessagesPageView.vue* que es justamente el que carga el panel de mensajes cuando se accede a él desde la cabecera.

Además, en este caso, tanto el componente *MessagesSentComp* como el *MessagesRecComp* en su parte correspondiente a *script setup* hacen llamadas a la API REST para realizar las operaciones sobre los mensajes, ya sea, cargar mensajes, enviar o borrar. Hay que destacar que la comunicación y envío de información entre los componentes es posible. Se hace de forma vertical bien desde el componente padre al hijo o bien desde el hijo al padre, defineProps() y defineEmits() [37] respectivamente. En el proyecto se ha hecho uso de esta metodología pero únicamente de padre a hijos, es decir, usando *defineProps()*.

También podemos pasar información entre componentes de distintos modelos de vista, que se corresponderían con sus equivalentes a distintas direcciones URL como vimos en el Código Fuente 4.6. Para ello tenemos que usar un almacén temporal, donde el componente guarda los datos y cualquier otro componente puede acceder a ellos aunque pertenezca a otro modelo de vista. La solución empleada en el proyecto ha sido usando la librería Pinia [38] que nos permite crear nuestro propio almacén para que cualquier componente pueda hacer uso de sus elementos.

Siguiendo con los componentes de las figuras anteriores el Código Fuente 4.8 muestra un extracto del componente *MessagesRecComp.vue* con las funciones que marcan como leído un mensaje y recuperan los mensajes recibidos.

Código Fuente 4.8: Funciones en Componente MessagesRecComp.

```
1  const readMsgBtn=async (msg)=>{
2      msg.messageRead=true
3      const readMSG= await msgService.updateMsg(msg)
4  }
5  const loadMsgRec = async ()=>{
6      allMsgRec.value= await msgService.loadMsgRec(
           loginstore.getUsername)
7      setTimeout(()=>{
8          reloadMsgRec()
9              }, 10000);
10 }
```

Se observa que se utiliza a *msgService* para las funciones *updateMsg* y *loadMsgRec*. *msgService* es una constante definida al crear el componente, que contiene el servicio creado para hacer las peticiones a la API, en este caso *MessagesService.js*. En el Código Fuente 4.9 se puede ver las dos llamadas a que hace referencia el Código Fuente 4.8.

Código Fuente 4.9: Servicio mensajes.

```
async updateMsg(msg){
        try{
            await axios
                    .put('http://127.0.0.1:8000/messages/updatemsg/'
                       ,msg)
                    .then((response) =>{
                        console.log(response);
                    });
            return true
        }catch(error)
        {
            console.log(error)
        }
    }
    async loadMsgRec(username){
        try{
            const res = await fetch('http://127.0.0.1:8000/messages/
               receivemsg/',{
                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    "toMessage":username
                })
            })
            const response = await res.json()
            return response
        }catch(error)
        {
            console.log(error)
        }
    }
```

Las llamadas a la API se realizan bien usando *fetch* que es propio de JavaScript o mediante Axios. En este caso la actualización mediante PUT ha sido vía Axios. Para realizar GET o PUT me ha resultado más cómodo usar Axios pero para las peticiones POST, el uso de *fetch* ha sido mucho más productivo y el código más estructurado. En general la estructura del Frontend en este proyecto contiene muy pocas páginas. La más compleja de todas es la correspondiente al perfil, ya que contiene los componentes más importantes y en concreto, la que se corresponde con las vistas de los mapas.

Cargar los mapas de Mapbox es muy sencillo, al estar integrado como librería, únicamente importando en el componente el correspondiente a Mapbox y crear un objeto *Map*. Sobre ese objeto *Map* se pueden ir añadiendo los marcadores, coordenadas y más objetos, si bien para el propósito del proyecto nos hemos limitado a los marcadores, *Popups* y trayectos. El paso más complejo ha sido generar el trayecto, para lo que hay que tener en cuenta los marcadores origen y destino. Una vez obtenidos serán usados en la función *calculateRoute*, de donde obtendremos la respuesta del servicio de Mapbox con el recorrido listo para ser almacenado en la BBDD. El Código Fuente 4.10 muestra la función *calculateRoute* del componente *MapBoxUserMap.vue*

Código Fuente 4.10: Calcular Ruta.

```
const calculateRoute=async ()=>{
    routeM="https://api.mapbox.com/directions/v5/mapbox/driving/"+
    markerO.getLngLat().lng+","+markerO.getLngLat().lat+";"+
    markerD.getLngLat().lng+","+markerD.getLngLat().lat+
    "?geometries=geojson&access_token="+mapboxgl.accessToken;
    const findRoute = await fetch(
        routeM,
        {method: 'GET'}
    );

    const json = await findRoute.json();
    const data = json.routes[0];
    const route = data.geometry.coordinates;


    const geojson = {
        type: 'Feature',
        properties:{},
        geometry:{
            type:'LineString',

            coordinates:route
        }
    };
    if (mapa.getSource('route')) {
    mapa.getSource('route').setData(geojson);
    }
    else {
    mapa.addLayer({
      id: 'route',
      type: 'line',
      source: {
        type: 'geojson',
        data: geojson
      },
      layout: {
        'line-join': 'round',
        'line-cap': 'round'
      },
      paint: {
        'line-color': '#3887be',
        'line-width': 5,
        'line-opacity': 0.75
      }
    });



    Array.from(geojson.geometry.coordinates).forEach((value)=>{

        fullPath.push([
            value[0],value[1]
        ]

        )
```

```
56      })
57
58      }
59  }
```

Destacar el uso de Bootstrap para darle un toque más vistoso a las páginas, ya que aunque con Vue 3 los colores son interesantes, queda mucho mejor los que ofrece Bootstrap. Igualmente se ha usado algo de CSS en algunas páginas, para dar efectos o mejoras visuales, si bien no se ha dedicado el tiempo necesario para hacer algo mucho más formal, principalmente por falta de formación sobre ese tema, cosa que expondré en las conclusiones finales.

En la Figura 4.5 se pueden observar algunos de los componentes de Bootstrap tanto en los *checkbox* como botones.

Figura 4.5: Vista Reservas.

4.5. Bases de datos

Las BBDD son uno de los componentes más importantes en un proyecto de este tipo, especialmente porque aspectos como la autenticación y almacenamiento de datos son muy importantes para el buen funcionamiento.

Pero gracias al uso de una API REST, en este caso basada en DRF, la elección de qué tipo de BBDD es mucho más fácil. Como se vio en la Sección 2, DRF permite el uso de múltiples BBDD y mediante el uso del *driver* correspondiente, en muy pocas líneas de código podemos cambiar sobre una u otra. Los esquemas y consultas SQL quedan todas en mano de DRF al hacer las migraciones de los modelos, liberando de esa tarea al desarrollador.

En el proyecto Seats2Share se usan dos tipos de BBDD:

- MongoDB.
- PostgreSQL.

4.5.1. MongoDB

MongoDB [39] es una BBDD NoSQL basada en documentos y era el formato elegido inicialmente como soporte de datos del proyecto Seats2Share. Desde el principio del desarrollo se estuvo trabajando con este tipo de BBDD, creando para cada aplicación de DRF una colección de documentos. Llamaremos colección al equivalente a las tablas de una BBDD relacional [40].

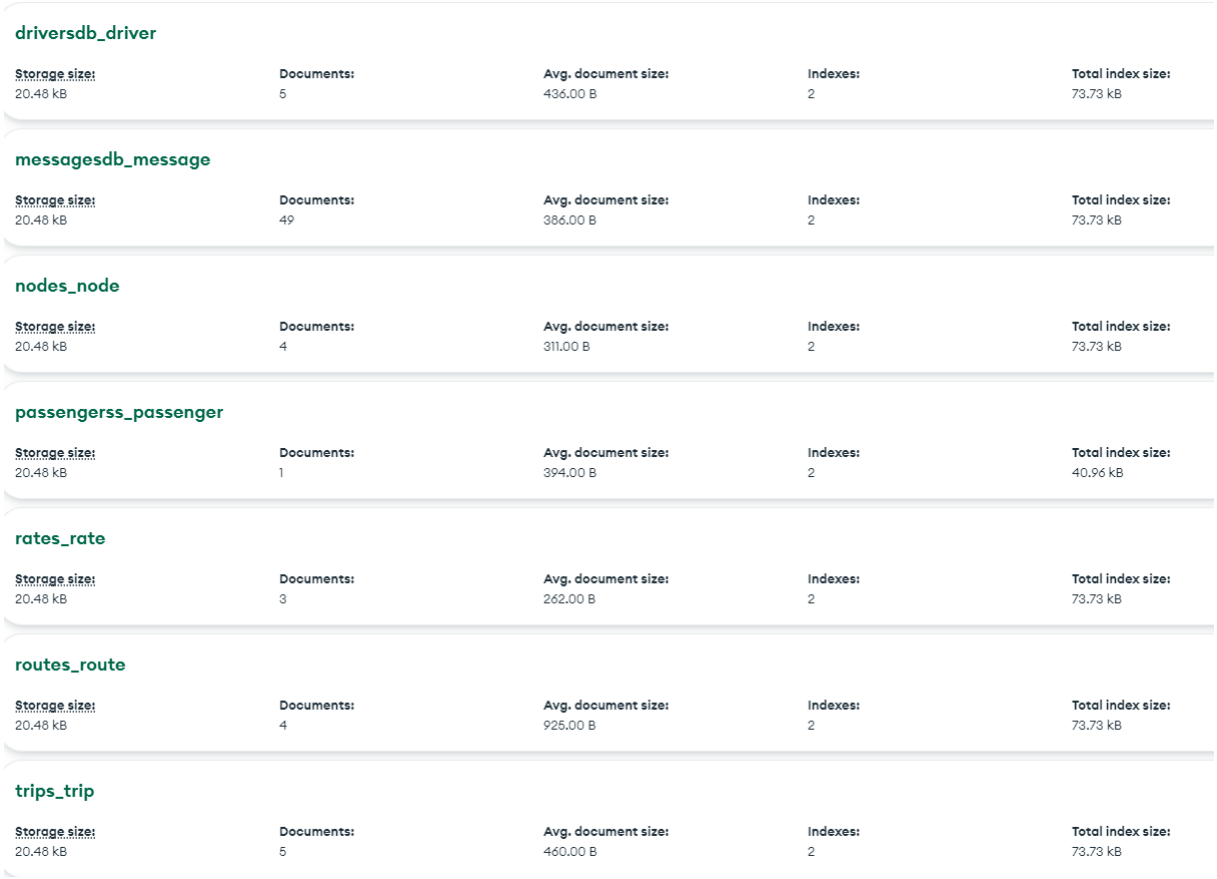


Figura 4.6: Colecciones en MongoDB.

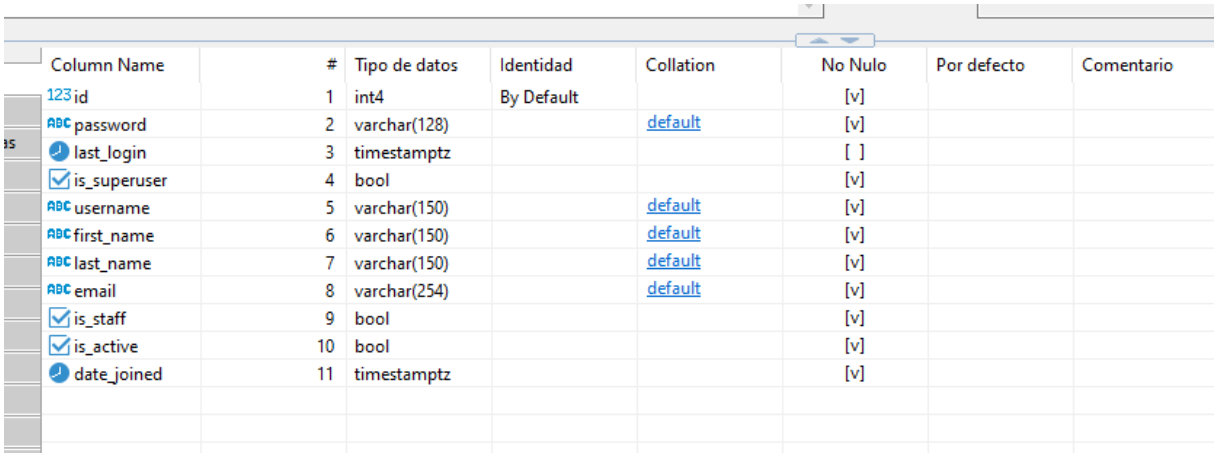
Por lo que nuestra BBDD contiene siete colecciones, una por cada aplicación como las vistas en la Figura 4.2 salvo la correspondiente a **Authsusers**, tal y como se muestra en la Figura 4.6.

A la hora de configurar MongoDB en DRF es tan sencillo como lo que aparece en el Código Fuente 4.11 anotado en el archivo *settings.py* de DRF.

Código Fuente 4.11: Extracto código para MongoDB en DRF

```
'data_db': {
        'ENGINE': 'djongo',
        'NAME': 'data_db',
        'CLIENT':{
            'host':'mongodb://localhost:27017/'
            #'host':'mongodb+srv://anavas99:anavas99@datadb.ivkrmvg.
    mongodb.net/?retryWrites=true&w=majority'
        }
     },
```

El comentar o descomentar el *host* nos permite cambiar MongoDB de modo local al modo online basado en la nube de MongoDB Atlas [41]. Se puede observar que trabajar con BBDD es bastante fácil si usamos DRF.



Column Name	#	Tipo de datos	Identidad	Collation	No Nulo	Por defecto	Comentario
id	1	int4	By Default		[v]		
password	2	varchar(128)		default	[v]		
last_login	3	timestamptz			[ ]		
is_superuser	4	bool			[v]		
username	5	varchar(150)		default	[v]		
first_name	6	varchar(150)		default	[v]		
last_name	7	varchar(150)		default	[v]		
email	8	varchar(254)		default	[v]		
is_staff	9	bool			[v]		
is_active	10	bool			[v]		
date_joined	11	timestamptz			[v]		

Figura 4.7: Tabla y campos en BBDD users_db.

4.5.2. PostgreSQL

El uso de **PostgreSQL** [42] ha sido obligado al integrar la autenticación en el proyecto, a pesar de que el sistema de autenticación es el integrado en DRF basado en JSON Web Token (JWT), no funcionaba bajo MongoDB. Por lo que la solución más práctica fue aprovechar el uso de múltiples BBDD de DRF y para los datos de los usuarios usar una BBDD diferente, que de hecho estaría aislada del resto de datos, garantizando así su seguridad y accesos únicamente cuando es requerido.

Esto supone que para el despliegue hay que tener en cuenta que es necesario un servidor específico para la BBDD de PostgreSQL. En el desarrollo se ha estado usando un servicio gratuito, alojado en la plataforma Render [43] a modo de pruebas, para confirmar que su despliegue es funcional y que al igual que con MongoDB, integrarlo en DRF es igual de fácil.

En este caso sí tenemos tabla al ser una BBDD relacional y los campos al usar el propio modelo de autenticación de DRF el JWT son los que se pueden observar en la Figura 4.7 que son los genéricos para usuarios. En cualquier caso, el uso de esta BBDD ha sido por utilizar JWT como medio de autenticación. El **RNC1** está relacionado con este asunto, ya que al pasar a otro sistema de autenticación habría que reconsiderar si seguir usando esta BBDD o no.

4.6. Conclusiones

Este ha sido un capítulo muy extenso en el que se han expuesto los argumentos para el diseño de Seats2Share. Desde la estructura en tres niveles, las tecnologías elegidas para cada uno de ellos y las decisiones tomadas para solventar los obstáculos encontrados. La elección de usar una API REST ya consolidada permite tener una estructura lista para ser accesible por cualquier dispositivo de forma rápida y fácil. Sin lugar a dudas donde más complicaciones nos hemos encontrado para seguir las directrices del análisis ha sido con Vue 3,ya que ha supuesto todo un reto encajar todos los componentes en el diseño previsto.

Capítulo 5

Implementación y pruebas

5.1. Introducción

En los capítulos anteriores nos hemos centrado en el análisis y diseño del proyecto, teniendo en cuenta las distintas tecnologías aplicadas, así como su integración a nivel de diseño. En este capítulo veremos el resultado de aplicar tanto el análisis como el diseño previsto, comenzando con las herramientas usadas durante el desarrollo del proyecto, cómo se ponen en funcionamiento el proyecto en sí y finalizando con múltiples pruebas de validación en las que se mostrará el cumplimiento de los requisitos funcionales.

5.2. Entorno de desarrollo

En esta sección iremos describiendo los distintos elementos usados para el desarrollo de este proyecto, tanto a nivel software como a nivel hardware.

5.2.1. Software

Se mostrarán a continuación los elementos software utilizados para el desarrollo del proyecto. Se ha priorizado el usar software gratuito y especialmente de código abierto. Para el software comercial de pago se ha hecho uso de la licencia de uso que nos proporciona la UNED.

5.2.1.1. Lenguajes de programación

En las primeras fases del desarrollo, se apostó por Django REST Framework (DRF) para la creación de la API REST, eso suponía el uso del lenguaje de programación interpretado Python. Se pretendía igualmente crear un módulo en lenguaje Java para la gestión de los grafos, si bien se descartó en las fases iniciales. La elección del Frontend supuso la aceptación de JavaScript como lenguaje a usar.

Los lenguajes empleados durante el desarrollo del proyecto son:

- **Python** [44]: Para la parte relacionada con DRF. Este lenguaje de programación es muy versátil y su sencillez de código nos permite crear una API REST basada

en DRF de forma muy sencilla. La versión utilizada durante todo el proceso de desarrollo ha sido la versión **3.11.1** y se han usado varias librerías externas de Python para añadir las funcionalidades requeridas en el desarrollo de la API REST.

- **JavaScript** [45]: Todo lo relacionado con el Frontend pasa por este lenguaje o cualquiera de sus subconjuntos como Typescript, pero todo pasa a ser JavaScript, ya que el navegador entiende principalmente HTML, CSS y JavaScript.
- **HTML** [46]: Al igual que con JavaScript, HTML es imperativo si se quiere hacer uso de una aplicación web que pueda ser soportada por cualquier navegador actual. La versión de HTML usada es la **5**.
- **CSS** [47]: Este lenguaje de marcas es usado conjuntamente con HTML en el Frontend. La versión usada en el proyecto ha sido la versión **3**.
- **JSON** [48]: Este formato de texto ha sido ampliamente usado en el proyecto. Es la comunicación básica entre los distintos niveles que forman el proyecto, intercambiando los mensajes en documentos de este tipo.

5.2.1.2. Frameworks

Los Frameworks son herramientas muy útiles para agilizar el desarrollo de aplicaciones y en este caso los usados son de amplia difusión y ya consolidados.

- **Vue 3** [49]: Es la pieza fundamental para el Frontend. La versión de Vue 3 usada ha sido la integrada en **Vue/cli 5.0.8** en su versión JavaScript. Existe otra versión de Vue 3 en Typescript, pero no se ha usado en el desarrollo del proyecto.
- **Django REST Framework (DRF)** [35]: Al igual que con Vue 3, sin DRF no podría existir el proyecto. Es el pilar principal que sustenta el proyecto. La versión de DRF utilizada ha sido la **versión 3.14.0**.
- **Django** [50]: Este Framework es pieza clave para el funcionamiento de DRF, aunque apenas se ha hecho uso del mismo en el proyecto, sin él DRF no puede funcionar. La versión utilizada es la **versión 4.1.9**.
- **NodeJS** [51]: Al igual que con Django, para que funcione correctamente Vue 3 es preciso disponer de este entorno de desarrollo. La versión utilizada es la **versión 18.15.0**.
- **Mapbox** [12]: Este elemento nos permite visualizar los mapas generados por los usuarios en el Frontend. Su integración con Vue 3 ha sido muy buena.

5.2.1.3. IDE

Antes de comenzar con el desarrollo del proyecto, al analizar los posibles lenguajes de programación a usar, los Integrated Development Environment (IDE) a usar estaban claros:

- PyCharm [52] en el caso del lenguaje Python.
- IntelliJ IDEA [53] en el caso del lenguaje Java.

Pero sin Visual Studio Code [54] no habría sido posible hacer este proyecto al menos de la forma tan fácil y rápida como este editor de código fuente permite. Por lo que se ha convertido en el único IDE usado en el desarrollo del proyecto. En Visual Studio Code se pueden añadir numerosas extensiones que facilitan la tarea del desarrollador, desde identificación por colores de la sintaxis del lenguaje de programación, pasando por extensiones para probar la API haciendo peticiones HTML. Lo más destacable es que todo el proyecto ha sido posible usando únicamente este IDE, lo cual facilita y mucho las tareas.

5.2.1.4. Herramientas de software

Se han usado varias herramientas software a lo largo del desarrollo, especialmente para ir comprobando el funcionamiento correcto de cada uno de los componentes que se iban realizando, destacamos aquí las herramientas usadas:

- **Postman** [55]: Esta herramienta nos permitía comprobar las peticiones HTTP que se realizaban a la API REST.
- **MongoDB Compass** [56]: Con esta herramienta podemos conectarnos con nuestra BBDD de MongoDB y visualizar así el estado de las colecciones.
- **DBeaver** [57]: Herramienta que nos permitía conectarnos con nuestra BBDD de PostgreSQL y visualizar el estado de la BBDD.
- **Lucidchart** [58]: Herramienta online con la que se han hecho los diagramas UML casos de uso y secuencia documentados en el Capítulo 3.
- **Astah UML** [59]: Herramienta de modelado en UML que ha sido usada para el diagrama de estados del Capítulo 3. Se ha usado la cuenta de estudiante ofrecida por la UNED.
- **GitHub desktop** [60]: Esta herramienta de control de versiones es fundamental en cualquier desarrollo y se ha usado ampliamente desde el origen del mismo. Se han creado dos repositorios privados, uno para la parte Backend y el otro para el Frontend.
- **Overleaf** [61]: Editor online de LaTeX con el que se ha hecho esta memoria.
- **Paint.net** [62]: Herramienta que nos permite editar imágenes, de uso gratuito y con la que se han realizado las figuras de la memoria.
- **Microsoft Word** [63]: Programa de Microsoft para editar textos, se han usado los iconos que ofrece el paquete Office para algunas figuras. Se ha usado el paquete de Office ofrecido por la UNED.
- **Microsoft Excel** [64]: Programa de Microsoft para hojas de cálculo, se ha usado para los diagramas de Gantt. Se ha usado el paquete de Office ofrecido por la UNED.

5.2.2. Hardware

Para el desarrollo del proyecto no ha sido necesario utilizar ningún hardware adicional del que ya tenía antes de comenzar con el proyecto:

- PC:
  - Procesador: Intel Core i5-13600K 3.50 GHz.
  - Memoria RAM: 32GB DDR4 3200 MHz.
  - Dos monitores BenQ GL2460 FullHD 24".
- Portátil:
  - Procesador: Intel Core i7-10510U 1.80 GHz.
  - Memoria RAM: 16GB LPDDR3 2133MHz.
  - Pantalla principal: FullHD 14".
  - Pantalla secundaria: ScreenPadTM Plus: 12.61".

5.3. Implementación

El carácter de la estructura en tres niveles nos permite desplegar cada uno de forma independiente para que una vez estén ejecutándose se puedan comunicar entre sí sin problema alguno. Así a la hora de poner en marcha el proyecto lo primero es comenzar con el Nivel 2, la API REST, sin la cual no hay comunicación entre el Nivel 1 y el Nivel 3.

5.3.1. Backend

Para que DRF pueda ejecutarse plenamente, es necesario ajustar la configuración de forma correcta, para ello tenemos el fichero *settings.py* el cual contiene toda la configuración del mismo. Veremos algunas de las partes más relevantes en el Código Fuente 5.1.

Código Fuente 5.1: Extracto settings.py.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django_filters',
    'corsheaders',
    'rest_framework',
    'rest_framework.authtoken',
    'apps.usersdb',
    'apps.driversdb',
    'apps.passengerss',
    'apps.routes',
    'apps.authsusers',
    'apps.logrecords',
```

```
    'apps.messagesdb',
    'apps.trips',
    'apps.rates',
    'apps.nodes'
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

DATABASE_ROUTERS = ['core.routers.db_routers.AuthRouter',
                    'apps.logrecords.logs_db_router.LogsRouter',
                    'apps.driversdb.drivers_db_router.DriversRouter',
                    'apps.passengerss.passengers_db_router.
   PassengersRouter',
                    'apps.routes.routes_db_router.RoutesRouter',
                    'apps.usersdb.users_db_router.UsersRouter',
                    'apps.messagesdb.messages_db_router.MessagesRouter',
                    'apps.trips.trips_db_router.TripsRouter',
                    'apps.rates.rates_db_router.RatesRouter',
                    'apps.nodes.nodes_db_router.NodesRouter',
                    ]
```

En **INSTALLED_APPS** tenemos las aplicaciones que DRF necesita para funcionar, así como las que hayamos instalado que sean de terceros, como *corsheaders* para la gestión de los hosts de acceso a la API desde el exterior. Se han de incluir además las aplicaciones que nosotros hemos creado, como *apps.routes*.

En **MIDDLEWARE** tenemos las aplicaciones que permiten a DRF comunicarse con otras aplicaciones para que todo funcione. Ya viene configurado así por defecto y a menos que una aplicación específica lo necesite no se tiene que tocar nada. En este caso se ha añadido la entrada para *corsheader* porque es necesaria.

En **DATABASE_ROUTERS** tenemos las entradas de los ficheros que tienen la configuración de conexión a las BBDD según las aplicaciones. Cada aplicación puede estar alojada en una BBDD distinta y su configuración está en esos ficheros.

Por otro lado, la correcta configuración del fichero *urls.py* en el core principal de DRF nos permite el acceso a todas nuestras Endpoint URL como se puede apreciar en el Código Fuente 5.2.

Código Fuente 5.2: Core urls.py.

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
```

```
    path('drivers/',include('apps.driversdb.urls')),
    path('pax/',include('apps.passengerss.urls')),
    path('routes/',include('apps.routes.urls')),
    path('auth/',include('apps.authsusers.urls')),
    path('messages/', include('apps.messagesdb.urls')),
    path('trips/', include('apps.trips.urls')),
    path('rates/', include('apps.rates.urls')),
    path('nodes/', include('apps.nodes.urls'))

]
```

Ya se explicó anteriormente las ocho aplicaciones que componen la API REST, pero veamos algunos detalles más sobre las aplicaciones:

- **authusers**: Aplicación que gestiona los datos de acceso de los usuarios. Utiliza como modelo base el User de Django tal y como se observa en el Código Fuente 5.3.

Código Fuente 5.3: User serializer.

```
from rest_framework import serializers
from django.contrib.auth.models import User

class UserAuthSerializer(serializers.ModelSerializer):
    class Meta(object):
        model = User
        fields = ['id','username','password','email']

```

- **driversdb**: Aplicación que modela el comportamiento de los conductores, por un lado creamos el modelo y con las vistas interactuamos con él. Veamos la clase *model.py* que contiene el Modelo de los conductores en el Código Fuente 5.4.

Código Fuente 5.4: Conductores model.py.

```
from django.db import models
from django.utils import timezone
import uuid

class Driver(models.Model):
    id = models.UUIDField(primary_key = True,
         default = uuid.uuid4,
         editable = False)
    driverCreated = models.DateTimeField(default=timezone.now)
    userIDowner = models.CharField(max_length=100, null=False,
    unique=False)
    driverStatus = models.BooleanField(default=True)
    driverBrand = models.CharField(max_length=50, null=True)
    driverModel = models.CharField(max_length=50, null=True)
    driverVColor = models.CharField(max_length=50, null=True)
    driverSeats = models.SmallIntegerField(null=True,default=3)
    driverDistance = models.SmallIntegerField(null=True, default=4)
    driverStartTime = models.CharField(max_length=15, null=True)
    driverEndTime = models.CharField(max_length=15, null=True)
    driverWaitLimitStart = models.SmallIntegerField(null=True,
    default=4)
    driverMaxDistanceAllowed = models.SmallIntegerField(null=True,
    default=2000)
    driverConfigState = models.BooleanField(default=True)
```

```
    driverFreeSeats = models.SmallIntegerField(null=True,default=3)
    driverSmoke=models.BooleanField(default=False)
    driverCity=models.CharField(max_length=100, null=True)
    driverPhone=models.CharField(max_length=50, null=True)
    driverPremium=models.BooleanField(default=False)
    driverVerified=models.BooleanField(default=False)
    objects = models.Manager()

```

- **passengers**: Esta aplicación realiza la misma función que la anterior, pero en el caso de los conductores. Su modelo ya fue mostrado en el Código Fuente 4.1.

- **routes**: Esta aplicación gestiona las rutas de los usuarios, veamos el Código Fuente 5.5 de las vistas que en líneas generales comparten todas las aplicaciones por igual, salvo en el nombre del modelo y serializador a usar.

Código Fuente 5.5: Rutas views.py.

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
from .models import Route
from .serializers import RouteSerializer
from django.db.models.query_utils import Q


@api_view(['GET'])
def routes_api_list_view(request):
    routes = Route.objects.using("data_db").all()
    serializer = RouteSerializer(routes, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)

@api_view(['POST'])
def add_route_api_view(request):

    route_serializer = RouteSerializer(data = request.data)

    if route_serializer.is_valid():
        route_serializer.create(route_serializer.data)

        return Response("success", status=status.HTTP_200_OK)
    return Response(route_serializer.errors, status=status.
   HTTP_400_BAD_REQUEST)

@api_view(['POST'])
def route_api_detail_view(request):
    routes = Route.objects.using("data_db").filter(userIDowner=
   request.data['userIDowner']).first()

    serializer = RouteSerializer(routes)
    return Response(serializer.data)

@api_view(['POST'])
def route_type_api_detail_view(request):
    routes = Route.objects.using("data_db").filter(routeType=
   request.data['routeType']).all()
```

```
38
39      serializer = RouteSerializer(routes, many=True)
40      return Response(serializer.data)
41
```

- **nodes**: Esta aplicación es la encargada de almacenar los nodos y por tanto ser usada para el grafo.

- **messages**: Esta aplicación se encarga de almacenar los mensajes que generan los mensajes, el modelo se muestra en el Código Fuente 5.6.

Código Fuente 5.6: Mensajes models.py.

```
1  from django.db import models
2  from django.utils import timezone
3  import uuid
4
5  class Message(models.Model):
6      id = models.UUIDField(primary_key = True,
7           default = uuid.uuid4,
8           editable = False)
9      messageCreated = models.DateTimeField(default=timezone.now)
10     messageToRead = models.DateTimeField(null=True)
11     messageFromRead= models.DateField(null=True)
12     messageToArchived = models.DateField(null=True)
13     messageFromArchived = models.DateField(null=True)
14     fromMessage = models.CharField(max_length=100, null=False,
       unique=False)
15     toMessage = models.CharField(max_length=100, null=False, unique
       =False)
16     bodyMessage = models.CharField(max_length=250, null=False,
       unique=False)
17     messageRead = models.BooleanField(default=False)
18     fromArchived = models.BooleanField(default=False)
19     toArchived = models.BooleanField(default=False)
20     messageStatus = models.CharField(max_length=10, default="OK")
21     messageShowFrom = models.CharField(max_length=10, default="YES"
       )
22     messageShowTo = models.CharField(max_length=10, default="YES")
23     messageAvoidReply =models.BooleanField(default=True)
24     objects = models.Manager()
25
```

- **trips**: Esta aplicación se encarga de almacenar las reservas creadas por los usuarios, su modelo lo podemos ver en el Código Fuente 5.7.

Código Fuente 5.7: Reservas models.py.

```
1  from django.db import models
2  from django.utils import timezone
3
4  import uuid
5
6  class Trip(models.Model):
7      id = models.UUIDField(primary_key = True,
8           default=uuid.uuid4,editable = False)
9      tripCreated = models.DateTimeField(default=timezone.now)
```

```
10      userIDowner = models.CharField(max_length=100, null=False,
        unique=False)
11      tripPax=models.CharField(max_length=100, null=True, unique=
        False)
12      tripAccepted=models.BooleanField(default=False)
13      tripRejected=models.BooleanField(default=False)
14      tripRejectedP=models.BooleanField(default=False)
15      tripStartTime=models.CharField(max_length=15, null=True)
16      tripEndTime=models.CharField(max_length=15, null=True)
17      tripActive=models.BooleanField(default=False)
18      tripDayMon=models.BooleanField(default=False)
19      tripDayTue=models.BooleanField(default=False)
20      tripDayWed=models.BooleanField(default=False)
21      tripDayThu=models.BooleanField(default=False)
22      tripDayFri=models.BooleanField(default=False)
23      tripDaySat=models.BooleanField(default=False)
24      tripDaySun=models.BooleanField(default=False)
25      tripPermanent=models.BooleanField(default=False)
26      tripInit = models.BooleanField(default=False)
27      tripBack = models.BooleanField(default=False)
28      tripArchived=models.BooleanField(default=False)
29      tripEnded=models.BooleanField(default=False)
30      tripOnVehicleD=models.BooleanField(default=False)
31      tripOnVehicleP=models.BooleanField(default=False)
32      tripReactivate=models.BooleanField(default=False)
33      tripStatus=models.CharField(max_length=10, default="OK")
34      tripShow=models.CharField(max_length=10, default="YES")
35      objects = models.Manager()
36
```

- **rates**: Con esta aplicación gestionamos las valoraciones y comentarios que los usuarios se hacen al finalizar un viaje. Veamos su modelo en el Código Fuente 5.8.

Código Fuente 5.8: Valoración models.py

```
1  from django.db import models
2  from django.utils import timezone
3  import uuid
4  class Rate(models.Model):
5      id = models.UUIDField(primary_key = True,
6           default=uuid.uuid4,editable = False)
7      rateCreated = models.DateTimeField(default=timezone.now)
8      userIDowner = models.CharField(max_length=100, null=False,
       unique=False)
9      rateUser=models.CharField(max_length=100, null=True, unique=
       False)
10     rateAnonymous=models.BooleanField(default=False)
11     rateValue=models.SmallIntegerField(default=3)
12     rateComment=models.CharField(max_length=250, null=True)
13     rateStatus=models.BooleanField(default=True)
14     rateDeleted=models.BooleanField(default=False)
15     rateDriverRated=models.CharField(max_length=10,default="NO")
16     ratePaxRated=models.CharField(max_length=10,default="NO")
17     rateShow=models.CharField(max_length=10,default="SI")
18     objects = models.Manager()
19
20
```

```
(env) PS G:\UNED\PFG\PFG_Dev\pfg_drf_api> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 31, 2023 - 15:13:32
Django version 4.1.9, using settings 'core.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figura 5.1: Servidor Django en ejecución.

Recordemos que las aplicaciones son totalmente individuales y aisladas entre sí, con sus modelos actúan como las tablas de BBDD relacionales, pero en este caso como colecciones en MongoDB. Al ejecutar el servidor de Django se pone en marcha el Nivel 2 y queda listo para recibir peticiones desde el Frontend como muestra la Figura 5.1.

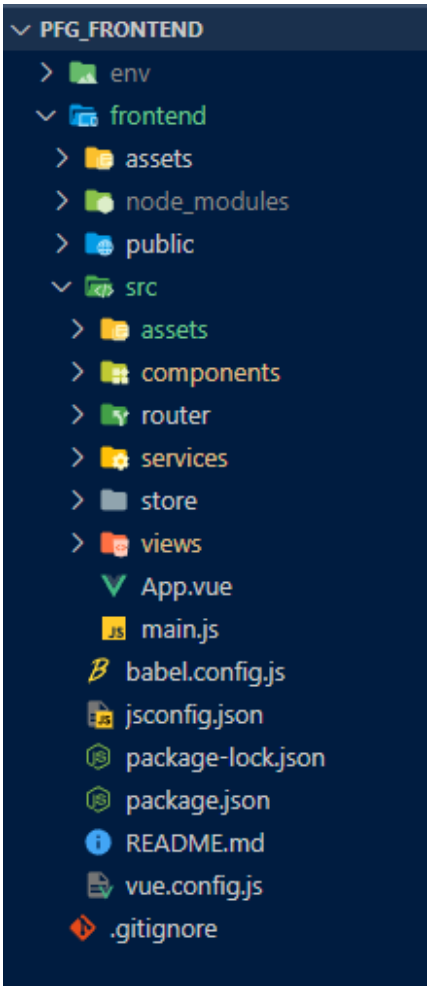


Figura 5.2: Estructura ficheros en Vue 3.

5.3.2. Frontend

La implementación del Frontend requiere de más elementos, por un lado es preciso la existencia de NodeJS instalado, así como vue/cli que es un instalador de Vue 3. La estructura de ficheros en Vue 3 nos quedaría como se muestra en la Figura 5.2.

Se puede observar en la Figura 5.2 que comenzando desde arriba tenemos los siguientes elementos.

- env: carpeta que contiene el entorno de desarrollo aislado del sistema, así evitamos que la instalación de las librerías o componentes sea de forma global.
- frontend: nombre de la aplicación Vue 3.
- assets: carpeta que almacenará imágenes, iconos o cualquier elemento extra que necesitemos.
- node_modules: en esa carpeta están instaladas todas las librerías que hemos necesitado para el entorno.
- public: contiene el único documento HTML que Vue 3 necesita, es el *indes.html* que toda página tiene por defecto.
- src: esta carpeta contiene todos los componentes y elementos necesarios para visualizar nuestro proyecto.
  - assets: aquí se colocan los iconos o fuentes específicas, así como imágenes para usar.
  - components: todos los componentes creados en Vue 3 se han de colocar en esta carpeta.
  - router: contiene un único fichero *index.js* que es un fichero JavaScript que contiene las rutas URL tal y como se mostró en Código Fuente 4.6.
  - services: Contiene los ficheros en JavaScript que realizan las peticiones HTTP al Nivel 2 en DRF.
  - store: Contiene los ficheros que sirve de almacén temporal para el paso de información entre componentes.
  - views: Contiene los ficheros que se utilizan como modelos de vista.
  - App.vue: Fichero que se carga en *index.html* y sobre el que se acoplan todos los componentes de los modelos de vista de la carpeta views.
  - main.js: Fichero JavaScript que contiene la configuración inicial de la aplicación.

El resto de ficheros son autogenerados por Vue 3 y sirven de configuración y respaldo. Veamos el contenido del fichero *main.js* en el Código Fuente 5.9.

Código Fuente 5.9: Vue 3 main.js.

```
1 /* eslint-disable */
2 import { createApp } from 'vue'
3 import App from './App.vue'
```

```
4  import router from './router'
5  import 'bootstrap'
6  import 'bootstrap/dist/css/bootstrap.min.css'
7  import './assets/fonts/poppins/poppins.css'
8  import { library } from '@fortawesome/fontawesome-svg-core'
9  import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'
10 import { faCar, faEarthEurope, faUsersViewfinder,
       faArrowRightFromBracket, faArrowRightToBracket, faUserPlus,
11          faPersonCirclePlus, faSmoking, faBanSmoking, faGear, faCarOn
               , faEnvelope, faStar as faStarSolid} from '@fortawesome/
               free-solid-svg-icons'
12 import { faUser, faStar as faStarRegular } from '@fortawesome/free-
       regular-svg-icons'
13 import { createPinia } from 'pinia';
14 import 'vuetify/styles'
15 import { createVuetify } from 'vuetify'
16 import * as components from 'vuetify/components'
17 import * as directives from 'vuetify/directives'
18 import mapboxgl from 'mapbox-gl'; // or "const mapboxgl = require('
       mapbox-gl');"
19
20 const vuetify = createVuetify({
21     components,
22     directives,
23   })
24
25 library.add(faCar, faUser, faEarthEurope,faUsersViewfinder,
       faArrowRightFromBracket,faArrowRightToBracket, faUserPlus,
26     faPersonCirclePlus, faSmoking, faBanSmoking, faGear, faCarOn,
          faEnvelope, faStarRegular, faStarSolid)
27
28 mapboxgl.accessToken = '
       pk.eyJ1IjoiYW5hdnByb2dyYW1zIiwiYSI6ImNsazVmY251bjAyMWUzcXM3dXZhe
29 XlkODAifQ.RrZDDFsepRR-qOJadp1YFg';
30
31 const app = createApp(App);
32 const pinia = createPinia();
33
34 app.component('font-awesome-icon', FontAwesomeIcon)
35 app.use(pinia);
36 app.use(router);
37 app.use(vuetify)
38 app.mount('#app');
```

Lo más destacado del Código Fuente 5.9 está en las últimas líneas, en las que se crea propiamente la aplicación a través de la definición *const app = createApp(App);*.

Sobre dicha *app* se irán añadiendo componentes a lo largo del ciclo de vida de la aplicación. Esos componentes son los modelos de vista que están compuestos a su vez por otros componentes con funcionalidades específicas, que según las necesidades se van añadiendo en la aplicación. No todos los componentes son accesibles para todos los usuarios, algunos tienen acceso público, por lo que cuando se accede a la aplicación web son visibles para el usuario y otros son sólo accesibles si el usuario está autenticado. En la Tabla 5.1 se muestran los modelos de vista que forman la aplicación.

Tabla 5.1: Modelos de vista en la aplicación.

Componente	Acción	Visibilidad
Header	Muestra la barra de navegación	Todos los usuarios
Home	Página de entrada	Todos los usuarios
Register	Formulario de registro	Todos los usuarios
Login	Formulario de acceso	Todos los usuarios
Profile	Muestra el perfil del usuario	Usuarios autenticados
Config	Formulario cambio de datos	Usuarios autenticados
LiveMap	Muestra en un mapa todos los usuarios	Usuarios autenticados
Trips	Muestra el panel de reservas	Usuarios autenticados
Messages	Muestra los mensajes	Usuarios autenticados

Cuando un usuario accede a la aplicación, Vue 3 inyecta el componente *Home* a la aplicación, a través de su modelo de vista. Según vaya interactuando el usuario, se irán cargando más o menos componentes al contenedor principal App. Ese es el funcionamiento de Vue 3, que permite entre otras cosas el uso de una única página web y la reactividad, es decir, los cambios sobre los componentes visuales sin necesidad de recargar la página.

5.3.2.1. BBDD

La implementación de las BBDD es una tarea que realiza Django REST Framework (DRF) cuando se crea o modifica el modelo de cualquier aplicación. Los modelos de las aplicaciones en DRF son los equivalentes a las tablas de las BBDD, por lo que una vez configuradas las aplicaciones, hay que hacer las migraciones correspondientes y finalmente migrarlas a las BBDD mediante el comando mostrado en la Figura 5.3.

```
(env) PS G:\UNED\PFG\PFG_Dev\pfg_drf_api> python manage.py migrate --database=data_db
```

Figura 5.3: Python migrate.

En este caso se migrarán los modelos a la BBDD de MongoDB *data_db*, creando las colecciones correspondientes, sin intervención por parte del desarrollador salvo la intalación del entorno de la BBDD.

Este método serviría para cualquier tipo de BBDD y es precisamente la mayor ventaja de crear una API REST con respecto a tener que plantear o diseñar el esquema de una BBDD.

En este caso con una simple instrucción quedan totalmente listas las BBDD para ser usadas por la aplicación.

5.4. Pruebas

Se presenta en esta sección una batería de pruebas tanto de la parte Backend como de la parte Frontend, para comprobar el funcionamiento según los requisitos previstos.

Igualmente exponemos unos errores detectados en el proyecto al realizar pruebas.

5.4.1. Conjunto de pruebas

Las pruebas realizadas durante el desarrollo del proyecto han sido constates, especialmente en la construcción de la API REST. La integración entre el Frontend y el Backend también ha supuesto un enorme esfuerzo en realizar pruebas de validación, para asegurar que todo el conjunto funciona según los **RF** Requisitos Funcionales previstos.

La primera batería de pruebas está dirigida a comprobar que la API REST funciona y ejecuta las peticiones HTTP recibidas. Para testear la API REST lo hacemos desde Visual Studio Code con una de las extensiones que más hemos usado durante el desarrollo del proyecto: Thunderclient [65]. Esta extensión nos permite hacer peticiones HTTP indicando el tipo de petición y en el cuerpo el elemento que queremos enviar en formato JSON.

P1: Registrar un usuario.

- Se procede a registrar un usuario con su nombre de usuario, correo y contraseña.
- El resultado esperado es recibir un **Status 200 OK** y el token asociado.

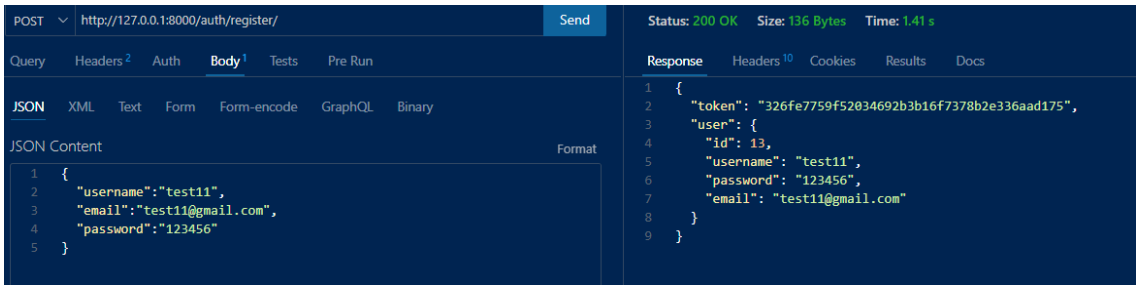


Figura 5.4: Prueba 1: Registrar usuario.

P2: Error al registrar usuario.

- Se procede a volver a registrar el mismo usuario, con igual nombre de usuario.
- El resultado esperado es un **Status 400 Bad request** y mensaje informativo del error.

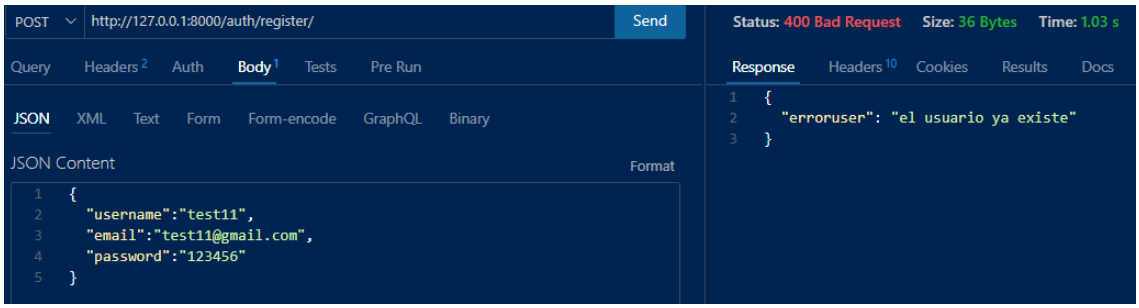


Figura 5.5: Prueba 2: Error al registrar usuario.

P3: Error al registrar usuario.

- Se procede a cambiar el nombre del usuario y mantener el mismo correo anterior.
- El resultado esperado es un **Status 400 Bad request** y mensaje informativo del error.

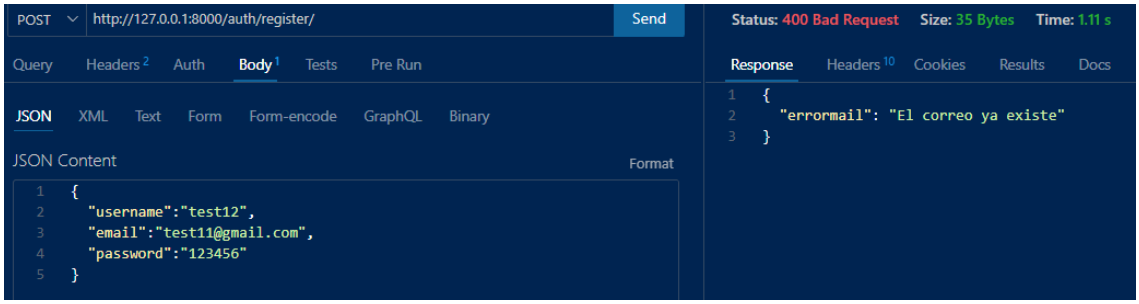


Figura 5.6: Prueba 3: Error al registrar usuario.

P4: Login.

- Se procede a enviar nombre de usuario y contraseña para acceder al sistema.
- El resultado esperado es **Status 200 OK** así como recibir el token datos de acceso.

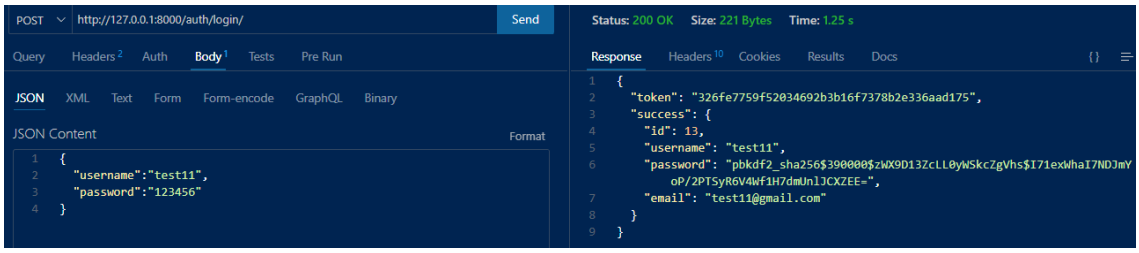


Figura 5.7: Prueba 4: Acceso al sistema.

P5: Error en login.

- Se procede a intentar acceder con un usuario incorrecto.
- El resultado esperado es **Status 400 Bad request** y mensaje informativo del error.

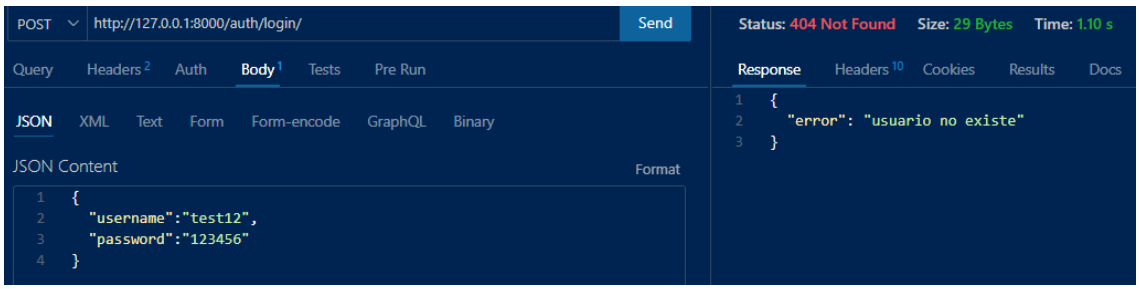


Figura 5.8: Prueba 5: Error acceso.

P6: Error en login.

- Se procede a ingresar una contraseña incorrecta.
- El resultado esperado es **Status 400 Bad request** y mensaje informativo del error.

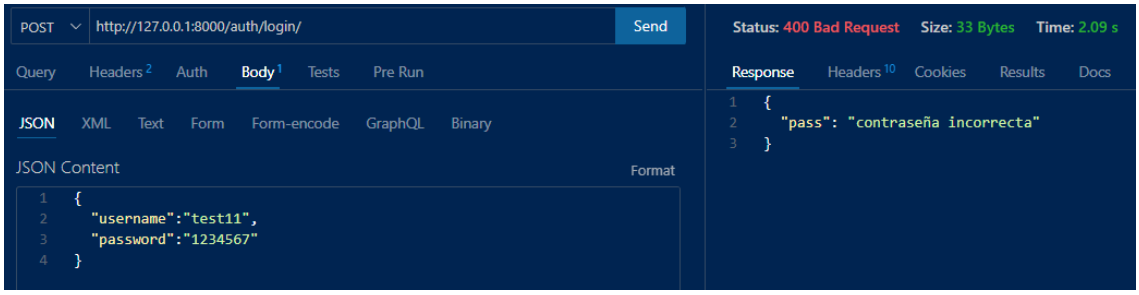


Figura 5.9: Prueba 6: Error acceso.

P7: Crear conductor.

- Se procede a crear un conductor.
- El resultado esperado es **Status 200 OK** y el mensaje *success*.

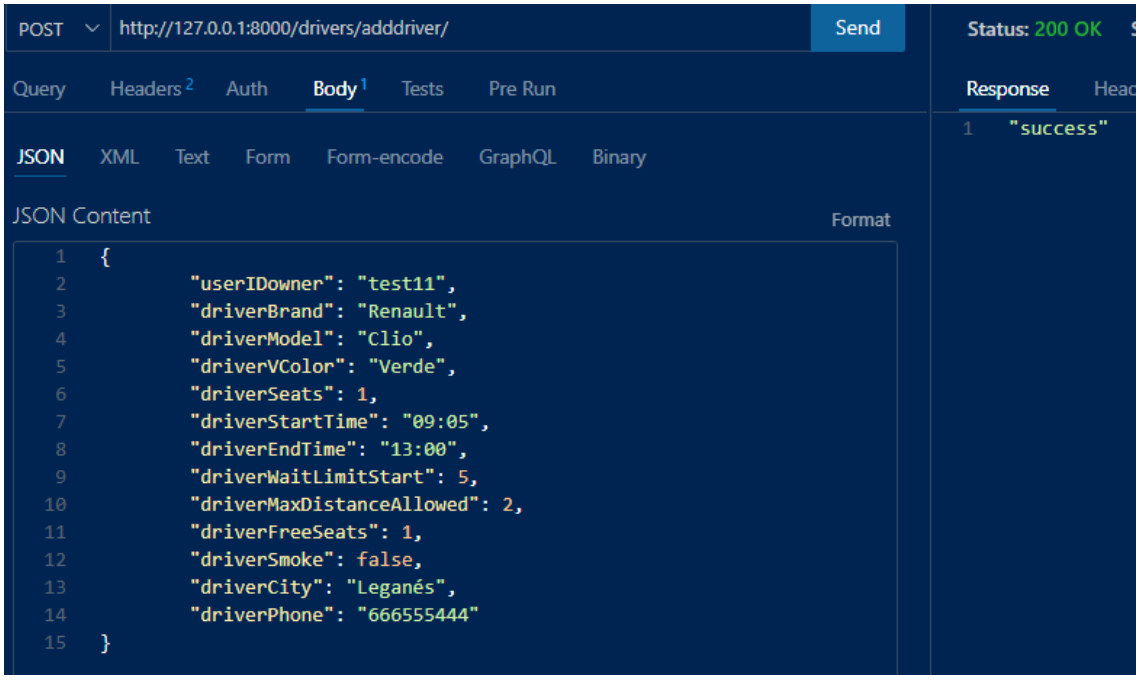


Figura 5.10: Prueba 7: Crear Conductor.

P8: Consultar conductor.

- Se procede a consultar un conductor.
- El resultado esperado es **Status 200 OK** y el objeto JSON con todos los datos del conductor.

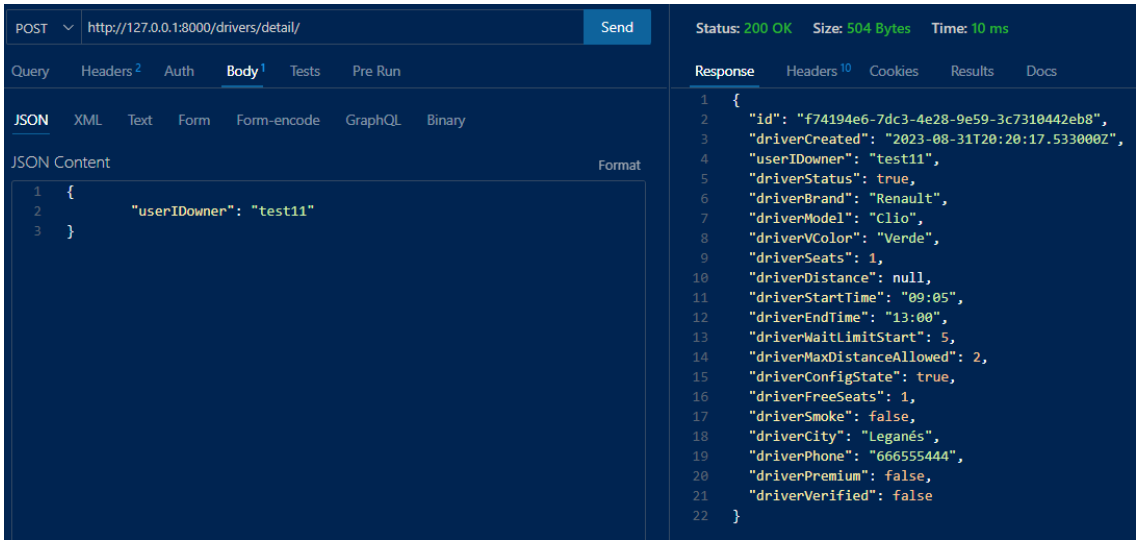


Figura 5.11: Prueba 8: Consultar Conductor.

P9: Modificar asientos libres conductor.

- Se procede a simular que sube un pasajero, dejando entonces el valor de asientos disponibles a 0.

- El resultado esperado es **Status 200 OK** y el mensaje *success*.

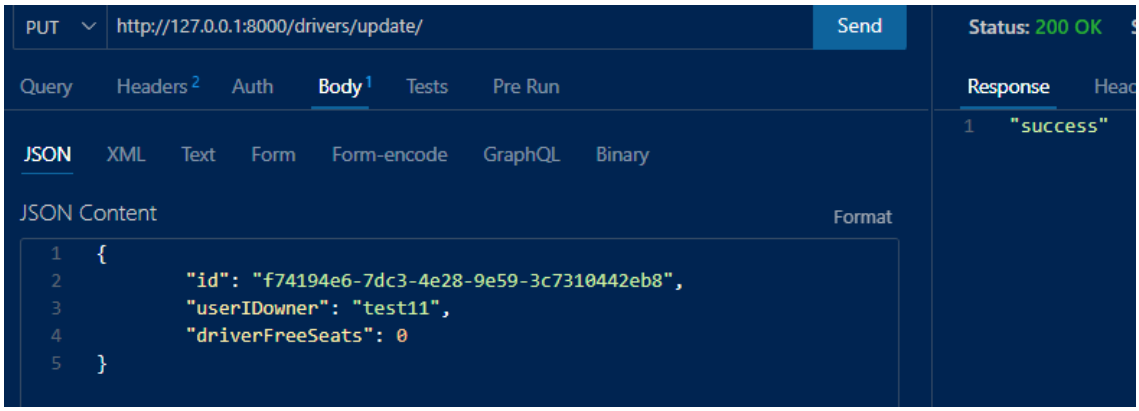


Figura 5.12: Prueba 9: Actualizar datos conductor.

P10: Verificar conductor.

- Se procede a comprobar el valor actualizado en el conductor.

- El resultado esperado es **Status 200 OK** el objeto JSON con todos los datos del conductor actualizados y *"driverFreeSeats": 0*.

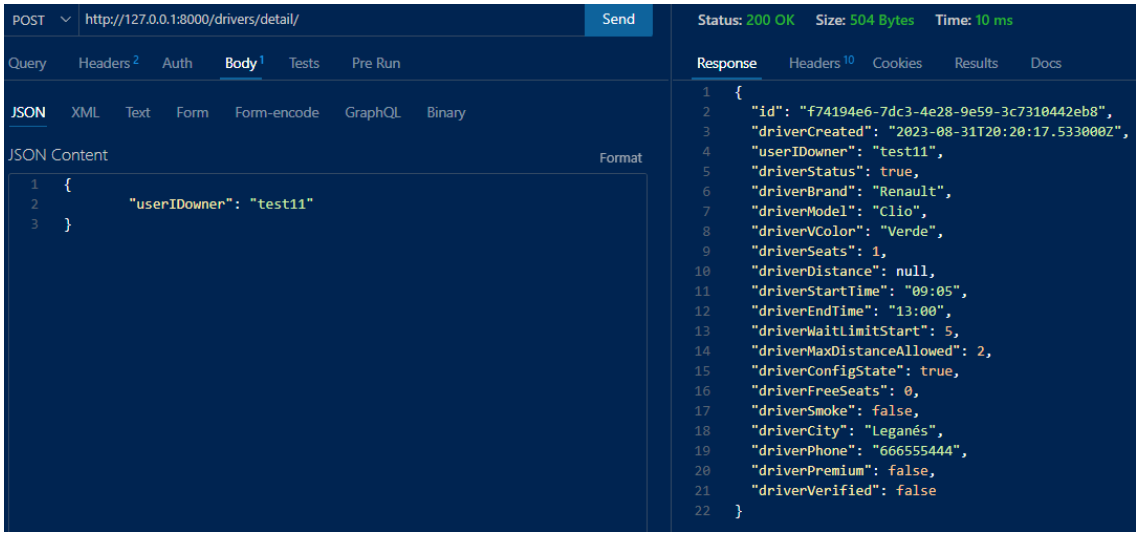


Figura 5.13: Prueba 10: Verificar conductor.

P11: Crear Ruta.

- Se procede a crear una ruta para el conductor, indicando las coordenadas previamente calculadas desde un mapa.

- El resultado esperado es **Status 200 OK** y el mensaje *success*. En Pruebas de Verificación esta misma consulta tendría un campo *routePath* con el recorrido calculado.

Figura 5.14: Prueba 11: Crear Ruta.

P12: Consultar Ruta.

- Se procede a buscar la ruta perteneciente al usuario.

- El resultado esperado es **Status 200 OK** el objeto JSON con todos los datos de la ruta. En Pruebas de Verificación esta misma consulta el campo *routePath* debería contener un array con las coordenadas del recorrido.

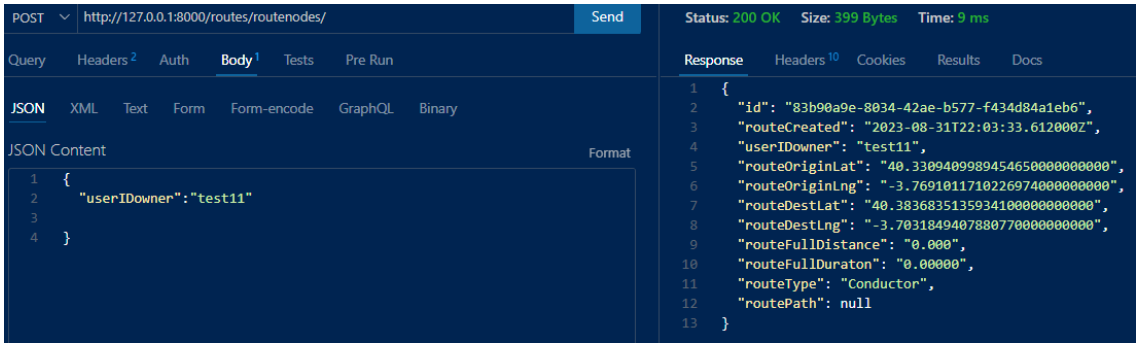


Figura 5.15: Prueba 12: Consultar Ruta.

P13: Crear Grafo.

- Se procede a crear el grafo con el pasajero test2 y los conductores que estén dentro de su área.

- El resultado esperado es **Status 200 OK** y el mensaje *success*.

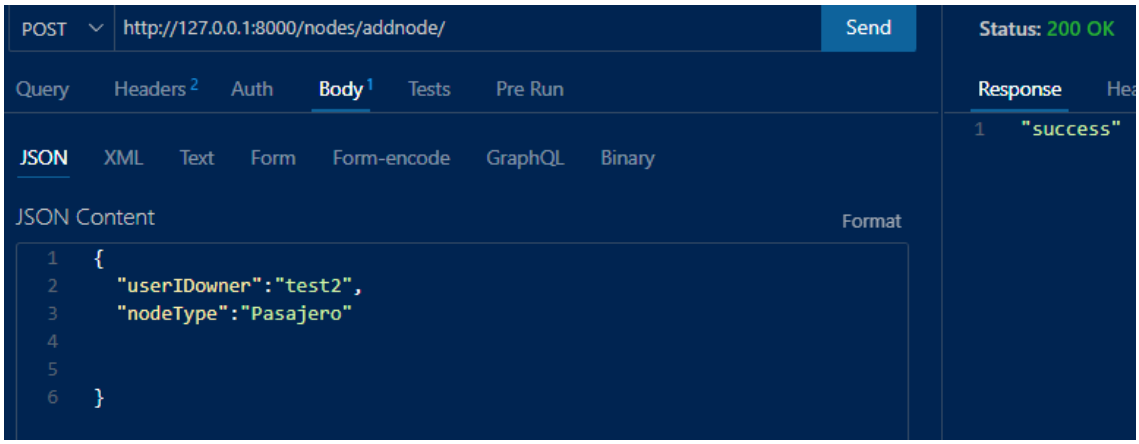


Figura 5.16: Prueba 13: Crear Grafo.

P14: Verificar Grafo.

- Se verifica la creación del grafo al consultar por el pasajero test2.

- El resultado esperado es **Status 200 OK** y un objeto JSON con los nodos creados en PI13 al generar el grafo de usuarios conductores dentro del área.

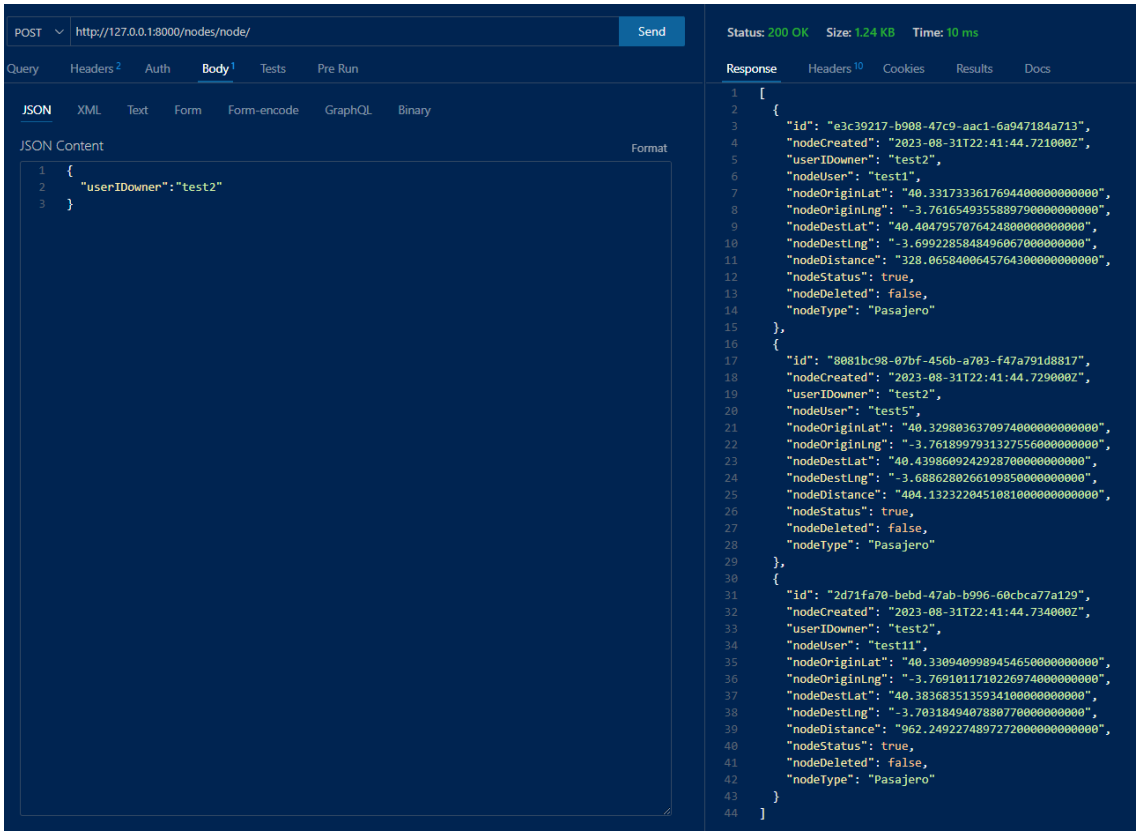


Figura 5.17: Prueba 14: Verificar Grafo.

Se dan por finalizadas las pruebas en la API REST al comprobar que los resultados obtenidos son los esperados.

Tanto los conductores como los pasajeros tienen el mismo código para la creación, consulta y modificación. Al verificar que funciona la aplicación **drivers**, se reescribe el código para **passengers** cambiando únicamente el modelo y serializador correspondiente.

Estos cambios serán introducidos también en las aplicaciones:

- **Messages**: Mismo código que en drivers para creación, consulta y modificación.
- **Trips**: Mismo código que en drivers para creación, consulta y modificación.
- **Rates**. Mismo código que en drivers para creación, consulta y modificación.

De esta forma se reduce el número de pruebas de manera considerable, al no tener que comprobar todas las operaciones en ocho aplicaciones de las cuáles 5 comparten el mismo código.

La siguiente batería de pruebas está dirigida a la comprobación del funcionamiento a nivel global de la aplicación, donde se comprueba que el Frontend y el Backend funcionan según lo esperado.

PV1: Registrar usuario.

- Se rellena el formulario de registro y se le da al botón **Registrar**.

- El resultado esperado es que aparezca la alerta con el mensaje y nos lleve al perfil.

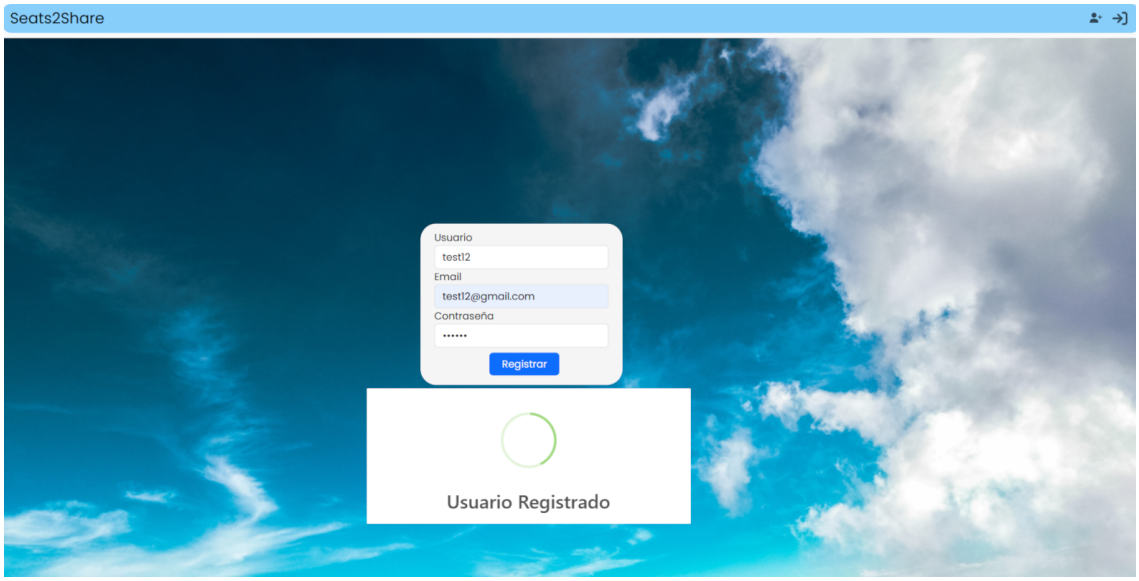


Figura 5.18: Prueba Verificación 1.1: Registrar Usuario.

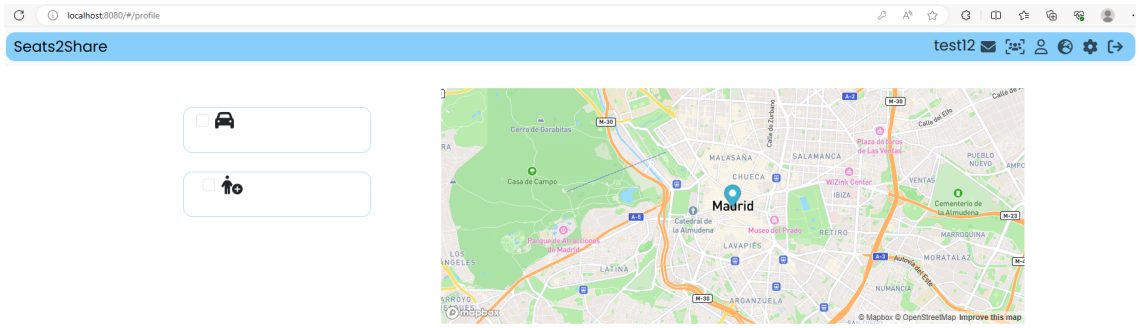


Figura 5.19: Prueba Verificación 1.2: Perfil.

PV2: Error al Registrar usuario.

- Se rellena el formulario de registro con los datos anteriores y se le da al botón **Registrar**.

- El resultado esperado es que aparezca una alerta indicando el error: **Ya existe ese usuario**.

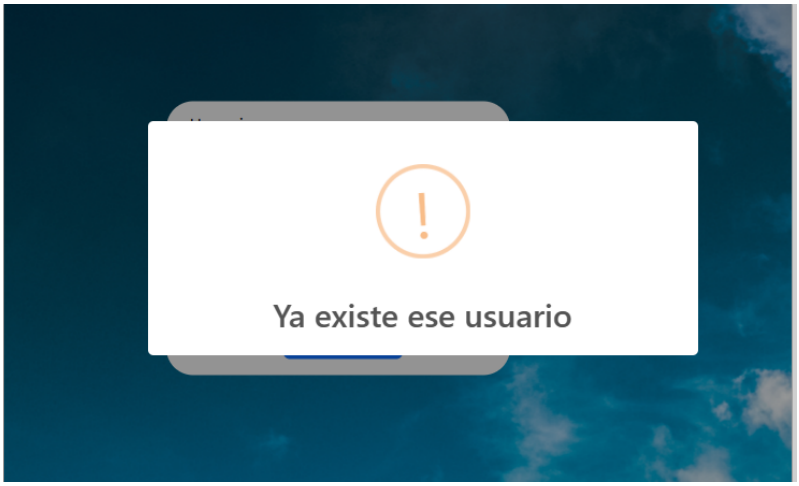


Figura 5.20: Prueba Verificación 2: Error al Registrar Usuario.

PV3: Login.

- Se rellena el formulario de login con los datos correctos y se le da al botón **Acceder**.
- El resultado esperado es que aparezca una alerta indicando el mensaje: **Acceso Correcto** y nos redirija a la página de perfil.

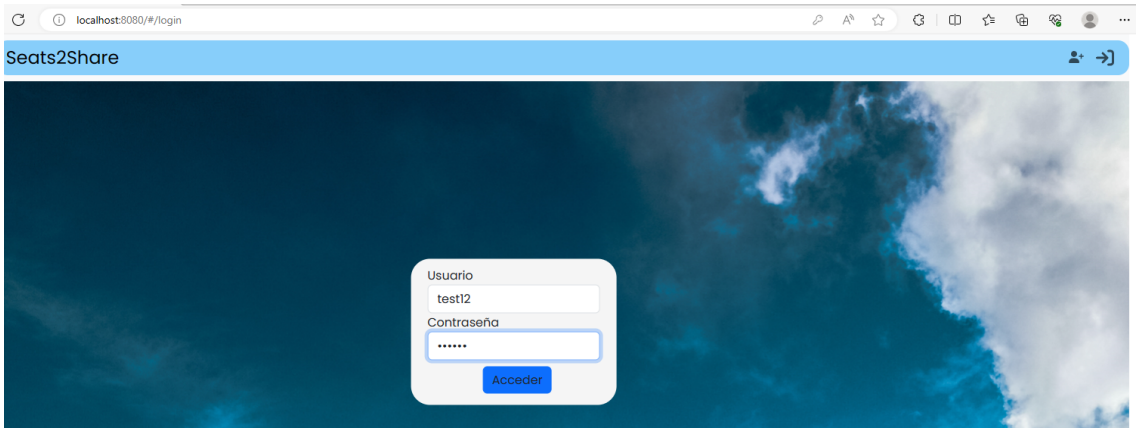


Figura 5.21: Prueba Verificación 3.1: Login.

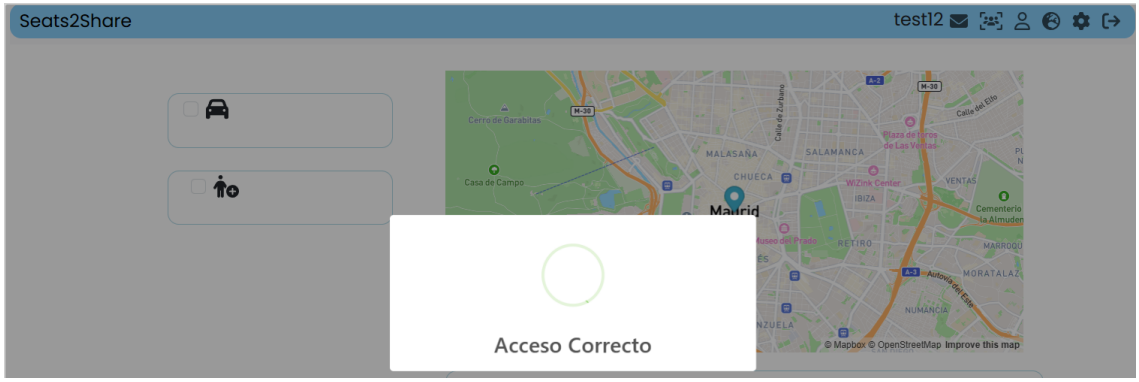


Figura 5.22: Prueba Verificación 3.2: Acceso Correcto.

PV4: Error en Login.

- Se rellena el formulario de login con la contraseña incorrecta y se le da al botón **Acceder**.

- El resultado esperado es que aparezca una alerta indicando el error: **Contraseña incorrecta**.

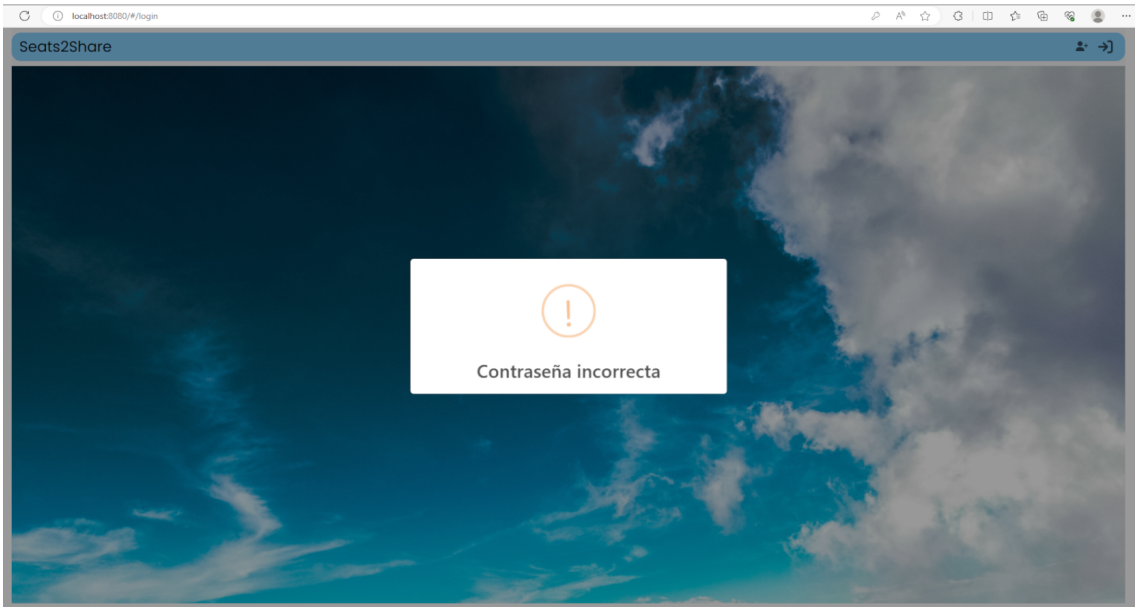


Figura 5.23: Prueba Verificación 4: Error acceso.

PV5: Registrar Pasajero.

- Se rellena el formulario de registro como pasajero, con las condiciones de recogida y se pulsa el botón **Añadir**.

- El resultado esperado es que aparezca una alerta indicando el mensaje: **Pasajero Añadido**. Se cambia a otra vista y se habilitan los botones de crear ruta.

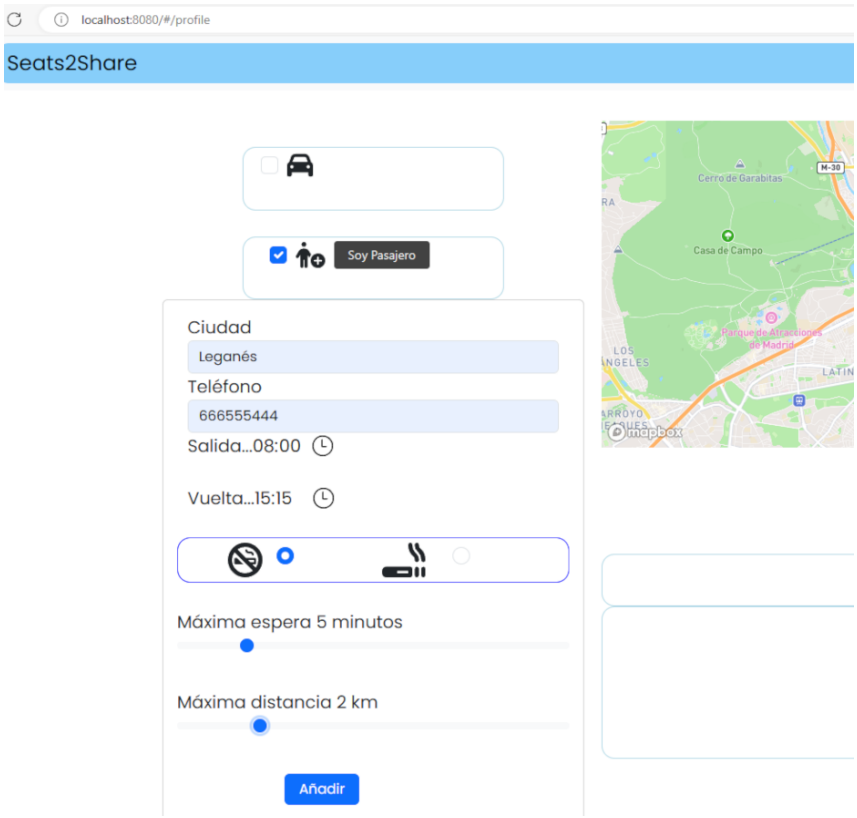


Figura 5.24: Prueba Verificación 5.1: Registrar Pasajero.

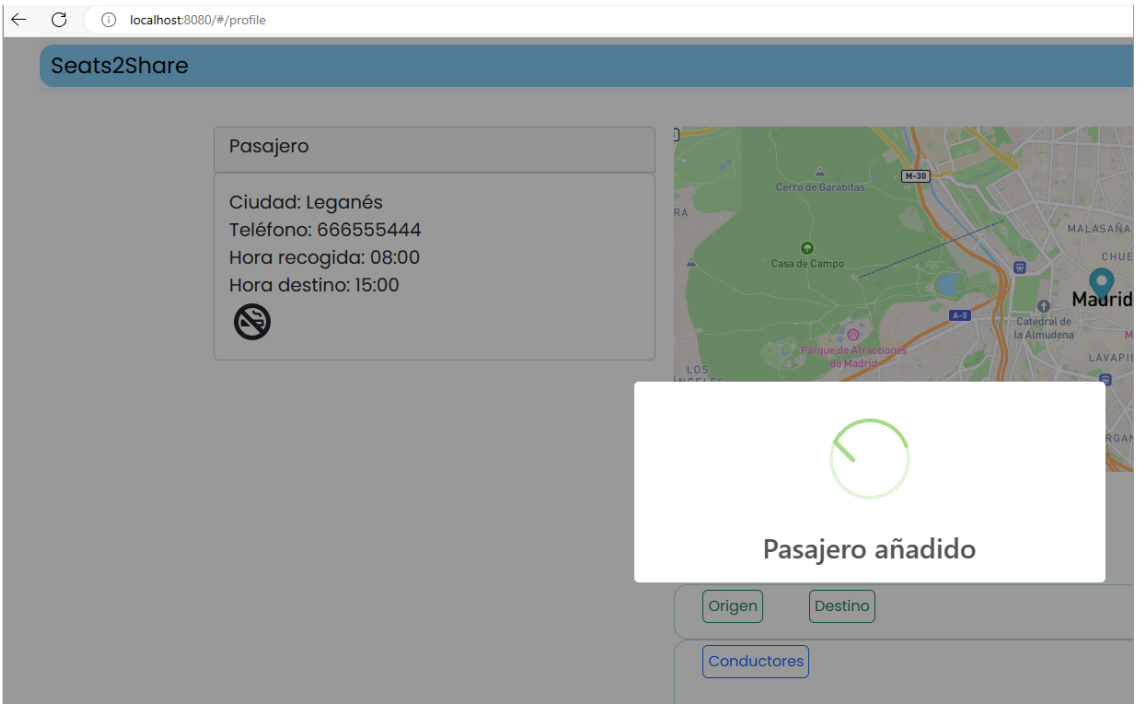


Figura 5.25: Prueba Verificación 5.2: Perfil Pasajero.

PV6: Registrar Conductor.

- Se rellena el formulario de registro como conductor, con las condiciones de recogida y se pulsa el botón **Añadir**.

- El resultado esperado es que aparezca una alerta indicando el mensaje: **Vehículo Añadido**. Se cambia a otra vista y se habilitan los botones de crear ruta.

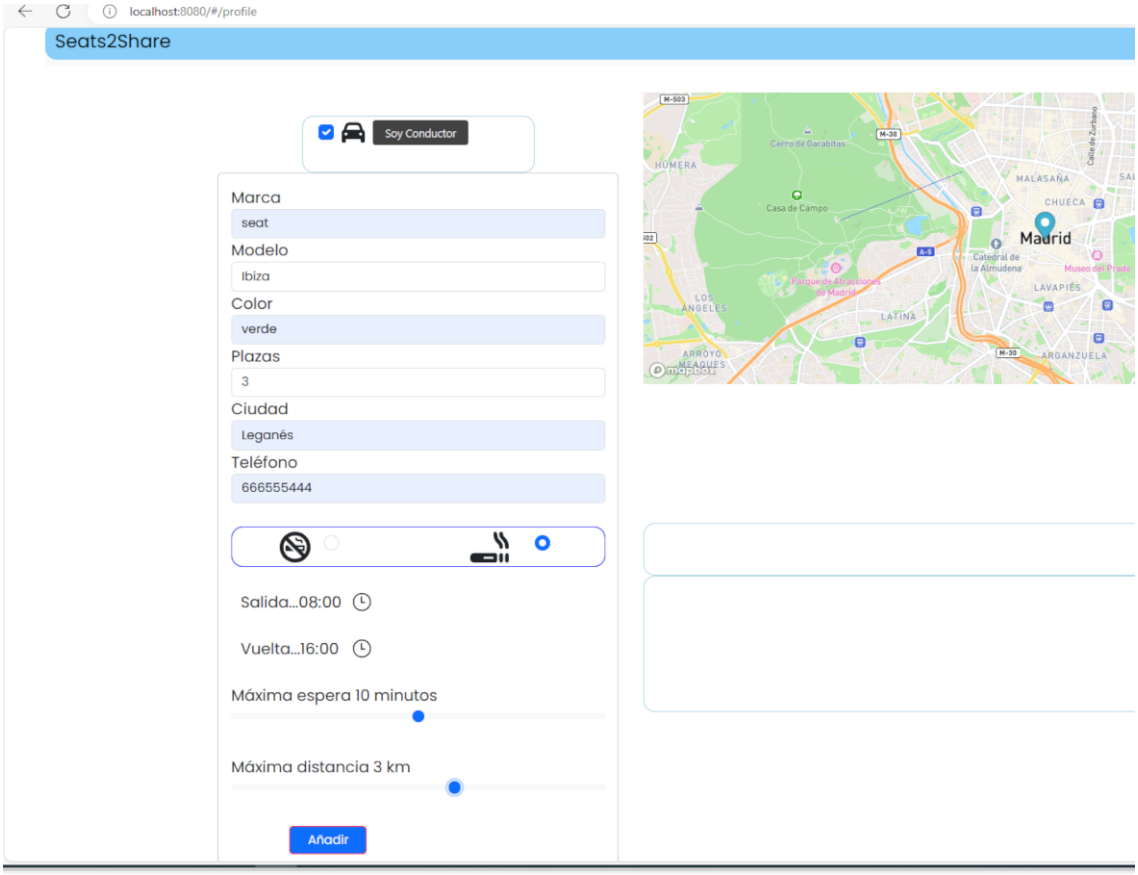


Figura 5.26: Prueba Verificación 6.1: Registrar Conductor.

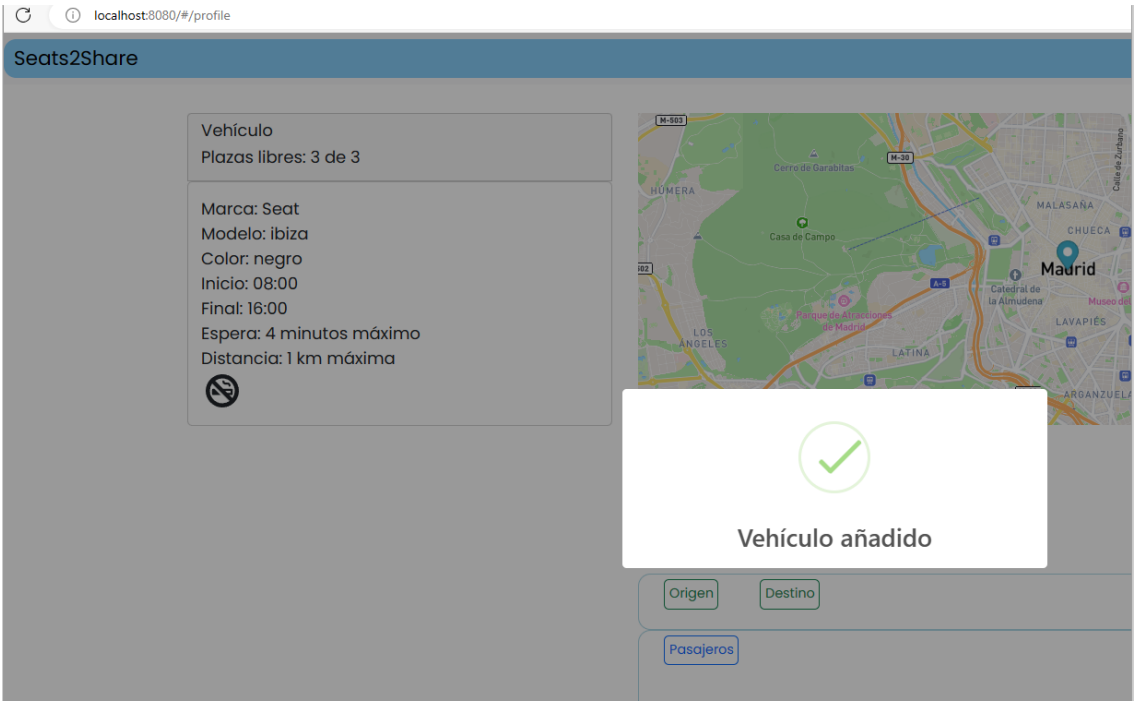


Figura 5.27: Prueba Verificación 6.2: Perfil Conductor.

PV7: Crear Ruta Conductor.

- Se marca en el mapa el origen mediante el botón Origen y el destino mediante el botón Destino.
- El resultado esperado es que se dibuje el recorrido y se quede centrado el mapa entre los puntos origen y destino. El botón Guardar se habilita.

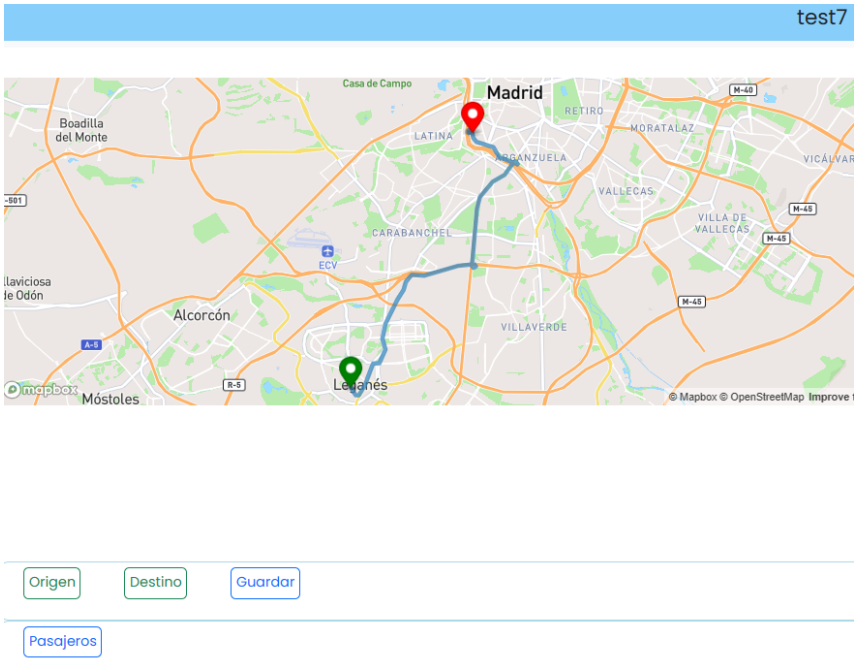


Figura 5.28: Prueba Verificación 7: Crear Ruta.

PV8: Guardar Ruta.

- Se pulsa el botón Guardar.
- El resultado esperado es que aparezca una alerta indicando el mensaje: **Ruta Añadida**.

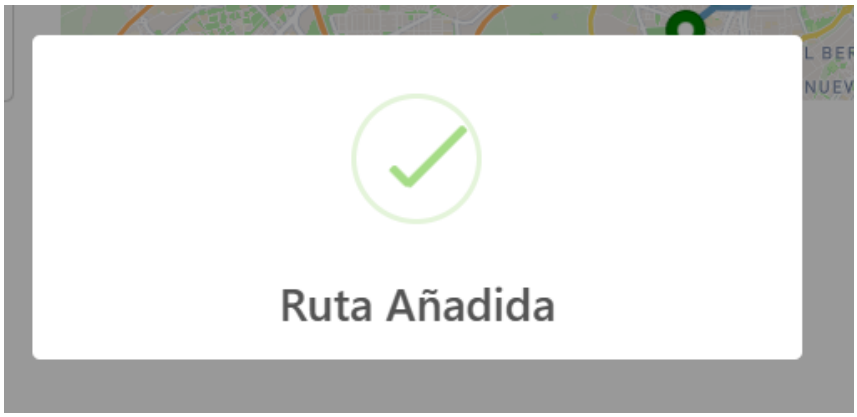


Figura 5.29: Prueba Verificación 8: Guardar Ruta.

PV9: Verificar Recorrido Guardado.

- Se quiere verificar si el recorrido se ha guardado comparando resultado con P11 y P12.
- El resultado esperado es que aparezcan las coordenadas del recorrido en el campo *routePath* en lugar de *null* como ocurría en P12.

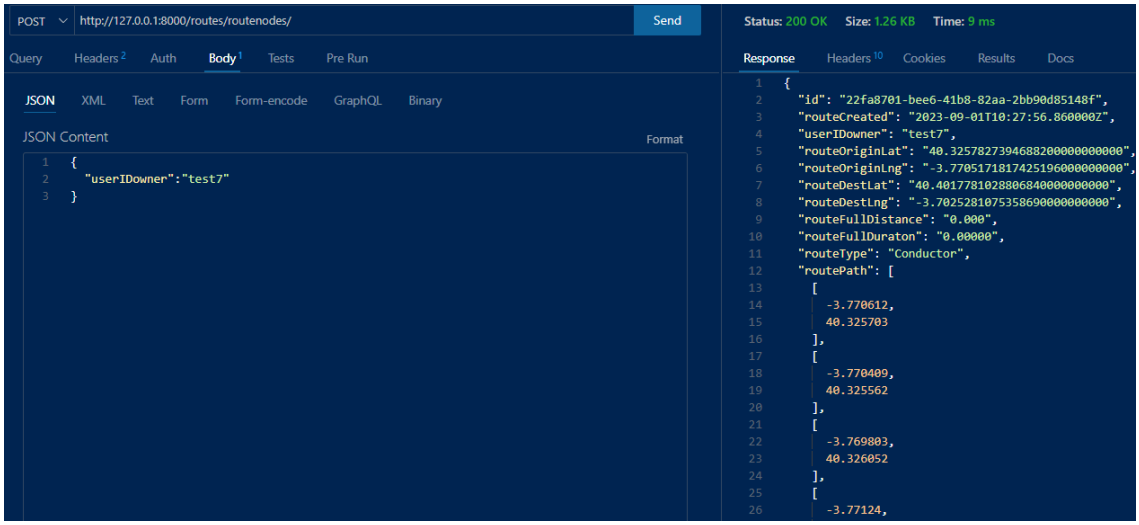


Figura 5.30: Prueba Verificación 9: Verificar Recorrido.

PV10: Verificar Grafo Generado.

- Se quiere verificar si se ha generado el grafo con los conductores en su área.
- El resultado esperado es que aparezcan las nodos de los conductores en su área según las condiciones establecidas en las distancias de aceptación.

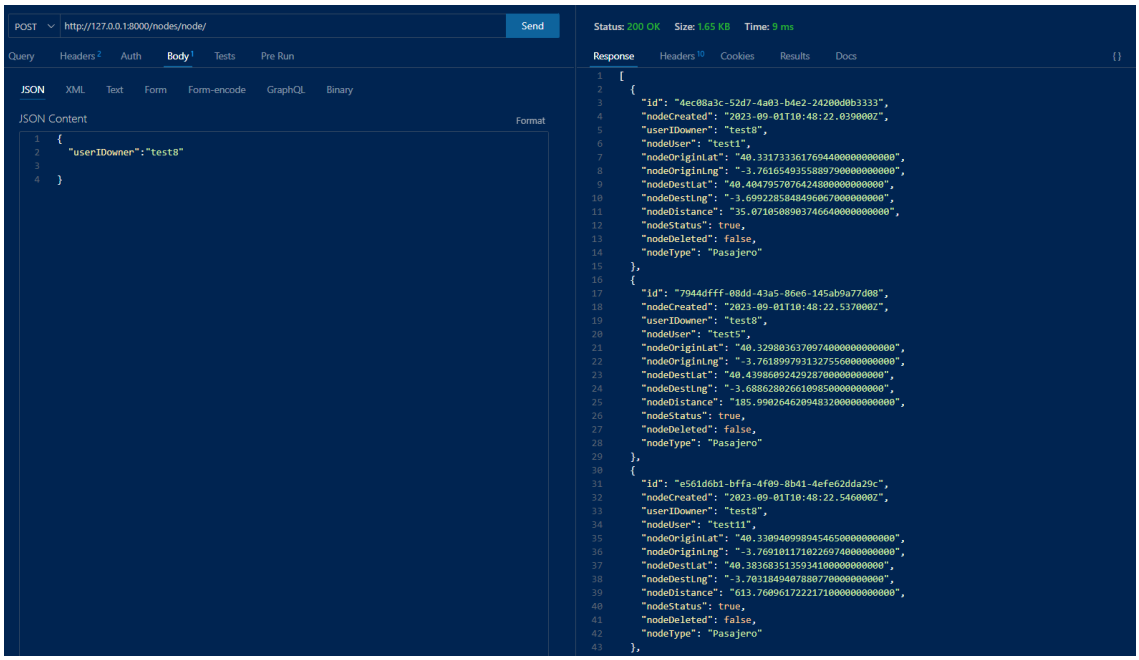


Figura 5.31: Prueba Verificación 10: Verificar Grafo.

PV11: Buscar Conductores Cercanos.

- Se ha pulsado el botón Conductores para ver los conductores cercanos que cumplen las restricciones de recogida.

- El resultado esperado es que aparezcan los conductores en el área de distancia máxima de recogida por parte del pasajero y que cumplan con las distancias de los conductores.

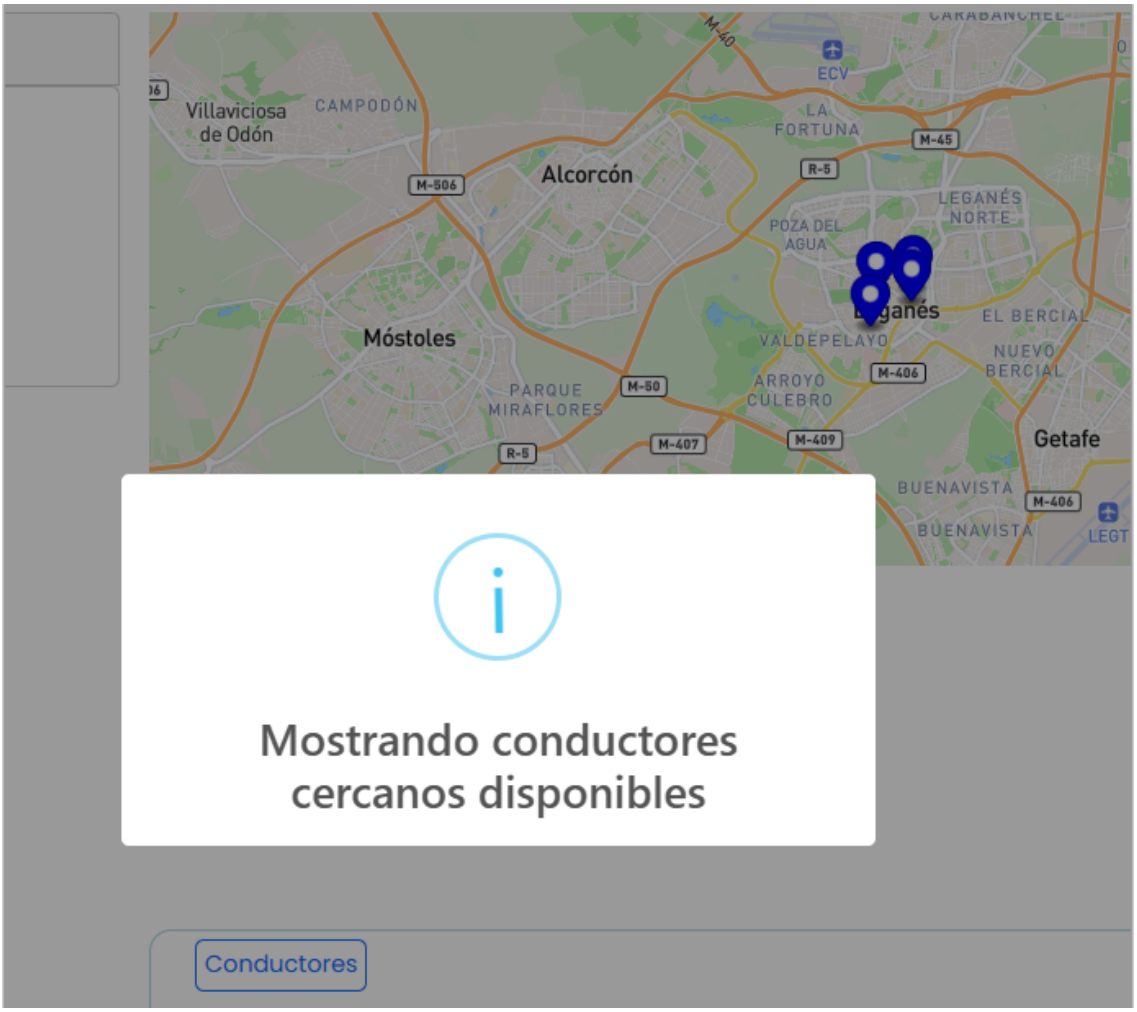


Figura 5.32: Prueba Verificación 11: Buscar Conductores.

PV12: Seleccionar conductor.

- Se ha pulsado sobre el marcador de un conductor.

- El resultado esperado es la información de las condiciones de horarios y plazas disponibles, así como el botón Reservar Plaza.

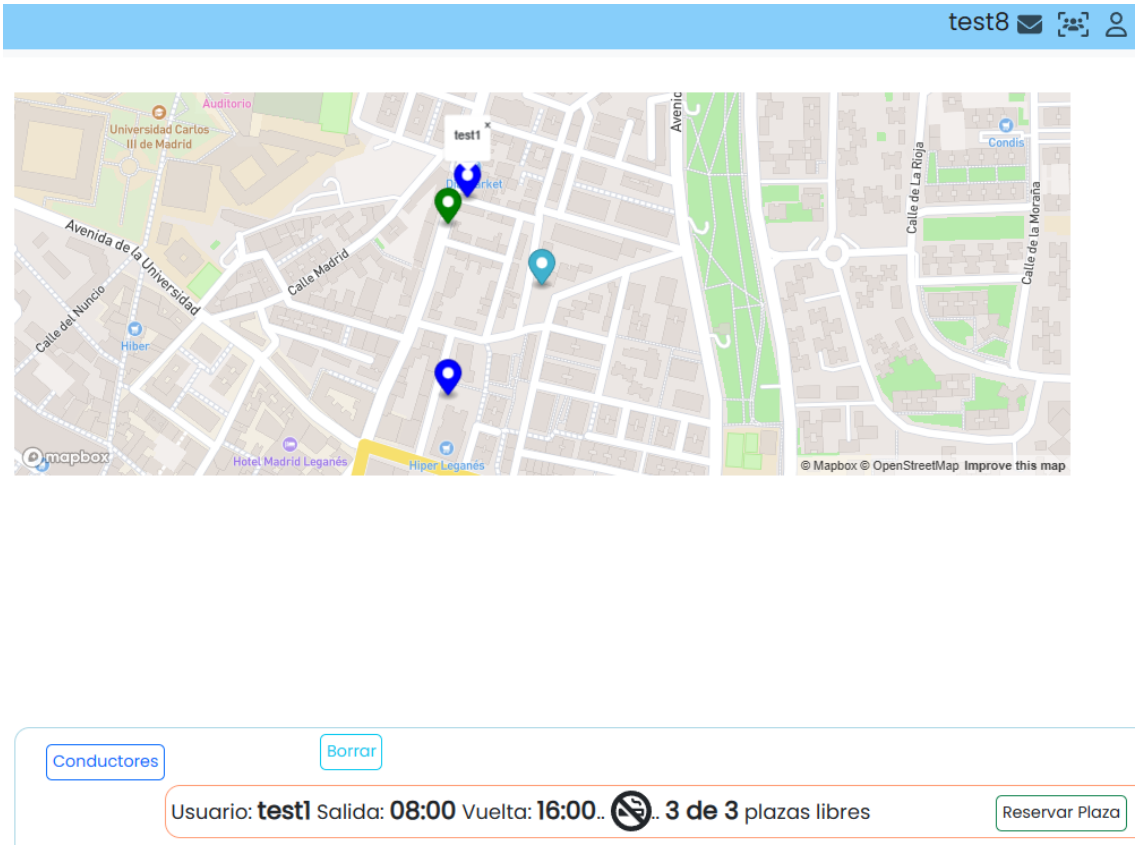


Figura 5.33: Prueba Verificación 12: Seleccionar Conductor.

PV13: Reservar Plaza.

- Se han establecido las condiciones de la reserva y se pulsa el botón Reservar.
- El resultado esperado es una alerta con el mensaje Reserva Realizada y que ésta aparezca en el panel de reservas tanto del pasajero como del conductor.

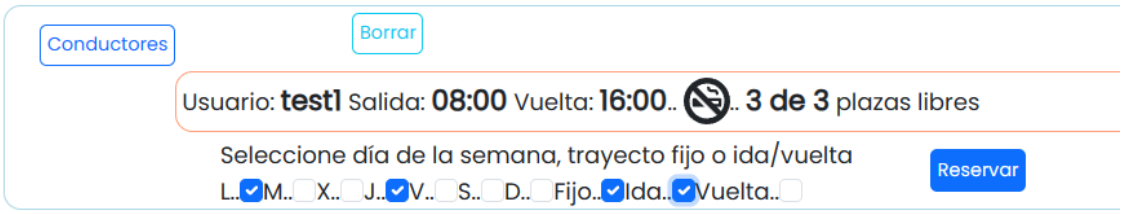


Figura 5.34: Prueba Verificación 13.1: Reservar Plaza.

Figura 5.35: Prueba Verificación 13.2: Mensaje Confirmación.

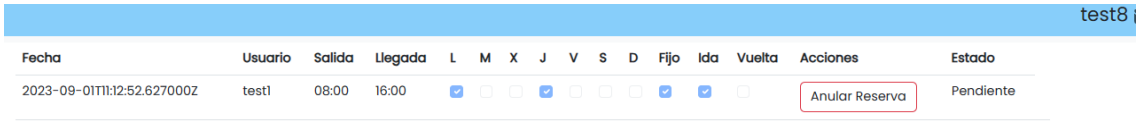


Figura 5.36: Prueba Verificación 13.3: Panel Reservas Pasajero.

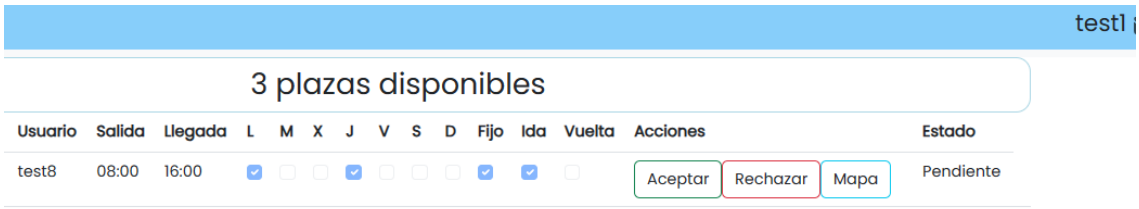


Figura 5.37: Prueba Verificación 13.4: Panel Reservas Conductor.

PV14: Aceptar Reserva.

- El conductor acepta la reserva.
- El resultado esperado es una alerta con el mensaje Reserva Aceptada y el cambio de estado del panel de reservas tanto del pasajero como del conductor.

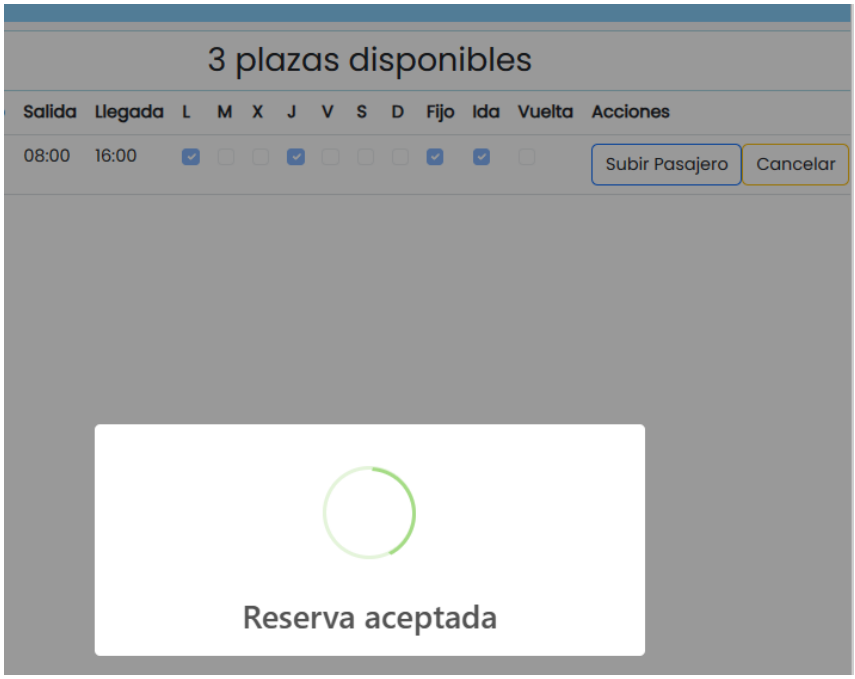


Figura 5.38: Prueba Verificación 14.1: Panel Reservas Conductor.

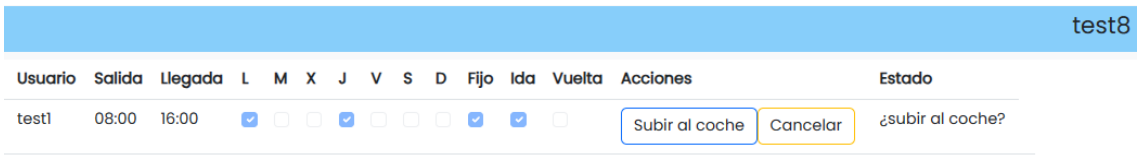


Figura 5.39: Prueba Verificación 14.2: Panel Reservas Pasajero.

PV15: Subir Pasajero.

- El conductor indica que el pasajero se sube al vehículo.
- El resultado esperado es una alerta con el mensaje *A la espera de confirmación del pasajero* y el estado ¿Confirmación Pasajero? hasta que el pasajero confirme esa acción.

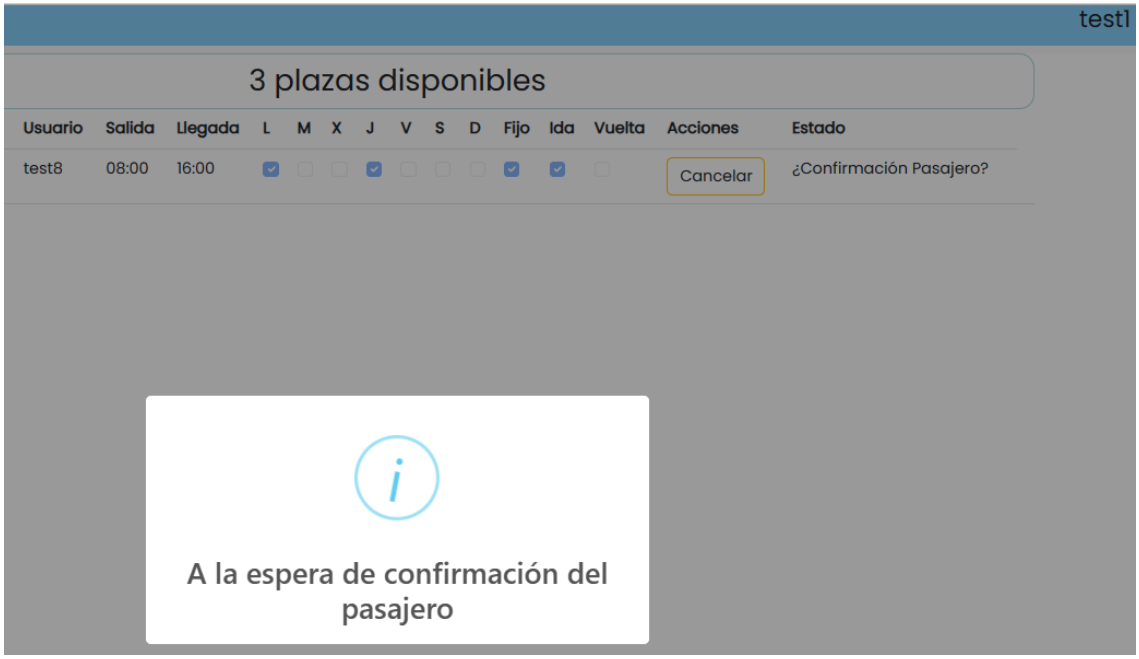


Figura 5.40: Prueba Verificación 15: Subir Pasajero.

PV16: Trayecto Activo.

- El pasajero confirma que está en el vehículo tras la PV15.
- El resultado esperado es una alerta con el mensaje Trayecto activo y el estado de la reserva es Activo. Se espera que el conductor tenga 2 plazas disponibles.

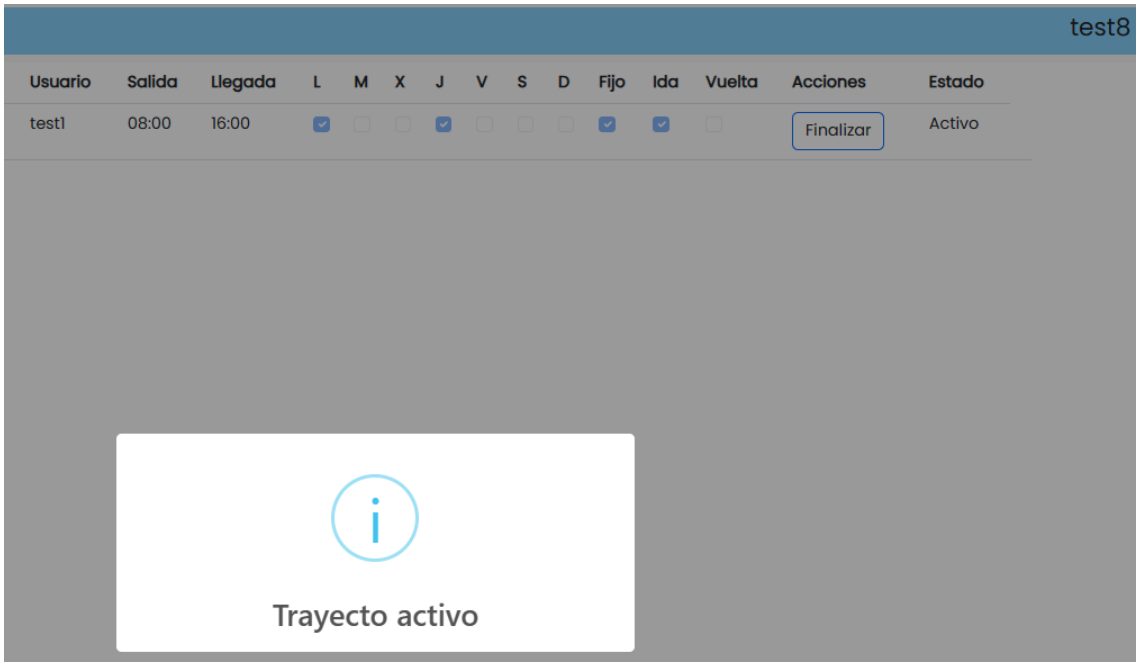


Figura 5.41: Prueba Verificación 16.1: Panel Reservas del pasajero Trayecto Activo.

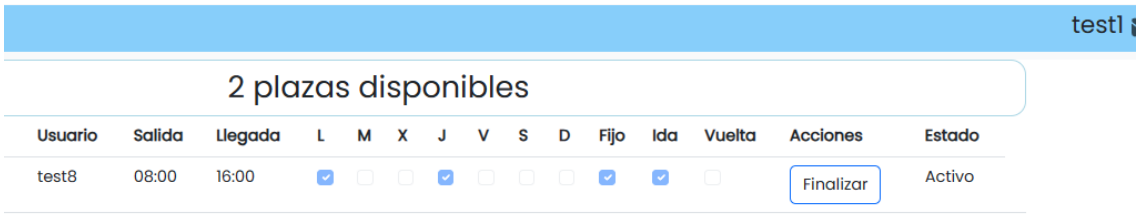


Figura 5.42: Prueba Verificación 16.2: Panel Reservas del conductor Trayecto Activo.

PV17: Finalizar Viaje.

- El pasajero o conductor pulsa el botón finalizar.
- El resultado esperado es un cambio en el estado de la reserva a inactivo. Se espera que el conductor tenga 3 plazas disponibles. Se puede finalizar la reserva y valorar el viaje.

Figura 5.43: Prueba Verificación 17: Finalizar Viaje.

PV18: Enviar Mensaje.

- El pasajero test8 envía un mensaje al conductor test1.
- El resultado esperado es el envío del mensaje. El pasajero lo tendrá en su lista de mensajes enviados y el conductor deberá tenerlo en su lista de mensajes recibidos.

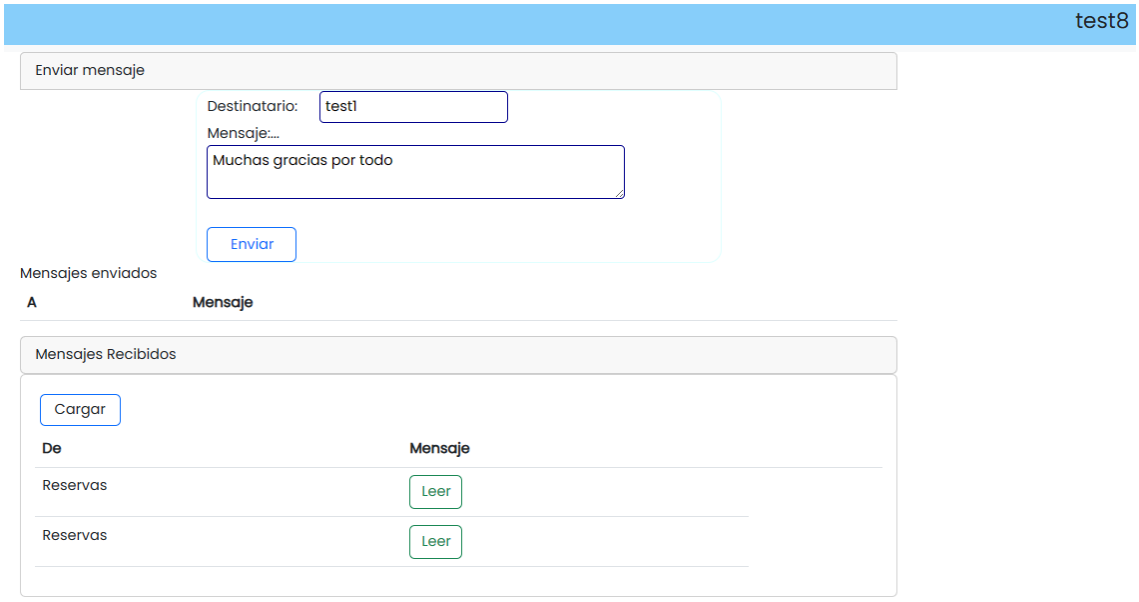


Figura 5.44: Prueba Verificación 18: Enviar Mensaje.

PV19: Recibir Mensaje.

- El conductor test1 ha recibido el mensaje de test1 y le ha dado al botón Leer.
- El resultado esperado es la recepción del mensaje . El conductor puede responder o borrar el mensaje.

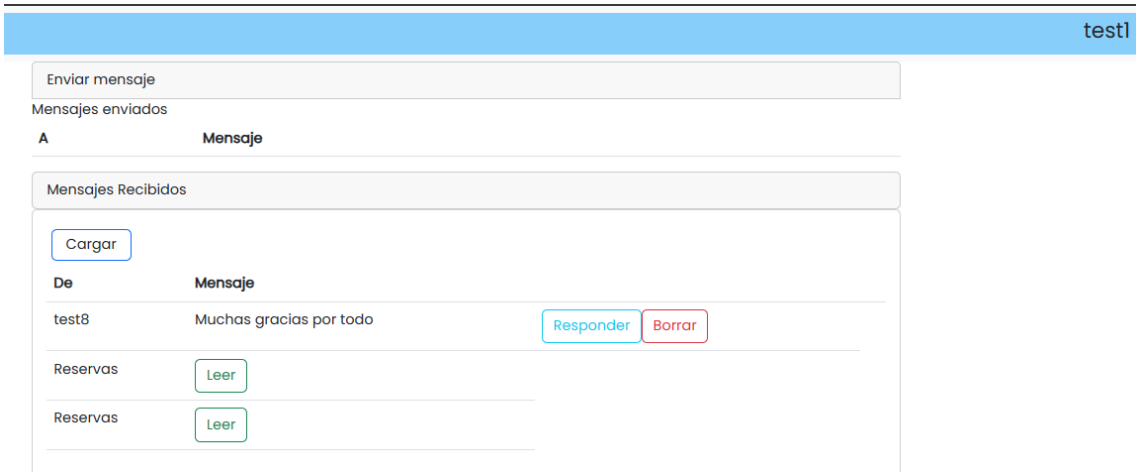


Figura 5.45: Prueba Verificación 19: Recibir Mensaje.

PV20: Valorar usuario.

- El conductor test1 ha pulsado el botón Valorar en el panel de reservas y ha valorado al pasajero test8.
- El resultado esperado es que en el perfil del pasajero test8 aparezca la valoración.

Figura 5.46: Prueba Verificación 20.1: Valorar usuario.

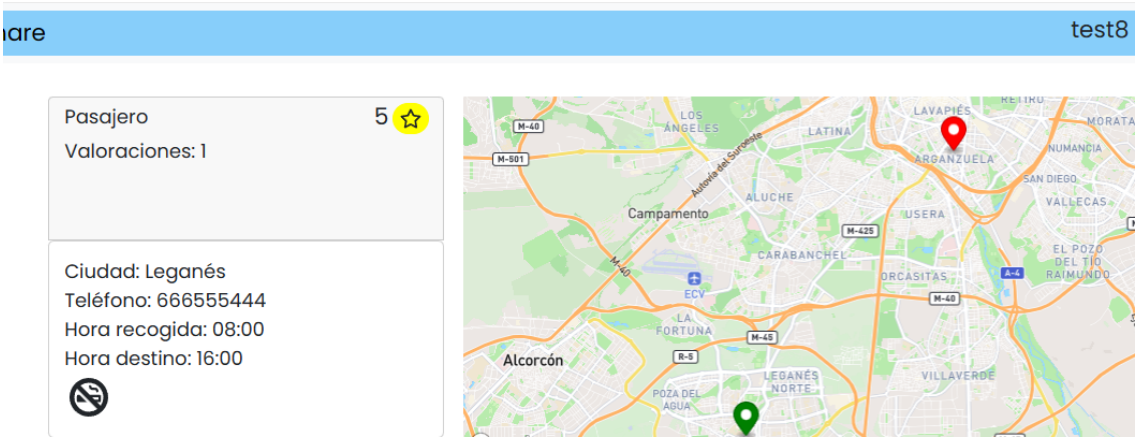


Figura 5.47: Prueba Verificación 20.2: Valoración en perfil.

PV21: Cambiar configuración.

- El pasajero test8 ha cambiado su horario de salida a las 13:00.

- El resultado esperado es que en el perfil del pasajero test8 se muestre el cambio.

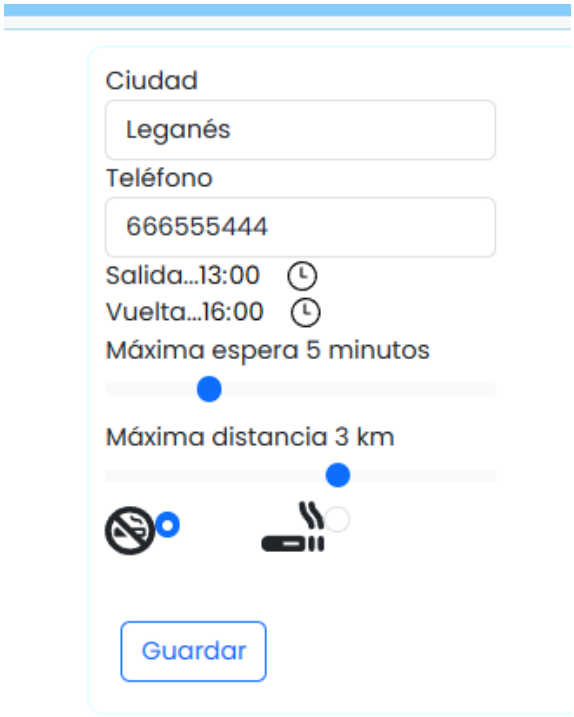


Figura 5.48: Prueba Verificación 21.1: Cambiar configuración.

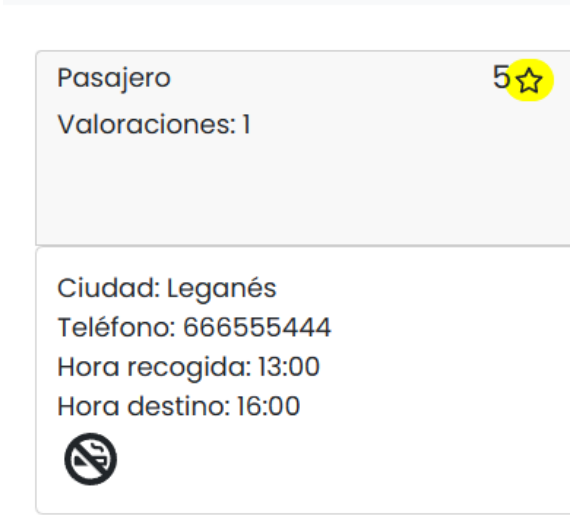


Figura 5.49: Prueba Verificación 21.2: Cambio en perfil.

Se da por finalizada la batería de pruebas.

Se han probado y verificado casi la totalidad de los requisitos funcionales. Se han omitido muchas de las comprobaciones de verificación de acciones tales como ¿Desea borrar? o ¿Actualizar Datos?, por citar unos ejemplos.

5.4.2. Errores encontrados

Por parte del Frontend los errores que se han detectado han sido pocos, pero sí hay uno que ha sucedido algunas veces y se muestra en la Figura 5.50.

Es causado porque el servicio de Mapbox no ha terminado de cargar el estilo del mapa o durante la carga hubo algún problema, no afecta al funcionamiento pero aparece esa ventana de error. La carga del mapa está bajo una sentencia *try catch* pero igualmente ha aparecido. Se cierra la ventana del error y la aplicación funciona sin problemas.

Otro fallo que puede aparecer en el Frontend es causado por Vue 3 y es que tras finalizar un viaje pueda aparecer por parte del conductor la opción de aceptar reserva sumado a reactivar. Como el panel se actualiza cada diez segundos desaparece sin problemas.

No se han detectado más fallos visuales que se repitan lo suficiente como para afectar a la experiencia de usuario.

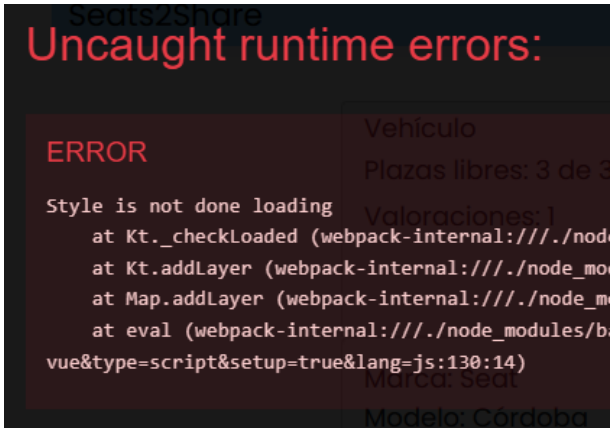


Figura 5.50: Error Style not load.

5.5. Conclusiones

En este capítulo se han descrito todos los pasos realizados para ajustar las fases de análisis y diseño al proyecto así como las pruebas de verificación a las que se ha visto sometido el proyecto. Comenzando con los componentes y herramientas usadas para el proceso de desarrollo del proyecto, continuando con la descripción de cómo se han implementado los distintos componentes que forman la aplicación.

Que las peticiones HTTP funcionen correctamente en la API es fundamental para evitar que la integración con el Frontend esté libre de errores. El capítulo finaliza con las pruebas y verificaciones realizadas para probar la funcionalidad de cada uno de los componentes del proyecto.

Comprobar todos los pasos que se han de dar para verificar los requisitos funcionales es una tarea larga y repetitiva, pero con la batería de pruebas realizada se puede afirmar que el proyecto es un sistema funcional y que cumple con los requisitos previstos en la fase de análisis.

Capítulo 6

Planificación del proyecto

6.1. Introducción

En este capítulo vamos a mostrar la planificación para el desarrollo del proyecto. La planificación es una de las partes del estudio del desarrollo de un proyecto que más cambios y alteraciones sufre. A lo largo del tiempo, surgen factores inesperados que al no formar parte de la planificación, alteran esta y por tanto el calendario previsto se ve modificado.

Se mostrán los diagramas de Gantt correspondientes a las planificaciones iniciales previstas y las reales. El capítulo finalizará con unas conclusiones sobre la planificación realizada ya que realizar una buena planificación es fundamental para la finalización de un proyecto en tiempo y forma.

6.2. Planificación

A la hora de planificar el proyecto se tuvo en cuenta que estaría formado por entidades independientes que tendrían que integrarse posteriormente.

El desarrollo se planificó en seis fases:

- **Fase 1**: Investigación sobre los componentes a desarrollar y los medios tecnológicos disponibles para obtener datos geográficos para ser usados de cara a la elaboración de un grafo para las rutas. A esta primera fase se le dio un tiempo estimado de mes y medio.

- **Fase 2**: Diseño y construcción de una API REST en Django REST Framework (DRF). Elemento principal del proyecto. En esta fase se estudia también el tipo de BBDD a utilizar. Se establece un tiempo de mes y medio para esta fase.

- **Fase 3**: Aplicación web cliente-servidor. En esta fase se desarrolla el Frontend y se irá integrando con la API a medida que el desarrollo avance. Para esta fase se planificó un tiempo de dos meses.

- **Fase 4**: Creación de rutas y grafos. En esta fase tanto el Frontend como el Backend ya están conectados y se aumentan las funcionalidades relacionadas con la estructura

del grafo para las rutas de los usuarios. Para esta fase se planificó un tiempo de mes y medio.

- **Fase 5**: Integración y pruebas de módulos. En esta fase se prueba todo el conjunto de funcionalidades que se han previsto en el análisis y diseño. Se estableció un tiempo de un mes para ello.
- **Fase 6**: Añadir funciones extras. Esta fase añade aquellas funcionalidades que no habían sido incluidas en las primeras etapas y conforme se integran se van probando en su trabajo conjunto. Se establece un tiempo de mes y medio para esta fase.
- **Memoria**: Finalizadas las fases se documenta todo el código y se elabora la memoria del proyecto. Se establece un tiempo de mes y medio para ello.

Paralelamente a estas fases de desarrollo, están las fases de análisis y diseño que transcurren durante todo el desarrollo del proyecto. Refinando, rehaciendo y modificando varias veces el diseño y codificación para poder cumplir con los requisitos inicialmente previstos.

El proyecto se planificó con un calendario entre septiembre del 2022 hasta julio del 2023. Varios acontecimientos externos retrasaron y modificaron dicho calendario extendiéndolo hasta septiembre del 2023, fecha prevista de su finalización.

Se ha aplicado el método ágil de desarrollo basado en iteraciones, primero con un prototipo limitado funcional y luego cada iteración añade funcionalidades extra o corrige una iteracción anterior.

Las iteraciones producidas se muestran a continuación:

- **IT1**: API REST básica en DRF que permite crear, consultar y modificar usuarios.
- **IT2**: Se añaden las aplicaciones **drivers** y **passengers**.
- **IT3**: Aplicación web hecha en Vue 3 con lectura de datos desde API REST.
- **IT4**: Conexión con BBDD en MongoDB.
- **IT5**: Aplicación web con formularios para Conductores y Pasajeros.
- **IT6**: Se añade visión de cartografía digital en aplicación web. Inicialmente con Leaflet-Routing-Machine.
- **IT7**: Se añade la aplicación **routes**.
- **PRO1**: Prototipo Funcional 1. Primer prototipo con capacidad funcional, es el que se muestra a los directores del proyecto.
- **IT8**: Aplicación web con uso de Mapbox.
- **PRO2** Prototipo Funcional 2. Mejora en la presentación de rutas. Este prototipo es el último que se muestra a los directores del proyecto.
- **IT9**: Se añade la aplicación **nodes**.

- **IT10**: Se incluye autenticación mediante JWT.
- **IT11**: Se añade BBDD PostgreSQL a la estructura del proyecto.
- **IT12**: Aplicaciones **trips** y **messages** añadidas.
- **IT13**: Aplicación **rates**.
- **PRO3**: Prototipo Final con la requisitos funcionales integrados.

Las iteraciones a partir de **IT6** incluyen las mejoras tanto en el Frontend como en el Backend.

6.3. Diagramas de Gantt

Los diagramas de Gantt nos muestran de forma clara la evolución de un proyecto y nos permiten comprobar la desviación sufrida en el eje temporal. La planificación inicial era muy conservadora en cuantos a los tiempos para las seis fases. En la Figura 6.1 podemos observar la planificación prevista inicialmente.

La Figura 6.2 nos muestra en cambio el tiempo empleado en cada fase y las iteraciones a lo largo del tiempo.

Figura 6.1: Diagrama de Gantt Planificación Prevista.

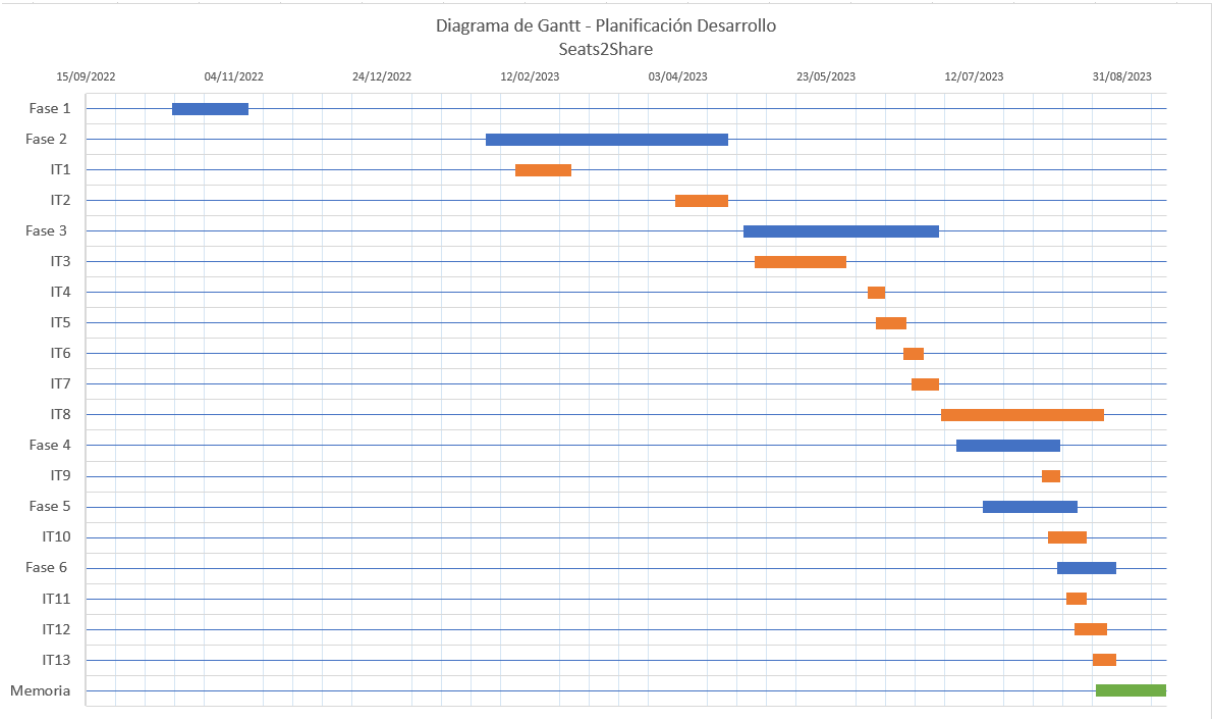


Figura 6.2: Diagrama de Gantt Planificación Real.

En la Figura 6.2 se muestran las fases en color azul, las iteraciones en color naranja y la memoria en color verde. Se puede observar la desviación temporal con respecto a la planificación inicial y el gran hueco en las fases iniciales del proyecto.

El número de horas empleadas para la elaboración del proyecto ha sido en torno a las 445 horas en total.

6.4. Conclusiones

En este capítulo se ha visto la planificación inicial del proyecto, las fases en las que se dividió su elaboración y las iteraciones que han ido surgiendo a lo largo de todo el proceso de desarrollo. El desvío en cuanto a la planificación inicial ha sido bastante evidente, aunque no en cuanto al número de horas previstas para su finalización.

Finalmente, los diagramas de Gantt mostrados reflejan las diferencias cuando se elabora una planificación al inicio de un proyecto y su evolución a lo largo del tiempo para dar lugar a una aplicación completa y funcional.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Introducción

En este capítulo se exponen las conclusiones una vez el proyecto ha sido finalizado. Sin duda los retos superados a lo largo del desarrollo de Seats2Share han permitido sentar las bases para los trabajos futuros que se pueden poner en marcha.

7.2. Conclusiones

Este proyecto tenía como principal motivación ser una herramienta para la toma de decisión, en este caso en viajes en vehículos compartidos con trayectos en común. Para ello se muestra al usuario aquellos usuarios cercanos a su posición que cumplen con sus condiciones para realizar un viaje.

Las sociedades actuales se están viendo forzadas a resolver varios retos y uno de ellos es la movilidad en grandes ciudades, debido al uso masivo de transporte privado con un único pasajero. La infraestructura de transporte público no puede asumir todo el flujo de personas que se mueven en una ciudad y el transporte comercial privado, taxi o VTC es de carácter individual en su mayor parte. Hay medidas para incentivar el uso de transportes más eficientes para el medio ambiente y al mismo tiempo reducir el número de vehículos en circulación. Una forma de reducción sería que los usuarios compartieran viajes si tienen trayectos coincidentes o cercanos a los que regularmente hacen, fomentando así la economía colaborativa y la movilidad sostenible.

Los objetivos que se marcaron para el desarrollo de la aplicación se han cubierto en su totalidad. Comenzando con el estudio de las distintas fuentes de datos de mapas digitalizados y la familiarización de los algoritmos de búsquedas de caminos más cortos que emplean para realizar los recorridos. Esto nos permitió obtener los datos necesarios para que fueran tratados por parte del Backend. Para el Backend se apostó por un entorno ya consolidado, Django REST Framework (DRF), el cual nos permite servir de intermediario entre la BBDD y el Frontend.

El diseño del proyecto en tres niveles, con una capa de presentación basada en Vue 3, una capa de aplicación basada en DRF y una capa de datos compuesta por dos BBDD: MongoDB y PostgreSQL; permite que la aplicación pueda ser mantenida y escalada sin

afectar a los demás niveles. Además la API REST del nivel de aplicación nos garantiza que cualquier dispositivo pueda acceder al mismo simplemente mediante peticiones HTTP, garantizando así un futuro uso de aplicación móvil, la cual se conectaría a dicha API para realizar las operaciones.

Para la capa de presentación, se diseñó e implementó la aplicación mediante Vue 3 que nos permite tener una interfaz clara, reactiva y funcional en cualquier dispositivo al tener un diseño adaptable a los distintos tamaños de pantallas. Las acciones previstas en los requisitos se pueden realizar con la intervención mínima por parte del usuario, con mensajes de confirmación en cada paso. Al mismo tiempo, Vue 3 nos permite un diseño por componentes que nos facilita la tarea de tener un entorno con bajo grado de acoplamiento. El número de componentes diseñados nos permite mejorar elementos específicos e individuales de la aplicación con la tranquilidad de no tener problemas de acoplamiento con otros componentes.

Se ha integrado un sistema de autenticación que permite un registro y acceso seguro a la aplicación, así, un usuario al registrarse o acceder, puede seleccionar su rol, ya sea conductor o pasajero y confeccionar su perfil en función de sus intereses, introduciendo los datos de su vehículo y plazas disponibles si es conductor, además de otras características como tiempo máximo de espera o distancia máxima de desvío aceptada de su ruta. El usuario puede indicar en un mapa sus puntos elegidos de origen y destino y consultar aquellos usuarios más cercanos que cumplan sus intereses, por ejemplo un pasajero vería los conductores próximos en su área y así poder realizar las reservas de viajes. Tienen a su disposición un mapa con todos los usuarios activos para así tener una mejor referencia de cara a la elección. Además se ha creado un sistema de reservas de plazas que permiten gestionar los distintos viajes en función de las plazas disponibles por parte del conductor.

Todas estas acciones se realizan en tiempo real, actualizándose en el acto en las BBDD permitiendo al usuario tomar las decisiones sobre sus desplazamientos sin esperas. A su vez se ha integrado un sistema de mensajería en la aplicación que permite a los usuarios coordinar sus viajes o puntos de recogida. Para una mayor seguridad y facilitar la selección de los usuarios, se ha creado también un sistema de valoración para que los usuarios puedan puntuarse en función de cómo fue su experiencia al viajar.

La aplicación Seats2Share es una aplicación totalmente funcional y lista para usar, por lo que damos por finalizado el desarrollo de una primera versión del producto.

7.3. Trabajos futuros

Este proyecto es una aplicación que puede ser ampliada en muchos aspectos y corregir algunos defectos que no han sido resueltos.

Sin tener que tocar la API REST, las primeras mejoras podrían ser por el lado Frontend, en especial para permitir un mejor comportamiento en dispositivos de pantalla pequeña como smartphones, haciendo que la aplicación sea aún más adaptable o *responsive*, ya que precisamente al ser una aplicación para viajes en vehículos, ese dispositivo estaría disponible para todos los usuarios. También una mejora visual en la interfaz, ya

que no se ha tenido en cuenta dicho aspecto en el diseño. Igualmente un panel para el administrador, para que pueda controlar todo de forma mucho más visual es una mejora muy interesante. Si además se le añade el usar una BBDD específica para grafos y la integramos para que se pueda visualizar se obtendría mucha información para análisis de comportamiento y uso.

El realizar una aplicación específica para dispositivos Android era uno de los extras que si la planificación se hubiera cumplido podría haber sido integrada en el proyecto, por lo que es una opción como trabajo futuro. La misma opción se podría decir de una aplicación específica para iOS. Integrar un sistema de autenticación más seguro sería otro paso más a tener en cuenta, sustituyendo el actual por uno como Firebase [24] que permitiría acceso y registro desde múltiples fuentes y redes sociales. Eso supondría algunos cambios en la API REST pero no en la BBDD ya que para la gestión de usuarios se usa PostgreSQL y es una BBDD de contrastada experiencia para ello.

Durante el proyecto no se consideró en ningún momento un sistema comercial o de pago para el producto, pero se podría integrar un sistema de acceso preferente o *premium* para usuarios, que sólo puedan escoger entre un número determinado, o filtros específicos, las posibilidades son numerosas en cuanto a la personalización de esas opciones si se integra el sistema por suscripción.

Un aspecto a tener en cuenta y que no se ha considerado en el proyecto por ser en principio un prototipo, es el tema de la privacidad y verificación de los usuarios para evitar que haya problemas legales. Así habría que adoptar un sistema en el cual el usuario tendría que acreditar su identidad real. De esta forma la persona tiene la certeza que el conductor o pasajero es la persona que en teoría dice ser, dando así una seguridad en el uso de la aplicación.

En cuanto a personalización del perfil, se pueden integrar imágenes de los usuarios o de los vehículos para una mejor identificación. Las comunicaciones y mensajes entre los usuarios no están cifradas, son simples mensajes de texto almacenados en la BBDD, una mejora sería cifrar esas comunicaciones. Tampoco se ha mostrado el aviso de cookies, si bien la aplicación no utiliza cookies salvo las obligatorias para almacenamiento, pero si se considera un sistema de métricas o análisis de usuarios, sí que habría que integrarlo. Al igual que no hay un aviso legal de las condiciones del servicio, ni datos de contacto, etc.

Hay también mejoras en cuanto al grafo de usuarios cercanos para mostrar, ya que con el diseño actual si el número de usuarios es muy elevado, el tiempo empleado en añadir nodos al usuario se iría incrementado y tendría un coste computacional cada vez mayor. Una opción sería dividir las ciudades en cuadrículas y agrupar en dichas cuadrículas los usuarios en función de su posición geográfica, así la consulta sólo sería en la cuadrícula a la que pertenece el usuario o en las adyacentes, omitiendo la consulta en el resto de cuadrículas y por tanto reduciendo considerablemente el número de elementos a buscar, mejorando la eficiencia y el tiempo empleado en ello.

Es evidente que aún hay mucho por hacer y esto no ha hecho más que empezar.

Bibliografía

[1] "Los líderes en materia de movilidad dialogan sobre la innovación en el transporte urbano de superficie - UCCI — ciudadesiberoamericanas.org." https://ciudadesiberoamericanas.org/2022/03/11/los-lideres-en-materia-de-movilidad-dialogan-sobre-la-innovacion-en-el-transporte-urbano-de-superficie/, Mar 2022.

[2] J. J. Vázquez, S. L. N. Alonso, and F. S. Ramos, "La economía colaborativa en el sector de la movilidad y el transporte: Hacia la configuración de un modelo sostenible,"

[3] O. Wiki, "Api — openstreetmap wiki,." https://wiki.openstreetmap.org/w/index.php?title=API&oldid=2414256, 2022. [Online; accessed 18-septiembre-2023].

[4] Google, "Las API de Google Maps Platform por plataforma — Google Maps Platform — Google for Developers — developers.google.com." https://developers.google.com/maps/apis-by-platform?hl=es-419, 2023. [Accessed 18-09-2023].

[5] OpenStreetMap, "ES:Acerca de OpenStreetMap - OpenStreetMap Wiki — wiki.openstreetmap.org." https://wiki.openstreetmap.org/wiki/ES:Acerca_de_OpenStreetMap, 2023. [Accessed 09-09-2023].

[6] L. E. G. Manrique, "Los mapas digitales y la invención de la realidad — Política Exterior — politicaexterior.com." https://www.politicaexterior.com/los-mapas-digitales-y-la-invencion-de-la-realidad/, 2021. [Accessed 11-09-2023].

[7] OpenStreetMaps, "Elements - OpenStreetMap Wiki — wiki.openstreetmap.org." https://wiki.openstreetmap.org/wiki/Elements, 2023. [Accessed 09-09-2023].

[8] Nominatin, "Overview - Nominatim Manual — nominatim.org." https://nominatim.org/release-docs/develop/api/Overview/, 2023. [Accessed 18-09-2023].

[9] Nominatim, "Basic Installation - Nominatim Manual — nominatim.org." https://nominatim.org/release-docs/develop/admin/Installation/#downloading-and-building-nominatim, 2023. [Accessed 18-09-2023].

[10] Leafletroutingmachine, "Easy, flexible routing for leaflet." https://www.liedman.net/leaflet-routing-machine/, 2023. [Accessed 18-09-2023].

[11] ProjectOSRM, "OSRM API Documentation — project-osrm.org." http://project-osrm.org/docs/v5.24.0/api/#, 2023. [Accessed 18-09-2023].

[12] Mapbox, "Mapbox GL JS — docs.mapbox.com." https://docs.mapbox.com/mapbox-gl-js/guides/, 2023. [Accessed 26-08-2023].

[13] Google, "Platform Pricing & API Costs - Google Maps Platform — mapsplatform.google.com." https://mapsplatform.google.com/pricing/, 2023. [Accessed 10-09-2023].

[14] J. Hamme, "Customizable route planning in external memory," *Bachelor Thesis, Institute of Theoretical Computer Science*, 2013.

[15] Mapbox, "Examples — Traffic Data — docs.mapbox.com." https://docs.mapbox.com/data/traffic/guides/example/, 2023. [Accessed 12-09-2023].

[16] TheOtherJesse, "Archivo:Law-of-haversines.svg - Wikipedia, la enciclopedia libre — es.wikipedia.org." https://es.wikipedia.org/wiki/Archivo:Law-of-haversines.svg, 2023. [Accessed 13-09-2023].

[17] B. Rouberol, "haversine — pypi.org." https://pypi.org/project/haversine/, 2023. [Accessed 26-08-2023].

[18] Repsol, "Qué es el carsharing y cómo funciona — Repsol — repsol.es." https://www.repsol.es/particulares/asesoramiento-consumo/que-es-el-carsharing/, 2023. [Accessed 18-09-2023].

[19] H. S. SL, "App para compartir coche en ciudades - HOOP Carpool — hoopcarpool.com." https://hoopcarpool.com/, 2023. [Accessed 18-09-2023].

[20] Europe-carpooling, "Compartir coche en Europa, carpool spain — europe-carpooling.es." https://www.europe-carpooling.es/, 2023. [Accessed 18-09-2023].

[21] blablacar, "Viaja de forma económica y sostenible." https://www.blablacar.es/, 2023. [Accessed 18-09-2023].

[22] C. Larman, L. M. H. Rodríguez, and H. C. Anaya, *UML y Patrones: Introducción al análisis y diseño orientado a objetos*, vol. 2. Prentice Hall, 1999.

[23] M. A. Chaves, "La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software," *InterSedes: Revista de las Sedes Regionales*, vol. 6, no. 10, pp. 1–13, 2005.

[24] Firebase, "Firebase Authentication — Acceso multiplataforma simple y sin costo — firebase.google.com." https://firebase.google.com/products/auth?hl=es, 2023. [Accessed 18-09-2023].

[25] Sigmajs, "Sigma.js — sigmajs.org." https://www.sigmajs.org/, 2022. [Accessed 18-09-2023].

[26] Lucidchart, "UML Use Case Diagram Tutorial — lucidchart.com." https://www.lucidchart.com/pages/uml-use-case-diagram, 2023. [Accessed 28-08-2023].

[27] MADEJA, "Guìa para la redacciòn de casos de uso — Marco de Desarrollo de la Junta de Andalucìa — juntadeandalucia.es." https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416, 2013. [Accessed 28-08-2023].

[28] Mozilla, "MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web — MDN — developer.mozilla.org." https://developer.mozilla.org/es/docs/Glossary/MVC, 2022. [Accessed 27-08-2023].

[29] S. Bastian, "Learning Django: REST Framework and MVT Architecture — medium.com." https://medium.com/geekculture/learning-django-rest-framework-and-mvt-architecture-1ca173163f1c, 2021. [Accessed 27-08-2023].

[30] chauvelinmariedelcasse, "How a View-model works in Vue.js? - GeeksforGeeks — geeksforgeeks.org." https://www.geeksforgeeks.org/how-a-view-model-works-in-vue-js/, 2023. [Accessed 27-08-2023].

[31] IBM, "¿Qué es la arquitectura de tres niveles? — IBM — ibm.com." https://www.ibm.com/es-es/topics/three-tier-architecture, 2023. [Accessed 30-08-2023].

[32] Mozilla, "Visión General Cliente-Servidor - Aprende desarrollo web — MDN — developer.mozilla.org." https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview, 2023. [Accessed 30-08-2023].

[33] AppMaster, "¿Cómo crear puntos finales y por qué los necesita? — AppMaster — appmaster.io." https://appmaster.io/es/blog/como-crear-puntos-finales-y-por-que-los-necesita, 2022. [Accessed 30-08-2023].

[34] M. Coppola, "Qué es una API REST, para qué sirve y ejemplos — blog.hubspot.es." https://blog.hubspot.es/website/que-es-api-rest, 2022. [Accessed 30-08-2023].

[35] T. Christie, "Home - Django REST framework — django-rest-framework.org." https://www.django-rest-framework.org/, 2023. [Accessed 18-09-2023].

[36] Vue, "Composition API FAQ — Vue.js — vuejs.org." https://vuejs.org/guide/extras/composition-api-faq.html#what-is-composition-api, 2023. [Accessed 30-08-2023].

[37] Vue, "Defineprops and definaemits — Vue.js — vuejs.org." https://vuejs.org/api/sfc-script-setup.html#using-custom-directives, 2023. [Accessed 30-08-2023].

[38] Tannazma, "Pinia — The intuitive store for Vue.js — pinia.vuejs.org." https://pinia.vuejs.org/, 2023. [Accessed 18-09-2023].

[39] MongoDB, "MongoDB: La Plataforma De Datos Para Aplicaciones — mongodb.com." https://www.mongodb.com/es, 2023. [Accessed 18-09-2023].

[40] MongoDB, "Databases and Collections MongoDB Manual — mongodb.com." https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections, 2023. [Accessed 30-08-2023].

[41] MongoDB, "MongoDB Atlas — Plataforma De Datos Multicloud Para Desarrolladores — MongoDB — mongodb.com." https://www.mongodb.com/es/atlas, 2023. [Accessed 18-09-2023].

[42] P. G. D. Group, "PostgreSQL — postgresql.org." https://www.postgresql.org/, 2023. [Accessed 18-09-2023].

[43] Render, "Cloud Application Hosting for Developers — Render — render.com." https://render.com/, 2023. [Accessed 18-09-2023].

[44] Python, "Welcome to Python.org — python.org." https://www.python.org/, 2023. [Accessed 18-09-2023].

[45] Mozilla, "JavaScript — MDN — developer.mozilla.org." https://developer.mozilla.org/es/docs/Web/JavaScript, 2023. [Accessed 18-09-2023].

[46] Mozilla, "HTML: Lenguaje de etiquetas de hipertexto — MDN — developer.mozilla.org." https://developer.mozilla.org/es/docs/Web/HTML, 2023. [Accessed 18-09-2023].

[47] Mozilla, "CSS — MDN — developer.mozilla.org." https://developer.mozilla.org/es/docs/Web/CSS, 2023. [Accessed 18-09-2023].

[48] Wikipedia, "JSON - Wikipedia, la enciclopedia libre — es.wikipedia.org." https://es.wikipedia.org/wiki/JSON, 2023. [Accessed 18-09-2023].

[49] VueJS, "Vue.js - The Progressive JavaScript Framework — Vue.js — vuejs.org." https://vuejs.org/, 2023. [Accessed 18-09-2023].

[50] Django, "Django — djangoproject.com." https://www.djangoproject.com/, 2023. [Accessed 18-09-2023].

[51] NodeJS, "Node.js — nodejs.org." https://nodejs.org/es, 2023. [Accessed 18-09-2023].

[52] Jetbrains, "PyCharm: el IDE de Python para desarrolladores profesionales, por JetBrains — jetbrains.com." https://www.jetbrains.com/es-es/pycharm, 2023. [Accessed 18-09-2023].

[53] Jetbrains, "IntelliJ IDEA: el IDE líder para Java y Kotlin — jetbrains.com." https://www.jetbrains.com/es-es/idea/, 2023. [Accessed 18-09-2023].

[54] Microsoft, "Visual Studio Code - Code Editing. Redefined — code.visualstudio.com." https://code.visualstudio.com/, 2023. [Accessed 18-09-2023].

[55] Postman, "Postman API Platform — postman.com." https://www.postman.com/, 2023. [Accessed 18-09-2023].

[56] MongoDB, "MongoDB Compass — mongodb.com." https://www.mongodb.com/es/products/compass, 2023. [Accessed 18-09-2023].

[57] DBeaver, "DBeaver Community — Free Universal Database Tool — dbeaver.io." https://dbeaver.io/, 2023. [Accessed 18-09-2023].

[58] Lucidchart, "Software de diagramación en línea y solución visual — Lucidchart — lucidchart.com." https://www.lucidchart.com/pages/es, 2023. [Accessed 18-09-2023].

[59] Astah, "Powerful and Fast UML Diagramming Software - Astah — astah.net." https://astah.net/products/astah-uml/, 2023. [Accessed 18-09-2023].

[60] GitHub, "GitHub Desktop — desktop.github.com." https://desktop.github.com/, 2023. [Accessed 18-09-2023].

[61] Overleaf, "Overleaf, Online LaTeX Editor — overleaf.com." https://www.overleaf.com/, 2023. [Accessed 18-09-2023].

[62] R. Brewster, "Paint.NET - Free Software for Digital Photo Editing — getpaint.net." https://www.getpaint.net/, 2023. [Accessed 18-09-2023].

[63] Microsoft, "Microsoft Word: software de procesamiento de textos — Microsoft 365 — microsoft.com." https://www.microsoft.com/es-es/microsoft-365/word, 2023. [Accessed 18-09-2023].

[64] Microsoft, "Software de hojas de cálculo Microsoft Excel — Microsoft 365 — microsoft.com." https://www.microsoft.com/es-es/microsoft-365/excel, 2023. [Accessed 18-09-2023].

[65] ThunderClient, "Thunder Client - Rest API Client Extension for VS Code — thunderclient.com." https://www.thunderclient.com/, 2023. [Accessed 18-09-2023].

Glosario y acrónimos

API Application Programming Interface. 3, 5, 6, 12, 35, 41, 42, 45, 46, 48, 50–56, 63, 64, 70, 86–88, 92, 93, 106, 107, 109, 110, 127, 129, 130

Array Tipo de dato estructurado que representaría una matriz. 11, 35

Axios Es una librería de NodeJS que nos permite hacer peticiones HTTP a un servidor basada en promesas. 39, 46

Backend Área del desarrollo web encargada de la parte lógica y que no es visible por el usuario. 33–35, 53, 63, 64, 70, 87, 89, 91, 106, 127

BBDD Bases de datos. 3, 19, 33, 34, 36–41, 46, 48–50, 53, 55, 60, 63, 87–89, 91–93, 106, 108, 109, 111, 118, 134, 138, 142, 147, 152, 155, 159, 163, 164

Bootstrap Librería de código abierto multiplataforma con múltiples plantillas de botores formularios y más componentes para usarlos en páginas webs.. 48

CH Contraction Hierarchies. 9

CSS Cascading Style Sheets. 48, 52

Django Es un framework de código abierto basado en el lenguaje Python. 52, 56, 60, 134, 138, 142, 147, 152, 155, 159, 164

DOM Document Object Model. 42

DRF Django REST Framework. 3, 33–38, 41, 48–52, 54, 55, 61, 63, 87, 88, 91, 106, 109, 127–129, 135, 167

DSS Decision Support System. 1

Endpoint Punto final. 35, 38, 41, 55, 129

Firebase Plataforma de Google con varios servicios entre ellos el de autenticación segura.. 19

Framework Marco de trabajo que facilita la creación de proyectos. 8, 34, 42, 52

Frontend Área del desarrollo web que es visible por el usuario. 3, 8, 33–35, 39, 41–43, 46, 51–53, 60, 61, 63, 64, 70, 85–87, 89, 91, 92, 106, 110, 111

GUI Graphical User Interface. 20, 33

HTML HyperText Markup Language. 35, 42, 45, 52, 53, 61, 169

HTTP Hypertext Transfer Protocol. 6, 35, 41, 53, 61, 64, 86, 92, 110, 111

IDE Integrated Development Environment. 52, 53

Java Lenguaje de programación orientado a objetos multiplataforma. 51, 52

JavaScript Lenguaje de programación que permite realizar operaciones complejas en páginas web. 35, 41, 42, 46, 51, 52, 61, 127

JSON JavaScript Object Notation. 35, 41, 64, 66–69, 111

JWT JSON Web Token. 50, 89

Mapbox Empresa que proporciona herramientas de acceso a fuentes de datos de mapas digitalizados a através de su API. 11, 12, 44, 46, 85, 88

MLD Multi-Level Dijkstra. 9–11

MongoDB Base de datos NoSQL orientada a objetos. 3, 40, 41, 48–50, 53, 60, 63, 88, 91, 106, 108, 111

MVC Modelo Vista Controlador. 33, 36

MVT Model View Template. 33

MVVM Model View ViewModel. 34

Neo4j Base de datos NoSQL orientada a grafos. 19

NodeJS Entorno basado en JavaScript en el lado servidor multiplataforma. 35, 61, 106

NoSQL Base de datos no sólo SQL. 48

OSRM Open Source Routing Machine. 8, 9, 11

PostgreSQL Sistema de gestión de bases de datos relacional de código abierto.. 3, 41, 48, 50, 53, 89, 91, 93, 106, 109

Python Lenguaje de alto nivel interpretado y multiplataforma. 13, 35, 40, 51, 52, 106, 107, 127

React Framework basado en JavaScript de código abierto y con el apoyo de Facebook.. 33, 41

REST REpresentational State Transfer. 3, 35, 41, 42, 45, 48, 50–54, 56, 63, 64, 70, 87, 88, 92, 93, 106, 107, 109, 127, 129, 130

SQL Structured Query Language. 41, 48

Typescript Lenguaje de programación subconjunto de Javascript mantenido por Microsoft. 52

UML Unified Modeling Language. 17, 20, 53

URL Uniform Resource Locators. 6, 35, 38, 39, 41, 42, 45, 55, 61, 254

Visual Studio Code Editor de código fuente desarrollado por Microsoft y multiplataforma que permite el uso de extensiones. 53, 64

Vue 3 Framework progresivo de una única página basado en el lenguaje JavaScript. 3, 33–35, 41, 42, 44, 48, 50, 52, 61, 63, 85, 88, 91, 92, 106, 167, 170

Anexo A: Manual de despliegue

Introducción

En este anexo se va describir la instalación de todos los elementos necesarios para que la aplicación se pueda probar y ejecutar en un entorno bajo el sistema operativo Windows 10. La aplicación puede ser instalada igualmente en otros sistemas operativos si se instalan las dependencias para cada uno de ellos.

Entornos Necesarios

Esta aplicación precisa de varios entornos que son necesarios instalar antes de hacer cualquier acción. Para el caso de Windows 10, es recomendable aunque no imprescindible la instalación de dos aplicaciones de la propia Tienda de Microsoft, que además son gratuitas. La Figura A.1 nos muestra el nombre de las aplicaciones. En caso de no querer instalarlas se puede usar la propia consola de comandos de Windows CMD Símbolo de sistema.

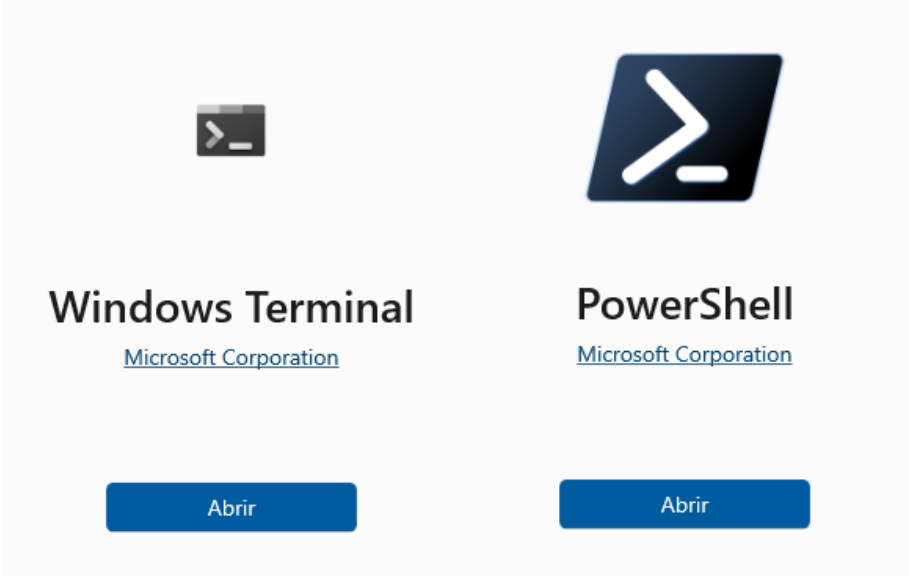


Figura A.1: Terminal en Windows.

A continuación hay que instalar los siguientes elementos:

- **Python** 3: Para que el Backend pueda funcionar, este lenguaje de programación se puede encontrar en este enlace.

- **NodeJS**: Para que el Frontend pueda funcionar, este entorno se encuentra disponible en este enlace.

- **Vue-CLI**: Instalador de Vue 3 disponible en enlace, si bien el código fuente ya incluye la carpeta node_modules que contiene todas las librerías necesarias y la instalación es más rápida.

- **MongoDB**: Si se quiere usar MongoDB en la propia máquina es necesario instalarse la versión Community Server. Otra opcion sería usar MongoDB Atlas que permite la BBDD online, enlace aquí.

- **PostgreSQL**: Se se quiere usar esta BBDD en la máquina local en este enlace está disponible la version 15 que es la usada en el proyecto. A la hora de instalar PostgreSQL hay que marcar en la instalación:

  - **PostgreSQL Server**, que nos permitirá tener el servidor de BBDD.
  - **pgAdmin4**, que nos permitirá manejar la BBDD mediante un entorno gráfico.
  - **StackBuilder**, complemento que contiene drivers y conectores para la BBDD.
  - **Command Line Tools**, nos permite interactuar con la BBDD desde consola

  Es más recomendable la instalación gratuita en un servicio como Render que permite crear una BBDD en pocos segundos lista para ser usada.

Todos los elementos se instalan siguiendo las instrucciones de sus propios fabricantes.

La instalación lleva un tiempo y es posible que haya que realizar varios reinicios del sistema si previamente no había estos elementos ya instalados.

En la carpeta del código fuente se encuentran dos carpetas diferenciadas:

- **pfg_drf_api**: En esta carpeta se encuentra el código fuente de la API REST en Django REST Framework (DRF).

- **pfg_frontend**: En esta carpeta se encuentra el código fuente del Frontend realizado en Vue 3.

Despliegue de la API REST

Si se han instalado las aplicaciones Terminal y Powershell, el siguiente paso es desde cada carpeta abrir una Terminal (con el botón derecho del ratón, la opción abrir en Terminal). Comenzamos con la carpeta **pfg_drf_api** y una vez dentro de la misma, abrimos terminal para que aparezca el resultado de la Figura A.2 si escribimos el comando **ls**

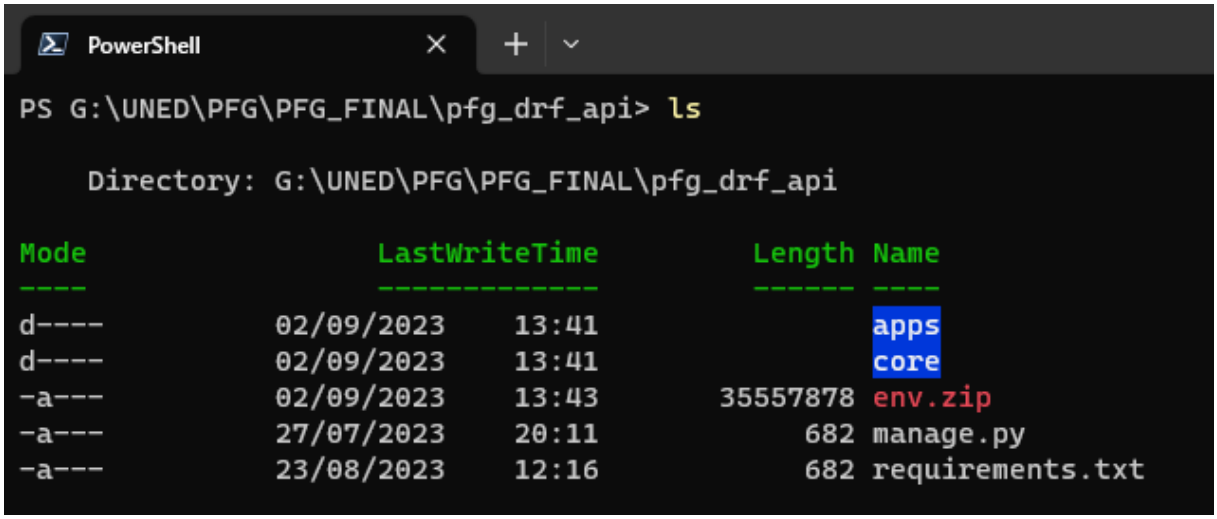


Figura A.2: Terminal en carpeta.

Para poner en funcionamiento la API REST necesitamos primero crear un entorno aislado para que las instalaciones de librerías de Python no sean globales al sistema.

Para ello en la Terminal hay que escribir el comando:

```
virtualenv env
```

Tras su ejecución se habrá creado una carpeta con nombre **env** en nuestra carpeta **pfg_drf_api**.

Descomprimimos el archivo *env.zip* que está en **pfg_drf_api** y sobreescribimos todos los ficheros que ya existen. Seguidamente se ejecuta el comando:

```
./env/scripts/activate
```

De esta forma estamos dentro de nuestro entorno y podemos continuar sin problema con la instalación siguiente mediante el comando:

```
pip install -r requirements.txt
```

Que debería indicar que todas las librerías necesarias para la API ya estaban instaladas, en caso de que alguna no lo estuviera el comando la instala. El resultado se puede observar en la Figura A.3.

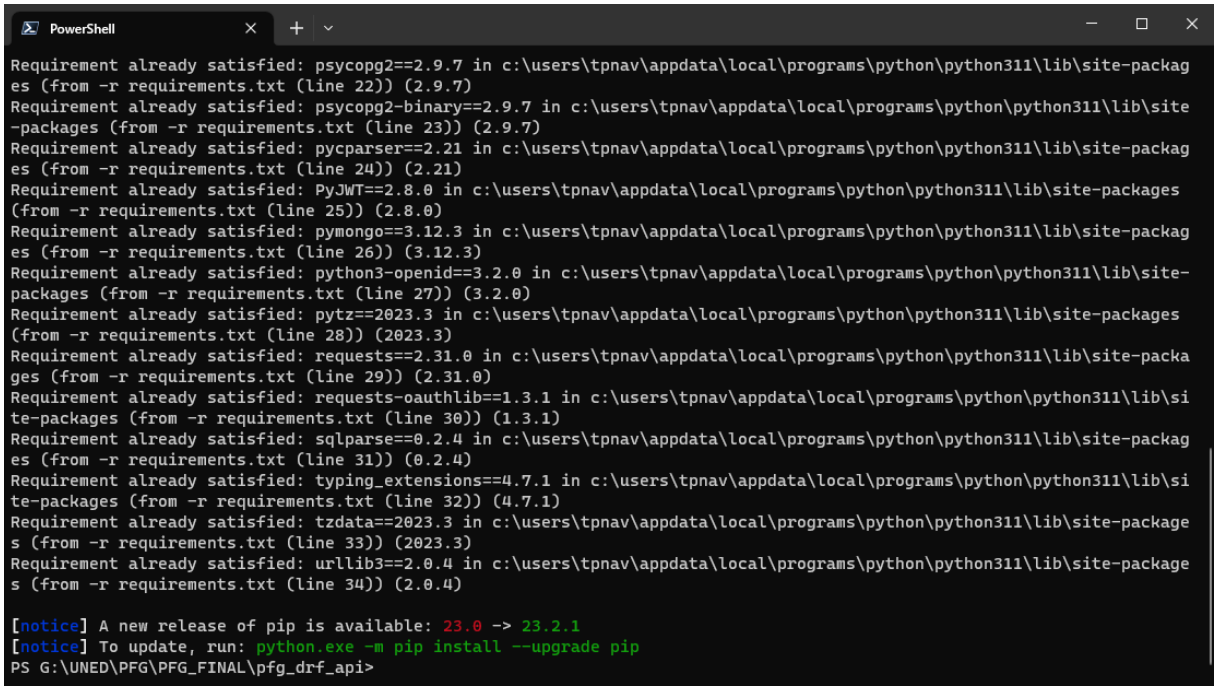


Figura A.3: Instalación librerías Python.

```
DATABASES = {
    'default':{},

    'data_db': {
        'ENGINE': 'djongo',
        'NAME': 'data_db',
        'CLIENT':{
            'host':'mongodb://localhost:27017/'
            #'host':'mongodb+srv://anavas99:anavas99@datadb.ivkrmvg.mongodb.net/?retryWrites=true&w=majo
        }
    },
    #'users_auth':{
    #    'ENGINE': 'django.db.backends.postgresql_psycopg2',
    #    'NAME': 'myproject',
    #    'USER': 'myuser',
    #    'PASSWORD': 'password',
    #    'HOST': 'localhost',
    #    'PORT': '',
    #}
    'users_auth':{
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR/'users_db.sqlite3',
    }
}

#DATABASES["users_auth"]=dj_database_url.parse("postgres://anavas99:ilQtfg8cHFdca6QSAip7Dx0hnJSuK5f7@dpg
```

Figura A.4: Configuración DATABASE en settings.py.

A continuación hay que cambiar los parámetros del fichero *settings.py* que está en la carpeta core según se muestra la Figura A.4.

Los datos que hay en DATABASES se corresponden a los tipos de BBDD que vamos a emplear, por defecto la BBDD de MongoDB está en la máquina local, si se se encuentra en MongoDB Atlas habría que comentar la línea 'host':'mongodb://localhost:27017/' y descomentar la línea comentada, escribiendo en ese caso la ruta que hayamos creado para

nuestra BBDD en MongoDB Atlas. La línea que aparece comentada es el enlace que se ha estado usando en pruebas durante el desarrollo y no estará disponible una vez se esté leyendo esta memoria.

Se incluye en el código una BBDD en SQLite3 lista con usuario administrador y algunos usuarios ya cargados para facilitar el despliegue, si se quiere mantener la configuración por defecto se puede saltar los cambios en el fichero que se describen a continuación, pasando directamente a makemigrations. Para la BBDD PostgreSQL habría que descomentar la entrada con nombre **users_auth** y establecer los datos que correspondan al haber creado la BBDD en su equipo.

Otra opción es usar por ejemplo la BBDD desde un servidor en Render y en nuestra BBDD creada allí copiar el enlace externo en la línea comentada DATABASES['users_auth']. Se ha dejado esta línea por haber realizado pruebas sobre esta BBDD en el desarrollo, pero esa dirección no estará disponible al leer este documento. Por defecto se ha dejado habilitada la BBDD en formato SQLite que es la que DRF define por defecto para tener un entorno de prueba sin problemas en cuanto a las conexiones con las BBDD. Es importante tener claro qué BBDD se van a usar y una vez decidido se continúa el despliegue.

Ahora vamos a preparar la estructura de modelos para migrar los datos a las BBDD. Para ello ejecutamos el comando:

```
python manage.py makemigrations
```

Seguidamente hacemos las migraciones a las BBDD de nuestros modelos con los comandos:

```
python manage.py migrate –database=users_auth
python manage.py migrate –database=data_db
```

Las BBDD ya están listas para recibir datos y ahora es preciso crear el usuario Administrador del sistema, para ello escribimos el comando siguiente y rellenamos los datos que correspondan.

```
python manage.py createsuperuser –database=users_auth
```

Si se está usando la BBDD por defecto de SQLite el usuario y password de accesso es: admin en ambos casos. Si hemos configurado nuestra BBDD en PostgreSQL tenemos que ejecutar el siguiente comando para crear el usuario administrador:

La API REST ya está lista y configurada para ponerla en funcionamiento, para ello ejecutamos el comando:

```
python manage.py runserver
```

La Figura A.5 nos muestra el resultado de dicho comando, indicando que todo está correcto.

```
PS G:\UNED\PFG\PFG_FINAL\pfg_drf_api> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 02, 2023 - 14:41:35
Django version 4.1.9, using settings 'core.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figura A.5: API REST en ejecución.

Se ha finalizado el despliegue de la API REST, está lista para recibir peticiones HTTP y es accesible desde el navegador mediante la dirección que aparece en la Figura A.5.

Una vez desplegada se puede volver a usar simplemente yendo a la carpeta y abriendo terminal con el botón derecho ejecutando el comando:

./env/scripts/activate

Y seguidamente:

python manage.py runserver

Con esos dos comandos se pone en marcha la API para ser usada en cualquier momento.

Despliegue del Frontend

El despliegue del Frontend es mucho más sencillo y rápido. Comenzamos abriendo la Terminal desde la carpeta mediante el botón derecho del ratón y darle a la orden Abrir en terminal. Al igual que con el despliegue anterior, hay que crear la variables de entorno para aislar la instalación con respecto al sistema global.

Ejecutamos el comando:

virtualenv env

y seguidamente el comando para activar las variables de entorno:

./env/scripts/activate

Pasamos ahora a ejecutar los siguientes comandos para iniciar el servidor:

cd frontend
npm run serve

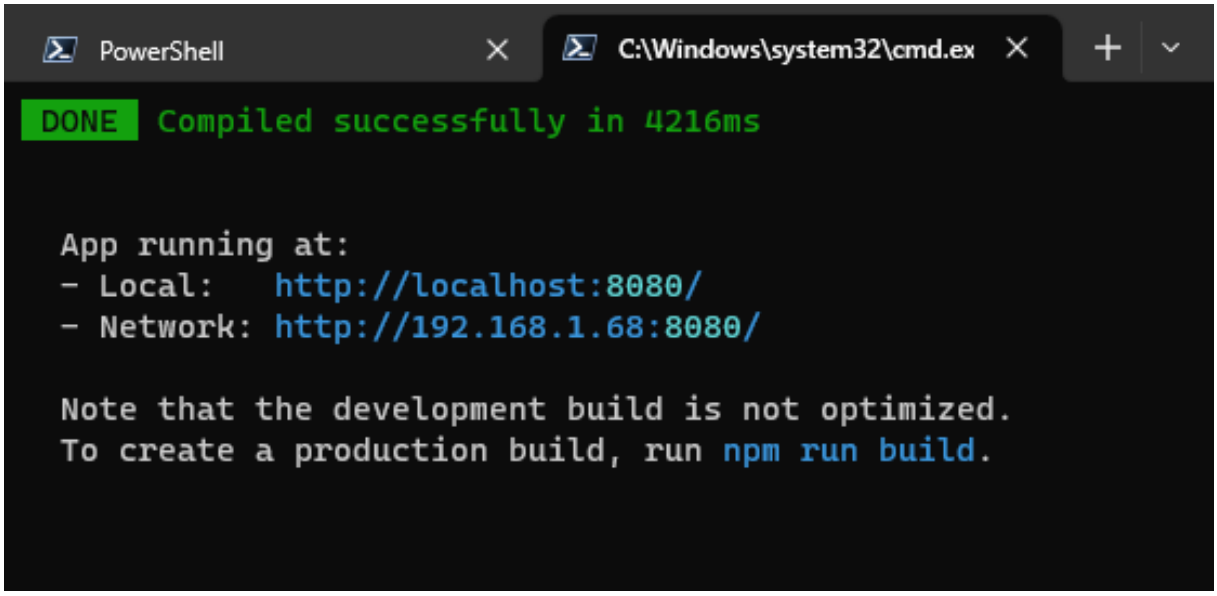


Figura A.6: Frontend en ejecución.

La Figura A.6 muestra el servidor ejecutándose y listo para ser accedido a través de cualquiera de esas dos direcciones HTTP. Cada vez que se vaya a usar el Frontend con activar las variables de entorno y ejecutar los comandos para iniciar el servidor, éste estará disponible.

Hay que indicar que si el servicio se detiene y no se ha reiniciado la máquina, al siguiente inicio del servidor el puerto de acceso será 8081, 8082,... por lo que se recomienda no apagar el servicio hasta que se vaya a apagar la máquina habrá que reiniciar la máquina para volver al puerto 8080.

Con estas últimas instrucciones el despliegue de la aplicación está terminado. La aplicación ya está lista y se pueden registrar usuarios y rutas.

Se han incluido unos archivos JSON para poder importarlos a la BBDD de MongoDB para tener ya configurados algunos conductores y pasajeros, así como sus rutas, grafos y valoraciones. No se incluyen los mensajes o reservas ya que con los archivos proporcionados está todo listo para tener un entorno estable y funcional.

Los datos de los usuarios creados en la BBDD son:

Administrador: usuario **admin** contraseña **admin**

Usuarios: test1, test2, ... hasta test13 todos con la contraseña 123456

Anexo B: Manual de usuario

Introducción

En este anexo vamos a ver la guía que permita el uso de la aplicación a nivel usuario.

Figura B.1: Home.

Home, registro y login

Al acceder por primera vez a la aplicación nos encontramos con la pantalla principal que contiene un pequeño carrusel de imágenes, el nombre de la aplicación y los iconos para registrarse o hacer login. Figura B.1.

Cuando se pasa el ratón sobre los iconos aparece un mensaje indicando la función que realiza. Figura B.2.

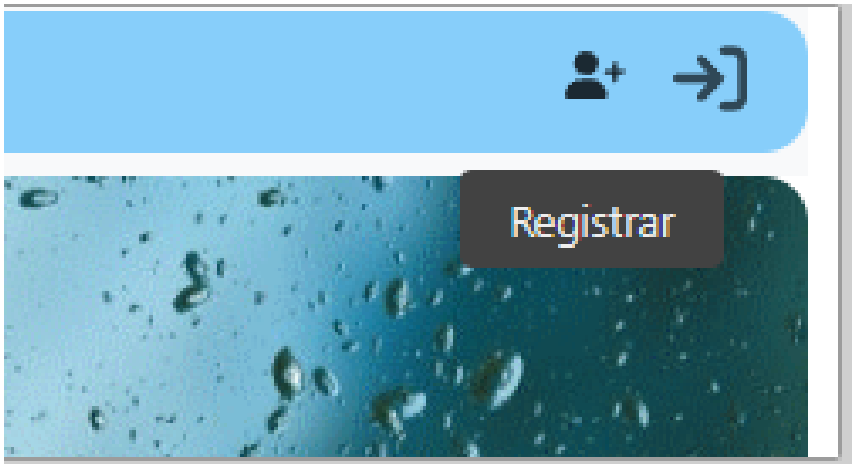


Figura B.2: Icono Registrar.

Al pulsar sobre Registrar nos lleva a la vista de registro con los formularios correspondientes. Figura B.3.

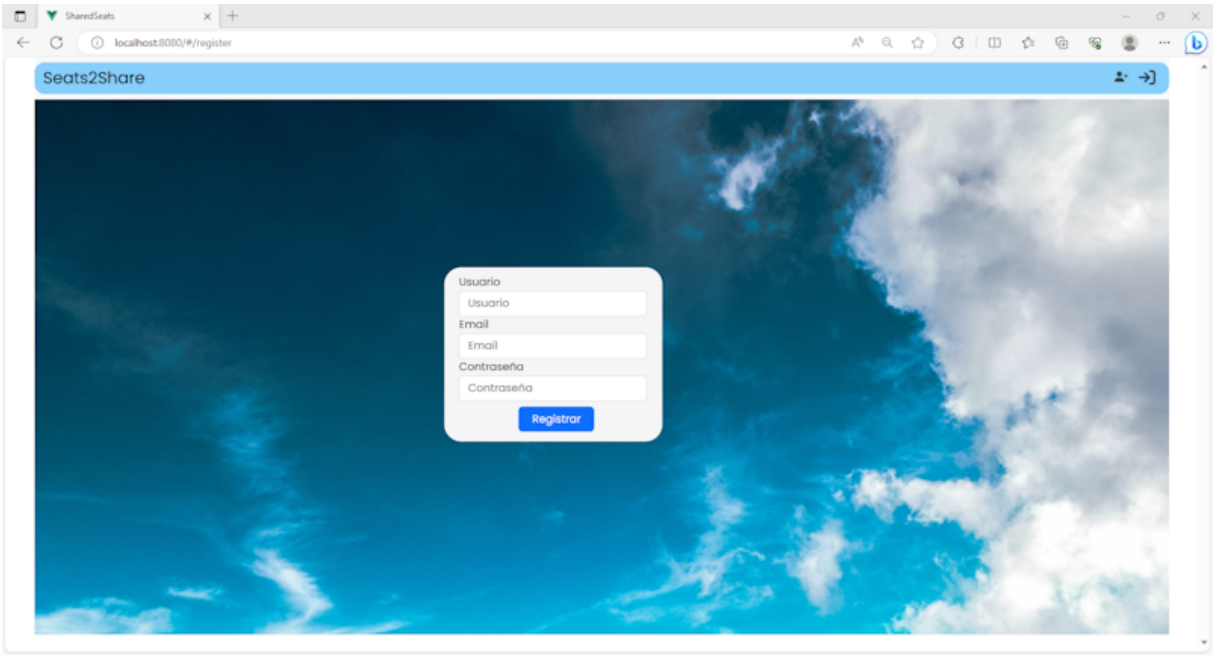


Figura B.3: Vista registro.

Al pulsar sobre Login nos lleva a la vista de login con los formularios correspondientes. Figura B.4.

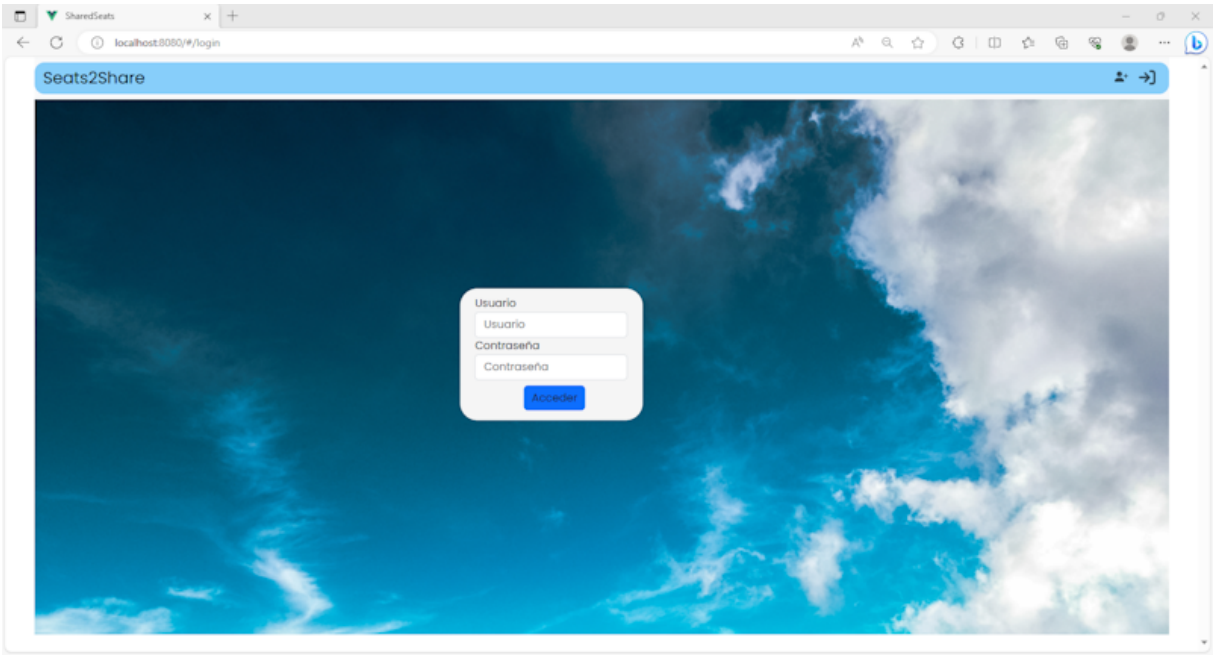


Figura B.4: Vista login.

Rol usuario

Si es la primera vez que accedemos la vista que nos aparece es la que vemos en la Figura B.5 para que indiquemos el rol de usuario que vamos a ejercer.

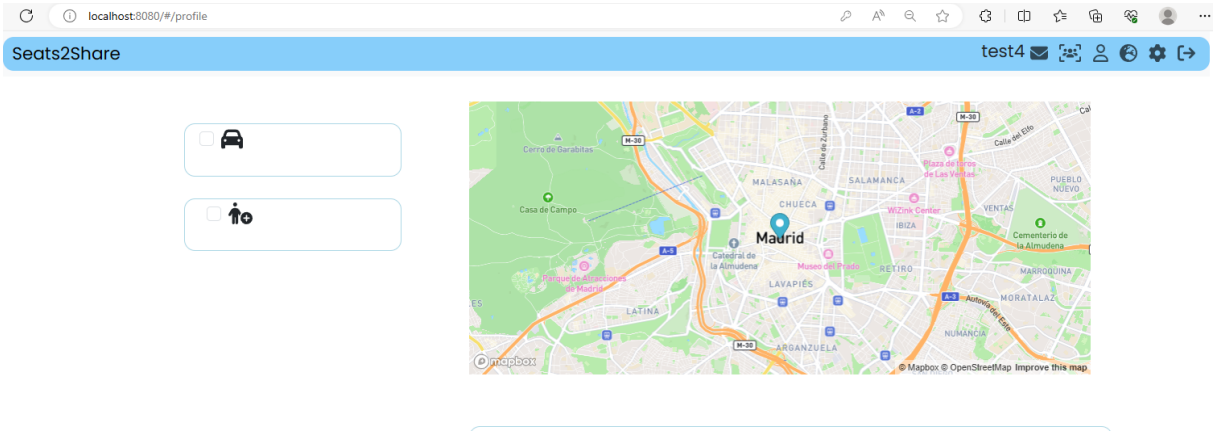


Figura B.5: Primer acceso.

En la barra de navegación superior derecha están los iconos principales como muestra la Figura B.6.

Figura B.6: Iconos principales.

Al pasar el ratón sobre ellos nos aparecen los nombres de las acciones, leyendo de izquierda a derecha:

- test4: El usuario activo y con acceso al sistema.

- Mensajes, nos llevará a la vista de los mensajes.

- Mis Viajes, nos lleva a la vista de panel de reservas.

- Perfil, nos lleva a la vista del perfil de usuario.

- Mapa, nos muestra en una vista un mapa con todos los usuarios que tienen ruta.

- Configuración, nos lleva a la vista configuración.

- Salir, nos permite salir de la aplicación.

En la parte izquierda de la pantalla tenemos los iconos que nos permiten definir qué tipo de usuario seremos, al pasar el ratón sobre ellos nos mostrarán la acción de cada uno de ellos. Figura B.7, en el caso de los pasajeros el mensaje es: Soy Pasajero.

Figura B.7: Elegir rol.

La Figura B.8 muestra el formulario para los conductores:

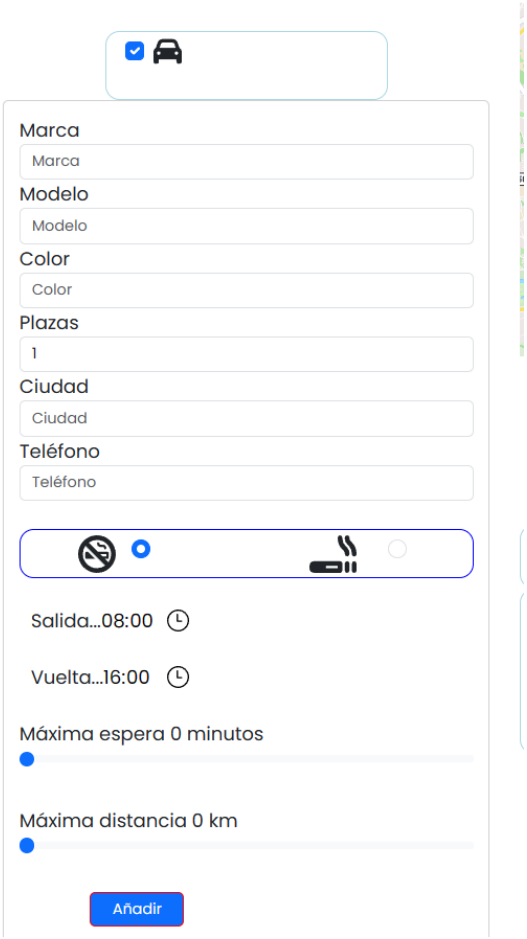


Figura B.8: Formulario conductor.

La Figura B.9 muestra el formulario para los pasajeros:

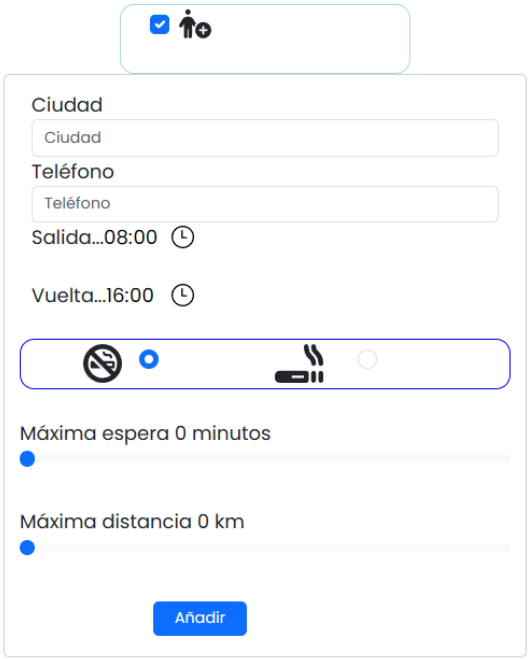


Figura B.9: Formulario pasajero.

En ambos formularios el botón **Añadir** registra al pasajero o conductor y habilita las opciones para el mapa. Figura B.10.

Crear ruta

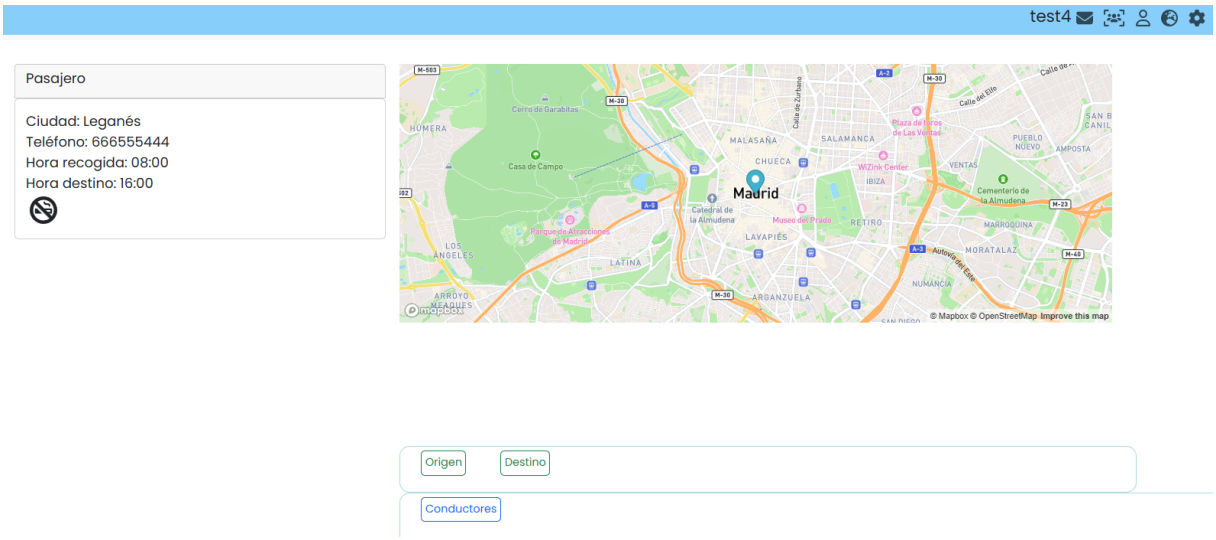


Figura B.10: Perfil listo crear ruta.

El usuario pueda ahora desplazarse por el mapa dejando el marcador azul en las zonas que vaya a definir como origen y destino. Los botones que aparecen cumplen las siguientes funciones:

- Origen: Una vez el usuario ha dejado el marcador azul en el lugar que desea como punto de origen, el usuario deberá pulsar ese botón.
- Destino: Una vez el usuario ha dejado el marcador azul en el lugar que desea como punto de destino, el usuario deberá pulsar ese botón. Pueden ocurrir dos casos:
  - El usuario es conductor: Al pulsar el botón destino se dibujará todo el recorrido de su trayecto en color azul, el destino en marcador rojo y el mapa quedará centrado en la vista entre los dos puntos.
  - El usuario es pasajero: Se mostrará únicamente los marcadores verde para el origen y rojo para el destino, sin recorrido y con el mapa centrado entre los dos puntos.
- Conductores: Al pulsar el botón se muestran los conductores que hay en la zona que cumplan los requisitos de distancias que hayan registrado en sus perfiles. Si no se ha guardado antes la ruta no mostrará ninguna información. Si no hay conductores cerca del área o que no cumplan los requisitos tampoco se mostrarán.

Para el caso de los conductores, el botón Conductores se llama Pasajeros y cumple las mismas funciones.

Cuando se ha pulsado el botón Destino, el botón Guardar está disponible y al pulsarlo se guardará la ruta en la BBDD. Figura B.11.

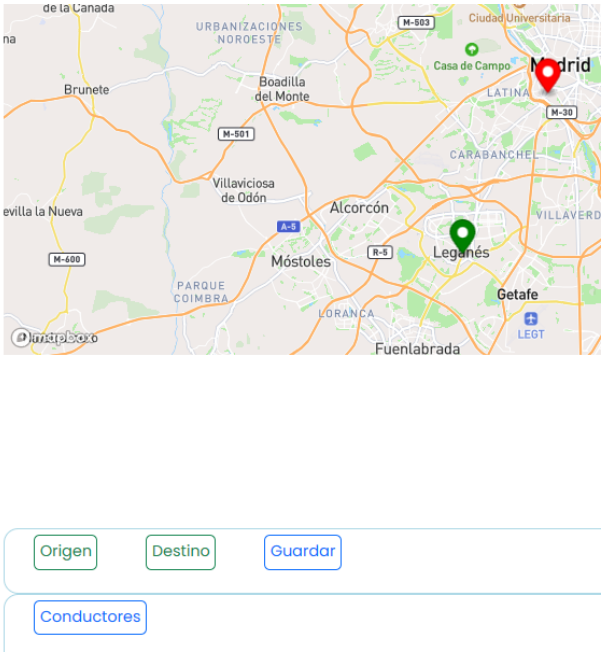


Figura B.11: Guardar Ruta.

Una vez añadida la ruta el único botón disponible es Conductores si el usuario es pasajero o Pasajeros si el usuario es conductor.

Ya podemos hacer uso del botón Conductores para ver aquellos que estén en nuestra zona, al pulsar el botón Conductores el mapa se centrará sobre nuestra posición de origen y podremos ver si hay conductores cercanos en nuestra zona. Figura B.12.

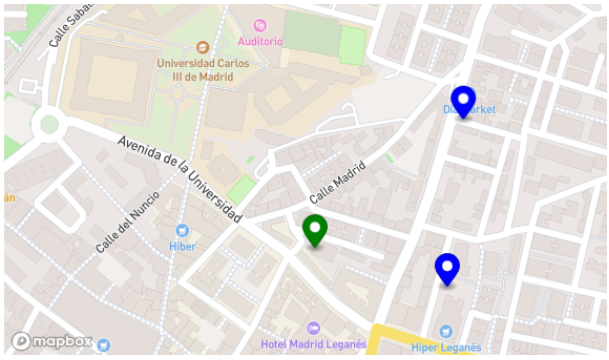
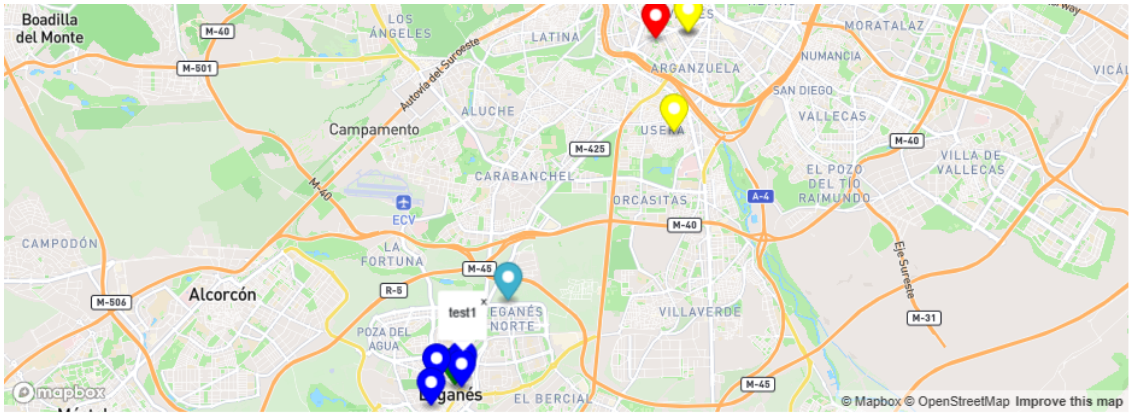


Figura B.12: Consultar Conductores.

El usuario activo es el marcador verde y los conductores son los marcadores en azul, al pulsar sobre los conductores nos saldrá una pequeña ventana *popup* indicando el nombre de usuario del conductor y se mostrará en pantalla la información de dicho conductor. Figura B.13.

Igualmente cada vez que se marca un conductor también se dibuja un marcador amarillo indicando el destino del conductor, para así poder compararlo con el nuestro, marcador rojo.



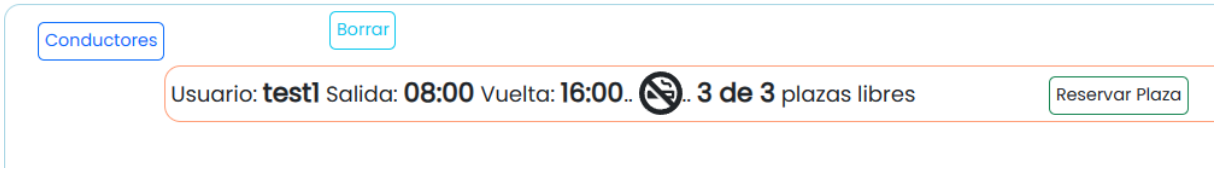


Figura B.13: Información Conductores.

Aparecen dos botones más, uno que indica Borrar, para borrar los resultados mostrados, ya sean conductores o pasajeros cercanos y el botón Reservar Plaza, que nos permitirá reservar plaza para un viaje con ese conductor.

Reservas

Una vez seleccionado un conductor y pulsado el botó Reservar Plaza, nos aparecerá el menú de reservas como muestra la Figura B.14.

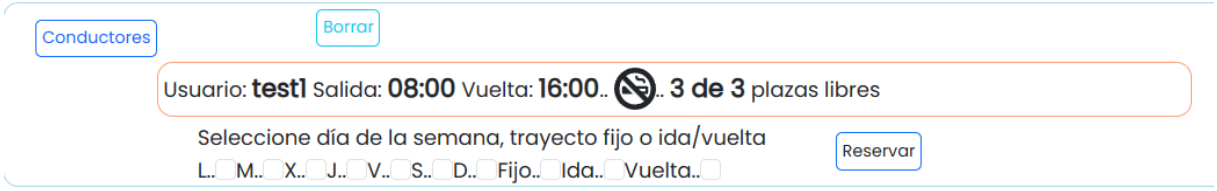


Figura B.14: Reservar Plaza.

El usuario pasajero puede seleccionar el día de la semana o si quiere que el trayecto sea de regreso en lugar de salida, o incluso que sea algo fijo, todos los lunes o todos los días.

Cuando esté conforme al pulsar el botón Reservar la reserva se habilita y el conductor recibirá un mensaje indicando que tiene una reserva. Dicha reserva estará disponible en el panel de reservas al que se accede pulsando el icono Mis Viajes.

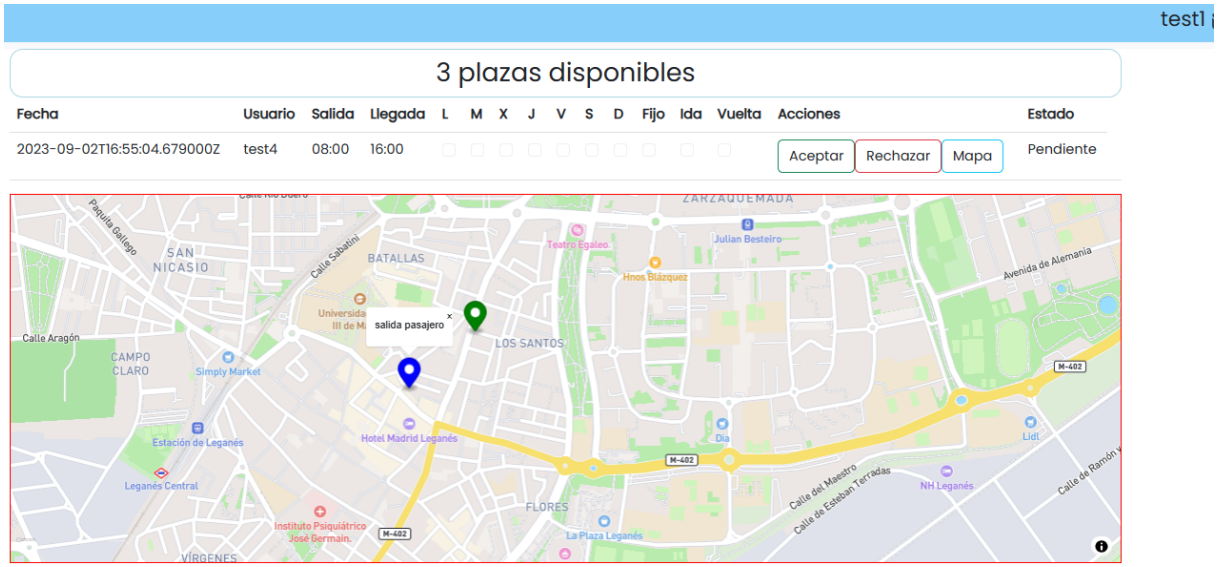


Figura B.15: Vista Mis Viajes.

La Figura B.15 muestra la vista Mis Viajes del conductor, los botones disponibles son:

- Aceptar: Al pulsar Aceptar la reserva queda en espera de subir al pasajero. Una reserva puede ser aceptada pero no ser activa hasta el momento de subir al pasajero, por lo que un conductor puede tener múltiples reservas aceptadas, pero activas tantas como plazas disponibles.

- Rechazar: Al pulsar Rechazar el conductor rechaza la reserva, se le abre un pequeño formulario para indicar el motivo del rechazo y si no lo rellena el pasajero recibirá un mensaje indicando que su reserva ha sido rechazada por el motivo que haya indicado el conductor o un mensaje genérico indicando que el conductor no ha explicado el motivo. Se habilita el botón Borrar para poder borrar la reserva.

- Mapa: Al pulsar Mapa, el conductor puede tener información visual de cuál es el punto de recogida del pasajero y así tener una capacidad de decisión sobre aceptar o no la reserva.

Se muestra información sobre el usuario que ha hecho la reserva, fecha, horario de salida, llegada y días seleccionados. Igualmente en la parte superior se muestran las plazas disponibles que tiene el conductor en ese momento.

Las reservas se quedan en estado pendiente hasta que el Conductor tome una decisión o el pasajero la anule al pulsar el botón Anular Reserva. Figura B.16.

Figura B.16: Anular Reserva.

Los cambios en el panel se ven reflejados a los diez segundos, cada vez que se va actualizando el panel, no son cambios instantáneos.

El resto de botones que aparecen en el panel de reservas se explican a continuación. Figura B.17.

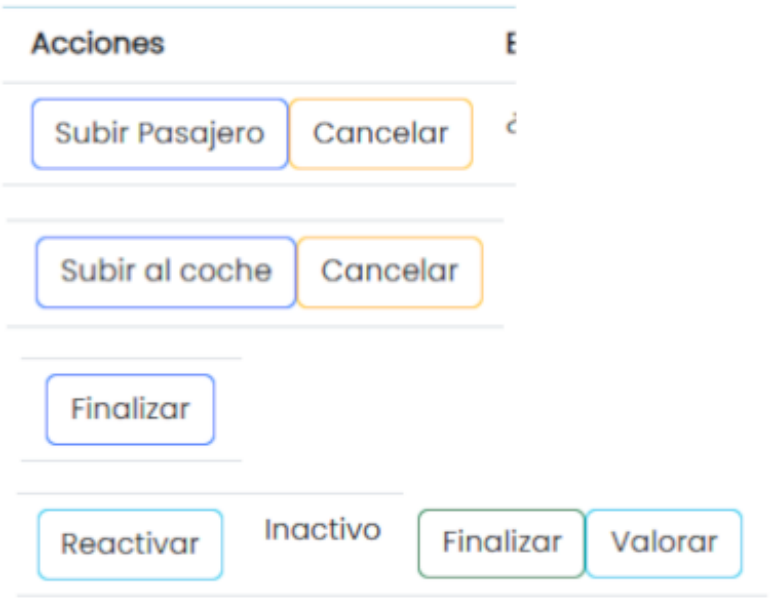


Figura B.17: Botones Reservas.

- Subir Pasajero: Este botón sólo lo pueden ver los conductores, al pulsarlo la reserva queda a la espera de confirmación por parte del pasajero para activar el viaje.
- Subir al coche: Este botón sólo lo pueden ver los pasajeros, al pulsarlo la reserva queda a la espera de confirmación por parte del conductor para activar el viaje.
- Cancelar: Cancela la reserva y la devuelve al estado pendiente de ser aceptada o rechazada.
- Finalizar (azul): Al pulsar el botón el viaje activo se finaliza.
- Reactivar: Al pulsar el botón la reserva se vuelve a quedar pendiente de subir pasajero o al coche.

- Finalizar (verde): El viaje ha finalizado y al pulsar el botón se archiva la reserva desapareciendo del panel.
- Valorar: Al pulsar sobre ese botón se abrirá un pequeño formulario para valorar el viaje y dejar un comentario.

Mensajes

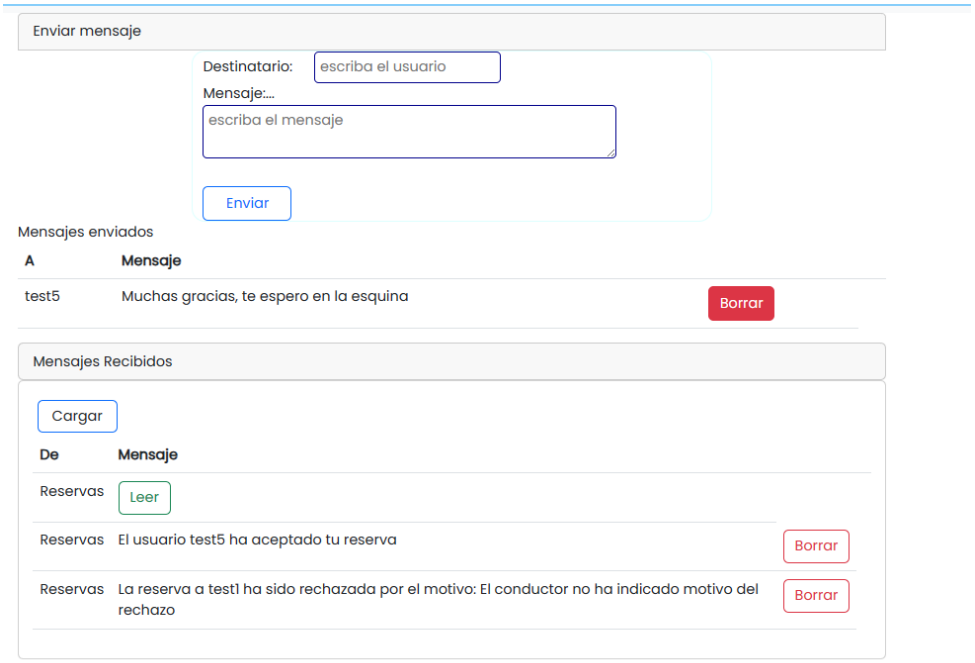


Figura B.18: Vista Mensajes.

Desde la barra de navegación superior se puede acceder a la vista de Mensajes, icono con el sobre. En esa vista podemos enviar y recibir mensajes de otros usuarios o del sistema de reservas. Figura B.18.

Para enviar un mensaje se pulsa sobre el texto: Enviar mensaje y aparecerá un pequeño formulario con dos cuadros de texto, uno para indicar el nombre del usuario a recibir el mensaje y otro cuadro de texto donde hay que escribir el mensaje.

Los botones disponibles son:

- Enviar: Una vez rellenado el formulario al pulsar sobre el botón se envía el mensaje.
- Borrar: Borra el mensaje, ya sea enviado o recibido.
- Cargar: Al pulsar el botón se comprueba si hay mensajes nuevos sin recibir, se comprueba automáticamente cada 30 segundos, pero se puede forzar la carga.
- Leer: Permite abrir un mensaje recibido y leer su contenido.

En la Figura B.19 se puede ver que los mensajes recibidos por un usuario se pueden responder al pulsar sobre el botón Responder, pero los mensajes recibidos por parte del sistema de Reservas no se pueden responder.

Figura B.19: Responder mensajes.

Los mensajes son un complemento para coordinar los viajes entre usuarios.

Valoración

Una vez los viajes han sido completados, los usuarios pueden valorar su experiencia en el viaje. Al pulsar el botón Valorar (azul) se abre el formulario de la Figura B.20. Se puede cerrar pulsando el botón Cerrar (no está en la imagen) o se puede enviar la valoración pulsando el botón Valorar (verde).

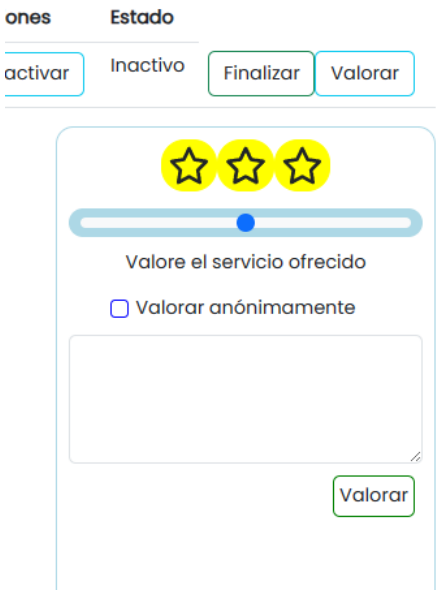


Figura B.20: Valorar viaje.

La valoración se muestra en el perfil de los usuarios. Figura B.21.

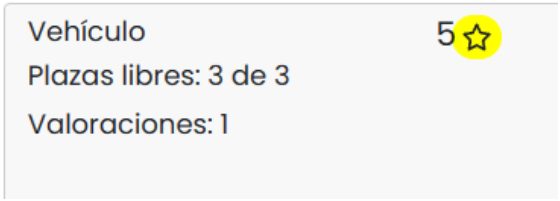


Figura B.21: Perfil con valoración.

No se pueden leer los comentarios realizados por los usuarios, esa función no ha podido ser añadida, pero se podrá realizar en una actualización posterior.

Vista Mapa

En la barra de navegación superior está el icono con forma de planeta, nos lleva a la Vista Mapa y en ella podemos ver en un mapa todos los usuarios con rutas activas. Figura B.22.

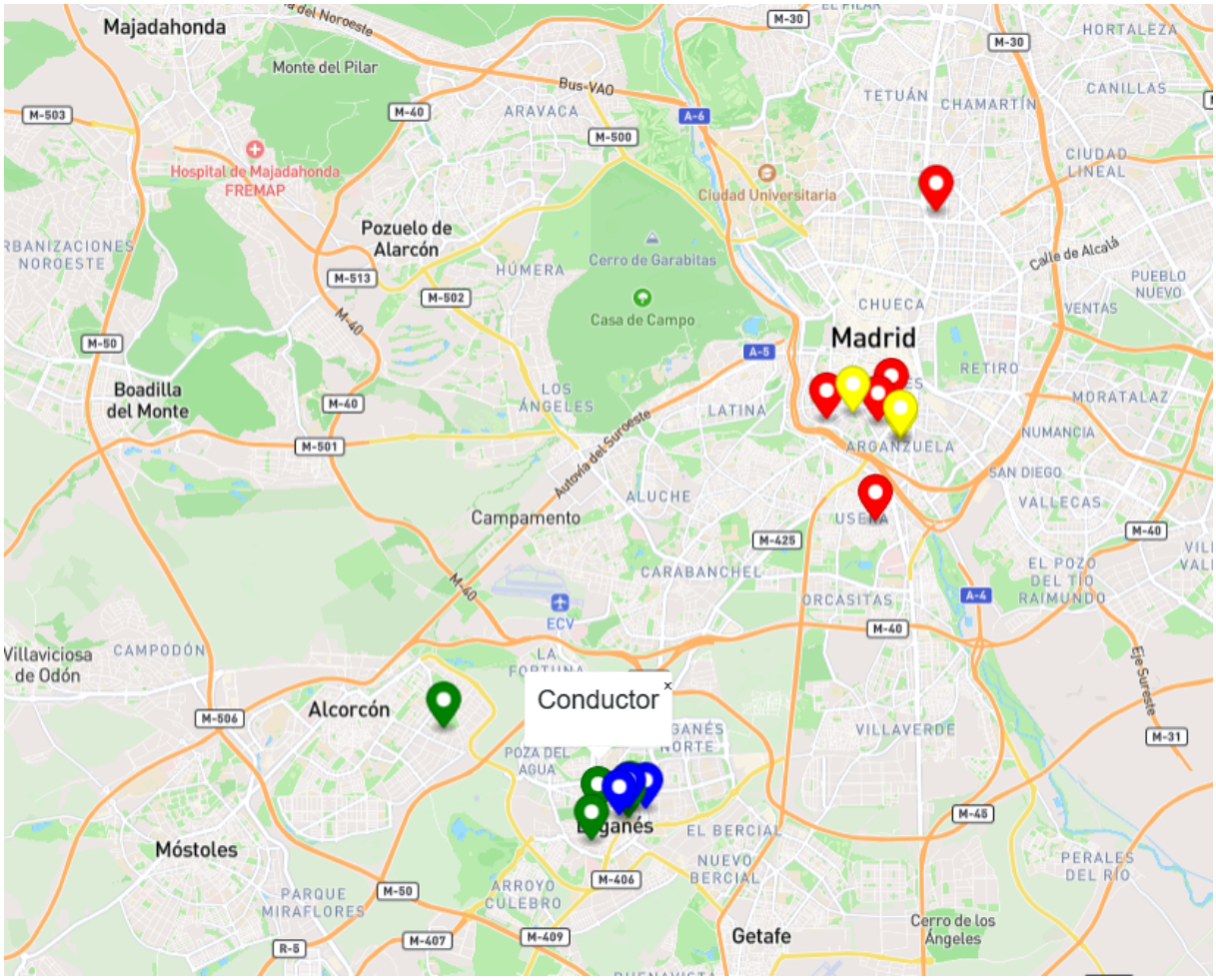


Figura B.22: Vista Mapa.

Al pulsar los marcadores nos sale una pequeña ventana o *popup* que nos indica si el usuario es conductor o pasajero. La leyenda de los marcadores por colores es:

- Marcador Verde: Puntos de salida de conductor.
- Marcador Azul: Puntos de salida de pasajeros.
- Marcador Rojo: Puntos de llegada de conductor.
- Marcador Amarillo: Puntos de llegada de pasajeros.

Esta vista puede ser útil para ver dónde hay conductores o pasajeros cercanos si no tenemos ninguno disponible en nuestra zona.

Vista Configuración

Al pulsar sobre el icono de configuración en el menú de navegación, nos vamos a la Vista Configuración, la cual nos permite cambiar los datos que hemos introducido al registrar el perfil.

La vista cambia en función de si el usuario es conductor Figura B.23 o pasajero B.24.

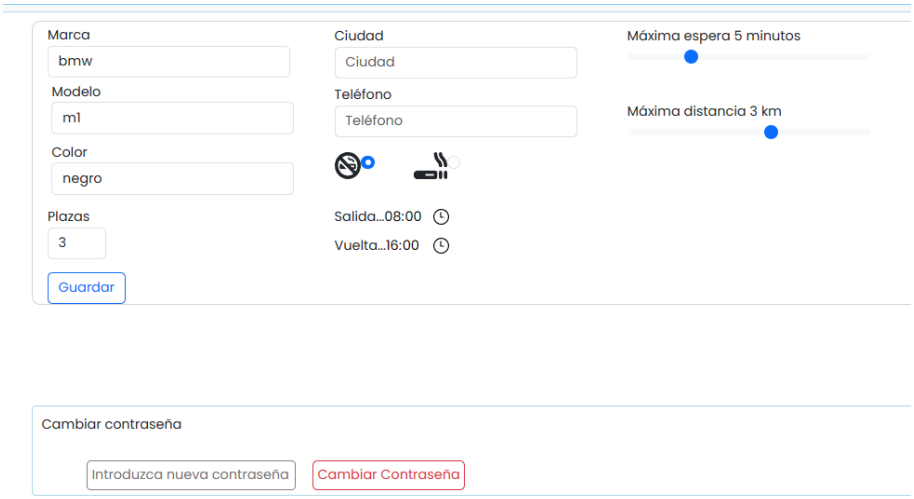


Figura B.23: Vista Configuración Conductor.

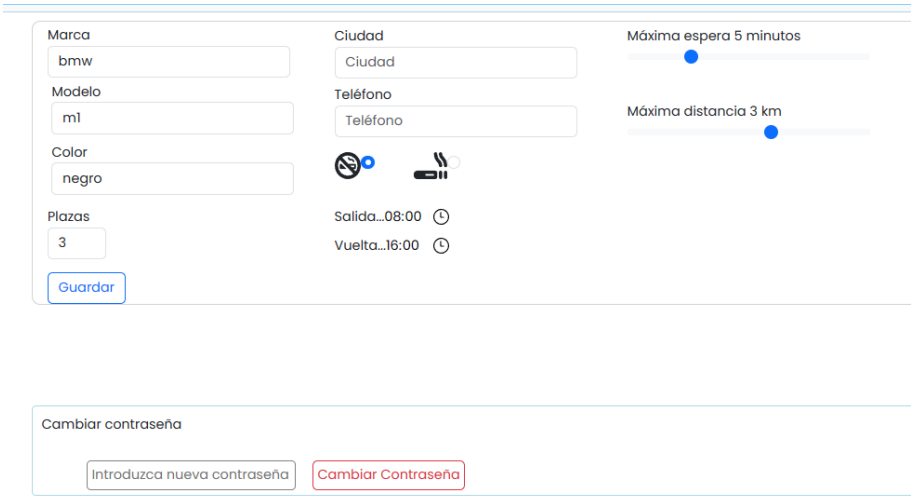


Figura B.24: Vista Configuración Pasajero.

Anexo C: Código Fuente

Introducción

En este anexo se mostrará todo el código fuente del proyecto. El código tanto en Python como en JavaScript está escrito en inglés para nombrar las variables, constantes, funciones o métodos. Los nombres de variables, constantes y funciones son autoexplicativos y se utiliza la notación de *camelCase* en la mayoría de los casos.

Se comenzará por el Backend para luego pasar al Backend, a su vez se indicará cuándo el código expuesto es autogenerado por la instalación del entorno o cuándo ha sido escrito por mí.

Backend: Django REST Framework

La instalación de DRF genera muchísimos ficheros que por defecto ya vienen completados por parte del entorno y que no hay que tocar apenas por parte del desarrollador. Se irán mostrando en las siguientes secciones las partes en las que se divide la API REST.

Directorio Raíz y carpeta Core

La Figura C.1 muestra el directorio raíz del proyecto.

Figura C.1 : Estructura raíz de ficheros Django REST Framework.

Se muestra a continuación el código fuente 7.3 del fichero *manage.py*. Dicho fichero no se puede modificar por parte del desarrollador y se muestra tal y como se genera al instalar Django REST Framework.

Fichero manage.py.

```
1 #!/usr/bin/env python
```

```python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          02/09/2023    20:10                __pycache__
d----          02/09/2023    13:41                routers
-a---          27/07/2023    20:11              0 __init__.py
-a---          03/09/2023     0:59             67 .env
-a---          02/09/2023    20:05            476 asgi.py
-a---          02/09/2023    20:07           5964 settings.py
-a---          02/09/2023    20:07            616 urls.py
-a---          02/09/2023    20:08            507 wsgi.py
```

Figura C.2 : Estructura de ficheros carpeta core.

En la Figura C.1 se muestra la estructura raíz de ficheros, en los que se observa la carpeta *env*. Dicha carpeta pertenece a las variables de entorno y no se mostrarán sus ficheros. Comencemos con los ficheros de la carpeta *core* que se muestran en la Figura C.2 .

Veamos el Código Fuente 7.3 del fichero *asgi.py*, el cual se genera por parte de DRF y no se ha modificado por parte del desarrollador.

Fichero asgi.py.

```python
# Esta clase pertenece a Django Rest Framework y no la hemos modificado

"""
ASGI config for core project.

It exposes the ASGI callable as a module-level variable named ``
    application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/
"""

import os
```

```
13
14 from django.core.asgi import get_asgi_application
15
16 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
17
18 application = get_asgi_application()
```

El siguiente fichero tampoco se ha modificado para el desarrollo del proyecto y forma parte de DRF. Es el fichero *wsgi.py*. Si se va a desplegar la API REST en un servicio externo sí que hay que modificar líneas de código, pero depende del servicio al que se envíe. Aquí se muestra para el funcionamiento básico en un entorno de instalación en Windows 10.

Fichero wsgi.py.

```
1  """
2  WSGI config for core project.
3  It exposes the WSGI callable as a module-level variable named ``
       application``.
4  For more information on this file, see
5  https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/
6  """
7  #Este fichero viene por defecto con DJango REST Framework
8  #y no ha sido modificado para el proyecto
9
10 import os
11
12 from django.core.wsgi import get\_wsgi\_application
13 os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'core.settings')
14 application = get\_wsgi\_application()
```

El siguiente fichero, *urls.py*, sí que contiene código escrito por mí. Este fichero contiene las rutas Endpoint de las aplicaciones.

Fichero urls.py.

```
1  from django.contrib import admin
2  from django.urls import path, include
3  # Esta clase contiene los Endpoints principales de la API REST
4  urlpatterns = [
5      path('admin/', admin.site.urls), #panel administrador
6      path('drivers/',include('apps.driversdb.urls')), #ruta para
       conductores
7      path('pax/',include('apps.passengerss.urls')), #ruta para pasajeros
8      path('routes/',include('apps.routes.urls')), #ruta para routes,
       rutas
9      path('auth/',include('apps.authsusers.urls')), #ruta para acceso y
       registro usuarios
10     path('messages/', include('apps.messagesdb.urls')), #ruta para
       mensajes
11     path('trips/', include('apps.trips.urls')), #ruta para las reservas
12     path('rates/', include('apps.rates.urls')), #ruta para las
       valoraciones
13     path('nodes/', include('apps.nodes.urls')) #ruta para los nodos
14 ]
```

Continuamos con el fichero *settings.py* el cual es el más importante y que contiene la configuración de la API REST. Contiene código generado al instalar el entorno (comentarios en inglés) que no ha sido modificado por mí y código escrito por mí con comentarios en español.

Fichero settings.py.

```
# Este archivo tiene todas las configuraciones
# de Django REST Framework
# Algunas vienen por defecto y no se ha modificado
# Otras han sido introducidas para el proyecto

"""
Django settings for core project.

Generated by 'django-admin startproject' using Django 4.2.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

import os
import dj_database_url
from pathlib import Path
from datetime import timedelta


# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-f&&ncibcn7kttag=8d^$b9s19zmh%*$)mx*yne2o*
    ktjkr88$i'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = [] # hosts permitidos, en despligue hay que poner cuales
     pueden acceder

# listado de hosts permitidos para el funcionamiento de la aplicacion
CORS_ORIGIN_WHITELIST = (
    'http://localhost:3000', #Django REST Framework
    'http://127.0.0.1:3000',
    'http://localhost:8080', #Frontend Vue 3
    'http://127.0.0.1:8080',
)


# Application definition
```

```
# Aplicaciones instaladas
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django_filters',
    'corsheaders',
    'rest_framework',
    'rest_framework.authtoken',
    'apps.driversdb',
    'apps.passengerss',
    'apps.routes',
    'apps.authsusers',
    'apps.messagesdb',
    'apps.trips',
    'apps.rates',
    'apps.nodes'

]

# Aplicaciones que permiten la comunicacion con programas externos
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

# Ubicacion fichero urls para las rutas
ROOT_URLCONF = 'core.urls'

# Configuracion de templates, no se ha usado en el proyecto, forma parte
    de Django
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

# Ubicacion fichero wsgi, no se ha usado en el proyecto, forma parte de
   Django
```

```
105 WSGI_APPLICATION = 'core.wsgi.application'
106
107
108 # Database
109 # https://docs.djangoproject.com/en/4.2/ref/settings/#databases
110
111 # Bases de datos
112 DATABASES = {
113     'default':{}, # Base de datos por defecto, vacia y sin usar
114
115     'data_db': { # Base de datos para las aplicaciones, menos usuarios,
    en MongoDB
116         'ENGINE': 'djongo',
117         'NAME': 'data_db',
118         'CLIENT':{
119             'host':'mongodb://localhost:27017/'
120             #'host':'mongodb+srv://anavas99:anavas99@datadb.ivkrmvg.
    mongodb.net/?retryWrites=true&w=majority'
121         }
122     },
123     #'users_auth':{ # Descomentar este codigo para usar una instalacion
    en local de postgreSQL
124     #     'ENGINE': 'django.db.backends.postgresql_psycopg2',
125     #     'NAME': 'myproject',
126     #     'USER': 'myuser',
127     #     'PASSWORD': 'password',
128     #     'HOST': 'localhost',
129     #     'PORT': '',
130     #}
131     'users_auth':{ # Base de datos por defecto para pruebas, con datos
    de usuarios
132         'ENGINE': 'django.db.backends.sqlite3',
133         'NAME': BASE_DIR/'users_db.sqlite3',
134     }
135 }
136
137 # Descomentar la linea siguiente para usar una base de datos de
    postgresql en un servicio externo.
138 #DATABASES["users_auth"]=dj_database_url.parse("postgres://anavas99:
    ilQtfg8cHFdca6QSAip7Dx0hnJSuK5f7@dpg-cjbug5bbq8nc7381vjf0-a.frankfurt
    -postgres.render.com/users_data_qahb")
139
140 # Ubicacion de ficheros de configuracion de bases de datos para las
    aplicaciones
141 DATABASE_ROUTERS = ['core.routers.db_routers.AuthRouter',
142                     'apps.driversdb.drivers_db_router.DriversRouter',
143                     'apps.passengerss.passengers_db_router.
    PassengersRouter',
144                     'apps.routes.routes_db_router.RoutesRouter',
145                     'apps.messagesdb.messages_db_router.MessagesRouter',
146                     'apps.trips.trips_db_router.TripsRouter',
147                     'apps.rates.rates_db_router.RatesRouter',
148                     'apps.nodes.nodes_db_router.NodesRouter',
149                     ]
150
151
152 # Configuracion para la autenticacion JWT. No implementada en el
    proyecto, pero lista para ser usada
```

```
153 SIMPLE_JWT = {
154     "ACCESS_TOKEN_LIFETIME": timedelta(minutes=60),
155     "REFRESH_TOKEN_LIFETIME": timedelta(days=7),
156     "ROTATE_REFRESH_TOKENS": False,
157     "BLACKLIST_AFTER_ROTATION": False,
158     "UPDATE_LAST_LOGIN": True,
159     "SIGNING_KEY": "complexsigningkey",  # generate a key and replace me
160     "ALGORITHM": "HS512",
161 }
162 
163 
164 # Password validation
165 # https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-
    validators
166 
167 # Configuracion de validacion de passwords, codigo generado por Django y
     no modificado
168 AUTH_PASSWORD_VALIDATORS = [
169     {
170         'NAME': 'django.contrib.auth.password_validation.
    UserAttributeSimilarityValidator',
171     },
172     {
173         'NAME': 'django.contrib.auth.password_validation.
    MinimumLengthValidator',
174     },
175     {
176         'NAME': 'django.contrib.auth.password_validation.
    CommonPasswordValidator',
177     },
178     {
179         'NAME': 'django.contrib.auth.password_validation.
    NumericPasswordValidator',
180     },
181 ]
182 
183 
184 # Internationalization
185 # https://docs.djangoproject.com/en/4.2/topics/i18n/
186 
187 # Configuracion idioma y horario
188 LANGUAGE_CODE = 'en-us'
189 
190 TIME_ZONE = 'UTC'
191 
192 USE_I18N = True
193 
194 USE_TZ = True
195 
196 
197 # Static files (CSS, JavaScript, Images)
198 # https://docs.djangoproject.com/en/4.2/howto/static-files/
199 
200 # configuracion archivos estaticos, solo valido cuando se realiza
    despliegue en servicio externo.
201 STATIC_URL = 'static/'
202 
203 # Configuracion de correo, no implementado pero listo para ser usado
```

```
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = os.environ.get('EMAIL_HOST_USER')
EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_HOST_PASSWORD')

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

# Codigo generado por Django no modificado
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

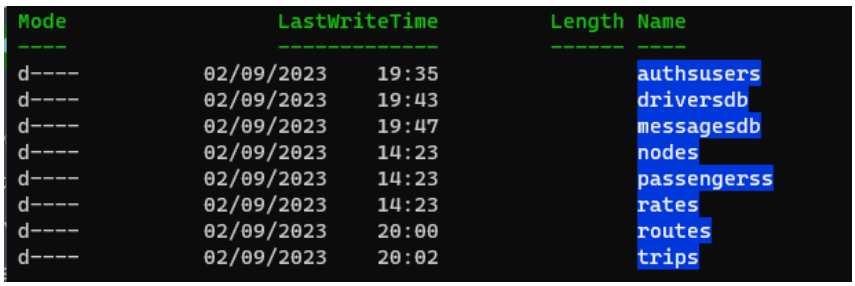
Continuamos con la carpeta *routers* que contiene el fichero *db_routers.py* cuyo contenido es el que Django sugiere para el uso correcto de múltiples BBDD. Este Código Fuente 7.3 se ha adaptado en el resto de las aplicaciones, cambiando el nombre de la clase y la BBDD a la que se conecta.

Fichero db_routers.py.

```
# Esta clase hace de router para base de datos, es requisito
# al usar multiples bases de datos. Permite la lectura, escritura,
    relacion y migracion

# Este es el codigo que Django Rest Framework proporciona para usar en
    multiples
# Bases de datos, solo se ha modificaco el nombre de la base de datos
class AuthRouter:


    route_app_labels = {"auth", "contenttypes", "sessions", "admin"}

    def db_for_read(self, model, **hints):

        if model._meta.app_label in self.route_app_labels:
            return "users_auth"
        return None

    def db_for_write(self, model, **hints):

        if model._meta.app_label in self.route_app_labels:
            return "users_auth"
        return None

    def allow_relation(self, obj1, obj2, **hints):

        if (
            obj1._meta.app_label in self.route_app_labels
            or obj2._meta.app_label in self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):

        if app_label in self.route_app_labels:
            return db == "users_auth"
        return None
```

Carpeta apps

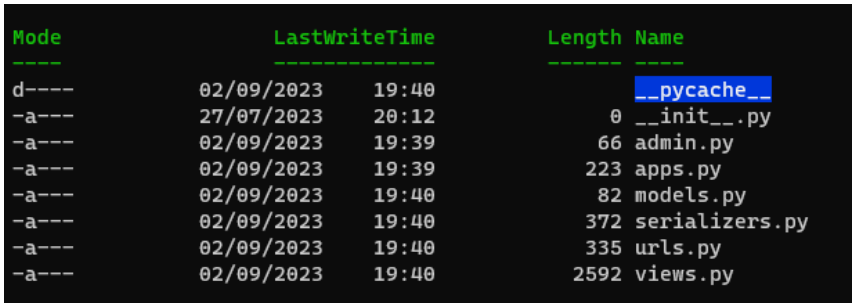
En la carpeta *apps* están las aplicaciones que hemos creado para el proyecto. Veremos el código fuente de cada una de ellas. La Figura C.3 muestra el contenido de la carpeta *apps*.



```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----           02/09/2023    19:35                authsusers
d----           02/09/2023    19:43                driversdb
d----           02/09/2023    19:47                messagesdb
d----           02/09/2023    14:23                nodes
d----           02/09/2023    14:23                passengerss
d----           02/09/2023    14:23                rates
d----           02/09/2023    20:00                routes
d----           02/09/2023    20:02                trips
```

Figura C.3 : Carpetas en apps.

Carpeta authsusers



```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----           02/09/2023    19:40                __pycache__
-a---           27/07/2023    20:12              0 __init__.py
-a---           02/09/2023    19:39             66 admin.py
-a---           02/09/2023    19:39            223 apps.py
-a---           02/09/2023    19:40             82 models.py
-a---           02/09/2023    19:40            372 serializers.py
-a---           02/09/2023    19:40            335 urls.py
-a---           02/09/2023    19:40           2592 views.py
```

Figura C.4 : Ficheros en carpeta authsusers.

En esta carpeta está la configuración de la aplicación *authsusers*, que utiliza como modelo el propio de Django, por lo que no se ha escrito apenas código en esta carpeta. Todas las aplicaciones necesitan tener unos ficheros para que DRF las pueda utilizar, pero en este caso algunos ficheros están sin código, son los siguientes:

- **admin.py**, este fichero no contiene código.
- **models.py** este fichero no contiene código.

Se muestra el Código Fuente del resto de ficheros.

Fichero authsusers apps.py.

```python
from django.apps import AppConfig

# Esta clase indica el nombre para que settings la reconozca
class AuthsusersConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.authsusers'
```

Fichero authsusers serializers.py.

```python
from rest_framework import serializers
from django.contrib.auth.models import User

```

```
# Esta clase utiliza el modelo User de Django para serializar directamente
# los usuarios al no utilizar un modelo especifico
class UserAuthSerializer(serializers.ModelSerializer):
    class Meta(object):
        model = User
        fields = ['id','username','password','email']
```

Fichero authsusers urls.py.

```
from django.urls import path
from .views import *

# Esta clase contiene las Endpoints URL

urlpatterns = [

    path('login/', login_auth_view, name='login_auth_view'), # ruta para el login
    path('register/', register_auth_view, name='register_auth_view'), # ruta para registrar usuarios
    path('token/',verify_token_view, name='verify_token_view'), # ruta para obtener el token

]
```

Fichero authsusers views.py.

```
# En esta clase se definen las vistas de acceso a la aplicacion

from rest_framework.decorators import api_view, authentication_classes, permission_classes
from rest_framework.authentication import SessionAuthentication, TokenAuthentication
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from .serializers import UserAuthSerializer
from rest_framework import status
from rest_framework.authtoken.models import Token
from django.contrib.auth.models import User

# Esta vista permite realizar el login y previene los accesos no permitidos
@api_view(['POST'])
def login_auth_view(request):
    if User.objects.using('users_auth').filter(username=request.data['username']).exists():
        user=User.objects.using('users_auth').get(username=request.data['username'])
        if(user.check_password(request.data['password'])):
            token=Token.objects.using('users_auth').get(user=user)
            serializer=UserAuthSerializer(user)
            return Response({"token":token.key,"success":serializer.data}, status=status.HTTP_200_OK)
        else:
            return Response({"pass":"contrasena incorrecta"}, status=status.HTTP_400_BAD_REQUEST)
    else:
```

```
24          return Response({"error":"usuario no existe"},status=status.
    HTTP_404_NOT_FOUND)
25
26
27  # Esta vista permite registrar al usuario y previene accesos no
    permitidos
28  @api_view(['POST'])
29  def register_auth_view(request):
30      serializer = UserAuthSerializer(data=request.data)
31      if serializer.is_valid():
32          if User.objects.using('users_auth').filter(email=request.data['
    email']).exists():
33              return Response({"errormail":"El correo ya existe"},status=
    status.HTTP_400_BAD_REQUEST)
34          else:
35              serializer.save()
36              user = User.objects.using('users_auth').get(username=request
    .data['username'])
37              user.set_password(request.data['password'])
38              user.save()
39              token = Token.objects.using('users_auth').create(user=user)
40              return Response({'token': token.key, 'user': serializer.data
    })
41      return Response({"erroruser":"el usuario ya existe"}, status=status.
    HTTP_400_BAD_REQUEST)
42  # Esta vista muestra el token asociado al usuario
43  @api_view(['GET'])
44  def verify_token_view(request):
45      print("veamos el token")
46      print(request.data['token'])
47   if(Token.objects.using('users_auth').filter(key=request.data['token']).
    exists()):
48          return Response("passed!")
49      else:
50          return Response("no pasa")
```

Carpeta driversdb

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         02/09/2023     20:54                __pycache__
-a---          27/07/2023     20:11              0 __init__.py
-a---          02/09/2023     19:39            171 admin.py
-a---          02/09/2023     19:39            219 apps.py
-a---          02/09/2023     19:41           1527 drivers_db_router.py
-a---          02/09/2023     19:42           1832 models.py
-a---          02/09/2023     19:44            410 serializers.py
-a---          02/09/2023     19:44            439 urls.py
-a---          02/09/2023     19:45           2087 views.py
```

Figura C.5 : Ficheros en carpeta driversdb.

Se muestra el Código Fuente de los ficheros de la Figura C.5 .

Fichero driversdb admin.py.

```
1  from django.contrib import admin
2  from .models import *
3
```

```
# Esta clase permite el acceso al administrador desde la web de
    administracion

admin.site.register(Driver)
```

Fichero driversdb apps.py.

```
from django.apps import AppConfig

# Esta clase indica el nombre para que settings la reconozca
class DriversConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.driversdb'
```

Se muestra el fichero *drivers_db_router.py* que sirve para añadir la aplicación a la BBDD. Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero driversdb drivers_db_router.py.

```
# Esta clase hace de router para base de datos, es requisito
# al usar multiples bases de datos. Permite la lectura, escritura,
    relacion y migracion

class DriversRouter:
    """
    A router to control all database operations on models in the
    auth and contenttypes applications.
    """

    route_app_labels = {"auth", "contenttypes"}

    def db_for_read(self, model, **hints):
        """
        Attempts to read auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def db_for_write(self, model, **hints):
        """
        Attempts to write auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def allow_relation(self, obj1, obj2, **hints):
        """
        Allow relations if a model in the auth or contenttypes apps is
        involved.
        """
        if (
            obj1._meta.app_label in self.route_app_labels
            or obj2._meta.app_label in self.route_app_labels
        ):
```

```
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        """
        Make sure the auth and contenttypes apps only appear in the
        'auth_db' database.
        """
        if app_label in self.route_app_labels:
            return db == "data_db"
        return None
```

Fichero driversdb models.py.

```
# Esta clase contiene el modelo de datos de la aplicacion, su equivalente
# a tablas en las bases de datos
# El nombre de los campos es autodecriptivo para facilitar su uso

from django.db import models
from django.utils import timezone

from django.contrib.auth.hashers import make_password
import uuid

# Modelo para el conductor
class Driver(models.Model):
    id = models.UUIDField(primary_key = True, # campo id
         default = uuid.uuid4,
         editable = False)
    driverCreated = models.DateTimeField(default=timezone.now) # campo creado en
    userIDowner = models.CharField(max_length=100, null=False, unique=False) # identificador unico de usuario
    driverStatus = models.BooleanField(default=True) # estado del conductor, True=Activo, False=Inactivo
    driverBrand = models.CharField(max_length=50, null=True) # marca del vehiculo
    driverModel = models.CharField(max_length=50, null=True) # modelo del vehiculo
    driverVColor = models.CharField(max_length=50, null=True) # color del vehiculo
    driverSeats = models.smallsizeIntegerField(null=True,default=3) # asientos disponibles
    driverDistance = models.smallIntegerField(null=True, default=4) # distancia en km asumible. No se usa
    driverStartTime = models.CharField(max_length=15, null=True) # hora de inicio
    driverEndTime = models.CharField(max_length=15, null=True) # hora de finalizacion
    driverWaitLimitStart = models.smallIntegerField(null=True, default=4) # tiempo en minutos de espera
    driverMaxDistanceAllowed = models.smallIntegerField(null=True, default=2000) # maxima distancia en metros
    driverConfigState = models.BooleanField(default=True) # marcador para ver si esta en configuracion o no. No implementado
    driverFreeSeats = models.smallIntegerField(null=True,default=3) # asientos libres en tiempo real
    driverSmoke=models.BooleanField(default=False) # valor True=Fumador,
```

```
    False=No fumador
    driverCity=models.CharField(max_length=100, null=True) # ciudad del conductor
    driverPhone=models.CharField(max_length=50, null=True) # telefono del conductor
    driverPremium=models.BooleanField(default=False) # valor True=Es Premium, False=No es premium. No implementado
    driverVerified=models.BooleanField(default=False) # valor True= Identidad verificada, False= no verificada. No implementado
    objects = models.Manager() # crea los objetos para el serializer

    # funcion que muestra los campos en el panel del administrador
    def __str__(self):
        return self.userIDowner+" "+self.driverBrand+" "+self.driverModel+" "+str(self.driverSeats)
```

Fichero driversdb serializers.py.

```
from rest_framework import serializers
from .models import *

# Esta clase se encarga de serializar el modelo de la aplicacion para
# su uso en la BBDD
class DriverSerializer(serializers.ModelSerializer):
    class Meta:
        model=Driver
        fields= '__all__'

    def create(self, validated_data):
        return Driver.objects.using("data_db").create(**validated_data)
```

Fichero driversdb urls.py.

```
from django.urls import path
from .views import *

# Esta clase contiene las Endpoints URL

urlpatterns = [

    path('drivers/', drivers_api_list_view, name='drivers_api_list'), # ruta que muestra los conductores
    path('adddriver/', add_driver_api_view, name='add_driver_api_view'),
    # ruta para incluir un conductor
    path('detail/',driver_api_detail_view, name='driver_api_detail_view'), # ruta que muestra el detail de un conductor
    path('update/',driver_api_update_view, name='driver_api_update_view') # ruta que permite modificar datos conductor

]
```

Fichero driversdb views.py.

```
# En esta clase se definen las vistas de acceso a la aplicacion
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
```

```
from .models import Driver
from ..passengerss.models import Passenger
from .serializers import DriverSerializer
from ..passengerss.serializers import PassengerSerializer
from django.db.models.query_utils import Q
# Esta vista muestra todos los conductores
@api_view(['GET'])
def drivers_api_list_view(request):
    drivers = Driver.objects.using("data_db").all()
    # if there is something in items else raise error
    serializer = DriverSerializer(drivers, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)
# Esta vista crea un conductor
@api_view(['POST'])
def add_driver_api_view(request):
    driver_serializer = DriverSerializer(data = request.data)
    if driver_serializer.is_valid():
        driver_serializer.create(driver_serializer.data)
        return Response("success", status=status.HTTP_200_OK)
    return Response(driver_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
# Esta vista busca un conductor y lo muestra
@api_view(['POST'])
def driver_api_detail_view(request):
    drivers = Driver.objects.using("data_db").filter(userIDowner=request
    .data['userIDowner']).first()
    # if there is something in items else raise error
    serializer = DriverSerializer(drivers)
    return Response(serializer.data)
# Esta vista actualiza los datos de un conductor
@api_view(['PUT'])
def driver_api_update_view(request):
    driver = Driver.objects.using("data_db").filter(userIDowner=request.
    data['userIDowner']).first()
    driver_serializer = DriverSerializer(driver, data=request.data)
    if(driver_serializer.is_valid()):
        driver_serializer.save()
        return Response("success", status=status.HTTP_200_OK)
    return Response(driver_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
```

Carpeta messagesdb

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          02/09/2023    20:54                __pycache__
-a---          13/08/2023    15:58              0 __init__.py
-a---          02/09/2023    19:46            170 admin.py
-a---          02/09/2023    19:46            221 apps.py
-a---          02/09/2023    19:46           1526 messages_db_router.py
-a---          02/09/2023    19:46           1549 models.py
-a---          02/09/2023    19:47            424 serializers.py
-a---          02/09/2023    19:47            595 urls.py
-a---          02/09/2023    19:49           2838 views.py
```

Figura C.6 : Ficheros en carpeta messagesdb.

Se muestra el Código Fuente de los ficheros de la Figura C.6 .

Fichero messagesdb admin.py.

```
from django.contrib import admin
from .models import *

# Esta clase permite el acceso al administrador desde la web de
    administracion
admin.site.register(Message)
```

Fichero messagesdb apps.py.

```
from django.apps import AppConfig

# Esta clase indica el nombre para que settings la reconozca
class MessagesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.messagesdb'
```

Se muestra el fichero *messages_db_router.py* que sirve para añadir la aplicación a la BBDD. Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero messagesdb messages_db_router.py.

```
# Esta clase hace de router para base de datos, es requisito
# al usar multiples bases de datos. Permite la lectura, escritura,
    relacion y migracion
class MessagesRouter:
    """
    A router to control all database operations on models in the
    auth and contenttypes applications.
    """

    route_app_labels = {"auth", "contenttypes"}

    def db_for_read(self, model, **hints):
        """
        Attempts to read auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def db_for_write(self, model, **hints):
        """
        Attempts to write auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def allow_relation(self, obj1, obj2, **hints):
        """
        Allow relations if a model in the auth or contenttypes apps is
        involved.
        """
        if (
```

```
            obj1._meta.app_label in self.route_app_labels
            or obj2._meta.app_label in self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        """
        Make sure the auth and contenttypes apps only appear in the
        'auth_db' database.
        """
        if app_label in self.route_app_labels:
            return db == "data_db"
        return None
```

Fichero messagesdb models.py.

```
from django.db import models
from django.utils import timezone

from django.contrib.auth.hashers import make_password
import uuid

# Esta clase contiene el modelo de datos de la aplicacion, su
    equivalente
# a tablas en las bases de datos
# El nombre de los campos es autodecriptivo para facilitar su uso
class Message(models.Model):  # modelo Mensaje
    id = models.UUIDField(primary_key = True, # campo id
         default = uuid.uuid4,
         editable = False)
    messageCreated = models.DateTimeField(default=timezone.now) # campo
    creado en
    messageToRead = models.DateTimeField(null=True) # campo destinatario
    fue leido. No implementado
    messageFromRead= models.DateField(null=True) # campo remitente fue
    leido. No implementado
    messageToArchived = models.DateField(null=True) # campo fecha
    archivo por remitente. No implementado
    messageFromArchived = models.DateField(null=True)  # campo fecha
    archivo por receptor. No implementado
    fromMessage = models.CharField(max_length=100, null=False, unique=
    False) # usuario remitente
    toMessage = models.CharField(max_length=100, null=False, unique=
    False) # usuario receptor
    bodyMessage = models.CharField(max_length=250, null=False, unique=
    False) # campo contenido mensaje
    messageRead = models.BooleanField(default=False) # mensaje leido.
    Value True=leido, False=No leido
    fromArchived = models.BooleanField(default=False) # mensaje
    archivado por remitente. value True=archivado, False=no archivado
    toArchived = models.BooleanField(default=False) # mensaje archivado
    por destinatario. value True=archivado, False=no archivado
    messageStatus = models.CharField(max_length=10, default="OK") #
    estado del mensaje para ser mostrado
    messageShowFrom = models.CharField(max_length=10, default="YES") #
    mostrar mensaje a remitente
    messageShowTo = models.CharField(max_length=10, default="YES") #
    mostrar mensaje a destinatario
```

```
28      messageAvoidReply =models.BooleanField(default=True) # permitir
        responder. value True=se permite, False=no se permite
29      objects = models.Manager() # permite crear los objetos en serializer
30
31      # muestra la informacion en el panel de administrador
32      def __str__(self):
33          return self.fromMessage+" "+self.toMessage+" "+self.bodyMessage
```

Fichero messagesdb serializers.py.

```
1  from rest_framework import serializers
2  from .models import *
3
4
5  # Esta clase se encarga de serializar el modelo de la aplicacion para
6  # su uso en la BBDD
7  class MessageSerializer(serializers.ModelSerializer):
8      class Meta:
9          model=Message
10         fields= '__all__'
11
12     def create(self, validated_data):
13
14         return Message.objects.using("data_db").create(**validated_data)
```

Fichero messagesdb urls.py.

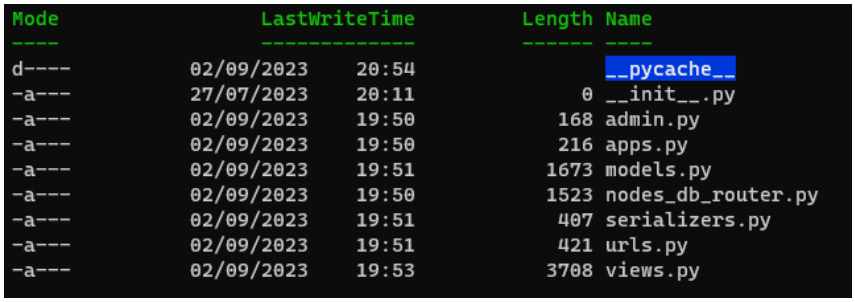
```
1  from django.urls import path
2  from .views import *
3  # Esta clase contiene las Endpoints URL
4
5  urlpatterns = [
6
7      path('messages/', messages_api_list_view, name='messages_api_list'),
       # ruta que muestra todos los mensajes
8      path('sendmsg/', sendmsg_api_view, name='sendmsg_api_view'),  # ruta
       que envia mensaje
9      path('receivemsg/',receivemsg_api_view, name='receivemsg_api_view'),
       # ruta que recibe mensajes
10     path('mymsg/',messages_mymsg_api_view, name='messages_mymsg_api_view
       '), # ruta mis mensajes recibidos
11     path('updatemsg/', messages_updatemsg_api_view, name='
       messages_updatemsg_api_view'), # ruta modificar mensaje
12     path('sentmsg/', sentmsg_api_view, name='sentmsg_api_view') # ruta
       mis mensajes enviados
13
14 ]
```

Fichero messagesdb views.py.

```
1  # En esta clase se definen las vistas de acceso a la aplicacion
2  from rest_framework.views import APIView
3  from rest_framework.response import Response
4  from rest_framework.decorators import api_view
5  from rest_framework import status
6  from rest_framework import permissions
7  from .models import Message
8  from .serializers import MessageSerializer
9  from django.db.models.query_utils import Q
```

```
10
11 #Esta vista muestra todos los mensajes en total
12 @api_view(['GET'])
13 def messages_api_list_view(request):
14     messages = Message.objects.using("data_db").all()
15     serializer = MessageSerializer(messages, many=True)
16     return Response(serializer.data, status=status.HTTP_200_OK)
17
18 #Esta vista envia un mensaje
19 @api_view(['POST'])
20 def sendmsg_api_view(request):
21     message_serializer = MessageSerializer(data = request.data)
22     if message_serializer.is_valid():
23         message_serializer.create(message_serializer.data)
24         return Response("success", status=status.HTTP_200_OK)
25     return Response(message_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
26
27 #Esta vista carga los mensajes recibidos de un usuario
28 @api_view(['POST'])
29 def receivemsg_api_view(request):
30     messages = Message.objects.using("data_db").filter(messageShowTo="
    YES",toMessage=request.data['toMessage']).all()
31     serializer = MessageSerializer(messages, many=True)
32     return Response(serializer.data)
33
34 #Esta vista muestra los mensajes enviados de un usuario
35 @api_view(['POST'])
36 def sentmsg_api_view(request):
37     messages = Message.objects.using("data_db").filter(messageShowFrom="
    YES",fromMessage=request.data['fromMessage']).all()
38     serializer = MessageSerializer(messages, many=True)
39     return Response(serializer.data)
40
41 # Esta vista muestra todos los mensajes enviados por un usuario
42 @api_view(['GET'])
43 def messages_mymsg_api_view(request):
44     messages = Message.objects.using("data_db").filter(messageShowFrom="
    YES",fromMessage=request.data['fromMessage']).all()
45     serializer = MessageSerializer(messages, many=True)
46     return Response(serializer.data)
47
48 # Esta vista modifica un mensaje
49 @api_view(['PUT'])
50 def messages_updatemsg_api_view(request):
51     message= Message.objects.using("data_db").filter(id=request.data['id
    ']).first()
52     message_serializer=MessageSerializer(message, data=request.data)
53
54     if(message_serializer.is_valid()):
55         message_serializer.save()
56         return Response("success", status=status.HTTP_200_OK)
57     return Response(message_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
```

Carpeta nodes



Mode	LastWriteTime	Length	Name
d----	02/09/2023 20:54		__pycache__
-a---	27/07/2023 20:11	0	__init__.py
-a---	02/09/2023 19:50	168	admin.py
-a---	02/09/2023 19:50	216	apps.py
-a---	02/09/2023 19:51	1673	models.py
-a---	02/09/2023 19:50	1523	nodes_db_router.py
-a---	02/09/2023 19:51	407	serializers.py
-a---	02/09/2023 19:51	421	urls.py
-a---	02/09/2023 19:53	3708	views.py

Figura C.7 : Ficheros en carpeta nodes.

Se muestra el Código Fuente de los ficheros de la Figura C.7 .

Fichero nodes admin.py.

```
from django.contrib import admin
from .models import *
# Esta clase permite el acceso al administrador desde la web de
    administracion

admin.site.register(Node);
```

Fichero nodes apps.py.

```
from django.apps import AppConfig

# Esta clase indica el nombre para que settings la reconozca

class RoutesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.nodes'
```

Fichero nodes models.py.

```
from django.db import models
from django.utils import timezone
import uuid


# Esta clase contiene el modelo de datos de la aplicacion, su
    equivalente
# a tablas en las bases de datos
# El nombre de los campos es autodecriptivo para facilitar su uso
class Node(models.Model):
    id = models.UUIDField(primary_key = True,
         default=uuid.uuid4,editable = False) #ID
    nodeCreated = models.DateTimeField(default=timezone.now) #fecha de
    creacion
    userIDowner = models.CharField(max_length=100, null=False, unique=
    False) #usuario propietario del nodo
    nodeUser=models.CharField(max_length=100, null=True, unique=False) #
    usuario del siguiente nodo
    nodeOriginLat=models.DecimalField(max_digits=30, decimal_places=25,
    null=True) #latitud nodo origen
    nodeOriginLng=models.DecimalField(max_digits=30, decimal_places=25,
    null=True) #longitud nodo origen
```

```
17      nodeDestLat=models.DecimalField(max_digits=30, decimal_places=25,
        null=True) #latitud nodo destino
18      nodeDestLng=models.DecimalField(max_digits=30, decimal_places=25,
        null=True) #longitud nodo destino
19      nodeDistance=models.DecimalField(max_digits=30, decimal_places=25,
        null=True) #distancia en metros entre nodos
20      nodeStatus=models.BooleanField(default=True) #estado del nodo: True
        en activo, False no activo
21      nodeDeleted=models.BooleanField(default=False) #marca si el nodo es
        borrado o no: True es borrado, False sin borrar
22      nodeType=models.CharField(max_length=10, default="Pasajero") #indica
        el tipo de nodo: Conductor o Pasajero
23      objects = models.Manager() #objeto para el serializer.py
24
25      def __str__(self):
26          return self.userIDowner
```

Se muestra el fichero *nodes_db_router.py* que sirve para añadir la aplicación a la BBDD. Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero nodes nodes_db_router.py.

```
1  # Esta clase hace de router para base de datos, es requisito
2  # al usar multiples bases de datos. Permite la lectura, escritura,
       relacion y migracion
3  class NodesRouter:
4      """
5      A router to control all database operations on models in the
6      auth and contenttypes applications.
7      """
8
9      route_app_labels = {"auth", "contenttypes"}
10
11     def db_for_read(self, model, **hints):
12         """
13         Attempts to read auth and contenttypes models go to auth_db.
14         """
15         if model._meta.app_label in self.route_app_labels:
16             return "data_db"
17         return None
18
19     def db_for_write(self, model, **hints):
20         """
21         Attempts to write auth and contenttypes models go to auth_db.
22         """
23         if model._meta.app_label in self.route_app_labels:
24             return "data_db"
25         return None
26
27     def allow_relation(self, obj1, obj2, **hints):
28         """
29         Allow relations if a model in the auth or contenttypes apps is
30         involved.
31         """
32         if (
33             obj1._meta.app_label in self.route_app_labels
```

```
            or obj2._meta.app_label in self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        """
        Make sure the auth and contenttypes apps only appear in the
        'auth_db' database.
        """
        if app_label in self.route_app_labels:
            return db == "data_db"
        return None
```

Fichero nodes serializers.py.

```
from rest_framework import serializers
from .models import *

# Esta clase se encarga de serializar el modelo de la aplicacion para
# su uso en la BBDD
class NodeSerializer(serializers.ModelSerializer):
    class Meta:
        model=Node
        fields= '__all__'

    def create(self, validated_data):

        return Node.objects.using("data_db").create(**validated_data)
```

Fichero nodes urls.py.

```
from django.urls import path
from .views import *

# Esta clase contiene las Endpoints URL
urlpatterns = [

    path('nodes/', nodes_api_list_view, name='nodes_api_list'), # ruta
    mostrar todos los nodos
    path('addnode/', add_node_api_view, name='add_node_api_view'), #
    ruta crear nodo
    path('node/',node_api_detail_view, name='node_api_detail_view'), #
    ruta mostrar detail nodo
    path('update/', update_node_api_view, name='update_node_api_view') #
    ruta modificar nodo

]
```

Fichero nodes views.py.

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
from .models import Node
from ..routes.models import Route
from ..driversdb.models import Driver
```

```
from ..passengerss.models import Passenger
import haversine as hs
from haversine import Unit
from .serializers import NodeSerializer
from django.db.models.query_utils import Q
from bson.decimal128 import Decimal128, create_decimal128_context
import decimal
# En esta clase se definen las vistas de acceso a la aplicacion

# Esta vista muestra todos los nodos del grafo
@api_view(['GET'])
def nodes_api_list_view(request):

    nodes = Node.objects.using("data_db").all()

    serializer = NodeSerializer(nodes, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)


# Esta vista crea el grafo de usuarios cercanos
@api_view(['POST'])
def add_node_api_view(request):

    if(request.data['nodeType']=="Pasajero"):

        drivers=Route.objects.using("data_db").filter(routeType="Conductor").all()
        print(drivers)
    else:
        drivers=Route.objects.using("data_db").filter(routeType="Pasajero").all()

    pax=Route.objects.using("data_db").filter(userIDowner=request.data['userIDowner']).first()
    for dis in drivers:
        if(request.data['nodeType']=="Pasajero"):
            dDistance = Driver.objects.using("data_db").filter(userIDowner=dis.userIDowner).first()
            maxDistance=dDistance.driverMaxDistanceAllowed*1000.00
        else:
            dDistance= Passenger.objects.using("data_db").filter(userIDowner=dis.userIDowner).first()
            maxDistance=dDistance.passengerMaxDistanceAllowed*1000.00

        paxPosition=(pax.routeOriginLat.to_decimal(),pax.routeOriginLng.to_decimal())
        driverPosition=(dis.routeOriginLat.to_decimal(),dis.routeOriginLng.to_decimal())
        realDistance=hs.haversine(paxPosition,driverPosition, unit=Unit.METERS)
        tempNode={
             "userIDowner":request.data['userIDowner'],
             "nodeUser":dis.userIDowner,
             "nodeOriginLat":dis.routeOriginLat.to_decimal(),
             "nodeOriginLng":dis.routeOriginLng.to_decimal(),
             "nodeDestLat":dis.routeDestLat.to_decimal(),
             "nodeDestLng":dis.routeDestLng.to_decimal(),
             "nodeDistance":realDistance,
```

```
            "nodeType":request.data['nodeType']
        }

        if(realDistance<=maxDistance):
            node_serializer = NodeSerializer(data = tempNode)
            if node_serializer.is_valid():
                node_serializer.create(node_serializer.data)

    return Response("success", status=status.HTTP_200_OK)
    return Response(node_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)


# Esta vista devuelve el grafo de usuarios cercanos
@api_view(['POST'])
def node_api_detail_view(request):
    nodes = Node.objects.using("data_db").filter(userIDowner=request.
    data['userIDowner']).all()

    serializer = NodeSerializer(nodes, many=True)
    return Response(serializer.data)


# Esta vista modifica el nodo
@api_view(['PUT'])
def update_node_api_view(request):
    node = Node.objects.using("data_db").filter(id=request.data['id']).
    first()
    node_serializer=NodeSerializer(node, data=request.data)
    if(node_serializer.is_valid()):
        node_serializer.save()
        return Response("success", status=status.HTTP_200_OK)
    return Response(node_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
```

Carpeta passengerss

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----         02/09/2023    20:54                __pycache__
-a---         27/07/2023    20:11              0 __init__.py
-a---         02/09/2023    19:53            172 admin.py
-a---         02/09/2023    19:54            226 apps.py
-a---         02/09/2023    19:55           1542 models.py
-a---         02/09/2023    19:54           1530 passengers_db_router.py
-a---         02/09/2023    19:55            426 serializers.py
-a---         02/09/2023    19:55            463 urls.py
-a---         02/09/2023    19:55           1795 views.py
```

Figura C.8 : Ficheros en carpeta passengerss.

Se muestra el Código Fuente de los ficheros de la Figura C.8 .

Fichero passengerss admin.py.

```
from django.contrib import admin
from .models import *
# Esta clase permite el acceso al administrador desde la web de
    administracion
```

```
4
5 admin.site.register(Passenger)
```

Fichero passengerss apps.py.

```
1 from django.apps import AppConfig
2
3 # Esta clase indica el nombre para que settings la reconozca
4
5 class PassengersConfig(AppConfig):
6     default_auto_field = 'django.db.models.BigAutoField'
7     name = 'apps.passengerss'
```

Fichero passengerss models.py.

```
1 from django.db import models
2 from django.utils import timezone
3
4 from django.contrib.auth.hashers import make_password
5 import uuid
6 # Esta clase contiene el modelo de datos de la aplicacion, su
    equivalente
7 # a tablas en las bases de datos
8 # El nombre de los campos es autodecriptivo para facilitar su uso
9 class Passenger(models.Model):
10     id = models.UUIDField(primary_key = True, # campo id
11          default = uuid.uuid4,
12          editable = False)
13     passengerCreated = models.DateTimeField(default=timezone.now) #
    campo creado en
14     userIDowner = models.CharField(max_length=100, null=False, unique=
    False) # identificador usuario
15     passengerStatus = models.BooleanField(default=True) # estado del
    pasajero, value True=Activo, False=inactivo
16     passengerOrigin = models.JSONField(null=True) # coordenadas origen
17     passengerDestination = models.JSONField(null=True) # coordenadas
    destino
18     passengerStartTime = models.CharField(max_length=15, null=True) #
    horario salida
19     passengerEndTime = models.CharField(max_length=15, null=True) #
    horario llegada
20     passengerWaitLimitStart = models.smallIntegerField(null=True,
    default=4) # tiempo espera en minutos
21     passengerMaxDistanceAllowed = models.smallIntegerField(null=True,
    default=2000) # distancia max en metros
22     passengerConfigState = models.BooleanField(default=True) # campo
    configuracion. No implementado
23     passengerSmoke=models.BooleanField(default=False) # pasajero fumador
    . value True=Fumador, False=No fumador
24     passengerCity=models.CharField(max_length=100, null=True) # ciudad
    del pasajero
25     passengerPhone=models.CharField(max_length=50, null=True) # telefono
     del pasajero
26     passengerPremium=models.BooleanField(default=False) # pasajero
    premium. No implementado
27     passengerVerified=models.BooleanField(default=False) # pasajero
    verificado. No implementado
28     objects = models.Manager()
29
```

```
30      def __str__(self):
31          return self.userIDowner
```

Se muestra el fichero *passengers_db_router.py* que sirve para añadir la aplicación a la BBDD. Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero passengerss passengers_db_router.py.

```
1  # Esta clase hace de router para base de datos, es requisito
2  # al usar multiples bases de datos. Permite la lectura, escritura,
       relacion y migracion
3
4  class PassengersRouter:
5      """
6      A router to control all database operations on models in the
7      auth and contenttypes applications.
8      """
9
10     route_app_labels = {"auth", "contenttypes"}
11
12     def db_for_read(self, model, **hints):
13         """
14         Attempts to read auth and contenttypes models go to auth_db.
15         """
16         if model._meta.app_label in self.route_app_labels:
17             return "data_db"
18         return None
19
20     def db_for_write(self, model, **hints):
21         """
22         Attempts to write auth and contenttypes models go to auth_db.
23         """
24         if model._meta.app_label in self.route_app_labels:
25             return "data_db"
26         return None
27
28     def allow_relation(self, obj1, obj2, **hints):
29         """
30         Allow relations if a model in the auth or contenttypes apps is
31         involved.
32         """
33         if (
34             obj1._meta.app_label in self.route_app_labels
35             or obj2._meta.app_label in self.route_app_labels
36         ):
37             return True
38         return None
39
40     def allow_migrate(self, db, app_label, model_name=None, **hints):
41         """
42         Make sure the auth and contenttypes apps only appear in the
43         'auth_db' database.
44         """
45         if app_label in self.route_app_labels:
46             return db == "data_db"
47         return None
```

Fichero passengerss serializers.py.

```
from rest_framework import serializers
from .models import *
# Esta clase se encarga de serializar el modelo de la aplicacion para
# su uso en la BBDD
class PassengerSerializer(serializers.ModelSerializer):
    class Meta:
        model=Passenger
        fields= '__all__'

    def create(self, validated_data):

        return Passenger.objects.using("data_db").create(**
   validated_data)
```

Fichero passengerss urls.py.

```
from django.urls import path
from .views import *

# Esta clase contiene las Endpoints URL
urlpatterns = [

    path('passengers/', passengers_api_list_view, name='
   passengers_api_list'), # ruta mostrar todos los pasajeros
    path('adddpassenger/', add_passenger_api_view, name='
   add_passenger_api_view'), # ruta crear un pasajero
    path('detail/',passenger_api_detail_view, name='
   passenger_api_detail_view'), # ruta detail de un pasajero
    path('update/', passenger_update_api_view, name='
   passenger_update_api_view') # ruta modificar pasajero

]
```

Fichero passengerss views.py.

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
from .models import Passenger
from .serializers import PassengerSerializer
from django.db.models.query_utils import Q
# En esta clase se definen las vistas de acceso a la aplicacion

# obtenemos todos los pasajeros
@api_view(['GET'])
def passengers_api_list_view(request):
    passengers = Passenger.objects.using("data_db").all()

    serializer = PassengerSerializer(passengers, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)

@api_view(['POST'])
def add_passenger_api_view(request):
    passenger_serializer = PassengerSerializer(data = request.data)
    if passenger_serializer.is_valid():
```

```
        passenger_serializer.create(passenger_serializer.data)
        return Response("success", status=status.HTTP_200_OK)
    return Response(passenger_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)

@api_view(['POST'])
def passenger_api_detail_view(request):
    passengers = Passenger.objects.using("data_db").filter(userIDowner=
    request.data['userIDowner']).first()

    serializer = PassengerSerializer(passengers)
    return Response(serializer.data)

@api_view(['PUT'])
def passenger_update_api_view(request):
    passenger = Passenger.objects.using("data_db").filter(userIDowner=
    request.data['userIDowner']).first()
    pax_serializer = PassengerSerializer(passenger, data=request.data)
    if(pax_serializer.is_valid()):
        pax_serializer.save()
        return Response("success", status=status.HTTP_200_OK)
    return Response(pax_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
```

Carpeta rates

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         02/09/2023    20:54                __pycache__
-a---          27/07/2023    20:11              0 __init__.py
-a---          02/09/2023    19:56            168 admin.py
-a---          02/09/2023    19:56            216 apps.py
-a---          02/09/2023    19:56           1200 models.py
-a---          02/09/2023    19:56           1523 rates_db_router.py
-a---          02/09/2023    19:56            399 serializers.py
-a---          02/09/2023    19:57            497 urls.py
-a---          02/09/2023    19:58           2206 views.py
```

Figura C.9 : Ficheros en carpeta rates.

Se muestra el Código Fuente de los ficheros de la Figura C.9 .

Fichero rates admin.py.

```
from django.contrib import admin
from .models import *
# Esta clase permite el acceso al administrador desde la web de
    administracion

admin.site.register(Rate);
```

Fichero rates admin.py.

```
from django.apps import AppConfig

# Esta clase indica el nombre para que settings la reconozca

class RoutesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.rates'
```

Fichero rates models.py.

```
from django.db import models
from django.utils import timezone
# Esta clase contiene el modelo de datos de la aplicacion, su
    equivalente
# a tablas en las bases de datos
# El nombre de los campos es autodecriptivo para facilitar su uso
from django.contrib.auth.hashers import make_password
import uuid

class Rate(models.Model): # Modelo valoracion
    id = models.UUIDField(primary_key = True, # campo id
         default=uuid.uuid4,editable = False)
    rateCreated = models.DateTimeField(default=timezone.now)  # campo
    creado en
    userIDowner = models.CharField(max_length=100, null=False, unique=
    False) # usuario que valora
    rateUser=models.CharField(max_length=100, null=True, unique=False) #
    usuario valorado
    rateAnonymous=models.BooleanField(default=False) # marcar anonimo,
    value True=anonimo, False=publico
    rateValue=models.smallIntegerField(default=3) # valor en numero de
    la valoracion rango 1 a 5
    rateComment=models.CharField(max_length=250, null=True) # comentario
    en valoracion
    rateStatus=models.BooleanField(default=True) # estado valoracion
    value True=activo, False=inactivo
    rateDeleted=models.BooleanField(default=False) # valoracion borrada,
    value True=borrada, False=no borrada
    rateDriverRated=models.CharField(max_length=10,default="NO") # el
    conductor ha valorado
    ratePaxRated=models.CharField(max_length=10,default="NO") # el
    pasajero ha volado
    rateShow=models.CharField(max_length=10,default="SI") # mostrar
    valoracion

    objects = models.Manager() # permite crear los objetos en el
    serializador

    def __str__(self):
        return self.userIDowner
```

Se muestra el fichero *rates_db_router.py* que sirve para añadir la aplicación a la BBDD. Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero rates rates_db_router.py.

```
# Esta clase hace de router para base de datos, es requisito
# al usar multiples bases de datos. Permite la lectura, escritura,
    relacion y migracion
class RatesRouter:
    """
    A router to control all database operations on models in the
    auth and contenttypes applications.
    """
```

```
8
9       route_app_labels = {"auth", "contenttypes"}
10
11      def db_for_read(self, model, **hints):
12          """
13          Attempts to read auth and contenttypes models go to auth_db.
14          """
15          if model._meta.app_label in self.route_app_labels:
16              return "data_db"
17          return None
18
19      def db_for_write(self, model, **hints):
20          """
21          Attempts to write auth and contenttypes models go to auth_db.
22          """
23          if model._meta.app_label in self.route_app_labels:
24              return "data_db"
25          return None
26
27      def allow_relation(self, obj1, obj2, **hints):
28          """
29          Allow relations if a model in the auth or contenttypes apps is
30          involved.
31          """
32          if (
33              obj1._meta.app_label in self.route_app_labels
34              or obj2._meta.app_label in self.route_app_labels
35          ):
36              return True
37          return None
38
39      def allow_migrate(self, db, app_label, model_name=None, **hints):
40          """
41          Make sure the auth and contenttypes apps only appear in the
42          'auth_db' database.
43          """
44          if app_label in self.route_app_labels:
45              return db == "data_db"
46          return None
```

Fichero rates serializers.py.

```
1  from rest_framework import serializers
2  from .models import *
3  # Esta clase se encarga de serializar el modelo de la aplicacion para
4  # su uso en la BBDD
5  class RateSerializer(serializers.ModelSerializer):
6      class Meta:
7          model=Rate
8          fields= '__all__'
9
10     def create(self, validated_data):
11
12         return Rate.objects.using("data_db").create(**validated_data)
```

Fichero rates urls.py.

```
1  from django.urls import path
2  from .views import *
```

```
3
4  # Esta clase contiene las Endpoints URL
5
6  urlpatterns = [
7
8      path('rates/', rates_api_list_view, name='rates_api_list'), # ruta
       mostrar todas las valoraciones
9      path('addrate/', add_rate_api_view, name='add_rate_api_view'), #
       ruta crear una valoracion
10     path('rate/',rate_api_detail_view, name='rate_api_detail_view'), #
       ruta mostrar detalle valoracion
11     path('myrates/',raterec_api_detail_view, name='
       raterec_api_detail_view'), # ruta mostrar mis valoraciones recibidas
12     path('update/', update_rate_api_view, name='update_rate_api_view') #
       ruta modificar valoracion
13
14 ]
```

Fichero rates views.py.

```
1  from rest_framework.views import APIView
2  from rest_framework.response import Response
3  from rest_framework.decorators import api_view
4  from rest_framework import status
5  from rest_framework import permissions
6  from .models import Rate
7  from .serializers import RateSerializer
8  from django.db.models.query_utils import Q
9  # En esta clase se definen las vistas de acceso a la aplicacion
10
11 # Esta vista muestra todas las valoraciones
12 @api_view(['GET'])
13 def rates_api_list_view(request):
14     rates = Rate.objects.using("data_db").all()
15     # if there is something in items else raise error
16     serializer = RateSerializer(rates, many=True)
17     return Response(serializer.data, status=status.HTTP_200_OK)
18
19 # Esta vista crea una valoracion
20 @api_view(['POST'])
21 def add_rate_api_view(request):
22
23     rate_serializer = RateSerializer(data = request.data)
24
25     if rate_serializer.is_valid():
26         rate_serializer.create(rate_serializer.data)
27
28         return Response("success", status=status.HTTP_200_OK)
29     return Response(rate_serializer.errors, status=status.
       HTTP_400_BAD_REQUEST)
30
31 # Esta vista devuelve las valoraciones hechas por un usuario
32 @api_view(['POST'])
33 def rate_api_detail_view(request):
34     rates = Rate.objects.using("data_db").filter(userIDowner=request.
       data['userIDowner']).all()
35
36     serializer = RateSerializer(rates, many=True)
37     return Response(serializer.data)
```

```
38
39 # Esta vista devuelve las valoraciones recibidas por un usuario
40 @api_view(['POST'])
41 def raterec_api_detail_view(request):
42     rates = Rate.objects.using("data_db").filter(rateUser=request.data['
   rateUser']).all()
43
44     serializer = RateSerializer(rates, many=True)
45     return Response(serializer.data)
46
47 # Esta vista modifica una valoracion
48 @api_view(['PUT'])
49 def update_rate_api_view(request):
50     rate = Rate.objects.using("data_db").filter(id=request.data['id']).
   first()
51     rate_serializer=RateSerializer(rate, data=request.data)
52     if(rate_serializer.is_valid()):
53         rate_serializer.save()
54         return Response("success", status=status.HTTP_200_OK)
55     return Response(rate_serializer.errors, status=status.
   HTTP_400_BAD_REQUEST)
```

Carpeta routes

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          02/09/2023    20:54                __pycache__
-a---          27/07/2023    20:11              0 __init__.py
-a---          02/09/2023    19:59            169 admin.py
-a---          02/09/2023    19:59            217 apps.py
-a---          02/09/2023    19:59           1714 models.py
-a---          02/09/2023    19:59           1524 routes_db_router.py
-a---          02/09/2023    19:59            402 serializers.py
-a---          02/09/2023    20:00            443 urls.py
-a---          02/09/2023    20:01           1838 views.py
```

Figura C.10 : Ficheros en carpeta routes.

Se muestra el Código Fuente de los ficheros de la Figura C.10 .

Fichero routes admin.py.

```
1 from django.contrib import admin
2 from .models import *
3 # Esta clase permite el acceso al administrador desde la web de
   administracion
4
5 admin.site.register(Route);
```

Fichero routes apps.py.

```
1 from django.apps import AppConfig
2
3 # Esta clase indica el nombre para que settings la reconozca
4
5 class RoutesConfig(AppConfig):
6     default_auto_field = 'django.db.models.BigAutoField'
7     name = 'apps.routes'
```

Fichero routes models.py.

```
from django.db import models
from django.utils import timezone

import uuid
# Esta clase contiene el modelo de datos de la aplicacion, su equivalente
# a tablas en las bases de datos
# El nombre de los campos es autodecriptivo para facilitar su uso
class Route(models.Model):
    id = models.UUIDField(primary_key = True,
         default=uuid.uuid4,editable = False) #ID
    routeCreated = models.DateTimeField(default=timezone.now) #fecha de creacion
    userIDowner = models.CharField(max_length=100, null=False, unique=False) #Identificador del usuario
    routeOriginLat = models.DecimalField(max_digits=30, decimal_places=25, null=True) #Latitud punto origen
    routeOriginLng = models.DecimalField(max_digits=30, decimal_places=25, null=True) #Longitud punto origen
    routeDestLat = models.DecimalField(max_digits=30, decimal_places=25, null=True) #Latitud punto destino
    routeDestLng = models.DecimalField(max_digits=30, decimal_places=25, null=True) #Longitud punto destino
    routeFullDistance = models.DecimalField(max_digits=20, decimal_places=3, default=0.0) #Distancia recorrido
    routeFullDuraton = models.DecimalField(max_digits=20, decimal_places=5, default=0.0) #Tiempo en segundos en recorrer el trayecto
    routeType = models.CharField(max_length=20, null=True) #Tipo de ruta: Conductor o Pasajero
    routePath = models.JSONField(null=True) #Coordenadas del trayecto
    objects = models.Manager() # Objetos para serializer.py

    #Metodo String que devuelve los valores identificativos, solo visible en panel admin
    def __str__(self):
        return self.userIDowner+" "+str(self.routeOriginLat)+" "+str(self.routeOriginLng)+" "+self.routeType
```

Se muestra el fichero *routes_db_router.py* que sirve para añadir la aplicación a la BBDD. Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero routes routes_db_router.py.

```
# Esta clase hace de router para base de datos, es requisito
# al usar multiples bases de datos. Permite la lectura, escritura, relacion y migracion
class RoutesRouter:
    """
    A router to control all database operations on models in the
    auth and contenttypes applications.
    """

    route_app_labels = {"auth", "contenttypes"}

```

```
    def db_for_read(self, model, **hints):
        """
        Attempts to read auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def db_for_write(self, model, **hints):
        """
        Attempts to write auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def allow_relation(self, obj1, obj2, **hints):
        """
        Allow relations if a model in the auth or contenttypes apps is
        involved.
        """
        if (
            obj1._meta.app_label in self.route_app_labels
            or obj2._meta.app_label in self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        """
        Make sure the auth and contenttypes apps only appear in the
        'auth_db' database.
        """
        if app_label in self.route_app_labels:
            return db == "data_db"
        return None
```

Fichero routes serializers.py.

```
from rest_framework import serializers
from .models import *
# Esta clase se encarga de serializar el modelo de la aplicacion para
# su uso en la BBDD
class RouteSerializer(serializers.ModelSerializer):
    class Meta:
        model=Route
        fields= '__all__'

    def create(self, validated_data):

        return Route.objects.using("data_db").create(**validated_data)
```

Fichero routes urls.py.

```
from django.urls import path
from .views import *

# Esta clase contiene las Endpoints URL

```

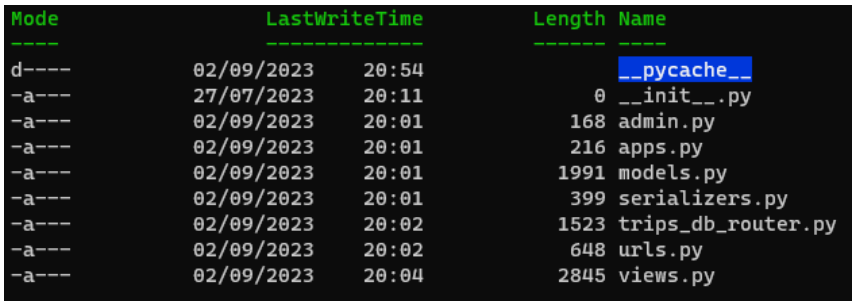
```
urlpatterns = [

    path('routes/', routes_api_list_view, name='routes_api_list'), # ruta mostrar todas las rutas
    path('addroute/', add_route_api_view, name='add_route_api_view'), # ruta crear una ruta
    path('routenodes/',route_api_detail_view, name='node_api_detail_view'), # ruta mostrar detalle una ruta
    path('nearest/', route_type_api_detail_view,name='route_type_api_detail_view') # ruta mostrar cercanas

]
```

Fichero routes views.py.

```

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
from .models import Route
from .serializers import RouteSerializer
from django.db.models.query_utils import Q

# En esta clase se definen las vistas de acceso a la aplicacion

# Esta vista devuelve todas las rutas
@api_view(['GET'])
def routes_api_list_view(request):
    routes = Route.objects.using("data_db").all()
    # if there is something in items else raise error
    serializer = RouteSerializer(routes, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)

# Esta vista crea una ruta
@api_view(['POST'])
def add_route_api_view(request):

    route_serializer = RouteSerializer(data = request.data)

    if route_serializer.is_valid():
        route_serializer.create(route_serializer.data)

        return Response("success", status=status.HTTP_200_OK)
    return Response(route_serializer.errors, status=status.HTTP_400_BAD_REQUEST)


# Esta vista ve la ruta del usuario
@api_view(['POST'])
def route_api_detail_view(request):
    routes = Route.objects.using("data_db").filter(userIDowner=request.data['userIDowner']).first()
    # if there is something in items else raise error
    serializer = RouteSerializer(routes)
    return Response(serializer.data)

# Esta vista devuelve todas las rutas de un tipo especifico
```

```
@api_view(['POST'])
def route_type_api_detail_view(request):
    routes = Route.objects.using("data_db").filter(routeType=request.
    data['routeType']).all()
    # if there is something in items else raise error
    serializer = RouteSerializer(routes, many=True)
    return Response(serializer.data)
```

Carpeta trips



```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         02/09/2023    20:54                __pycache__
-a---          27/07/2023    20:11              0 __init__.py
-a---          02/09/2023    20:01            168 admin.py
-a---          02/09/2023    20:01            216 apps.py
-a---          02/09/2023    20:01           1991 models.py
-a---          02/09/2023    20:01            399 serializers.py
-a---          02/09/2023    20:02           1523 trips_db_router.py
-a---          02/09/2023    20:02            648 urls.py
-a---          02/09/2023    20:04           2845 views.py
```

Figura C.11 : Ficheros en carpeta trips.

Se muestra el Código Fuente de los ficheros de la Figura C.11 .

Fichero trips admin.py.

```
from django.contrib import admin
from .models import *
# Esta clase permite el acceso al administrador desde la web de
    administracion

admin.site.register(Trip);
```

Fichero trips apps.py.

```
from django.apps import AppConfig

# Esta clase indica el nombre para que settings la reconozca

class RoutesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.trips'
```

Fichero trips models.py.

```
from django.db import models
from django.utils import timezone

from django.contrib.auth.hashers import make_password
import uuid
# Esta clase contiene el modelo de datos de la aplicacion, su
    equivalente
# a tablas en las bases de datos
# El nombre de los campos es autodecriptivo para facilitar su uso
class Trip(models.Model): # modelo para reservas
    id = models.UUIDField(primary_key = True, # campo id
         default=uuid.uuid4,editable = False)
```

```
12      tripCreated = models.DateTimeField(default=timezone.now) # campo
    creado en
13      userIDowner = models.CharField(max_length=100, null=False, unique=
    False) # identificador usuario conductor
14      tripPax=models.CharField(max_length=100, null=True, unique=False) #
    identificador usuario pasajero
15      tripAccepted=models.BooleanField(default=False) # reserva aceptada
16      tripRejected=models.BooleanField(default=False) # reserva rechazada
17      tripRejectedP=models.BooleanField(default=False) # reserva anulada
    por pasajero
18      tripStartTime=models.CharField(max_length=15, null=True) # horario
    inicio reserva
19      tripEndTime=models.CharField(max_length=15, null=True) # horario fin
        reserva
20      tripActive=models.BooleanField(default=False) # reserva activa
21      tripDayMon=models.BooleanField(default=False) # lunes
22      tripDayTue=models.BooleanField(default=False) # martes
23      tripDayWed=models.BooleanField(default=False) # miercoles
24      tripDayThu=models.BooleanField(default=False) # jueves
25      tripDayFri=models.BooleanField(default=False) # viernes
26      tripDaySat=models.BooleanField(default=False) # sabado
27      tripDaySun=models.BooleanField(default=False) # domingo
28      tripPermanent=models.BooleanField(default=False) # reserva fija
29      tripInit = models.BooleanField(default=False) # ida
30      tripBack = models.BooleanField(default=False) # vuelta
31      tripArchived=models.BooleanField(default=False) # reserva archivada
32      tripEnded=models.BooleanField(default=False) # reserva finalizada
33      tripOnVehicleD=models.BooleanField(default=False) # conductor listo
    viaje
34      tripOnVehicleP=models.BooleanField(default=False) # pasajero listo
    viaje
35      tripReactivate=models.BooleanField(default=False) # reactivar
    reserva
36      tripStatus=models.CharField(max_length=10, default="OK") # estado de
        la reserva
37      tripShow=models.CharField(max_length=10, default="YES") # mostrar
    reserva
38      objects = models.Manager() # permite crear los objetos par el
    serializador
39
40      def __str__(self):
41          return self.userIDowner+" "+self.tripPax
```

Fichero trips serializers.py.

```
1  from rest_framework import serializers
2  from .models import *
3  # Esta clase se encarga de serializar el modelo de la aplicacion para
4  # su uso en la BBDD
5  class TripSerializer(serializers.ModelSerializer):
6      class Meta:
7          model=Trip
8          fields= '__all__'
9
10     def create(self, validated_data):
11
12         return Trip.objects.using("data_db").create(**validated_data)
```

Se muestra el fichero *trips_db_router.py* que sirve para añadir la aplicación a la BBDD.

Este Código Fuente 7.3 mantiene la estructura y código ofrecida por Django para el uso de múltiples BBDD. Se ha cambiado únicamente el código asociado al nombre de clase y el nombre de la BBDD.

Fichero trips trips_db_router.py.

```
# Esta clase hace de router para base de datos, es requisito
# al usar multiples bases de datos. Permite la lectura, escritura,
   relacion y migracion
class TripsRouter:
    """
    A router to control all database operations on models in the
    auth and contenttypes applications.
    """

    route_app_labels = {"auth", "contenttypes"}

    def db_for_read(self, model, **hints):
        """
        Attempts to read auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def db_for_write(self, model, **hints):
        """
        Attempts to write auth and contenttypes models go to auth_db.
        """
        if model._meta.app_label in self.route_app_labels:
            return "data_db"
        return None

    def allow_relation(self, obj1, obj2, **hints):
        """
        Allow relations if a model in the auth or contenttypes apps is
        involved.
        """
        if (
            obj1._meta.app_label in self.route_app_labels
            or obj2._meta.app_label in self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        """
        Make sure the auth and contenttypes apps only appear in the
        'auth_db' database.
        """
        if app_label in self.route_app_labels:
            return db == "data_db"
        return None
```

Fichero trips urls.py.

```
from django.urls import path
from .views import *
```

```
# Esta clase contiene las Endpoints URL

urlpatterns = [

    path('trips/', trips_api_list_view, name='trips_api_list'), # ruta mostrar todas las reservas
    path('addtrip/', add_trip_api_view, name='add_trip_api_view'), # ruta crear una reserva
    path('trip/',trip_api_detail_view, name='trip_api_detail_view'), # ruta mostrar detalle una reserva
    path('update/', update_trip_api_view, name='update_trip_api_view'), # ruta moficiar una reserva
    path('booktrip/', book_trip_api_detail, name='book_trip_api_detail'), # ruta reservar una reserva
    path('deltrip/', delete_trip_view, name='delete_trip_view'), # ruta borrar una reserva
    path('tripid/', trip_unique_api_detail_view,name='trip_unique_api_detail_view') # ruta mostrar id una reserva

]
```

Fichero trips views.py.

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import permissions
from django_filters.rest_framework import DjangoFilterBackend
from .models import Trip
from .serializers import TripSerializer
from django.db.models.query_utils import Q
# En esta clase se definen las vistas de acceso a la aplicacion

# Esta vista devuelve todas las reservas
@api_view(['GET'])
def trips_api_list_view(request):

    trips = Trip.objects.using("data_db").filter(tripShow="YES").all()


    # if there is something in items else raise error
    serializer = TripSerializer(trips, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)

# Esta vista borra una reserva
@api_view(['POST'])
def delete_trip_view(request):
    trip = Trip.objects.using("data_db").filter(id=request.data['id']).first()
    trip.delete()
    return Response("success", status=status.HTTP_200_OK)

# Esta vista crea una reserva
@api_view(['POST'])
def add_trip_api_view(request):

    trip_serializer = TripSerializer(data = request.data)

```

```
36      if trip_serializer.is_valid():
37          trip_serializer.create(trip_serializer.data)
38
39          return Response("success", status=status.HTTP_200_OK)
40      return Response(trip_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
41
42
43  # Esta vista devuelve todas las reservas de un usuario
44  @api_view(['POST'])
45  def trip_api_detail_view(request):
46      trips = Trip.objects.using("data_db").filter(tripShow="YES",
    userIDowner=request.data['userIDowner']).all()
47
48      serializer = TripSerializer(trips, many=True)
49      return Response(serializer.data)
50
51  # Esta vista devuelve una reserva especifica
52  @api_view(['POST'])
53  def trip_unique_api_detail_view(request):
54      trip = Trip.objects.using("data_db").filter(tripShow="YES", id=
    request.data['id']).first()
55
56      serializer = TripSerializer(trip)
57      return Response(serializer.data)
58
59  # Esta vista devuelve las reservas activas
60  @api_view(['POST'])
61  def book_trip_api_detail(request):
62      trips = Trip.objects.using("data_db").filter(tripShow="YES", tripPax
    =request.data['tripPax']).all()
63
64      serializer = TripSerializer(trips, many=True)
65      return Response(serializer.data)
66
67  # Esta vista modifica una reserva
68  @api_view(['PUT'])
69  def update_trip_api_view(request):
70      trip = Trip.objects.using("data_db").filter(id=request.data['id']).
    first()
71      trip_serializer=TripSerializer(trip, data=request.data)
72      if(trip_serializer.is_valid()):
73          trip_serializer.save()
74          return Response("success", status=status.HTTP_200_OK)
75      return Response(trip_serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
```

Frontend: Vue 3

Al igual que con DRF, la instalación de Vue 3 genera unos ficheros por defecto que apenas se han de modificar por parte del desarrollador y que han de estar para un correcto funcionamiento.

Directorio Raíz y carpeta public

La Figura C.12 muestra el directorio raíz del proyecto.

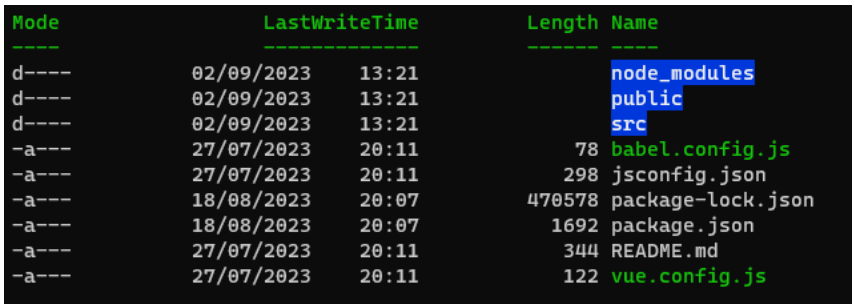


Figura C.12 : Estructura raíz de ficheros en Vue 3.

Los ficheros del directorio raíz son ficheros de configuración y no se han de modificar por parte del desarrollador. El fichero *package.json* contiene las librerías que son necesarias para el funcionamiento y aquellas que hemos instalado nosotros. El resto de ficheros son de configuración y se generan solos sin intervención por nuestra parte.

Fichero babel.config.js.

```
module.exports = {
  presets: [
    '@vue/cli-plugin-babel/preset'
  ]
}
```

Fichero jsconfig.json.

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "esnext",
    "baseUrl": "./",
    "moduleResolution": "node",
    "paths": {
      "@/*": [
        "src/*"
      ]
    },
    "lib": [
      "esnext",
      "dom",
      "dom.iterable",
      "scripthost"
    ]
```

```
  }
}
```

Fichero package.json.

```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^6.4.2",
    "@fortawesome/free-brands-svg-icons": "^6.4.2",
    "@fortawesome/free-regular-svg-icons": "^6.4.2",
    "@fortawesome/free-solid-svg-icons": "^6.4.2",
    "@fortawesome/vue-fontawesome": "^3.0.3",
    "@popperjs/core": "^2.11.8",
    "axios": "^1.4.0",
    "bootstrap": "^5.3.0-alpha3",
    "core-js": "^3.8.3",
    "geolib": "^3.3.4",
    "haversine-distance": "^1.2.1",
    "leaflet-routing-machine": "^3.2.12",
    "mapbox-gl": "^2.15.0",
    "pinia": "^2.1.4",
    "sweetalert2": "^11.7.12",
    "uuid": "^9.0.0",
    "vue": "^3.3.4",
    "vue-router": "^4.0.3",
    "vue3-simple-alert": "^1.0.4",
    "vuetify": "^3.3.13"
  },
  "devDependencies": {
    "@babel/core": "^7.12.16",
    "@babel/eslint-parser": "^7.12.16",
    "@vue-leaflet/vue-leaflet": "^0.9.0",
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-eslint": "~5.0.0",
    "@vue/cli-plugin-router": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "eslint": "^7.32.0",
    "eslint-plugin-vue": "^8.0.3",
    "leaflet": "^1.9.4"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/vue3-essential",
      "eslint:recommended"
    ],
    "parserOptions": {
```

```
54        "parser": "@babel/eslint-parser"
55      },
56      "rules": {}
57    },
58    "browserslist": [
59      "> 1%",
60      "last 2 versions",
61      "not dead",
62      "not ie 11"
63    ]
64  }
```

Fichero vue.config.js.

```
1  const { defineConfig } = require('@vue/cli-service')
2  module.exports = defineConfig({
3    transpileDependencies: true
4  })
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          27/07/2023    20:11           4286 favicon.ico
-a---          19/08/2023    16:03            795 index.html
```

Figura C.13 : Carpeta public.

En la Figura C.13 se muestra el contenido de la carpeta public, el cual sólo contiene un fichero HTML y un fichero de icono. Se mostrará en el Código Fuente 7.3 el contenido del fichero *index.html*.

Fichero index.html.

```
1  <!DOCTYPE html>
2  <html lang="es">
3    <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width,initial-scale
           =1.0, maximum-scale=1">
7      <link rel="icon" href="<%= BASE_URL %>favicon.ico">
8
9      <link href='https://api.mapbox.com/mapbox-gl-js/v2.15.0/mapbox-
           gl.css' rel='stylesheet' />
10
11     <!--title><%= htmlWebpackPlugin.options.title %></!--title-->
12     <title>SharedSeats</title>
13   </head>
14   <body>
15     <noscript>
16       <strong>We're sorry but <%= htmlWebpackPlugin.options.title %>
             doesn't work properly without JavaScript enabled. Please
           enable it to continue.</strong>
17     </noscript>
18     <div id="app"></div>
19     <!-- built files will be auto injected -->
20   </body>
21 </html>
```

Carpeta src

La Figura C.14 muestra las carpetas y ficheros de la carpeta src.

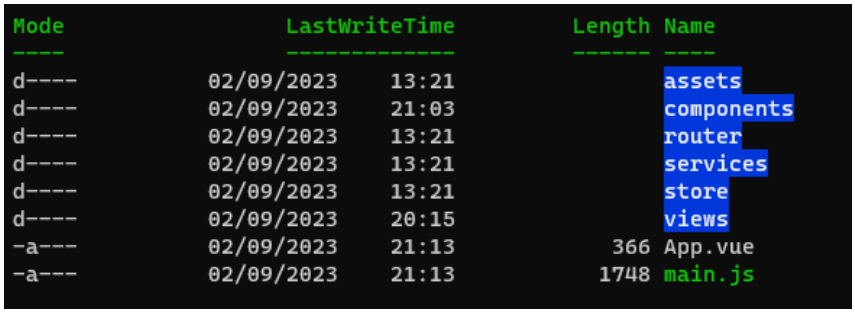


Figura C.14 : Ficheros y carpetas en carpeta src.

Se muestra el Código Fuente de los ficheros *App.vue* y *main.js* que son los principales ficheros para que Vue 3 funcione.

Fichero App.vue.

```
<template>
  <div class="headerC">
    <HeaderComp />
  </div>
</template>

<script setup>
import HeaderComp from './components/HeaderComp.vue'
/** Aplicacion principal sobre la que se cargan todas las vistas */
</script>

<style>
.headerC{
  font-family: "Poppins";
  background-color:white;
  margin: 0 auto;
  width: 95%;
}
</style>
```

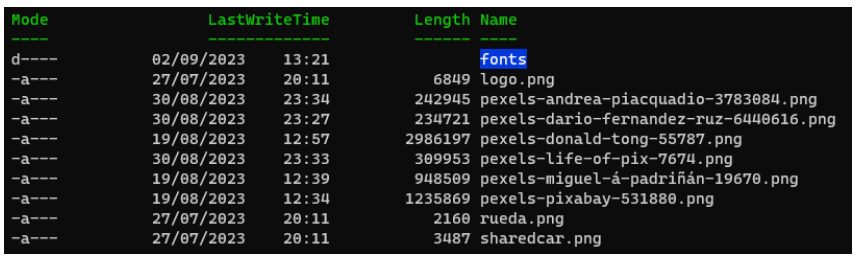
Fichero main.js.

```
/*** Archivo principal de configuracion donde se cargan todos
 * los componentes necesarios para poner en marcha la aplicacion
 */
/* eslint-disable */
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import 'bootstrap'
import 'bootstrap/dist/css/bootstrap.min.css'
import './assets/fonts/poppins/poppins.css'
import { library } from '@fortawesome/fontawesome-svg-core'
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'
import { faCar, faEarthEurope, faUsersViewfinder,
   faArrowRightFromBracket, faArrowRightToBracket, faUserPlus,
```

```
14          faPersonCirclePlus, faSmoking, faBanSmoking, faGear, faCarOn
                , faEnvelope, faStar as faStarSolid} from '@fortawesome/
                free-solid-svg-icons'
15  import { faUser, faStar as faStarRegular } from '@fortawesome/free-
        regular-svg-icons'
16  import { createPinia } from 'pinia';
17  import 'vuetify/styles'
18  import { createVuetify } from 'vuetify'
19  import * as components from 'vuetify/components'
20  import * as directives from 'vuetify/directives'
21
22  // importamos la libreria necesaria para los mapas
23  import mapboxgl from 'mapbox-gl'; // or "const mapboxgl = require('
        mapbox-gl');"
24  // constante para iconos
25  const vuetify = createVuetify({
26      components,
27      directives,
28    })
29  // distintos iconos usados en el frontend
30  library.add(faCar, faUser, faEarthEurope,faUsersViewfinder,
        faArrowRightFromBracket,faArrowRightToBracket, faUserPlus,
31      faPersonCirclePlus, faSmoking, faBanSmoking, faGear, faCarOn,
            faEnvelope, faStarRegular, faStarSolid)
32
33  mapboxgl.accessToken = '
        pk.eyJ1IjoiYW5hdnByb2dyYW1zIiwiYSI6ImNsazVmY251bjAyMWUzcXM3dX'+
34  'ZheXlkODAifQ.RrZDDFsepRR-qOJadp1YFg';
35
36  const app = createApp(App);
37  const pinia = createPinia();
38
39  app.component('font-awesome-icon', FontAwesomeIcon)
40  app.use(pinia);
41  app.use(router);
42  app.use(vuetify)
43  app.mount('#app');
```

Carpeta assets

La Figura C.15 muestra los ficheros de la carpeta assets, que contiene los archivos de imágenes empleados en la aplicación y las fuentes del texto.

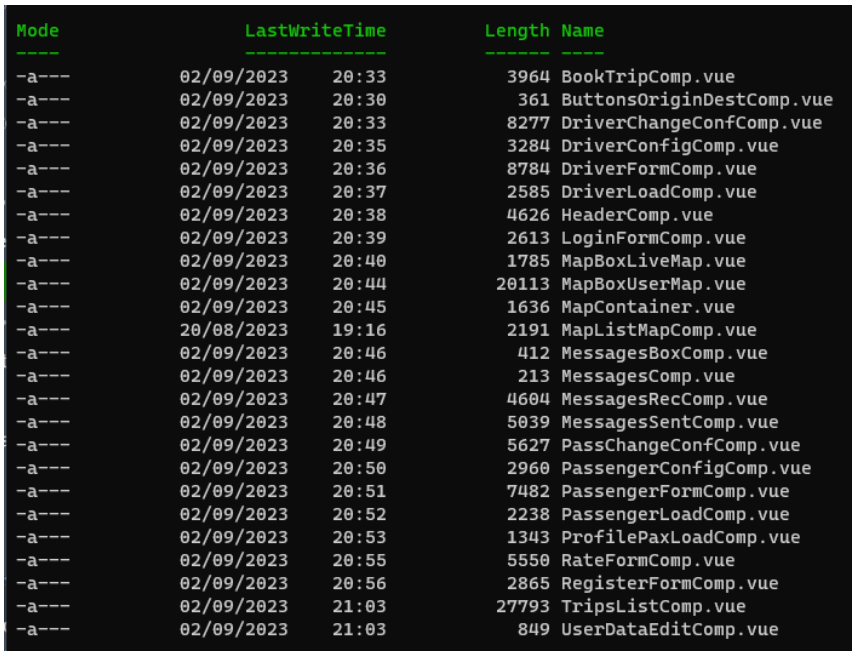


Mode	LastWriteTime	Length	Name
d----	02/09/2023 13:21		fonts
-a---	27/07/2023 20:11	6849	logo.png
-a---	30/08/2023 23:34	242945	pexels-andrea-piacquadio-3783084.png
-a---	30/08/2023 23:27	234721	pexels-dario-fernandez-ruz-6440616.png
-a---	19/08/2023 12:57	2986197	pexels-donald-tong-55787.png
-a---	30/08/2023 23:33	309953	pexels-life-of-pix-7674.png
-a---	19/08/2023 12:39	948509	pexels-miguel-á-padriñán-19670.png
-a---	19/08/2023 12:34	1235869	pexels-pixabay-531880.png
-a---	27/07/2023 20:11	2160	rueda.png
-a---	27/07/2023 20:11	3487	sharedcar.png

Figura C.15 : Ficheros en carpeta assets.

Carpeta components

La Figura C.16 muestra los ficheros de la carpeta components, que son los componentes que se han usado en la aplicación.



Mode	LastWriteTime	Length	Name
-a---	02/09/2023 20:33	3964	BookTripComp.vue
-a---	02/09/2023 20:30	361	ButtonsOriginDestComp.vue
-a---	02/09/2023 20:33	8277	DriverChangeConfComp.vue
-a---	02/09/2023 20:35	3284	DriverConfigComp.vue
-a---	02/09/2023 20:36	8784	DriverFormComp.vue
-a---	02/09/2023 20:37	2585	DriverLoadComp.vue
-a---	02/09/2023 20:38	4626	HeaderComp.vue
-a---	02/09/2023 20:39	2613	LoginFormComp.vue
-a---	02/09/2023 20:40	1785	MapBoxLiveMap.vue
-a---	02/09/2023 20:44	20113	MapBoxUserMap.vue
-a---	02/09/2023 20:45	1636	MapContainer.vue
-a---	20/08/2023 19:16	2191	MapListMapComp.vue
-a---	02/09/2023 20:46	412	MessagesBoxComp.vue
-a---	02/09/2023 20:46	213	MessagesComp.vue
-a---	02/09/2023 20:47	4604	MessagesRecComp.vue
-a---	02/09/2023 20:48	5039	MessagesSentComp.vue
-a---	02/09/2023 20:49	5627	PassChangeConfComp.vue
-a---	02/09/2023 20:50	2960	PassengerConfigComp.vue
-a---	02/09/2023 20:51	7482	PassengerFormComp.vue
-a---	02/09/2023 20:52	2238	PassengerLoadComp.vue
-a---	02/09/2023 20:53	1343	ProfilePaxLoadComp.vue
-a---	02/09/2023 20:55	5550	RateFormComp.vue
-a---	02/09/2023 20:56	2865	RegisterFormComp.vue
-a---	02/09/2023 21:03	27793	TripsListComp.vue
-a---	02/09/2023 21:03	849	UserDataEditComp.vue

Figura C.16 : Ficheros en carpeta components.

Fichero BookTripComp.vue.

```
1
2  <!-- Este es el componente principal de las reservas-->
3
4  <template>
5      <div class="bookTripBox">
6
7      Seleccione dia de la semana, trayecto fijo o ida/vuelta
8      <!-- Reservas {{ tripstore.getTripData }} -->
9      <form>
10         <div class="form-group">
11             <label>L..</label>
12             <input class="form-check-input" type="checkbox" v-model=
                   "booker.tripDayMon"/>
13             <label>M..</label>
14             <input class="form-check-input" type="checkbox" v-model=
                   "booker.tripDayTue"/>
15             <label>X..</label>
16             <input class="form-check-input" type="checkbox" v-model=
                   "booker.tripDayWed"/>
17             <label>J..</label>
18             <input class="form-check-input" type="checkbox" v-model=
                   "booker.tripDayThu"/>
19             <label>V..</label>
20             <input class="form-check-input" type="checkbox" v-model=
                   "booker.tripDayFri"/>
21             <label>S..</label>
22             <input class="form-check-input" type="checkbox" v-model=
                   "booker.tripDaySat"/>
```

```
23              <label>D..</label>
24              <input class="form-check-input" type="checkbox" v-model=
                    "booker.tripDaySun"/>
25              <label>Fijo..</label>
26              <input class="form-check-input" type="checkbox" v-model=
                    "booker.tripPermanent"/>
27              <label>Ida..</label>
28              <input class="form-check-input" type="checkbox" v-model=
                    "booker.tripInit"/>
29              <label>Vuelta..</label>
30              <input class="form-check-input" type="checkbox" v-model=
                    "booker.tripBack"/>
31              <button type="button" class="btn btn-outline-primary"
                    @click="bookTrip" id="bookTripbtn">Reservar</button>
32          </div>
33
34
35      </form>
36  </div>
37  </template>
38
39  <script setup>
40  /*  Importamos las librerias a usar */
41  /* eslint-disable */
42
43
44  import { useLoginStore } from '@/store/loginstore';
45  import { usePaxStore } from '@/store/paxstore';
46  import { useTripStore } from '@/store/tripstore';
47  import {ref} from 'vue';
48  import axios from 'axios'
49  import { useRouter } from 'vue-router';
50  import Swal from 'sweetalert2';
51  import MessagesService from '@/services/MessagesService';
52
53  /*  Definimos las constantes   */
54  const loginstore = useLoginStore();
55  const paxstore = usePaxStore();
56  const tripstore = useTripStore();
57  const router = useRouter()
58  const messageService = new MessagesService();
59
60  /*  Esta variable alamcenari el valor de la reserva seleccionada
        */
61  let booker=ref({
62      userIDowner:'',
63      tripPax:loginstore.getUsername,
64      tripStartTime:'',
65      tripEndTime:'',
66      tripDayMon:false,
67      tripDayTue:false,
68      tripDayWed:false,
69      tripDayThu:false,
70      tripDayFri:false,
71      tripDaySat:false,
72      tripDaySun:false,
73      tripPermanent:false,
74      tripInit:false,
```

```
    tripBack:false
})

/* Funciin que crea las reservas   */
const bookTrip=async ()=>{

    let temppax=paxstore.getPaxData
    let tempowner=tripstore.getTripData

   booker.value.userIDowner=tempowner
   booker.value.tripStartTime=temppax.passengerStartTime
   booker.value.tripEndTime=temppax.passengerEndTime
   await axios
   .post('http://127.0.0.1:8000/trips/addtrip/', booker.value)
   .then((response)=>{
    console.log(response)
   })
   Swal.fire({
        position: 'center',
        icon: 'success',
        title: 'Reserva realizada',
        showConfirmButton: false,
        timer: 1500
    })
   let bodyMSG="El usuario "+booker.value.tripPax + " ha solicitado
      una reserva"

   let messageData=ref({
    fromMessage:"Reservas",
    toMessage:booker.value.userIDowner,
    bodyMessage:bodyMSG
})
   const sendmsg= await messageService.sendMessageTo(messageData
    )


    router.push('/trips');

}
</script>

<style scoped>

.bookTripBox{

    position: relative;
    padding: 5px;
    width: 115%;
    top:50px;
    left:175px;
}

#bookTripbtn{
    position: relative;
    padding: 5px;
    right:-110px;
    top:-20px;
}
```

```
</style>
```

Fichero ButtonsOriginDestComp.vue.

```
<template>

    <button class="btn btn-primary" id="origincoordsbtn">Origen</
       button>
    <button class="btn btn-primary" id="destcoordsbtn">Destino</
       button>

</template>

<script setup>
/* Componente boton de origen y destino */

</script>

<style scoped>
 #origincoordsbtn{
    align-content: start;
    margin-right: 25%;
 }
</style>
```

Fichero DriverChangeConfComp.vue.

```
<!-- Formulario para la cambiar los datos de los conductores -->

<template>
    <div class="container">
        <form>
            <div class="flex-item item-1">
                <div class="form-group ">
                <label>Marca</label><input type="text" id="
                   driverBrand" name="ddriverBrand" class="form-
                   control"
                    placeholder="Marca" minlength="2" maxlength="100
                       " required v-model="driverData.driverBrand">
            </div>
            <div class="flex-item item-2">

            <div class="form-group">
                <label>Modelo</label><input type="text" id="
                   driverModel" name="ddriverModel" class="form-
                   control"
                    placeholder="Modelo" minlength="2" maxlength="
                       100" required v-model="driverData.driverModel
                       ">
            </div></div>
            <div class="flex-item item-3">
            <div class="form-group">
                <label>Color</label><input type="text" id="
                   driverVColor" name="ddriverVColor" class="form-
                   control"
                    placeholder="Color" minlength="2" maxlength="100
                       " required v-model="driverData.driverVColor">
            </div></div>
```

```
22              </div>
23              <div class="flex-item item-4">
24
25                  <div class="form-group">
26                  <label>Plazas</label><input type="number" id="
                        driverSeats" name="ddriverSeats" class="form-
                        control"
27                          placeholder="Plazas" min="1" max="7" required v-
                            model="driverData.driverSeats">
28              </div></div>
29              <div class="flex-item item-5">
30              <div class="form-group">
31                  <label>Ciudad</label><input type="text" id="
                        driverCity" name="ddriverCity" class="form-
                        control"
32                          placeholder="Ciudad" v-model="
                            driverData.driverCity">
33              </div></div>
34              <div class="flex-item item-6">
35              <div class="form-group">
36                  <label>Telefono</label><input type="text" id="
                        driverPhone" name="ddriverPhone" class="form-
                        control"
37                          placeholder="Telefono" v-model="
                            driverData.driverPhone">
38              </div></div>
39              <div class="flex-item item-7">
40                  <div class="form-group">
41                  <div class="chooseSmokingGroup">
42
43
44                      <div class="nosomokingGroup">
45
46                          <font-awesome-icon icon="fa-solid fa-ban-
                                smoking" size="2xl" />
47                          <input class="form-check-input" type="radio"
                                name="radioNoSmoking" id="radioNoSmoking
                                "
48                              :checked="noSmokingSelection" >
49                      </div>
50                      <div class="smokingGroup">
51
52                          <font-awesome-icon icon="fa-solid fa-smoking
                                " size="2xl" />
53                          <input class="form-check-input" type="radio"
                                name="radioNoSmoking" id="radioSmoking"
54                              :checked="!noSmokingSelection" >
55                      </div>
56                  </div></div></div>
57
58              <div class="flex-item item-8">
59                  <div class="form-group">
60                      <label>Salida...</label><input type="time" id="
                            driverStartTime" name="ddriverStartTime"
61                              v-model="driverData.driverStartTime">
62                  </div></div>
63                  <div class="flex-item item-9">
64                  <div class="form-group">
```

```
65                      <label>Vuelta... </label><input type="time" id="
                           driverEndTime" name="ddriverEndTime"
66                          v-model="driverData.driverEndTime">
67                  </div></div>
68
69              <div class="flex-item item-10">
70
71
72              <div class="form-group">
73                  <label>Maxima espera {{
                       driverData.driverWaitLimitStart }} minutos</label
                       ><input type="range"
74                      class="form-range" min="0" max="20" id="
                           driverWaitLimitStart" name="
                           ddriverWaitLimitStart"
75                      v-model="driverData.driverWaitLimitStart">
76              </div></div>
77              <div class="flex-item item-11">
78              <div class="form-group">
79                  <label>Maxima distancia {{
                       driverData.driverMaxDistanceAllowed }} km</label
                       ><input type="range"
80                      class="form-range" min="0" max="5" id="
                           driverMaxDistanceAllowed" name="
                           ddriverMaxDistanceAllowed"
81                      v-model="driverData.driverMaxDistanceAllowed">
82              </div></div>
83
84
85
86          </form>
87          <div class="buttonbox">
88              <button class="btn btn-outline-primary" @click="
                    updateDriverData">Guardar</button>
89          </div>
90      </div>
91  </template>
92
93  <script setup>
94  /*   Importamos las librerias */
95  /* eslint-disable */
96
97
98  import {ref, onMounted} from 'vue';
99  import { useLoginStore } from '@/store/loginstore';
100 import axios from 'axios';
101 import Swal from 'sweetalert2';
102
103 /* Definimos las constantes  */
104 const loginstore = useLoginStore();
105
106 /*  definimos las variables a usar */
107 let idNoSmoking="radioNoSmoking"
108 let idSmoking="radioSmoking"
109 let noSmokingSelection=true
110 let smokingSelection=false
111
112
```

```
let driverData=ref({
    driverBrand:'',
    driverModel:'',
    driverVColor:'',
    driverSeats:1,
    driverStartTime:'08:00',
    driverEndTime:'16:00',
    driverWaitLimitStart:0,
    driverMaxDistanceAllowed:0,
    driverFreeSeats:1,
    driverCity:'',
    driverPhone:'',
    driverSmoke:smokingSelection
})

/*  funcion que se ejecuta al cargar vista    */
onMounted(async ()=>{

      /* Cargamos los conductores  */
    const res = await fetch('http://127.0.0.1:8000/drivers/detail/',{
                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    "userIDowner": loginstore.getUsername
                })
            })
            const response = await res.json()
            driverData.value=response
    }

    );

/*  Modificamos el conductor  */
const updateDriverData=async ()=>{
    Swal.fire({
        title: '?Actualizar datos?',
        showDenyButton: true,
        showCancelButton: false,
        confirmButtonText: 'Si',
        denyButtonText: 'No',
        }).then(async (result) => {

        if (result.isConfirmed) {
            await axios
            .put('http://127.0.0.1:8000/drivers/update/',
               driverData.value)
            .then((response) =>{
                console.log(response);
            });
            Swal.fire({
                        position: 'center',
                        icon: 'success',
                        title: 'Datos actualizados',
                        showConfirmButton: false,
```

```
                        timer: 1500
                        });
        }else if (result.isDenied) {

        }})



}

</script>

<style scoped>
.container{
    margin:0 auto;
    border: 1px solid lightgray;
    border-radius: 10px;
    height: 100%;


}

.flex-item{


    width: 300px;
    position: relative;
}

.item-1{
    position: relative;
    padding: 5px;
}
.item-2{
    position: relative;
    padding: 5px;
}

.item-3{
    position: relative;
    padding: 5px;
}

.item-4{
    position: relative;
    padding: 5px;
    width: 80px;

}
.item-5{
    position: absolute;
    padding: 5px;
    left: 750px;
    top: 75px

}
.item-6{
    position: absolute;
```

```
    padding: 5px;
    left: 750px;
    top: 145px

}
.item-7{
    position: absolute;
    padding: 5px;
    left: 750px;
    top: 225px;

}
.item-8{
    position: absolute;
    padding: 5px;
    left: 750px;
    top: 290px;

}
.item-9{
    position: absolute;
    padding: 5px;
    left: 750px;
    top: 325px;

}
.item-10{
    position: absolute;
    padding: 5px;
    left: 1100px;
    top: 75px;

}
.item-11{
    position: absolute;
    padding: 5px;
    left: 1100px;
    top: 165px;

}
.smokingGroup{
    position: absolute;

    left:100px;
    top:5px;
}
.buttonbox{
    position: relative;
    right: 0px;
    padding: 5px;
    top:5px;
}



</style>
```

Fichero DriverConfigComp.vue.

```
<!-- Vista que carga el perfil o el formulario de registro de conductor
 -->
<template>
<div class="container">
<div class="collapse" :class="{'show':loginstore.getDriver}">

    <DriverLoadComp v-bind:dData="dData"/>

</div>
<div class="collapse" :class="{'show':!loginstore.getDriver}" v-if="!loginstore.getPassenger">


        <br>
    <div class="driverSelection">
    <input class="form-check-input" type="checkbox" value="" id="flexCheckDefault" v-model="isDriver">
    <div class="iconDriver">
    <label class="form-check-label" for="flexCheckDefault" id="iconLabelDriver">
        <font-awesome-icon icon="fa-solid fa-car" size="xl" v-model="isDriver"/><v-tooltip
                activator="parent"
                location="right"
                >Soy Conductor</v-tooltip>
      </label>
    </div>
    </div>
      <div class="collapse" :class="{'show':isDriver}">
        <DriverFormComp/>
        <br>

    </div>

</div>


</div>



</template>

<script setup>
/* Importamos las librerias   */
/* eslint-disable */

import { useLoginStore } from '@/store/loginstore';
import { useDriverStore } from '@/store/driverstore';
import {ref} from 'vue';
import { onMounted } from 'vue';
import DriverFormComp from './DriverFormComp.vue';
import DriverLoadComp from './DriverLoadComp.vue';

/* Definimos las constantes   */

const loginstore = useLoginStore();
```

```
54  const driverstore = useDriverStore();
55
56  /* Definimos las variables  */
57  let dData=driverstore.getDriverData
58
59  let rData={
60
61  }
62
63  let isDriver=ref(false)
64  onMounted(async ()=>{
65
66      reloadDataDriver();
67
68  })
69
70  /* Funcion que recarga los datos de conductor cada cierto tiempo
        */
71  const autoReloadDataDrivers=async()=>{
72      await reloadDataDriver()
73  }
74
75  /* Funcion que carga los datos del conductor    */
76  const reloadDataDriver=async()=>{
77      try{
78
79          const res = await fetch('http://127.0.0.1:8000/drivers/
                detail/',{
80          method: 'POST',
81                  headers: {
82                      'Accept':'application/json',
83                      'Content-Type': 'application/json'
84                  },
85                  body: JSON.stringify({
86                      userIDowner:loginstore.username
87                  })
88          })
89
90          const response = await res.json()
91
92          if(response.driverStatus){
93
94              dData=response
95              loginstore.addDriver(true)
96              driverstore.addDriverStoreData(response)
97
98          }
99          else{
100             loginstore.addDriver(false)
101
102         }
103
104         setTimeout(()=>{
105             autoReloadDataDrivers()
106         }, 30000);
107
108
109     }catch(error){
```

```
110         console.log(error)
111     }
112 }
113
114
115
116 </script>
117
118 <style scoped>
119 .driverSelection{
120
121
122     margin: 0 auto;
123     border: 1px solid lightblue;
124     border-radius: 15px;
125     width: 300px;
126     padding: 5px;
127
128 }
129
130 #flexCheckDefault{
131     position: relative;
132     grid-column: 1;
133
134     padding: 5px;
135     left:15px;
136 }
137
138 #iconLabelDriver{
139     position: relative;
140     left:45px;
141     top:-30px;
142     grid-column: 2;
143
144     position: relative;
145 }
146 </style>
```

Fichero DriverFormComp.vue.

```
1  <!-- Formulario de registro de los conductores
2   -->
3  <template>
4  <div class="card card-body">
5      <form >
6          <div class="form-group">
7                              <label>Marca</label><input type="text"
                                   id="driverBrand" name="ddriverBrand"
                                   class="form-control" placeholder="
                                   Marca"
8                              minlength="2" maxlength="100" required v
                                   -model="driverData.driverBrand">
9                              </div>
10                             <div class="form-group">
11                             <label>Modelo</label><input type="text"
                                   id="driverModel" name="ddriverModel"
                                   class="form-control" placeholder="
                                   Modelo"
```

```
12                          minlength="2" maxlength="100" required v
                               -model="driverData.driverModel">
13                          </div>
14                          <div class="form-group">
15                            <label>Color</label><input type="text"
                                  id="driverVColor" name="
                                 ddriverVColor" class="form-control"
                                  placeholder="Color"
16                          minlength="2" maxlength="100" required v
                               -model="driverData.driverVColor">
17                          </div>
18                          <div class="form-group">
19                            <label>Plazas</label><input type="
                                 number" id="driverSeats" name="
                                 ddriverSeats" class="form-control"
                                 placeholder="Plazas"
20                          min="1" max="7" required v-model="
                               driverData.driverSeats">
21                          </div>
22                          <div class="form-group">
23                            <label>Ciudad</label><input type="text
                                 " id="driverCity" name="ddriverCity
                                 " class="form-control" placeholder=
                                 "Ciudad"
24                            v-model="driverData.driverCity">
25                          </div>
26                          <div class="form-group">
27                            <label>Telefono</label><input type="
                                 text" id="driverPhone" name="
                                 ddriverPhone" class="form-control"
                                 placeholder="Telefono"
28                            v-model="driverData.driverPhone">
29                          </div>
30                          <br>
31                          <div class="form-group">
32                            <div class="chooseSmokingGroup">
33
34
35                            <div class="nosomokingGroup">
36
37                              <font-awesome-icon icon="fa-solid fa
                                    -ban-smoking" size="2xl" />
38                              <input class="form-check-input" type
                                    ="radio" name="radioNoSmoking" id
                                    ="radioNoSmoking" :checked="
                                    noSmokingSelection" @click="
                                    setNoSmokingSelection(idNoSmoking
                                    )" >
39                            </div>
40                           <div class="smokingGroup">
41
42                            <font-awesome-icon icon="fa-solid fa-
                                 smoking" size="2xl" />
43                            <input class="form-check-input" type="
                                 radio" name="radioNoSmoking" id="
                                 radioSmoking" :checked="!
                                 noSmokingSelection" @click="
                                 setNoSmokingSelection(idSmoking)">
```

```
                        </div>
                      </div>

                      </div>
                      <br>
                      <div class="container">
                      <div class="form-group">
                        <label>Salida...</label><input type="
                            time" id="driverStartTime" name="
                            ddriverStartTime" v-model="
                            driverData.driverStartTime">
                      </div>
                      <br>
                      <div class="form-group">
                        <label>Vuelta... </label><input type="
                            time" id="driverEndTime" name="
                            ddriverEndTime"
                      v-model="driverData.driverEndTime">
                      </div>
                    </div>
                    <br>
                      <div class="form-group">
                        <label>Maxima espera {{
                            driverData.driverWaitLimitStart }}
                            minutos</label><input type="range"
                            class="form-range" min="0" max="20"
                             id="driverWaitLimitStart" name="
                            ddriverWaitLimitStart"
                      v-model="driverData.driverWaitLimitStart
                          ">
                      </div>
                      <br>
                      <div class="form-group">
                        <label>Maxima distancia {{
                            driverData.driverMaxDistanceAllowed
                             }} km</label><input type="range"
                            class="form-range" min="0" max="5"
                            id="driverMaxDistanceAllowed" name=
                            "ddriverMaxDistanceAllowed"
                        v-model="
                            driverData.driverMaxDistanceAllowed
                            ">
                      </div>
                      <br>
                      <div class="driverButtonsLower">

                            <button class="btn btn-primary" id
                               ="ddriverAdded" @click.prevent=
                               "sendDataDriver" v-if="
                               addBtnDriver">Anadir</button>



                      </div>

```

```
81
82      </form>
83
84  </div>
85  </template>
86
87  <script setup>
88  /* Importamos librerias   */
89  /* eslint-disable */
90  import {ref} from 'vue';
91  import { useLoginStore } from '@/store/loginstore';
92  import Swal from 'sweetalert2';
93   /* Definimos constantes y variables   */
94  const loginstore = useLoginStore();
95
96  let idNoSmoking="radioNoSmoking"
97  let idSmoking="radioSmoking"
98  let noSmokingSelection=true
99  let smokingSelection=false
100
101 /* Esta variable contiene los datos del conductor   */
102 let driverData=ref({
103     driverBrand:'',
104     driverModel:'',
105     driverVColor:'',
106     driverSeats:1,
107     driverStartTime:'08:00',
108     driverEndTime:'16:00',
109     driverWaitLimitStart:0,
110     driverMaxDistanceAllowed:0,
111     driverFreeSeats:1,
112     driverCity:'',
113     driverPhone:'',
114     driverSmoke:smokingSelection
115 })
116
117 let addBtnDriver=ref(true);
118
119 const setNoSmokingSelection=(id)=>{
120   if((noSmokingSelection)&&(id=="radioSmoking")){
121
122
123     smokingSelection=true
124     noSmokingSelection=false
125   }
126   if((smokingSelection)&&(id=="radioNoSmoking")){
127
128     smokingSelection=false
129     noSmokingSelection=true
130   }
131
132
133 }
134  /* Creamos el conductor */
135  const sendDataDriver=async()=>{
136     try{
137       const res = await fetch('http://127.0.0.1:8000/drivers/
            adddriver/',{
```

```
        method:'POST',
        headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
body: JSON.stringify({
  userIDowner:loginstore.getUsername,
  driverBrand:driverData.value.driverBrand,
  driverModel:driverData.value.driverModel,
  driverVColor:driverData.value.driverVColor,
  driverSeats:driverData.value.driverSeats,
  driverStartTime:driverData.value.driverStartTime,
  driverEndTime:driverData.value.driverEndTime,
  driverWaitLimitStart:driverData.value.
  driverWaitLimitStart,
  driverMaxDistanceAllowed:driverData.value.
  driverMaxDistanceAllowed,
  driverFreeSeats:driverData.value.driverSeats,

})
      })
      const response = await res.json()
      loginstore.addDriver(true)
      Swal.fire({
                  position: 'center',
                  icon: 'success',
                  title: 'Vehiculo anadido',
                  showConfirmButton: false,
                  timer: 1500
                });


    } catch(error){
      Swal.fire({
                  position: 'center',
                  icon: 'warning',
                  title: 'Error al guardar',
                  showConfirmButton: false,
                  timer: 1500
                });
      console.log(error)
    }

}

</script>

<style scoped>

.chooseSmokingGroup{
    columns: 2;
    margin: 0 auto;
    border:1px solid blue;
    border-radius: 15px;
    padding: 5px;
    justify-content: stretch;
}

```

```
.nosomokingGroup{
  position: relative;
  grid-column: 1;
  left: 25%;
  width: 50%;
}

#radioNoSmoking{
  position: relative;
  grid-column: 1;
  left:15%
}

#radioSmoking{
  position: relative;
  grid-column: 1;
  left:30%
}


.smokingGroup{
  position: relative;
  left: 25%;
  grid-column: 2;


  width: 50%;

}
#ddriverAdded{
    position: relative;
    border: 1px solid red;
    padding: 5px;
    left:75px;
    width: 100px;
    margin: 0 auto;
}


</style>
```

Fichero DriverLoadComp.vue.

```
<!-- Plantilla con datos del conductor -->
<template>
<div class="card card-header">
    <h5>Vehiculo</h5>
    <h5>Plazas libres: {{ dData.driverFreeSeats }} de {{
       dData.driverSeats }}</h5>
    <label v-if="ratesReceived!=0">Valoraciones: {{ ratesReceived }}
        </label><font-awesome-icon icon="fa-regular fa-star" id="
       star1" v-if="ratesReceived!=0"/>
    <label id="labelstar" v-if="ratesReceived!=0">{{ rateAverage
       }}</label>
</div>
<div class="card card-body">
    <label>Marca: {{ dData.driverBrand }}</label>
    <label>Modelo: {{ dData.driverModel }}</label>
```

```
12      <label>Color: {{ dData.driverVColor }}</label>
13      <label>Inicio: {{ dData.driverStartTime }}</label>
14      <label>Final: {{ dData.driverEndTime }}</label>
15      <label>Espera: {{ dData.driverWaitLimitStart }} minutos maximo</
            label>
16      <label>Distancia: {{ dData.driverMaxDistanceAllowed }} km maxima
            </label>
17
18      <div class="smokeBox">
19          <font-awesome-icon icon="fa-solid fa-ban-smoking" size="2xl"
                v-if=!dData.driverSmoke />
20          <font-awesome-icon icon="fa-solid fa-smoking" size="2xl"  v-
                if=dData.driverSmoke />
21      </div>
22
23
24
25  </div>
26
27  </template>
28
29  <script setup>
30  /* Cargamos librerias  */
31  /* eslint-disable */
32  import RateService from '@/services/RateService'
33  import {ref, onMounted} from 'vue'
34  import { useLoginStore } from '@/store/loginstore';
35
36  /* Definimos constantes y variables  */
37  const rateservices = new RateService();
38  const loginstore = useLoginStore();
39
40  let rateData=ref([])
41
42  let ratesReceived=ref(0)
43  let rateAverage=ref(0)
44  let allRates=ref([])
45
46  defineProps({
47      dData: Object
48  })
49
50  onMounted(async ()=>{
51      rateData.value=await rateservices.loadMyRates(
            loginstore.getUsername)
52      ratesReceived.value=rateData.value.length
53      setAverageRates()
54
55  })
56
57  const setAverageRates=()=>{
58      rateData.value.forEach((rate, id)=>{
59          allRates.value.push(rate.rateValue)
60      })
61      let averageValue=0;
62      for(let i=0; i< ratesReceived.value; i++){
63          averageValue+=allRates.value[i];
64      }
```

```
65      rateAverage.value=averageValue/ratesReceived.value
66
67  }
68
69  </script>
70
71  <style scoped>
72  .smokeBox{
73
74      position: relative;
75      padding: 5px;
76
77
78  }
79
80  #star1{
81      border: 2px solid yellow;
82      background-color: yellow;
83      border-radius: 25px;
84      padding: 5px;
85      width: 25px;
86      position: relative;
87      top:-95px;
88      left:310px;
89
90  }
91
92  #labelstar{
93
94      width: 15%;
95      position: relative;
96      font-size:larger;
97      text-align: center;
98      top:-130px;
99      left:270px;
100
101
102 }
103
104
105 </style>
```

Fichero HeaderComp.vue.

```
1   <!-- Barra de navegacion con enlaces -->
2   <template>
3   <div>
4         <nav class="navbar navbar-expand-lg bg-body-tertiary" >
5         <div class="container-fluid" id="navBarHeader">
6           <a class="navbar-brand" href="#" id="headerTitle">
              Seats2Share</a>
7           <button class="navbar-toggler" type="button" data-bs-toggle=
              "collapse" data-bs-target="#navbarNav" aria-controls="
              navbarNav" aria-expanded="false" aria-label="Toggle
              navigation">
8             <span class="navbar-toggler-icon"></span>
9           </button>
10          <div class="collapse navbar-collapse justify-content-end" id
```

```
            ="navbarNav">

          <ul class="navbar-nav">
            <li class="nav-item">
              <label id="usernameLabelNavBar">{{
                 loginstore.getUsername }}</label>
            </li>
            <li class="nav-item">
              <router-link to="/messages" class="nav-link" v-show=
                 loginstore.getAuthLog ><font-awesome-icon icon="fa-
                 solid fa-envelope" size="xl" />
                <v-tooltip
                  activator="parent"
                  location="bottom"
                  >Mensajes</v-tooltip>

              </router-link>

            </li>
            <li class="nav-item">
              <router-link to="/trips" class="nav-link" v-show=
                 loginstore.getAuthLog><font-awesome-icon icon="fa-
                 solid fa-users-viewfinder" size="xl"/>
                <v-tooltip
                  activator="parent"
                  location="bottom"
                  >Mis Viajes</v-tooltip>

              </router-link>
            </li>
            <li class="nav-item">
              <router-link to="/profile" class="nav-link" v-show=
                 loginstore.getAuthLog ><font-awesome-icon icon="fa-
                 regular fa-user" size="xl"/>
                <v-tooltip
                  activator="parent"
                  location="bottom"
                  >Perfil</v-tooltip>

              </router-link>

            </li>
            <li class="nav-item">
              <router-link to="/livemap" class="nav-link" v-show=
                 loginstore.getAuthLog ><font-awesome-icon icon="fa-
                 solid fa-earth-europe" size="xl" />
                <v-tooltip
                  activator="parent"
                  location="bottom"
                  >Mapa</v-tooltip>

              </router-link>

            </li>
            <li class="nav-item">
              <router-link to="/config/user" class="nav-link" v-show
                 =loginstore.getAuthLog ><font-awesome-icon icon="fa
                 -solid fa-gear" size="xl" />
```

```
57                  <v-tooltip
58                    activator="parent"
59                    location="bottom"
60                    >Configuracion</v-tooltip>
61
62                </router-link>
63
64              </li>
65
66
67
68
69              <li class="nav-item">
70                <router-link to="/register" class="nav-link" v-show=!
                     loginstore.getAuthLog><font-awesome-icon icon="fa-
                     solid fa-user-plus" />
71                  <v-tooltip
72                    activator="parent"
73                    location="bottom"
74                    >Registrar</v-tooltip>
75                </router-link>
76              </li>
77              <li class="nav-item">
78                <router-link to="/login" class="nav-link" v-show=!
                     loginstore.getAuthLog><font-awesome-icon icon="fa-
                     solid fa-arrow-right-to-bracket" size="xl" />
79                  <v-tooltip
80                    activator="parent"
81                    location="bottom"
82                    >Login</v-tooltip>
83                </router-link>
84              </li>
85              <li class="nav-item">
86                <router-link to="/logout" class="nav-link" v-show=
                     loginstore.getAuthLog><font-awesome-icon icon="fa-
                     solid fa-arrow-right-from-bracket" size="xl"/>
87                  <v-tooltip
88                    activator="parent"
89                    location="bottom"
90                    >Salir</v-tooltip>
91
92                </router-link>
93              </li>
94
95            </ul>
96          </div>
97        </div>
98        </nav>
99        <section class="section">
100       <router-view/>
101       </section>
102
103
104     </div>
105 </template>
106
107 <script setup>
108 /* eslint-disable */
```

```
import { useLoginStore } from '@/store/loginstore';
const loginstore = useLoginStore();
let logState=loginstore.getAuthLog;
</script>

<style scoped>

#navBarHeader{
  margin:0 auto;

  background-color:lightskyblue;
  border-radius: 15px;
}

#headerTitle{

  font-size:x-large;
}

#usernameLabelNavBar{
  font-size: x-large;
}
</style>
```

Fichero LoginFormComp.vue.

```
<!-- Formulario Login -->
<template>
    <div class="container">
        <div class="logformbox">


        <form>
            <div class="form-group">
                <label for="inputUsername">Usuario</label>
                <input type="text" class="form-control" id="
                   inputUsername" placeholder="Usuario" v-model="
                   usernamelog">

            </div>
            <div class="form-group">
                <label for="inputPassword">Contrasena</label>
                <input type="password" class="form-control" id="
                   inputPassword" placeholder="Contrasena" v-model="
                   passwordlog">

            </div>
            <button type="submit" class="btn btn-primary mt-2"
               @click.prevent="logAuth" id="idlogbtn">Acceder</
               button>

        </form>
    </div>
    </div>
</template>

<script setup>
/* eslint-disable */
```

```
27      import {ref} from 'vue';
28      import { useRouter } from 'vue-router';
29      import AuthService from '../services/AuthService'
30      import { useLoginStore } from '@/store/loginstore';
31      const loginstore = useLoginStore();
32      import Swal from 'sweetalert2';
33
34      const router = useRouter()
35      let usernamelog=ref("");
36      let passwordlog=ref("");
37
38
39      /*  Comprobamos el acceso al sistema  */
40      const logAuth = async () =>{
41
42          const auth = new AuthService()
43          const loginSuccess = await auth.login(usernamelog.value,
                passwordlog.value)
44
45
46          if(loginSuccess){
47
48              loginstore.addToken(auth.getJwt());
49              loginstore.addUsername(usernamelog.value);
50              loginstore.addAuthLog(true);
51              router.push('/profile');
52              Swal.fire({
53                    position: 'center',
54                    icon: 'success',
55                    title: 'Acceso Correcto',
56                    showConfirmButton: false,
57                    timer: 1500
58                  });
59          }
60          else{
61              router.push('/login');
62              Swal.fire({
63                    position: 'center',
64                    icon: 'warning',
65                    title: auth.getError(),
66                    showConfirmButton: false,
67                    timer: 1500
68                  });
69          }
70      }
71
72
73
74  </script>
75
76  <style scoped>
77  @media(max-width: 1200px){
78      .logformbox{
79          width: 250px;
80      }
81  }
82
83  .logformbox{
```

```
84
85      border-radius: 5px;
86      padding: 10px;
87      margin: 0 auto;
88
89
90
91  }
92
93  #idlogbtn{
94      align-content: center;
95      position: relative;
96      padding: 5px;
97      width: 40%;
98      margin: 5px;
99      left: 30%;
100
101 }
102
103 </style>
```

Fichero MapBoxLiveMap.vue.

```
1   <!-- Mapa con todos los usuarios -->
2   <template>
3
4       <div id="map"></div>
5   </template>
6
7   <script setup>
8   /* Importamos librerias  */
9    /* eslint-disable */
10   import axios from 'axios';
11   import { onMounted } from 'vue';
12   import mapboxgl from 'mapbox-gl';
13   import { useLoginStore } from '@/store/loginstore';
14
15   const loginstore=useLoginStore()
16  // eslint-disable-next-line
17  let map=[];
18  let waypoint=[];
19  let markerMapO=[];
20  let markerMapD=[];
21  onMounted(async () => {
22      map = new mapboxgl.Map({
23      container: 'map', // container ID
24      style: 'mapbox://styles/mapbox/streets-v12', // style URL
25      center: [-3.7035825,40.4167047], // starting position [lng, lat]
26      zoom: 12, // starting zoom
27      });
28      await axios.get('http://127.0.0.1:8000/routes/routes/')
29      .then(response => waypoint=response.data);
30
31      waypoint.forEach((markerP, id)=>{
32          if(markerP.userIDowner==loginstore.getUsername){
33              map.setCenter([markerP.routeOriginLng,
                    markerP.routeOriginLat])
34          }
```

```
        let colorA='green'
        let colorB='red'
        if(markerP.routeType=="Pasajero"){
            colorA='blue'
            colorB='yellow'
        }
        markerMapO = new mapboxgl.Marker({color: colorA})
    .setLngLat([markerP.routeOriginLng,markerP.routeOriginLat])
    .setPopup(new mapboxgl.Popup().setHTML("<h5>"+markerP.routeType+
       "<h5>"))
    .addTo(map);
    markerMapD = new mapboxgl.Marker({color: colorB})
    .setLngLat([markerP.routeDestLng,markerP.routeDestLat])
    .setPopup(new mapboxgl.Popup().setHTML("<h5>"+markerP.routeType+
       "<h5>"))
    .addTo(map);
    })


})

</script>
<style scoped>
    #map{
        position: absolute;
        top:10%;
        bottom: 0;
        width: 100%;
    }
</style>
```

Fichero MapBoxUserMap.vue.

```
<!-- Mapa del usuario para crear rutas -->
<template>
    <div class="grid-layout">

    <div class="upperbox" v-if="!loginstore.getRoute">
        <button class="btn btn-outline-success" id="origincoordsbtn"
            @click="getOriginC" v-if="(loginstore.getDriver ||
           loginstore.getPassenger) && (!loginstore.getRoute)" >
           Origen</button>
     <button class="btn btn-outline-success" id="destcoordsbtn"
        @click="getDestinC" v-if="(loginstore.getDriver ||
        loginstore.getPassenger) && (!loginstore.getRoute)" >Destino
        </button>
     <button class="btn btn-outline-primary" id="guardarRoutesbtn"
        @click="saveRouteBTN" v-if="((loginstore.getDriver ||
        loginstore.getPassenger) && readyToSave) && (!
        loginstore.getRoute)" >Guardar</button>

    </div>
    <div class="midbox"  >



    <button class="btn btn-outline-primary" id="driversmapbtn"
       @click="driversmapBTN" v-if="loginstore.getPassenger">
```

```
        Conductores</button>
16      <button class="btn btn-outline-primary" id="paxmapbtn" @click="
            paxmapBTN" v-if="loginstore.getDriver">Pasajeros</button>
17      <button class="btn btn-outline-info" id="clearMapbtn" @click="
            clearMapBTN" v-if="(loginstore.getDriver ||
            loginstore.getPassenger) && ((showLabelDriverInfor) || (
            showPaxInfor)) ">Borrar</button>
18      <label id="labelDriverInfo" v-if="showLabelDriverInfor &&
            loginstore.getPassenger"> Usuario: <label id="boldUserLabel"
            >{{ nearUser.userIDowner }} </label> Salida: <label id="
            boldUserLabel">{{ nearUser.driverStartTime }}
19          </label> Vuelta: <label id="boldUserLabel">{{
                nearUser.driverEndTime }}</label>..
20          <font-awesome-icon icon="fa-solid fa-ban-smoking" size="2xl"
                v-if=!nearUser.driverSmoke />
21          <font-awesome-icon icon="fa-solid fa-smoking" size="2xl"  v-
                if=nearUser.driverSmoke />..
22          <label id="boldUserLabel" v-if="loginstore.getPassenger">{{
                nearUser.driverFreeSeats }} de {{ nearUser.driverSeats }}
                </label> plazas libres
23      </label>
24      <button type="button" class="btn btn-outline-success" @click="
            bookTrip" id="bookSeatDriverbtn" v-if="showLabelDriverInfor
            && !readyToBookbtn && loginstore.getPassenger" >Reservar
            Plaza</button>
25      <br>
26      <div class="collapse" :class="{'show':readyToBookbtn}">
27      <BookTripComp/>
28       </div>
29      </div>
30       <div class="lowerbox">
31          <div id="map"></div>
32       </div>
33
34
35
36  </div>
37  </template>
38
39  <script setup>
40  /* importamos librerias   */
41   /* eslint-disable */
42   import { onMounted, ref} from 'vue';
43   import axios from 'axios'
44   import { getCenter } from 'geolib';
45   import { useLoginStore } from '@/store/loginstore';
46   import mapboxgl from 'mapbox-gl';
47   import Swal from 'sweetalert2';
48
49  import BookTripComp from './BookTripComp.vue'
50  import { useTripStore } from '@/store/tripstore';
51
52  /* Definimos variables */
53  // eslint-disable-next-line
54  let centerM = [-3.7035825,40.4167047];
55  let userCenter = centerM;
56  let mapa=[];
57
```

```
/* Marcadores */
let markerO=[];
let markerD=[];
let marker=[];
let routeM="";
let distanceR="";
let readyToSave=ref(false);
let driversNear=[];
let paxNear=[];
let markerMapOD=[];
let markerMapDD=[];
let markerMapOP=[];
let markerMapDP=[];
/* Informacion usuarios cercanos */
let nearUser=ref([]);
let showLabelDriverInfor=ref(false);
let showPaxInfor=ref(false);

const loginstore=useLoginStore();
const tripstore=useTripStore();

let readyToBookbtn=ref(false);

let fullPath=[]
let tempCoords={
    lat:'',
    lng:''
}
let rData={

}

let temppath=[]


/*Funcion al cargar la vista */
onMounted(async () => {
    try{
        mapa = new mapboxgl.Map({
        container: 'map', // container ID
        style: 'mapbox://styles/mapbox/streets-v12', // style URL
        center: centerM, // starting position [lng, lat]
        zoom: 12, // starting zoom
        });

        marker = new mapboxgl.Marker()
        .setLngLat(centerM)
        .addTo(mapa);
        mapa.on('drag', async function() {
        centerM=mapa.getCenter();

        marker.setLngLat(centerM);




});
    }catch(error){
```

```
116
117     }
118
119
120 const resroute = await fetch('http://127.0.0.1:8000/routes/
        routenodes/',{
121         method: 'POST',
122                 headers: {
123                     'Accept':'application/json',
124                     'Content-Type': 'application/json'
125                 },
126                 body: JSON.stringify({
127                     userIDowner:loginstore.getUsername
128                 })
129         })
130
131
132         const responseroute = await resroute.json()
133
134         if(responseroute.routeCreated!=null){
135
136             rData=responseroute
137
138             loginstore.addRoute(true)
139
140
141             Array.from(rData.routePath).forEach((value)=>{
142                 temppath.push(value)
143             })
144
145             markerO = new mapboxgl.Marker({color: 'green'})
146     .setLngLat([rData.routeOriginLng, rData.routeOriginLat])
147     .addTo(mapa);
148     markerD = new mapboxgl.Marker({color: 'red'})
149     .setLngLat([rData.routeDestLng, rData.routeDestLat])
150     .addTo(mapa);
151
152     let fixCenter=getCenter([
153             {latitude: rData.routeOriginLat, longitude:
                rData.routeOriginLng},
154             {latitude:rData.routeDestLat,
                longitude:rData.routeDestLng}
155         ]);
156
157         mapa.setCenter([fixCenter.longitude, fixCenter.latitude])
158         userCenter=mapa.getCenter();
159
160         const fitZoom = [[rData.routeOriginLng, rData.routeOriginLat
            ],[rData.routeDestLng, rData.routeDestLat]];
161         mapa.fitBounds(fitZoom,{
162             padding: {top: 55, bottom:25, left: 15, right: 5}
163         })
164
165
166
167
168        /*  mapa.setZoom(mapa.getZoom()-1); */
169     const geojson = {
```

```
        type: 'Feature',
        properties:{},
        geometry:{
            type:'LineString',

            coordinates:temppath
        }
    };
        mapa.addLayer({
        id: 'route',
        type: 'line',
        source: {
            type: 'geojson',
            data: geojson
        },
        layout: {
            'line-join': 'round',
            'line-cap': 'round'
        },
        paint: {
            'line-color': '#3887be',
            'line-width': 5,
            'line-opacity': 0.75
        }})
        }
        else{

            loginstore.addRoute(false)

        }



})

/*Funcion reservar plaza */
const bookTrip=async ()=>{


    tripstore.addTripStoreData(nearUser.value.userIDowner)
    readyToBookbtn.value=true;


}

/* Funcion limpiar mapa */
const clearMapBTN=()=>{


    let counter=mapa._markers.length-1
    while(mapa._markers.length>3){
        mapa._markers[counter].remove()
        counter--;
    }
    nearUser.value=[]
    showLabelDriverInfor.value=false;
    showPaxInfor.value=false;
    readyToBookbtn.value=false;
```

```
}

/* Funcion mostrar conductores */
const driversmapBTN=async()=>{

    /* nearestUsers("Conductor") */
    nearestUsers(loginstore.getUsername)
    Swal.fire({
        position: 'center',
        icon: 'info',
        title: 'Mostrando conductores cercanos disponibles',
        showConfirmButton: false,
        timer: 2500
    })

}

/* Funcion mostrar pasajeros */
const paxmapBTN=()=>{
    showPaxInfor.value=true
    nearestPax(loginstore.getUsername)
    Swal.fire({
        position: 'center',
        icon: 'info',
        title: 'Mostrando pasajeros cercanos disponibles',
        showConfirmButton: false,
        timer: 2500
    })
}

/* Funcion pasajeros mas cercanos */
const nearestPax=async(routeType)=>{
    try{


        await axios
                /* .post('http://127.0.0.1:8000/routes/nearest/',{
                    routeType: routeType
                }) */
                .post('http://127.0.0.1:8000/nodes/node/',{
                    userIDowner: routeType
                })
                .then((response) =>{
                    //if(routeType=="Conductor"){
                        driversNear=response.data;
                        driversNear.forEach((markerP, id)=>{

                        const popup= new mapboxgl.Popup({offset:25})
                            .setHTML(
                            `
                                ${markerP.nodeUser}


                            `
                        ).on('open', function(){

                            markerClick(markerP.nodeUser, "Pasajero"
                                , markerP.nodeDestLng,
```

```
                                markerP.nodeDestLat)
284                         })
285
286
287
288                         markerMapOD = new mapboxgl.Marker({color:'
                                blue'})
289                     .setLngLat([markerP.nodeOriginLng,
                            markerP.nodeOriginLat])
290                     .setPopup(popup)
291                     .addTo(mapa);
292
293
294                     /* markerMapDD = new mapboxgl.Marker({color: '
                            yellow'})
295                     .setLngLat([markerP.routeDestLng,
                            markerP.routeDestLat])
296                     .setPopup(new mapboxgl.Popup().setHTML("<h5>"+
                            markerP.routeType+"<h5><br><h5>"+
                            markerP.userIDowner+"<h5>"))
297                     .addTo(mapa); */
298                     })
299                     //}
300
301
302                 });
303
304         mapa.setCenter([rData.routeOriginLng, rData.routeOriginLat])
305
306
307         //const responseroute = await resroute.json()
308
309         /* if(routeType=="Conductor"){
310             console.log("veamos conductores")
311             driversNear=responseroute.data
312         }
313
314         console.log(driversNear) */
315     }catch(error){
316
317     }
318 }
319
320 /* Funcion conductores mas cercanos */
321 const nearestUsers=async(routeType)=>{
322     try{
323         await axios
324                 .post('http://127.0.0.1:8000/nodes/node/',{
325                     userIDowner: routeType
326                 })
327                 .then((response) =>{
328                         driversNear=response.data;
329                         driversNear.forEach((markerP, id)=>{
330
331                         const popup= new mapboxgl.Popup({offset:25})
                                .setHTML(
                                 `
332
333                                     ${markerP.nodeUser}
```

```
                                `
                            ).on('open', function(){
                                markerClick(markerP.nodeUser, "Conductor
                                    ", markerP.nodeDestLng,
                                    markerP.nodeDestLat)
                            })
                            markerMapOD = new mapboxgl.Marker({color:'
                                blue'})
                        .setLngLat([markerP.nodeOriginLng,
                            markerP.nodeOriginLat])
                        .setPopup(popup)
                        .addTo(mapa);
                        })
                });
        mapa.setCenter([rData.routeOriginLng, rData.routeOriginLat])


        //const responseroute = await resroute.json()

        /* if(routeType=="Conductor"){
            console.log("veamos conductores")
            driversNear=responseroute.data
        }

        console.log(driversNear) */
    }catch(error){

    }
}

/* El usuario ha hecho click en un marcador */
const markerClick=async (id, type, lng, lat)=>{
    if(type=="Conductor"){
        const resroute = await fetch('http://127.0.0.1:8000/drivers/
            detail/',{
        method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    userIDowner:id
                })
        })
        markerMapDD = new mapboxgl.Marker({color: 'yellow'})
                    .setLngLat([lng,lat])
                    .setPopup(new mapboxgl.Popup().setHTML(id+"<br>"
                        +type))
                    .addTo(mapa);
        const fitZoom = [[rData.routeOriginLng, rData.routeOriginLat
            ],[lng, lat]];
        mapa.fitBounds(fitZoom,{
            padding: {top: 25, bottom:25, left: 15, right: 5}
        })

        /* mapa.setCenter(userCenter)
        mapa.setZoom(mapa.getZoom()-1) */
        const responseroute = await resroute.json()
```

```
385
386         nearUser.value=responseroute
387
388         showLabelDriverInfor.value=true;
389     }
390
391
392 }
393
394 /* Establecer punto de origen */
395 const getOriginC=()=>{
396     if(!loginstore.getRoute){
397         markerO = new mapboxgl.Marker({color: 'green'})
398     .setLngLat(mapa.getCenter())
399     .addTo(mapa);
400     }
401     else{
402         Swal.fire({
403         position: 'center',
404         icon: 'warning',
405         title: 'Ya hay un origen: Vaya a configuracion para
                cambiarlo',
406         showConfirmButton: false,
407         timer: 1500
408     })}
409
410 }
411
412 /* Guardar ruta */
413 const saveRouteBTN=async ()=>{
414
415     if(!loginstore.getRoute &&(loginstore.getDriver ||
            loginstore.getPassenger)){
416         let rType=''
417         if(loginstore.getDriver){
418             rType='Conductor'
419         }
420         if(loginstore.getPassenger){
421             rType='Pasajero'
422         }
423         let routeO=
424         {
425         "userIDowner": loginstore.getUsername,
426         "routeOriginLat": markerO.getLngLat().lat,
427         "routeOriginLng": markerO.getLngLat().lng,
428         "routeDestLat": markerD.getLngLat().lat,
429         "routeDestLng": markerD.getLngLat().lng,
430         "routeType": rType,
431         "routePath":fullPath}
432
433         let nodesTemp=
434         {
435             "userIDowner": loginstore.getUsername,
436             "nodeType": rType
437         }
438
439         await axios
440             .post('http://127.0.0.1:8000/routes/addroute/',
```

```
441             routeO)
442             .then((response) =>{
443 
444                         console.log(response);
445                     });
446                     //useUserS.getUserData.userHasRoute=true;
447                 //const userupdate = useUserS.getUserData.username;
448                 //await axios.put('http://127.0.0.1:8000/api/
                        updateuser/'+userupdate+'/',useUserS.getUserData)
                        ;
449             await axios
450                 .post('http://127.0.0.1:8000/nodes/addnode/',
451                 nodesTemp)
452                 .then((response)=>{
453                     console.log(response)
454                 });
455 
456                 Swal.fire({
457             position: 'center',
458             icon: 'success',
459             title: 'Ruta Anadida',
460             showConfirmButton: false,
461             timer: 1500
462         });
463         loginstore.addRoute(true)
464 
465     }
466     else{
467         if(loginstore.getDriver || loginstore.getPassenger){
468             Swal.fire({
469             position: 'center',
470             icon: 'warning',
471             title: 'Ya hay una ruta guardada',
472             showConfirmButton: false,
473             timer: 1500
474         })
475         }
476         else{
477             Swal.fire({
478             position: 'center',
479             icon: 'warning',
480             title: 'Debe ser conductor o pasajero antes',
481             showConfirmButton: false,
482             timer: 1500
483         })
484         }
485 
486     }
487 
488 
489 
490 
491 }
492 
493 
494 /* Establecer punto de destino */
495 const getDestinC=async ()=>{
496     try{
```

```
497         if(!loginstore.getRoute){
498
499
500         let noOrigin=markerO.getLngLat().lat
501
502         markerD = new mapboxgl.Marker({color: 'red'})
503         .setLngLat(mapa.getCenter())
504         .addTo(mapa);
505         marker.remove();
506         let fixCenter=getCenter([
507             {latitude: markerO.getLngLat().lat, longitude:
                    markerO.getLngLat().lng},
508             {latitude:markerD.getLngLat().lat, longitude:
                    markerD.getLngLat().lng}
509         ]);
510
511         mapa.setCenter([fixCenter.longitude, fixCenter.latitude])
512         mapa.setZoom(mapa.getZoom()-0.75);
513         if(!loginstore.getPassenger&&loginstore.getDriver){
514         calculateRoute();
515         }
516         }else{
517         Swal.fire({
518         position: 'center',
519         icon: 'warning',
520         title: 'Ya hay un destino: Vaya a configuracion para
                cambiarlo',
521         showConfirmButton: false,
522         timer: 1500
523         })}
524         readyToSave.value=true
525     }catch(error){
526
527     }
528
529
530 }
531
532 /* Calcular el recorrido */
533 const calculateRoute=async ()=>{
534     routeM="https://api.mapbox.com/directions/v5/mapbox/driving/"+
535     markerO.getLngLat().lng+","+markerO.getLngLat().lat+";"+
536     markerD.getLngLat().lng+","+markerD.getLngLat().lat+
537     "?geometries=geojson&access_token="+mapboxgl.accessToken;
538     const findRoute = await fetch(
539         routeM,
540         {method: 'GET'}
541     );
542
543     const json = await findRoute.json();
544     console.log(json)
545     const data = json.routes[0];
546     console.log(data)
547
548     const route = data.geometry.coordinates;
549     console.log(route)
550
551     const geojson = {
```

```
        type: 'Feature',
        properties:{},
        geometry:{
            type:'LineString',

            coordinates:route
        }
    };
    if (mapa.getSource('route')) {
    mapa.getSource('route').setData(geojson);
    }
    else {
    mapa.addLayer({
      id: 'route',
      type: 'line',
      source: {
        type: 'geojson',
        data: geojson
      },
      layout: {
        'line-join': 'round',
        'line-cap': 'round'
      },
      paint: {
        'line-color': '#3887be',
        'line-width': 5,
        'line-opacity': 0.75
      }
    });

    const fitZoom = [[marker0.getLngLat().lng, marker0.getLngLat().
       lat],[markerD.getLngLat().lng, markerD.getLngLat().lat]];
        mapa.fitBounds(fitZoom,{
            padding: {top: 35, bottom:25, left: 15, right: 5}
        })

    Array.from(geojson.geometry.coordinates).forEach((value)=>{

        fullPath.push([
            value[0],value[1]
        ]

        )
    })

    }
}









</script>
<style scoped>
```

```
@media(max-width: 450px){
    .grid-layout{
        width:350px;

    }
}

.grid-layout{
    box-sizing: border-box;
    margin: 0 auto;
    width: 100%;
    height: 90vh;
    grid-template-rows: 60% 7% 25%;
    justify-content: center;
    display: grid;

}

.upperbox{
    grid-row: 2;
    grid-template-columns: 100% 100% 100%;
    margin: 0 auto;
    width: 300%;

    border: 1px solid lightblue;
    border-radius: 15px;
    justify-content: center;
    padding: 5px;
}

#origincoordsbtn{
    grid-column: 1;
    position: relative;
    padding: 5px;
    left:25px;

    }

#destcoordsbtn{
    grid-column: 2;
    position: relative;
    padding: 5px;
    left:75px;
}

#guardarRoutesbtn{
    position: relative;
    padding: 5px;
    grid-column: 3;

    left:125px;
}

.midbox{
    grid-row:3;
    border: 1px solid lightblue;
    border-radius: 15px;
    padding: 5px;
```

```
    width: 350%;
    grid-template-columns: 100%;
    grid-template-rows: 15% 15%;
    height:175px;

}

#driversmapbtn{

    position: relative;
    padding: 5px;
    left:25px;
}

#paxmapbtn{
    position: relative;
    padding: 5px;
    left:25px;
}

#labelDriverInfo{
    position: relative;
    border: 1px solid lightsalmon;
    border-radius: 15px;
    padding: 5px;
    width: 80%;
    top:50px;
    left:25px;
}

#boldUserLabel{

    font-weight: bold;
    font-size: x-large;
}

#bookSeatDriverbtn{
    position: relative;
    padding: 5px;
    right: 150px;
    top:50px;
}

#clearMapbtn{
    position: absolute;
    padding: 5px;
    left:300px;

}

.lowerbox{
    grid-row:1;
    width: 290%;
    height: 500px;


}

```

```
725
726
727 #map{
728         position: relative;
729
730         height: 75%;
731         top:25px;
732
733     }
734
735
736 </style>
```

Fichero MapListMapComp.vue.

```
1  <template>
2      <div id="map"></div>
3  </template>
4
5  <script setup>
6  /* eslint-disable */
7  import mapboxgl from 'mapbox-gl';
8  import { onMounted, ref} from 'vue';
9  import TripsService from '../services/TripsService'
10 import { useLoginStore } from '@/store/loginstore';
11 import { useTripStore } from '@/store/tripstore';
12 const tripsActions= new TripsService()
13
14 const loginstore = useLoginStore();
15 const tripstore = useTripStore();
16 let centerM = [-3.7035825,40.4167047];
17 let mapa=[];
18 let marker=[];
19 let markerDO=[];
20 let markerDD=[];
21 let markerPO=[];
22 let markerPD=[];
23
24
25
26
27 onMounted(async () => {
28
29     mapa = new mapboxgl.Map({
30     container: 'map', // container ID
31     style: 'mapbox://styles/mapbox/streets-v12', // style URL
32     center: centerM, // starting position [lng, lat]
33     zoom: 12, // starting zoom
34     });
35
36     marker = await tripsActions.showMapInfoPax(tripstore.getTripPax,
           loginstore.getUsername)
37
38
39
40     markerDO = new mapboxgl.Marker({color:'green'})
41     .setLngLat([marker[0].routeOriginLng, marker[0].routeOriginLat])
42     .setPopup(new mapboxgl.Popup().setHTML("mi salida"))
```

```
    .addTo(mapa)
    markerPD = new mapboxgl.Marker({color:'blue'})
    .setLngLat([marker[1].routeOriginLng, marker[1].routeOriginLat])
    .setPopup(new mapboxgl.Popup().setHTML("salida pasajero"))
    .addTo(mapa)
    markerDO = new mapboxgl.Marker({color:'red'})
    .setLngLat([marker[0].routeDestLng, marker[0].routeDestLat])
    .setPopup(new mapboxgl.Popup().setHTML("mi vuelta"))
    .addTo(mapa)
    markerPD = new mapboxgl.Marker({color:'yellow'})
    .setLngLat([marker[1].routeDestLng, marker[1].routeDestLat])
    .setPopup(new mapboxgl.Popup().setHTML("vuelta pasajero"))
    .addTo(mapa)

    mapa.setCenter([marker[0].routeOriginLng, marker[0].
       routeOriginLat])


    /* marker = new mapboxgl.Marker()
    .setLngLat(centerM)
    .addTo(mapa);
    mapa.on('drag', async function() {
  centerM=mapa.getCenter();

    marker.setLngLat(centerM); */




})

</script>

<style scoped>
#map{
    position: relative;
    display: flex;
    height: 100%;
    width: 100%;
}

</style>
```

Fichero MessagesBoxComp.vue.

```
<template>
<div class="container">
<MessagesSentComp/>
<MessagesRecComp/>
</div>
</template>

<script setup>
import MessagesSentComp from './MessagesSentComp.vue';
import MessagesRecComp from './MessagesRecComp.vue';
/* Componente para los mensajes */

</script>
```

```
<style scoped>
.container{
    display: flex;
    flex-direction: column;
    width: 50%;
    justify-content: center;
}
</style>
```

Fichero MessagesComp.vue.

```
<template>
    <div class="container">
        <MessagesRecComp/>
    </div>
</template>

<script setup>
import MessagesRecComp from './MessagesRecComp.vue';
/* Contenedor de los mensajes */
</script>
```

Fichero MessagesRecComp.vue.

```
<!-- Mensajes recibidos -->
<template>
<div class="card card-header">
    <label data-bs-toggle="collapse" href="#recmsglist">Mensajes
       Recibidos</label>

</div>
<div class="collapse" id="recmsglist">
    <div class="card card-body">
        <button type=button class="btn btn-outline-primary" id="
           btnRecMsg" @click="loadMsgRec" >Cargar</button>
        <table class="table">
        <thead>
            <tr>
                <th scope="col">De</th>
                <th scope="col">Mensaje</th>
                <th colspan=""></th>
                <th colspan=""></th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="msg in allMsgRec.slice().reverse()" :key="
               msg.id">
                <td>{{ msg.fromMessage }}</td>
                <td v-if="msg.messageRead"> {{ msg.bodyMessage }}</
                   td>
                <td v-if="!msg.messageRead">
                    <a class="btn btn-outline-success" @click="
                       readMsgBtn(msg)" >Leer</a>
                </td>
                <td v-if="msg.messageRead">
                    <a class="btn btn-outline-info" v-if="!
                       msg.messageAvoidReply"  @click="replyMsgBtn(
                       msg)">Responder</a>
```

```
28
29                      <a class="btn btn-outline-danger" @click="
                            deleteMsgBtn(msg)">Borrar</a>
30                  </td>
31              </tr>
32          </tbody>
33      </table>
34  </div>
35  </div>
36
37  </template>
38
39  <script setup>
40  /* eslint-disable */
41  import {ref, onMounted} from 'vue';
42  import { useLoginStore } from '@/store/loginstore';
43  import { useMessageStore} from '@/store/messagesstore';
44  import MessagesService from '@/services/MessagesService';
45  import axios from 'axios';
46  import Swal from 'sweetalert2';
47  const loginstore = useLoginStore();
48  const messagesStore = useMessageStore();
49  const msgService = new MessagesService()
50
51  const avoitReply="Reservas"
52
53  //let allMsgRec=messagesStore.getMessagesData
54  let allMsgRec=ref([])
55
56  onMounted(async ()=>{
57      loadMsgRec()
58  })
59
60  /* Recargar mensajes */
61  const reloadMsgRec = async()=>{
62      loadMsgRec()
63  }
64
65  /* responder mensajes */
66  const replyMsgBtn=async(msg)=>{
67      const { value: replyMsgTemp} = await Swal.fire({
68          title: 'Responder a '+msg.fromMessage,
69          input: 'textarea',
70
71          showCancelButton: true
72      })
73
74      if(replyMsgTemp){
75
76          let msgData=ref({
77          fromMessage:msg.toMessage,
78          toMessage:msg.fromMessage,
79          bodyMessage:replyMsgTemp
80          })
81
82          const replymsgaction= await msgService.sendMessageTo(msgData
                )
83          console.log(replymsgaction)
```

```
84          if(replymsgaction){
85              Swal.fire({
86                          position: 'center',
87                          icon: 'success',
88                          title: 'Mensaje enviado',
89                          showConfirmButton: false,
90                          timer: 1500
91                          });
92          } else{
93              Swal.fire({
94                          position: 'center',
95                          icon: 'warning',
96                          title: 'Error al enviar mensaje',
97                          showConfirmButton: false,
98                          timer: 1500
99                          });
100         }
101     }else{
102
103     }
104 }
105
106 /*Borrar mensajes */
107 const deleteMsgBtn=async(msg)=>{
108     Swal.fire({
109     title: 'Borrar Mensaje?',
110     showDenyButton: true,
111     showCancelButton: false,
112     confirmButtonText: 'Si',
113     denyButtonText: 'No',
114     }).then(async (result) => {
115     /* Read more about isConfirmed, isDenied below */
116     if (result.isConfirmed) {
117         msg.messageShowTo="NO"
118         const updateMSG = await msgService.updateMsg(msg);
119         if(updateMSG){
120             Swal.fire({
121                     position: 'center',
122                     icon: 'warning',
123                     title: 'Mensaje borrado',
124                     showConfirmButton: false,
125                     timer: 1500
126                     });
127             await loadMsgRec();
128         }
129
130     } else if (result.isDenied) {
131
132     }
133     })
134
135
136 }
137
138 /*Marcar leido un mensaje */
139 const readMsgBtn=async (msg)=>{
140     msg.messageRead=true
141     const readMSG= await msgService.updateMsg(msg)
```

```
}

/* Cargar mensajes recibidos */
const loadMsgRec = async ()=>{
    allMsgRec.value= await msgService.loadMsgRec(
       loginstore.getUsername)
    setTimeout(()=>{
        reloadMsgRec()
            }, 10000);

}
</script>

<style scoped>
#btnRecMsg{
    align-content: left;
    position: relative;
    padding: 5px;
    width: 90px;
    margin: 5px;
}



</style>
```

Fichero MessagesSentComp.vue.

```
<!-- Mensajes enviados -->
<template>
    <div class="mainsendbox">
    <div class="card card-header">
        <label data-bs-toggle="collapse" href="#sentmsglist">Enviar
           mensaje</label>
    </div>
    <div class="sendboxinfo">
    <div class="collapse" id="sentmsglist">

            <form>
                <div class="container">
                    <div class="form-group">
                        <label>Destinatario:</label><input type="
                           text" id="destMsg" name="destMsg"
                        placeholder="escriba el usuario" v-model="
                           messageData.toMessage">
                    </div>
                    <div class="form-group">
                        <label>Mensaje:... </label>
                        <br>
                        <textarea id="msgMsg" name="msgMsg" rows="2"
                             cols="30" placeholder="escriba el
                           mensaje"
                        v-model="messageData.bodyMessage"></textarea
                           >
                    </div>
                        <br>
                        <button class="btn btn-outline-primary"
                           @click.prevent="sendMessageto" id="
```

```
                            btnSendMSG">Enviar</button>

                </div>

            </form>
        </div>
            <div class="tablemsglist">
                <label>Mensajes enviados</label>
            <table class="table">
        <thead>
            <tr>
                <th scope="col">A</th>
                <th scope="col">Mensaje</th>
                <th colspan=""></th>
                <th colspan=""></th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="msg in allMsgSen.slice().reverse()" :key="
                msg.id">
                <td>{{ msg.toMessage }}</td>
                <td>{{ msg.bodyMessage }}</td>
                <td>
                    <a class="btn btn-danger" @click="deleteMsgBtn(
                        msg)">Borrar</a>
                </td>
            </tr>
        </tbody>
    </table>
        </div>
    </div>
</div>
</template>

<script setup>
/* eslint-disable */
import {ref, onMounted} from 'vue';
import MessagesService from '@/services/MessagesService';
import { useLoginStore } from '@/store/loginstore';
import Swal from 'sweetalert2';
import axios from 'axios'

let allMsgSen=ref([])

const loginstore = useLoginStore();
const msgService = new MessagesService()

let messageData=ref({
    fromMessage:loginstore.getUsername,
    toMessage:'',
    bodyMessage:''
})

onMounted(async ()=>{
    loadMsgSen()
})

/* Morrar mensaje enviado */
```

```
79  const deleteMsgBtn=async(msg)=>{
80      Swal.fire({
81      title: 'Borrar Mensaje?',
82      showDenyButton: true,
83      showCancelButton: false,
84      confirmButtonText: 'Si',
85      denyButtonText: 'No',
86      }).then(async (result) => {
87      /* Read more about isConfirmed, isDenied below */
88      if (result.isConfirmed) {
89          msg.messageShowFrom="NO"
90          const updateMSG = await msgService.updateMsg(msg);
91          if(updateMSG){
92              Swal.fire({
93                      position: 'center',
94                      icon: 'warning',
95                      title: 'Mensaje borrado',
96                      showConfirmButton: false,
97                      timer: 1500
98                      });
99              await loadMsgSen();
100         }
101
102     } else if (result.isDenied) {
103
104     }
105     })
106
107
108 }
109
110 /* recargar mensajes */
111 const reloadMsgSen = async()=>{
112     loadMsgSen()
113 }
114
115 /* Cargar mensajes enviados */
116 const loadMsgSen = async ()=>{
117     allMsgSen.value= await msgService.loadMsgSen(
        loginstore.getUsername)
118     setTimeout(()=>{
119         reloadMsgSen()
120     }, 10000);
121
122 }
123
124 /* Enviar mensaje */
125 const sendMessageto=async()=>{
126
127     try{
128
129         const response=msgService.sendMessageToUsers(messageData)
130         if(response){
131             Swal.fire({
132                   position: 'center',
133                   icon: 'success',
134                   title: 'Mensaje enviado',
135                   showConfirmButton: false,
```

```
                    timer: 1500
                });
        }
        messageData.value.toMessage=''
        messageData.value.bodyMessage=''
        loadMsgSen();

    } catch(error){
        Swal.fire({
                    position: 'center',
                    icon: 'warning',
                    title: 'Error al enviar mensaje',
                    showConfirmButton: false,
                    timer: 1500
                });
    }
}
</script>
<style scoped>
.mainsendbox{
    display: flex;

    flex-direction: column;
    width: 100%;

}

.container{
    border: 1px solid lightcyan;
    border-radius: 15px;
    width: 60%;
}

#destMsg{
    border:1px solid darkblue;
    position: relative;
    border-radius: 5px;
    text-align: left;
    padding: 5px;
    left:25px
}

#msgMsg{
    border: 1px solid darkblue;
    position: relative;
    border-radius: 5px;
    text-align: left;
    padding: 5px;

    width: 83%;
}


#btnSendMSG{
    width: 100px;
}

</style>
```

Fichero PassChangeConfComp.vue.

```
<!-- Formulario de configuracion de los pasajeros -->
<template>
    <div class="container">


    <div class="passSetConfigBox">
    <form>
        <div class="flex-item item-12">
            <div class="form-group">
                <label>Ciudad</label><input type="text" id="paxCity"
                    name="ppaxCity" class="form-control"
                    placeholder="Ciudad" v-model="
                       paxData.passengerCity">
            </div>
        </div>
        <div class="flex-item item-13">
            <div class="form-group">
                <label>Telefono</label><input type="text" id="
                   paxPhone" name="ppaxPhone" class="form-control"
                    placeholder="Telefono" v-model="
                       paxData.passengerPhone">
            </div></div>
    <div class="flex-item item-14">
        <div class="form-group">
            <label>Salida...</label><input type="time" id="
               paxStartTime" name="paaxStartTime"
                v-model="paxData.passengerStartTime">
        </div></div>
        <div class="flex-item item-15">
                <div class="form-group">
                    <label>Vuelta... </label><input type="time" id="
                       paxEndTime" name="ppaxEndTime"
                        v-model="paxData.passengerEndTime">
                </div></div>
                <div class="flex-item item-16">
        <div class="form-group">
            <label>Maxima espera {{ paxData.passengerWaitLimitStart
               }} minutos</label><input type="range"
                class="form-range" min="0" max="20" id="
                   paxWaitLimitStart" name="ppaxWaitLimitStart"
                v-model="paxData.passengerWaitLimitStart">
        </div></div>
        <div class="flex-item item-17">
        <div class="form-group">
            <label>Maxima distancia {{
               paxData.passengerMaxDistanceAllowed }} km</label><
               input type="range"
                class="form-range" min="0" max="5" id="
                   paxMaxDistanceAllowed" name="
                   ppaxMaxDistanceAllowed"
                v-model="paxData.passengerMaxDistanceAllowed">
        </div></div>
        <div class="flex-item item-18">
                <div class="form-group">
                <div class="chooseSmokingGroup">


```

```
                    <div class="nosomokingGroup">

                        <font-awesome-icon icon="fa-solid fa-ban-smoking" size="2xl" />
                        <input class="form-check-input" type="radio" name="radioNoSmoking" id="radioNoSmoking"
                            :checked="noSmokingSelection" v-model="noSmokingSelection">
                    </div>
                    <div class="smokingGroup">

                        <font-awesome-icon icon="fa-solid fa-smoking" size="2xl" />
                        <input class="form-check-input" type="radio" name="radioNoSmoking" id="radioSmoking"
                            :checked="!noSmokingSelection" v-model="noSmokingSelection">
                    </div>
                </div></div></div>
    </form>
    <div class="buttonbox">
            <button class="btn btn-outline-primary" @click="updatePaxData">Guardar</button>
        </div>
    </div>
</div>
</template>

<script setup>
    /* eslint-disable */
import {ref, onMounted} from 'vue';
import { useLoginStore } from '@/store/loginstore';
import axios from 'axios';
import Swal from 'sweetalert2';
const loginstore = useLoginStore();
let idNoSmoking="radioNoSmoking"
let idSmoking="radioSmoking"
let noSmokingSelection=true
let smokingSelection=false

/* Datos de los pasajeros */
let paxData=ref({
    passengerCity:'',
    passengerPhone:'',
    passengerStartTime:'08:00',
    passengerEndTime:'16:00',
    passengerWaitLimitStart:0,
    passengerMaxDistanceAllowed:0,
    passengerSmoke:smokingSelection
})

onMounted(async ()=>{
    const res = await fetch('http://127.0.0.1:8000/pax/detail/',{
                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
```

```
                },
                body: JSON.stringify({
                    "userIDowner": loginstore.getUsername
                })
            })
            const response = await res.json()
            paxData.value=response
    }

    );

    /* Modificar datos pasajeros */
    const updatePaxData=async ()=>{
    Swal.fire({
        title: 'Actualizar datos?',
        showDenyButton: true,
        showCancelButton: false,
        confirmButtonText: 'Si',
        denyButtonText: 'No',
        }).then(async (result) => {

        if (result.isConfirmed) {

            await axios
            .put('http://127.0.0.1:8000/pax/update/',paxData.value)
            .then((response) =>{
                console.log(response);
            });
            Swal.fire({
                        position: 'center',
                        icon: 'success',
                        title: 'Datos actualizados',
                        showConfirmButton: false,
                        timer: 1500
                        });
        }else if (result.isDenied) {

        }})}


</script>

<style scoped>
    .passSetConfigBox{
        padding: 10px;
        border: 1px solid lightcyan;
        border-radius: 10px;
        width: 70%;

    }

    .flex-item{
        width: 250px;
    }

    .smokingGroup{
    position: relative;

```

```
    left:100px;
    top:-35px;
}
    .buttonbox{
        padding: 10px;
    }
    .flex-item item-12{
        position: absolute;
        right:350px;
    }
</style>
```

Fichero PassengerConfigComp.vue.

```
<!-- Componente de perfil y formulario alta de pasajero -->
<template>
<div class="boxpassenger">

    <div class="collapse" :class="{'show':loginstore.getPassenger}">

        <PassengerLoadComp v-bind:pData="pData"/>
    </div>
    <div class="collapse" :class="{'show':!loginstore.getPassenger}"
        v-if="!loginstore.getDriver">

        <br>
        <div class="passengerSelection">


        <input class="form-check-input" type="checkbox" value="" id=
           "flexCheckDefaultP" v-model="isPassenger">
        <div class="iconPassenger">
        <label class="form-check-label" for="flexCheckDefaultP" id="
           iconLabelPassenger">


         <font-awesome-icon icon="fa-solid fa-person-circle-plus"
            size="xl" v-model="isPassenger"/><v-tooltip
                  activator="parent"
                  location="right"
                  >Soy Pasajero</v-tooltip>
      </label>
    </div>
    </div>
      <div class="collapse" :class="{'show':isPassenger}">
        <PassengerFormComp/>
        </div>
    </div>
</div>
</template>

<script setup>
/* eslint-disable */
import { useLoginStore } from '@/store/loginstore';
import { usePaxStore } from '@/store/paxstore';
import {ref} from 'vue';
import { onMounted } from 'vue';
```

```
41 import PassengerLoadComp from './PassengerLoadComp.vue';
42 import PassengerFormComp from './PassengerFormComp.vue'
43
44 const loginstore = useLoginStore();
45 const paxstore = usePaxStore();
46 let isPassenger=ref(false)
47 let pData=paxstore.getPaxData
48
49 /* Funcion que carga detalles del pasajero */
50 onMounted(async ()=>{
51     try{
52
53         const res = await fetch('http://127.0.0.1:8000/pax/detail/'
               ,{
54         method: 'POST',
55                 headers: {
56                     'Accept':'application/json',
57                     'Content-Type': 'application/json'
58                 },
59                 body: JSON.stringify({
60                     userIDowner:loginstore.username
61                 })
62         })
63
64         const response = await res.json()
65
66         if(response.passengerCreated!=null){
67
68             pData=response
69             paxstore.addPaxStoreData(response)
70             loginstore.addPassenger(true)
71
72         }
73         else{
74
75             loginstore.addPassenger(false)
76
77         }
78
79
80     }catch(error){
81         console.log(error)
82     }
83
84 })
85
86 </script>
87
88 <style scoped>
89
90 .passengerSelection{
91     grid-template-columns: auto auto;
92     box-sizing: border-box;
93     margin: 0 auto;
94     border: 1px solid lightblue;
95     border-radius: 15px;
96     width: 300px;
97     padding: 5px;
```

```
}

#flexCheckDefaultP{
    position: relative;
    grid-column: 1;

    padding: 5px;
    left:25px;
}

#iconLabelPassenger{
    position: relative;
    left:55px;
    top:-30px;
    grid-column: 2;

    position: relative;
}

</style>
```

Fichero PassengerFormComp.vue.

```
<!-- Formulario de registro de pasajeros -->
<template>
    <div class="card card-body">
        <form >


                <div class="container">
                <div class="form-group">
                            <label>Ciudad</label><input type="text
                                " id="passengerCity" name="
                                dpassengerCity" class="form-control
                                " placeholder="Ciudad"
                            v-model="passengerData.passengerCity">
                </div>
                <div class="form-group">
                            <label>Telefono</label><input type="
                                text" id="passengerPhone" name="
                                dpassengerPhone" class="form-
                                control" placeholder="Telefono"
                            v-model="passengerData.passengerPhone"
                                >
                </div>
                <div class="form-group">
                    <label>Salida...</label><input type="time" id="
                        passengerStartTime" name="dpassengerStartTime
                        " v-model="passengerData.passengerStartTime">
                </div>
                <br>
                <div class="form-group">
                    <label>Vuelta...</label><input type="time" id="
                        passengerEndTime" name="dpassengerEndTime"
                v-model="passengerData.passengerEndTime">
                </div>
                </div>
                <br>
```

```
                <div class="form-group">
                        <div class="chooseSmokingGroupPax">


                        <div class="nosomokingGroupPax">

                        <font-awesome-icon icon="fa-solid fa-ban-smoking" size="2xl" />
                        <input class="form-check-input" type="radio" name="radioNoSmokingPax" id="radioNoSmokingPax" :checked="noSmokingSelection" @click="setNoSmokingSelection(idNoSmoking)" >
                        </div>
                        <div class="smokingGroupPax">

                        <font-awesome-icon icon="fa-solid fa-smoking" size="2xl" />
                        <input class="form-check-input" type="radio" name="radioNoSmokingPax" id="radioSmokingPax" :checked="!noSmokingSelection" @click="setNoSmokingSelection(idSmoking)">
                        </div>
                    </div>
                    <br>
                </div>
                <div class="form-group">
                    <label>Maxima espera {{ passengerData.passengerWaitLimitStart }} minutos</label><input type="range" class="form-range" min="0" max="30" id="passengerWaitLimitStart" name="dpassengerWaitLimitStart"
                v-model="passengerData.passengerWaitLimitStart">
                </div>
                <br>
                <div class="form-group">
                    <label>Maxima distancia {{ passengerData.passengerMaxDistanceAllowed }} km</label><input type="range" class="form-range" min="0" max="10" id="passengerMaxDistanceAllowed" name="dpassengerMaxDistanceAllowed"
                    v-model="passengerData.passengerMaxDistanceAllowed">
                </div>
                <br>
                <div class="row justify-content-center">
                    <div class="col-6">
                        <button class="btn btn-primary" id="dpassengerAdded" @click.prevent="sendDatapassenger" v-if="addBtnpassenger">Anadir</button>
                    </div>


```

```
                </div>



        </form>

    </div>
    </template>

    <script setup>
    /* eslint-disable */
    import {ref} from 'vue';
    import { useLoginStore } from '@/store/loginstore';
    import { usePaxStore } from '@/store/paxstore';
    import Swal from 'sweetalert2';
    const loginstore = useLoginStore();
    const paxstore = usePaxStore();
    let idNoSmoking="radioNoSmoking"
    let idSmoking="radioSmoking"
    let noSmokingSelection=true
    let smokingSelection=false

    /* Variable que contiene los campos de los pasajeros */
    let passengerData=ref({

        passengerStartTime:'08:00',
        passengerEndTime:'16:00',
        passengerWaitLimitStart:0,
        passengerMaxDistanceAllowed:0,
        passengerCity:'',
        passengerPhone:'',
        passsengerSmoke:smokingSelection
    })

    let addBtnpassenger=ref(true);
    let editBtnpassenger=ref(true);


    const setNoSmokingSelection=(id)=>{
    if((noSmokingSelection)&&(id=="radioSmoking")){


            smokingSelection=true
            noSmokingSelection=false
        }
        if((smokingSelection)&&(id=="radioNoSmoking")){

            smokingSelection=false
            noSmokingSelection=true
        }


}
    /* funcion para crear pasajero */
     const sendDatapassenger=async()=>{
        try{
          const res = await fetch('http://127.0.0.1:8000/pax/
              adddpassenger/',{
```

```
            method:'POST',
            headers: {
                        'Accept':'application/json',
                        'Content-Type': 'application/json'
                    },
body: JSON.stringify({
userIDowner:loginstore.getUsername,
passengerStartTime:passengerData.value.passengerStartTime,
passengerEndTime:passengerData.value.passengerEndTime,
passengerWaitLimitStart:passengerData.value.passengerWaitLimitStart,
passengerMaxDistanceAllowed:passengerData.value.
passengerMaxDistanceAllowed,
passengerCity:passengerData.value.passengerCity,
passengerPhone:passengerData.value.passengerPhone,
passsengerSmoke:passengerData.value.passsengerSmoke
})
          })
          const response = await res.json()
          loginstore.addPassenger(true)
          Swal.fire({
                      position: 'center',
                      icon: 'success',
                      title: 'Pasajero anadido',
                      showConfirmButton: false,
                      timer: 1500
                    });

        paxstore.addPaxStoreData(passengerData.value);
        } catch(error){
          Swal.fire({
                      position: 'center',
                      icon: 'warning',
                      title: 'Error al guardar',
                      showConfirmButton: false,
                      timer: 1500
                    });
          console.log(error)
        }
        console.log(passengerData.value)
    }

    </script>

    <style scoped>

    .chooseSmokingGroupPax{
    columns: 2;
    margin: 0 auto;
    border:1px solid blue;
    border-radius: 15px;
    padding: 5px;
    justify-content: stretch;
}

.nosomokingGroupPax{
  position: relative;
  grid-column: 1;
  left: 25%;
```

```
  width: 50%;
}

#radioNoSmokingPax{
  position: relative;
  grid-column: 1;
  left:15%
}

#radioSmokingPax{
  position: relative;
  grid-column: 1;
  left:15%
}



    #dpassengerAdded{
        align-content:start;
        padding: 5px;
        width:40%;
        margin: 5px;
    }

    #dpassengerEddit{
        align-content:end;
        padding: 5px;
        width: 40%;
        margin: 5px;
    }
    </style>
```

Fichero PassengerLoadComp.vue.

```
<!-- Plantilla que carga datos de pasajeros -->
<template>
<div class="card card-header">
    <h5>Pasajero</h5><label v-if="ratesReceived!=0">Valoraciones: {{
        ratesReceived }} </label><font-awesome-icon icon="fa-regular
        fa-star" id="star1" v-if="ratesReceived!=0"/><label v-if="
       ratesReceived!=0" id="labelstar">{{ rateAverage }}</label>
</div>

<div class="card card-body">
    <label>Ciudad: {{ pData.passengerCity }}</label>
    <label>Telefono: {{ pData.passengerPhone }}</label>
    <label>Hora recogida: {{ pData.passengerStartTime }}</label>
    <label>Hora destino: {{ pData.passengerEndTime }}</label>
    <div class="smokeBox">
        <font-awesome-icon icon="fa-solid fa-ban-smoking" size="2xl"
            v-if=!pData.PassengerSmoke />
        <font-awesome-icon icon="fa-solid fa-smoking" size="2xl"  v-
           if=pData.PassengerSmoke />
    </div>
</div>

</template>

```

```
<script setup>
/* eslint-disable */
import RateService from '@/services/RateService'
import {ref, onMounted} from 'vue'
import { useLoginStore } from '@/store/loginstore';

const rateservices = new RateService();
const loginstore = useLoginStore();

let rateData=ref([])

let ratesReceived=ref(0)
let rateAverage=ref(0)
let allRates=ref([])

defineProps({
    pData: Object
})

onMounted(async ()=>{
    rateData.value=await rateservices.loadMyRates(
       loginstore.getUsername)
    ratesReceived.value=rateData.value.length
    setAverageRates()

})

const setAverageRates=()=>{
    rateData.value.forEach((rate, id)=>{
        allRates.value.push(rate.rateValue)
    })
    let averageValue=0;
    for(let i=0; i< ratesReceived.value; i++){
        averageValue+=allRates.value[i];
    }
    rateAverage.value=averageValue/ratesReceived.value

}

</script>

<style scoped>
.smokeBox{

    position: relative;
    padding: 5px;


}
#star1{
    border: 2px solid yellow;
    background-color: yellow;
    border-radius: 25px;
    padding: 5px;
    width: 25px;
    position: relative;
    top:-65px;
    left:340px;
```

```
77
78  }
79
80  #labelstar{
81
82      width: 15%;
83      position: relative;
84      font-size:larger;
85      text-align: center;
86      top:-100px;
87      left:300px;
88
89
90  }
91
92  </style>
```

Fichero ProfilePaxLoadComp.vue.

```
1   <!-- Componente para carga de datos de pasajero -->
2   <template>
3   <div class="container">
4       <div class="collapse" :class="{'show':loginstore.getPassenger}">
5           <PassengerLoadComp v-bind:pData="pData"/>
6       </div>
7
8
9   </div>
10  </template>
11
12  <script setup>
13  /* eslint-disable */
14  import { useLoginStore } from '@/store/loginstore';
15  import {ref} from 'vue';
16  import { onMounted } from 'vue';
17  import PassengerLoadComp from './PassengerLoadComp.vue';
18
19  const loginstore = useLoginStore();
20  let pData={
21
22  }
23  onMounted(async ()=>{
24      try{
25
26          const res = await fetch('http://127.0.0.1:8000/pax/detail/'
                ,{
27          method: 'POST',
28                  headers: {
29                      'Accept':'application/json',
30                      'Content-Type': 'application/json'
31                  },
32                  body: JSON.stringify({
33                      userIDowner:loginstore.username
34                  })
35          })
36
37          const response = await res.json()
38
```

```
        if(response.passengerCreated!=null){

            pData=response
            loginstore.addPassenger(true)

        }
        else{

            loginstore.addPassenger(false)

        }


    }catch(error){
        console.log(error)
    }

})

</script>
```

Fichero RateFormComp.vue.

```
<!-- Formulario para valoracion -->
<template>

<div class="starsbox">



    <div class="regularstars">
        <div class="starfive">
            <label for="star5" ><font-awesome-icon icon="fa-regular
               fa-star" size="2xl" id="item-star-1" v-if="rangeValue
               ==5"/></label>
        </div>
        <div class="starfour">
            <label for="star4" ><font-awesome-icon icon="fa-regular
               fa-star" size="2xl" id="item-star-1" v-if="rangeValue
               >=4"/></label>
        </div>
        <div class="starthree">
            <label for="star3" ><font-awesome-icon icon="fa-regular
               fa-star" size="2xl" id="item-star-1" v-if="rangeValue
               >=3"/></label>
        </div>
        <div class="startwo">
            <label for="star2" ><font-awesome-icon icon="fa-regular
               fa-star" size="2xl" id="item-star-1" v-if="rangeValue
               >=2"/></label>
        </div>
        <div class="starone">
            <label for="star1" ><font-awesome-icon icon="fa-regular
               fa-star" size="2xl" id="item-star-1" /></label>
        </div>



```

```
27
28
29
30
31      </div>
32
33      <input type="range" class="form-range" min="1" max="5" id="
            rateRange" v-model="rangeValue">
34      <label class="labelrange">Valore el servicio ofrecido</label>
35      <input class="form-check-input" type="checkbox" id="
            checkanonymous" v-model="rateData.rateAnonymous">
36      <label class="labelcheck">Valorar anonimamente</label>
37      <textarea class="form-control" id="rateTextArea" rows="4" v-
            model="rateData.rateComment"></textarea>
38      <button type="button" class="btn btn-outline-primary" id="
            btnRateSend" @click="sendRate">Valorar</button>
39
40
41
42
43
44
45  </div>
46  </template>
47
48  <script setup>
49  /* eslint-disable */
50  import {ref} from 'vue'
51  import { useTripStore } from '@/store/tripstore';
52  import { useLoginStore } from '@/store/loginstore';
53  import RateService from '@/services/RateService'
54  import Swal from 'sweetalert2';
55  const tripstore = useTripStore();
56  const loginstore = useLoginStore();
57
58  const tripdata= tripstore.getTripFull
59  const rateservices = new RateService();
60
61
62  let rangeValue=ref(3)
63
64  /* almacena la valoracion */
65  let rateData=ref({
66      userIDowner:loginstore.getUsername,
67      rateUser:'',
68      rateAnonymous:false,
69      rateValue:rangeValue,
70      rateComment:''
71
72  })
73
74  /* Envia valoracion */
75  const sendRate=async()=>{
76      if(loginstore.getDriver){
77          rateData.value.rateUser=tripdata.tripPax
78      }else{
79
80          rateData.value.rateUser=tripdata.userIDowner
```

```
81      }
82      const sentRate= await rateservices.sendRate(rateData.value)
83      if(sentRate){
84          Swal.fire({
85                          position: 'center',
86                          icon: 'success',
87                          title: 'Gracias por valorar',
88                          showConfirmButton: false,
89                          timer: 2500
90                        });
91      }else{
92          Swal.fire({
93                          position: 'center',
94                          icon: 'warning',
95                          title: 'ocurrio un error al valorar',
96                          showConfirmButton: true
97
98                        });
99      }
100
101 }
102
103 </script>
104
105 <style scoped>
106 .starsbox{
107     border: 1px solid lightblue;
108     border-radius: 15px;
109     width: 25%;
110     display: flex;
111     flex-direction: column;
112     height: 45vh;
113     padding: 10px;
114
115 }
116
117
118 .regularstars{
119
120     display: flex;
121     flex-direction: row;
122     justify-content: center;
123
124 }
125
126 .solidstars{
127
128     display: flex;
129     flex-direction: row-reverse;
130     position: relative;
131     top:-32px;
132     visibility: hidden;
133 }
134
135 #rateRange{
136
137     position: relative;
138     padding: 10px;
```

```
    top:15px;
    background-color: lightblue;
    border-radius: 15px;
}

.labelrange{
    position: relative;
    padding: 10px;

    justify-content: center;
    text-align: center;
    top:15px;
}

#item-star-1{
    border: 2px solid yellow;
    border-radius: 20px;
    padding: 5px;
    background-color: yellow;

}

#checkanonymous{
    padding: 5px;
    border: 2px solid blue;
    justify-content: left;
    position: relative;
    top:20px;
    left:35px;
}

.labelcheck{
    padding: 5px;

    position: relative;
    top:-5px;
    text-align: center;

}

#btnRateSend{
    padding: 5px;
    width: 70px;
    border: 1px solid green;
    position: relative;
    left:225px;
    top:10px;
}

/*
#star1:checked + label {

    color: blue;
}

#star2:checked + label {

    color: white;
```

```
}

#star3:checked + label {

    color: white;
}

#star4:checked + label {

    color: white;
}

#star5:checked + label {

    color: white;
} */


/*
#item-star-1:hover{
    border: 1px solid yellow;
    border-radius: 5;
    background-color: yellow;

}

#item-star-2:hover{
    border: 1px solid yellow;
    border-radius: 5;
    background-color: yellow;
}

#item-star-3:hover{
    border: 1px solid yellow;
    border-radius: 5;
    background-color: yellow;

}

#item-star-4:hover
{
    border: 1px solid yellow;
    border-radius: 5;
    background-color: yellow;
}

#item-star-5:hover{
    border: 1px solid yellow;
    border-radius: 5;
    background-color: yellow;
} */

/*
.rangebox{
    display: flex;
    flex-direction: row-reverse;
    border: 1px solid red;
    position: relative;
```

```
    left:2px;
    width: 100%;
    padding: 5px;
} */
</style>
```

Fichero RegisterFormComp.vue.

```
<!-- Formulario de registro como usuario -->
<template>
    <div class="container">
        <div class="regformbox">


            <form>
                <div class="form-group">
                    <label for="inputUsername">Usuario</label>
                    <input type="text" class="form-control" id="
                       inputUsername" placeholder="Usuario" v-model=
                       "usernamelog">

                </div>
                <div class="form-group">
                    <label for="inputEmail">Email</label>
                    <input type="text" class="form-control" id="
                       inputEmail" placeholder="Email" v-model="
                       emaillog">

                </div>
                <div class="form-group">
                    <label for="inputPassword">Contrasena</label>
                    <input type="password" class="form-control" id="
                       inputPassword" placeholder="Contrasena" v-
                       model="passwordlog">

                </div>
                <button tye="submit" class="btn btn-primary mt-2"
                   @click.prevent="registerAuth" id="btnregform">
                   Registrar</button>

            </form>
        </div>
        </div>
</template>

<script setup>
/* eslint-disable */
    import {ref} from 'vue';
    import { useRouter } from 'vue-router';
    import Swal from 'sweetalert2';
    import AuthService from '@/services/AuthService';
    import { useLoginStore } from '@/store/loginstore';
    const loginstore = useLoginStore();
    const router = useRouter()
    let usernamelog=ref("");
    let emaillog=ref("");
    let passwordlog=ref("");

```

```
43      /** Comprueba que el usuario se puede registrar */
44      const registerAuth= async() =>{
45          const auth = new AuthService()
46          const registerSuccess = await auth.register(
               usernamelog.value, emaillog.value, passwordlog.value)
47          if(registerSuccess){
48
49              loginstore.addAuthLog(true)
50              loginstore.addUsername(usernamelog.value)
51              loginstore.addToken(auth.getJwt())
52              router.push('/profile');
53              Swal.fire({
54                    position: 'center',
55                    icon: 'success',
56                    title: 'Usuario Registrado',
57                    showConfirmButton: false,
58                    timer: 1500
59                  });
60
61          }
62          else{
63              Swal.fire({
64                    position: 'center',
65                    icon: 'warning',
66                    title: auth.getError(),
67                    showConfirmButton: false,
68                    timer: 1500
69                  });
70          }
71      }
72
73  </script>
74
75  <style scoped>
76  .regformbox{
77
78      border-radius: 5px;
79      padding: 10px;
80      margin: 0 auto;
81
82      width:100%;
83  }
84
85  #btnregform{
86      align-content: center;
87      position: relative;
88      padding: 5px;
89      width: 40%;
90      margin: 5px;
91      left: 30%;
92
93  }
94
95  </style>
```

Fichero TripsListComp.vue.

```
1   <!-- Vista de reservas en un listado -->
```

```
<template>
    <div class="container">

        <label v-if="loginstore.getDriver" id="lblfreeseats">{{
            freeSeats }} plazas disponibles</label>
        <!-- Viajes <button type="button" class="btn btn-outline-
            info" @click="reloadTrips">Actualizar</button> -->
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">Fecha</th>
                    <th scope="col">Usuario</th>
                    <th scope="col">Salida</th>
                    <th scope="col">Llegada</th>
                    <th scope="col">L</th>
                    <th scope="col">M</th>
                    <th scope="col">X</th>
                    <th scope="col">J</th>
                    <th scope="col">V</th>
                    <th scope="col">S</th>
                    <th scope="col">D</th>
                    <th scope="col">Fijo</th>
                    <th scope="col">Ida</th>
                    <th scope="col">Vuelta</th>
                    <th scope="col">Acciones</th>
                    <th scope="col">Estado</th>
                </tr>
            </thead>
            <tbody>
                <tr v-for="trip in trips.slice().reverse() " :key="
                    trip.id">
                    <td>{{ trip.tripCreated }}</td>
                    <td v-if="loginstore.getPassenger">{{
                        trip.userIDowner }}</td>
                    <td v-if="loginstore.getDriver">{{ trip.tripPax
                        }}</td>
                    <td>{{ trip.tripStartTime }}</td>
                    <td>{{ trip.tripEndTime }}</td>
                    <td>
                        <input class="form-check-input" type="
                            checkbox" v-model="trip.tripDayMon"
                            disabled />
                    </td>
                    <td>
                        <input class="form-check-input" type="
                            checkbox" v-model="trip.tripDayTue"
                            disabled />
                    </td>
                    <td>
                        <input class="form-check-input" type="
                            checkbox" v-model="trip.tripDayWed"
                            disabled/>
                    </td>
                    <td>
                        <input class="form-check-input" type="
                            checkbox" v-model="trip.tripDayThu"
                            disabled/>
                    </td>
```

```
47                      <td>
48                          <input class="form-check-input" type="
                                checkbox" v-model="trip.tripDayFri"
                                disabled/>
49                      </td>
50                      <td>
51                          <input class="form-check-input" type="
                                checkbox" v-model="trip.tripDaySat"
                                disabled />
52                      </td>
53                      <td>
54                          <input class="form-check-input" type="
                                checkbox" v-model="trip.tripDaySun"
                                disabled/>
55                      </td>
56                      <td>
57                          <input class="form-check-input" type="
                                checkbox" v-model="trip.tripPermanent"
                                disabled/>
58                      </td>
59                      <td>
60                          <input class="form-check-input" type="
                                checkbox" v-model="trip.tripInit"
                                disabled/>
61                      </td>
62                      <td>
63                          <input class="form-check-input" type="
                                checkbox" v-model="trip.tripBack"
                                disabled/>
64                      </td>
65                      <td>
66                          <button type="button" class="btn btn-outline
                                -success" v-if="(!trip.tripAccepted)&&(
                                loginstore.getDriver)&&(!trip.tripActive)
                                &&(!trip.tripRejected)&&(!
                                trip.tripRejectedP)" @click="acceptBook(
                                trip)" >Aceptar</button>
67                          <button type="button" class="btn btn-outline
                                -danger" v-if="(!trip.tripAccepted)&&(
                                loginstore.getDriver)&&(!trip.tripActive)
                                &&(!trip.tripRejected)&&(!
                                trip.tripRejectedP)" @click="rejectTrip(
                                trip)">Rechazar</button>
68                          <button type="button" class="btn btn-outline
                                -info"  v-if="(!trip.tripAccepted)&&(
                                loginstore.getDriver)&&(!trip.tripActive)
                                &&(!trip.tripRejected)&&(!
                                trip.tripRejectedP)" @click="showMapPax(
                                trip.tripPax)" >Mapa</button>
69                          <button type="button" class="btn btn-outline
                                -danger" v-if="(!trip.tripAccepted)&&(
                                loginstore.getPassenger)&&(!
                                trip.tripActive)&&(!trip.tripRejected)
                                &&(!trip.tripRejectedP)" @click="
                                rejectTripP(trip)">Anular Reserva</button
                                >
70                          <button type="button" class="btn btn-outline
                                -primary" v-if="((trip.tripAccepted)&&(
```

```
                                loginstore.getDriver)&&(!
                                trip.tripOnVehicleD)&&(!
                                trip.tripReactivate))" @click="takeSeatD(
                                trip)" >Subir Pasajero</button>
71                          <button type="button" class="btn btn-outline
                                -warning" v-if="((trip.tripAccepted)&&(
                                loginstore.getDriver)&&(!
                                trip.tripOnVehicleD)&&(!
                                trip.tripReactivate))" @click="cancelTrip
                                (trip)" >Cancelar</button>
72                          <button type="button" class="btn btn-outline
                                -warning" v-if="((trip.tripAccepted)&&(
                                loginstore.getDriver)&&(
                                trip.tripOnVehicleD)&&(!trip.tripActive))
                                " @click="cancelNoSeat(trip)" >Cancelar</
                                button>
73                          <button type="button" class="btn btn-outline
                                -primary" v-if="((trip.tripAccepted)&&(
                                loginstore.getPassenger)&&(!
                                trip.tripOnVehicleP)&&(!trip.tripActive)
                                &&(!trip.tripReactivate))" @click="
                                takeSeatP(trip)" >Subir al coche</button>
74                          <button type="button" class="btn btn-outline
                                -warning" v-if="((trip.tripAccepted)&&(
                                loginstore.getPassenger)&&(!
                                trip.tripOnVehicleP)&&(!trip.tripActive)
                                &&(!trip.tripReactivate))" @click="
                                cancelTrip(trip)" >Cancelar</button>
75                          <button type="button" class="btn btn-outline
                                -warning" v-if="((trip.tripAccepted)&&(
                                loginstore.getPassenger)&&(
                                trip.tripOnVehicleP)&&(!trip.tripActive))
                                " @click="cancelNoSeatP(trip)" >Cancelar
                                </button>
76
77                          <button type="button" class="btn btn-outline
                                -primary" v-if="trip.tripActive" @click="
                                endTrip(trip)">Finalizar</button>
78                          <button type="button" class="btn btn-outline
                                -info" v-if="trip.tripReactivate" @click=
                                "reactivateTrip(trip)" >Reactivar</button
                                >
79
80
81                      </td>
82                      <td>
83                          <label v-if="!trip.tripAccepted && !
                                trip.tripActive &&(!trip.tripRejected)
                                &&(!trip.tripRejectedP)">Pendiente</label
                                ><label v-if="trip.tripActive">Activo</
                                label><label v-if="!trip.tripAccepted &&
                                trip.tripActive && trip.tripEnded">
                                Finalizado</label>
84                          <label v-if="trip.tripReactivate">Inactivo</
                                label>
85                          <label v-if="trip.tripRejected">Rechazado</
                                label><label v-if="((trip.tripAccepted)
                                &&(loginstore.getDriver)&&(!
```

```
                            trip.tripOnVehicleD)&&(!
                            trip.tripReactivate))">recoger pasajero
                            ?</label>
86                      <label v-if="((trip.tripAccepted)&&(
                            loginstore.getDriver)&&(
                            trip.tripOnVehicleD)&&(!trip.tripActive))
                            ">Confirmacion Pasajero?</label>
87                      <label v-if="((trip.tripAccepted)&&(
                            loginstore.getPassenger)&&(!
                            trip.tripOnVehicleP)&&(!trip.tripActive)
                            &&(!trip.tripReactivate))">subir al coche
                            ?</label>
88                      <label v-if="((trip.tripAccepted)&&(
                            loginstore.getPassenger)&&(
                            trip.tripOnVehicleP)&&(!trip.tripActive))
                            ">Confirmacion Conductor?</label>
89                      <label v-if="(trip.tripRejectedP)&&(
                            loginstore.getDriver)">Anulada por
                            pasajero</label>
90                      <label v-if="(trip.tripRejectedP)&&(
                            loginstore.getPassenger)">Anulada</label>
91
92
93                  </td>
94                  <td>
95                      <button type="button" class="btn btn-outline
                            -success" v-if="(trip.tripEnded)" @click=
                            "storeTrip(trip)">Finalizar</button>
96                      <button type="button" class="btn btn-outline
                            -info" v-if="trip.tripReactivate" @click=
                            "rateTrip(trip)" >Valorar</button>
97                      <button type="button" class="btn btn-outline
                            -danger" v-if="((trip.tripRejected)||(
                            trip.tripRejectedP))" @click="delTrip(
                            trip.id)" >Borrar</button>
98                  </td>
99
100
101                 </tr>
102             </tbody>
103         </table>
104         <div class="collapse" :class="{'show':showMapping}" v-if="
                showMapping" id="boxMap">
105             <MapListMapComp />
106             <button type="button" class="btn btn-info" id="
                    btnCloseMap" @click="clsMapInfo">Cerrar</button>
107         </div>
108         <div class="collapse" :class="{'show':showRates}" v-if="
                showRates" id="boxRate">
109             <RateFormComp />
110             <button type="button" class="btn btn-info" id="
                    btnCloseRate" @click="clsRateInfo">Cerrar</button>
111         </div>
112
113     </div>
114     </template>
115
116     <script setup>
```

```
117     /* eslint-disable */
118     import {ref, onMounted, onUnmounted} from 'vue';
119     import { useLoginStore } from '@/store/loginstore';
120     import { useMessageStore} from '@/store/messagesstore';
121     import { useTripStore } from '@/store/tripstore';
122     import axios from 'axios';
123     import Swal from 'sweetalert2';
124     import TripsService from '../services/TripsService'
125     import MessagesService from '@/services/MessagesService';
126     import MapListMapComp from '@/components/MapListMapComp.vue';
127     import RateFormComp from '@/components/RateFormComp.vue'
128 
129     const loginstore = useLoginStore();
130     const tripstore = useTripStore();
131 
132     const tripsActions= new TripsService()
133     const msgService = new MessagesService()
134 
135     let tempReason=ref('')
136 
137     let dataMap=ref([])
138 
139     let msgData=ref({
140         fromMessage:loginstore.getUsername,
141         toMessage:'',
142         bodyMessage:''
143     })
144 
145     let timerReload=ref(null)
146 
147     let freeSeats=ref(0);
148 
149     let showMapping=ref(false)
150     let showRates=ref(false)
151 
152     let trips=ref([])
153 
154     let checkBothAcceptedTO=null
155 
156 
157 
158 
159     /* al cargar la vista */
160     onMounted(async ()=>{
161 
162         await reloadTrips()
163     })
164 
165     /* al cambiar de vista */
166     onUnmounted(async()=>{
167         clearTimeout(timerReload.value)
168     })
169 
170     /** funcion para recargar reservas */
171     const autoReloadTrips=async()=>{
172 
173         reloadTrips();
174     }
```

```
175
176     /* cierra el formulario valorar */
177     const clsRateInfo=()=>{
178         showRates.value=false;
179     }
180
181     /* cierra el mapa de apoyo */
182     const clsMapInfo=()=>{
183         showMapping.value=false;
184     }
185
186     /*** muestra el mapa de apoyo */
187     const showMapPax=async (trip)=>{
188         tripstore.addTripPax(trip)
189         //dataMap=await tripsActions.showMapInfoPax(trip,
                loginstore.getUsername)
190         showMapping.value=true;
191     }
192
193     /** valora la reserva */
194     const rateTrip=async(trip)=>{
195         showRates.value=true;
196         tripstore.addTripFull(trip)
197     }
198
199     /*** reactivar la reserva */
200     const reactivateTrip=async(trip)=>{
201
202
203         Swal.fire({
204         title: 'Reactivar Reserva?',
205         showDenyButton: true,
206         showCancelButton: false,
207         confirmButtonText: 'Si',
208         denyButtonText: 'No',
209         }).then(async (result) => {
210
211         if (result.isConfirmed) {
212             trip.tripEnded=false
213             trip.tripReactivate=false
214             const storetrip = await tripsActions.updateTrip(trip)
215             Swal.fire({
216                         position: 'center',
217                         icon: 'success',
218                         title: 'Viaje Reactivado',
219                         showConfirmButton: false,
220                         timer: 1500
221                         });
222             if(storetrip){
223                 if(loginstore.getDriver){
224
225                 trips.value= await tripsActions.loadDriverBooks(
                        loginstore.getUsername)
226             }
227             else{
228
229                 trips.value= await tripsActions.loadPaxBooks(
                        loginstore.getUsername)
```

```
230             }
231
232
233             }else{
234
235             }
236         }else if (result.isDenied) {
237
238         }
239
240         })
241
242     }
243
244     /*** guarda la reserva */
245     const storeTrip=async(trip)=>{
246         Swal.fire({
247         title: 'Finalizar y archivar Viaje?',
248         showDenyButton: true,
249         showCancelButton: false,
250         confirmButtonText: 'Si',
251         denyButtonText: 'No',
252         }).then(async (result) => {
253         /* Read more about isConfirmed, isDenied below */
254         if (result.isConfirmed) {
255                 trip.tripShow="NO"
256             const storetrip = await tripsActions.updateTrip(trip)
257             Swal.fire({
258                         position: 'center',
259                         icon: 'success',
260                         title: 'Viaje Archivado',
261                         showConfirmButton: false,
262                         timer: 1500
263                         });
264             if(storetrip){
265                 if(loginstore.getDriver){
266
267                 trips.value= await tripsActions.loadDriverBooks(
                        loginstore.getUsername)
268             }
269             else{
270
271                 trips.value= await tripsActions.loadPaxBooks(
                        loginstore.getUsername)
272             }
273
274
275             }else{
276
277             }
278         }else if (result.isDenied) {
279
280         }
281
282         })
283
284
285
```

```
286     }
287
288     /** borra la reserva */
289     const delTrip=async(id)=>{
290         Swal.fire({
291         title: 'Borrar reserva?',
292         showDenyButton: true,
293         showCancelButton: false,
294         confirmButtonText: 'Si',
295         denyButtonText: 'No',
296         }).then(async (result) => {
297         /* Read more about isConfirmed, isDenied below */
298         if (result.isConfirmed) {
299             const deleteTrip =tripsActions.delTrip(id)
300             if(deleteTrip){
301                 Swal.fire({
302                         position: 'center',
303                         icon: 'info',
304                         title: 'Reserva borrada',
305                         showConfirmButton: false,
306                         timer: 1500
307                         });
308
309             }
310             await reloadTrips();
311         }else if (result.isDenied) {
312
313         }
314
315         })
316
317
318
319
320     }
321
322
323     /*** recarga el panel de reservas */
324     const reloadTrips=async()=>{
325
326         try{
327             if(loginstore.getDriver){
328
329
330
331                 trips.value= await tripsActions.loadDriverBooks(
                        loginstore.getUsername)
332                 freeSeats.value=await tripsActions.seatAvailable(
                        loginstore.getUsername)
333
334             }
335             if(loginstore.getPassenger){
336
337                 trips.value= await tripsActions.loadPaxBooks(
                        loginstore.getUsername)
338
339
340             }
```

```
341             clearTimeout(timerReload.value)
342             timerReload.value=setTimeout(()=>{
343                 autoReloadTrips()
344             }, 15000);
345 
346 
347 
348         }catch(error)
349         {
350             console.log(error)
351         }
352     }
353 
354     /** rechaza la reserva */
355     const rejectTrip=async (trip)=>{
356         Swal.fire({
357         title: 'Rechazar Viaje?',
358         showDenyButton: true,
359         showCancelButton: false,
360         confirmButtonText: 'Si',
361         denyButtonText: 'No',
362         }).then(async (result) => {
363         /* Read more about isConfirmed, isDenied below */
364         if (result.isConfirmed) {
365 
366             const reasonReject = await Swal.fire({
367                 title:'Motivo rechazo',
368                 input:'text',
369                 inputLabel: '',
370                 inputPlaceholder: 'escriba el motivo o dejelo en
                        blanco'
371             })
372 
373 
374             if(reasonReject.value!=""){
375 
376                 tempReason.value=reasonReject.value
377             } else{
378 
379                 tempReason.value="El conductor no ha indicado motivo
                        del rechazo"
380             }
381 
382             trip.tripRejected=true
383             const rejectBookTrip= await tripsActions.updateTrip(trip
                    )
384             Swal.fire({
385                         position: 'center',
386                         icon: 'warning',
387                         title: 'Viaje rechazado',
388                         showConfirmButton: false,
389                         timer: 1500
390                         });
391             if(rejectBookTrip){
392 
393                 trips.value= await tripsActions.loadDriverBooks(
                        loginstore.getUsername)
394             }
```

```
395             let bodyTemp="La reserva a "+ trip.userIDowner +" ha
                    sido rechazada por el motivo: "+ tempReason.value
396             msgData.value.fromMessage="Reservas"
397             msgData.value.bodyMessage=bodyTemp
398             msgData.value.toMessage=trip.tripPax
399             const response= msgService.sendMessageTo(msgData)
400         }
401         else if (result.isDenied) {
402 
403         }
404 
405         })
406     }
407 
408     /** anula la reserva */
409     const rejectTripP=async (trip)=>{
410         Swal.fire({
411         title: 'Anular Reserva?',
412         showDenyButton: true,
413         showCancelButton: false,
414         confirmButtonText: 'Si',
415         denyButtonText: 'No',
416         }).then(async (result) => {
417         /* Read more about isConfirmed, isDenied below */
418         if (result.isConfirmed) {
419 
420             const reasonReject = await Swal.fire({
421                 title:'Motivo anulacion',
422                 input:'text',
423                 inputLabel: '',
424                 inputPlaceholder: 'escriba el motivo o dejelo en
                        blanco'
425             })
426 
427 
428             if(reasonReject.value!=""){
429 
430                 tempReason.value=reasonReject.value
431             } else{
432 
433                 tempReason.value="El pasajero no ha indicado motivo
                        de la anulacion"
434             }
435 
436             trip.tripRejectedP=true
437             const rejectBookTrip= await tripsActions.updateTrip(trip
                    )
438             Swal.fire({
439                         position: 'center',
440                         icon: 'warning',
441                         title: 'Reserva anulada',
442                         showConfirmButton: false,
443                         timer: 1500
444                         });
445             if(rejectBookTrip){
446 
447                 trips.value= await tripsActions.loadDriverBooks(
                        loginstore.getUsername)
```

```
448             }
449             let bodyTemp="La reserva de "+ trip.tripPax +" ha sido
                    anulada por el motivo: "+ tempReason.value
450             msgData.value.fromMessage="Reservas"
451             msgData.value.bodyMessage=bodyTemp
452             msgData.value.toMessage=trip.userIDowner
453             const response= await msgService.sendMessageTo(msgData)
454         }
455         else if (result.isDenied) {
456 
457         }
458 
459         })
460     }
461 
462     /*** finaliza la reserva */
463     const endTrip=async (trip)=>{
464 
465         trip.tripEnded=true
466         trip.tripActive=false
467         trip.tripReactivate=true
468         trip.tripOnVehicleD=false
469         trip.tripOnVehicleP=false
470         const endBookTrip= await tripsActions.updateTrip(trip)
471         if(endBookTrip){
472 
473                 if(loginstore.getDriver){
474                 const increaseSeat=await tripsActions.increaseSeat(
                        trip.userIDowner)
475                 trips.value= await tripsActions.loadDriverBooks(
                        loginstore.getUsername)
476                 }
477                 else{
478                     const increaseSeat=await
                            tripsActions.increaseSeat(trip.userIDowner)
479                     trips.value= await tripsActions.loadPaxBooks(
                            loginstore.getUsername)
480             }
481             Swal.fire({
482                       position: 'center',
483                       icon: 'success',
484                       title: 'Viaje finalizado',
485                       showConfirmButton: false,
486                       timer: 1500
487                     });
488         }else{
489 
490         }
491 
492 
493 
494 
495     }
496 
497     /*** tomar asiento por parte del pasajero */
498     const takeSeatP=async(trip)=>{
499         const seatavailabe=await tripsActions.seatAvailable(
               trip.userIDowner)
```

```
500         if(seatavailabe){
501                     const okDri=await tripsActions.getTrip(trip)
502 
503             if(okDri.tripOnVehicleD){
504 
505                 trip.tripOnVehicleP=true
506                 const acceptBookTrip = await tripsActions.updateTrip
                        (trip)
507                 checkAcceptedTrip(trip)
508             }else{
509                 Swal.fire({
510                             position: 'center',
511                             icon: 'info',
512                             title: 'A la espera de confirmacion del
                                conductor',
513                             showConfirmButton: false,
514                             timer:2500
515                         });
516                 trip.tripOnVehicleP=true
517                 const acceptBookTrip = await tripsActions.updateTrip
                        (trip)
518             }
519         }else{
520             Swal.fire({
521                       position: 'center',
522                       icon: 'warning',
523                       title: 'No hay plazas disponibles. El
                             conductor debe liberar plazas. Pongase en
                             contacto con el.',
524                       showConfirmButton: true
525                     });
526         }
527 
528 
529 
530     }
531 
532     /*** cancelar el pasajero no se mete en coche */
533     const cancelNoSeatP=async(trip)=>{
534         trip.tripOnVehicleP=false
535         const acceptBookTrip = await tripsActions.updateTrip(trip)
536     }
537 
538     /** el conductor da cancelar  */
539     const cancelNoSeat=async(trip)=>{
540         trip.tripOnVehicleD=false
541         const acceptBookTrip = await tripsActions.updateTrip(trip)
542     }
543 
544     /*** se cancela el viaje */
545     const cancelTrip=async(trip)=>{
546         trip.tripAccepted=false
547 
548         const acceptBookTrip = await tripsActions.updateTrip(trip)
549     }
550 
551     /** comprobar si el pasajero ha confirmado dentro vehiculo */
552     const takeSeatD=async(trip)=>{
```

```
553         const seatavailabe=await tripsActions.seatAvailable(
                trip.userIDowner)
554         if(seatavailabe){
555                 const okPax=await tripsActions.getTrip(trip)
556 
557             if(okPax.tripOnVehicleP){
558                 trip.tripOnVehicleD=true
559 
560                 checkAcceptedTrip(trip)
561             }else{
562                 Swal.fire({
563                         position: 'center',
564                         icon: 'info',
565                         title: 'A la espera de confirmacion del
                            pasajero',
566                         showConfirmButton: false,
567                         timer:2500
568                         });
569                 trip.tripOnVehicleD=true
570                 const acceptBookTrip = await tripsActions.updateTrip
                    (trip)
571             }
572         }else{
573             Swal.fire({
574                       position: 'center',
575                       icon: 'warning',
576                       title: 'Todas las plazas estan ocupadas!.
                          Libere alguna plaza o aumente las plazas en
                           configuracion',
577                       showConfirmButton: true
578                     });
579         }
580 
581 
582 
583 
584     }
585 
586 
587     /*** el trayecto esta activo */
588     const checkAcceptedTrip=async(trip)=>{
589 
590         trip.tripActive=true
591         trip.tripEnded=false
592             const acceptBookTrip = await tripsActions.updateTrip(
                    trip)
593             if(acceptBookTrip){
594                 const updatingSeats= await tripsActions.updateSeats(
                        trip.userIDowner)
595                 if(updatingSeats){
596                     Swal.fire({
597                       position: 'center',
598                       icon: 'info',
599                       title: 'Trayecto activo',
600                       showConfirmButton: false,
601                       timer:2500
602                     });
603                 }else{
```

```
604
605                 }
606
607             }
608         /* if((trip.tripOnVehicleD)&&(trip.tripOnVehicleP)){
609             Swal.fire({
610                       position: 'center',
611                       icon: 'info',
612                       title: 'Trayecto activo',
613                       showConfirmButton: false,
614                       timer:2500
615                     });
616             trip.tripActive=true
617             trip.tripEnded=false
618             const acceptBookTrip = await tripsActions.updateTrip(
                    trip)
619             if(acceptBookTrip){
620                 const updatingSeats=await tripsActions.updateSeats(
                        loginstore.getUsername)
621             }
622
623         } */
624     }
625
626     /*** aceptar reserva */
627     const acceptBook=async (trip)=>{
628
629         trip.tripAccepted=true
630
631
632         const acceptBookTrip = await tripsActions.updateTrip(trip)
633         if(acceptBookTrip){
634             Swal.fire({
635                       position: 'center',
636                       icon: 'success',
637                       title: 'Reserva aceptada',
638                       showConfirmButton: false,
639                       timer: 1500
640                     });
641
642
643             let bodyTemp="El usuario "+ loginstore.getUsername+ " ha
                    aceptado tu reserva"
644             msgData.value.fromMessage="Reservas"
645             msgData.value.bodyMessage=bodyTemp
646             msgData.value.toMessage=trip.tripPax
647             const response= await msgService.sendMessageTo(msgData)
648         }else{
649             Swal.fire({
650                       position: 'center',
651                       icon: 'warning',
652                       title: 'Ocurrio un error al aceptar',
653                       showConfirmButton: false,
654                       timer: 1500
655                     });
656         }
657         /* const updatingSeats=await tripsActions.updateSeats(
               loginstore.getUsername)
```

```
658
659         if(updatingSeats){
660             trip.tripAccepted=true
661             trip.tripActive=true
662             const acceptBookTrip = await tripsActions.updateTrip(
                    trip)
663
664                 if(acceptBookTrip){
665                     trips.value= await tripsActions.loadDriverBooks(
                            loginstore.getUsername)
666
667                     Swal.fire({
668                       position: 'center',
669                       icon: 'success',
670                       title: 'Viaje aceptado',
671                       showConfirmButton: false,
672                       timer: 1500
673                     });
674                 } else{
675
676                 }
677
678             }else{
679                 Swal.fire({
680                       position: 'center',
681                       icon: 'warning',
682                       title: 'No se ha podido realizar la accion. El
                              vehiculo esta completo',
683                       showConfirmButton: false,
684                       timer: 3500
685                     });
686             }
687
688
689
690     } */}
691     </script>
692
693     <style scoped>
694
695     #lblfreeseats{
696         border: 1px solid lightblue;
697         border-radius: 15px;
698         padding: 5px;
699         font-size:30px;
700         margin: 0 auto;
701         position: relative;
702         width:100%;
703         text-align: center;
704     }
705     #boxMap{
706         border: 1px solid red;
707         width: 100%;
708         height: 40vh;
709         display: flex;
710         flex-direction: row;
711     }
712
```

```
713     #btnCloseMap{
714         position: absolute;
715         right:175px;
716         bottom: 180px;
717     }
718 
719     #boxRate{
720 
721         width: 100%;
722         height: 40vh;
723         display: flex;
724         flex-direction: row-reverse;
725     }
726 
727     #btnCloseRate{
728         position: absolute;
729         right:150px;
730         top: 550px;
731     }
732 
733     </style>
```

Fichero UserDataEditComp.vue.

```
1  <!-- Configuracion de cambio de contrasena -->
2  <template>
3  <div class="container">
4      <label id="labelitem1">Cambiar contrasena</label>
5      <input type="password" placeholder="Introduzca nueva contrasena"
           id="passboxcomp1">
6      <button class="btn btn-outline-danger" id="buttonpassword">
          Cambiar Contrasena</button>
7  </div>
8  </template>
9 
10 <script setup>
11 
12 </script>
13 
14 <style scoped>
15 .container{
16     border: 1px solid lightblue;
17     border-radius: 5px;
18     width: 100%;
19     height: 10vh;
20 
21     padding: 5px;
22 }
23 
24 #passboxcomp1{
25     padding: 5px;
26     position: absolute;
27     border: 1px solid grey;
28     border-radius: 5px;
29     left: 405px;
30     width: 250px;
31     top:75px;
32 }
```

```
33
34 #labelitem1{
35     padding: 5px;
36 }
37
38 #buttonpassword{
39     position: absolute;
40     padding: 5px;
41     left:675px;
42     top:75px;
43 }
44 </style>
```

Carpeta router

La Figura C.17 muestra el fichero *index.js* de la carpeta router, que se encarga de generar las direcciones URL de la aplicación.

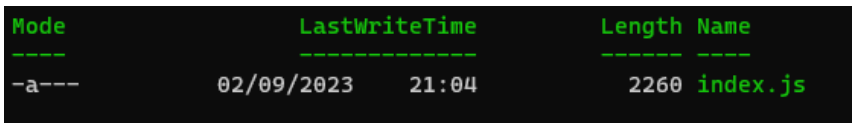


Figura C.17 : Fichero en la carpeta router.

Se muestra el Código Fuente 7.3 del fichero *index.js*.

Fichero index.js.

```
1  /*** en este fichero se definen las rutas asignadas para las vistas
       */
2  /* eslint-disable */
3  import { createRouter, createWebHashHistory } from 'vue-router'
4
5
6  import LiveMapView from '../views/LiveMapView.vue'
7
8
9  import RegAccessView from '../views/RegAccessView.vue'
10 import LoginAccessView from '../views/LoginAccessView.vue'
11 import ConfigUser from '../views/ConfigUser.vue'
12 import HomeViewTemp from '../views/HomeViewTemp.vue'
13 import LogOutView from '../views/LogOutView.vue'
14
15 import { useLoginStore } from '@/store/loginstore';
16 import TripsListView from '../views/TripsListView.vue'
17 import ProfileUserView from '../views/ProfileUserView.vue'
18 import MessagesPageView from '../views/MessagesPageView.vue'
19
20
21
22
23 const routes = [
24   {
25     path: '/',
26     name: 'home',
```

```
27      component: HomeViewTemp,
28      meta:{
29        needAuth:false
30      }
31
32    },
33    {
34      path: '/messages',
35      name:'messagesview',
36      component: MessagesPageView,
37      meta:{
38        needAuth:true
39      }
40    },
41
42    {
43      path: '/register',
44      name: 'registeru',
45      component: RegAccessView,
46      meta:{
47        needAuth:false
48      }
49    },
50    {
51      path: '/profile',
52      name: 'profileview',
53      component: ProfileUserView,
54      meta:{
55        needAuth:true
56      }
57    },
58    {
59      path:'/config/user',
60      name:'configuser',
61      component: ConfigUser,
62      meta:{
63        needAuth:true
64      }
65    },
66    {
67      path: '/livemap',
68      name: 'livemap',
69      component: LiveMapView,
70      meta:{
71        needAuth:true
72      }
73    },
74    {
75      path: '/login',
76      name: 'login',
77      component: LoginAccessView,
78      meta:{
79        needAuth:false
80      }
81    },
82    {
83      path: '/trips',
84      name: 'trips',
```

```
85      component: TripsListView,
86      meta:{
87        needAuth:true
88      }
89    },
90    {
91      path: '/logout',
92      name: 'logout',
93      component: LogOutView,
94      meta:{
95        needAuth:true
96      }
97    }
98  ]
99
100 const router = createRouter({
101   history: createWebHashHistory(),
102   routes
103 })
104
105 router.beforeEach((to, from, next) =>{
106
107
108   const loginstore = useLoginStore();
109   const isAuth = loginstore.getAuthLog
110   const checkAuth = to.meta.needAuth
111
112   if(checkAuth && !isAuth){
113     next('login')
114   } else{
115     next()
116   }
117 })
118
119
120
121 export default router
```

Carpeta services

La Figura C.18 muestra los ficheros de la carpeta services.

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          02/09/2023    21:04           2543 AuthService.js
-a---          02/09/2023    21:05           3480 MessagesService.js
-a---          02/09/2023    21:06           3312 RateService.js
-a---          02/09/2023    21:08           7311 TripsService.js
```

Figura C.18 : Ficheros en la carpeta services.

Se muestra el Código Fuente de los ficheros de la Figura C.18 .

Fichero AuthService.js.

```
1 /* eslint-disable */
```

```
import { ref } from 'vue'



/** servio de autenticacion para los usuarios */
class AuthService {
    constructor() {
        this.jwt = ref('')
        this.error = ref('')
        this.username=ref('')
    }

    getJwt() {
        return this.jwt
    }

    getError() {
        return this.error
    }

    getUsername(){
        return this.username
    }
    // registrar usuario
    async register(username, email, password){
        try{
            const res = await fetch('http://127.0.0.1:8000/auth/
               register/',{

                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    username:username,
                    email:email,
                    password:password
                })
            })

            const response = await res.json();
            if('erroruser' in response){
                this.error="Ya existe ese usuario"
                return false
            }
            if('errormail' in response){
                this.error="Ya existe ese correo"
                return false
            }

            this.jwt=res.token

            return true

        }catch(error){
            console.log(error)
        }
```

```
    }
    // funcion para acceso o login
    async login(username, password){
        try {
            const res = await fetch('http://127.0.0.1:8000/auth/
                login/',{

                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    username:username,
                    password:password
                })
            })

            const response = await res.json()

            if('error' in response){
                this.error="Usuario incorrecto"
                return false
            }
            if('pass' in response){

                this.error="Contrasena incorrecta"
                return false
            }
            this.jwt = response.token

            this.username=response.success.username;
            return true

        }catch(error){
            console.log(error)
        }
    }


}

export default AuthService
```

Fichero MessagesService.js.

```
/* eslint-disable */
import { ref } from 'vue'

import axios from 'axios';

/** Esta clase se encarga de las funciones asociadas a los mensajes
 * actua de enlace con la API REST
 */

class MessagesService{
    // enviar mensaje sin responder
    async sendMessageTo(messageData){
```

```
13          try{
14
15              const res = await fetch('http://127.0.0.1:8000/messages/
                    sendmsg/',{
16                  method:'POST',
17                  headers:{
18                      'Accept':'application/json',
19                      'Content-Type': 'application/json'
20                  },
21                  body: JSON.stringify({
22                      "fromMessage":messageData.value.fromMessage,
23                      "toMessage":messageData.value.toMessage,
24                      "bodyMessage":messageData.value.bodyMessage
25
26                      })
27              })
28              const response = await res.json()
29              console.log(response)
30              return true
31          } catch(error){
32
33          }
34
35      }
36      // enviar mensaje que se pueda responder
37      async sendMessageToUsers(messageData){
38          try{
39
40              const res = await fetch('http://127.0.0.1:8000/messages/
                    sendmsg/',{
41                  method:'POST',
42                  headers:{
43                      'Accept':'application/json',
44                      'Content-Type': 'application/json'
45                  },
46                  body: JSON.stringify({
47                      "fromMessage":messageData.value.fromMessage,
48                      "toMessage":messageData.value.toMessage,
49                      "bodyMessage":messageData.value.bodyMessage,
50                      "messageAvoidReply":false
51                      })
52              })
53              const response = await res.json()
54              console.log(response)
55              return true
56          } catch(error){
57
58          }
59
60      }
61
62
63      // cargar mensajes enviados
64      async loadMsgSen(username){
65          try{
66              const res = await fetch('http://127.0.0.1:8000/messages/
                    sentmsg/',{
67                  method: 'POST',
```

```
68                  headers: {
69                      'Accept':'application/json',
70                      'Content-Type': 'application/json'
71                  },
72                  body: JSON.stringify({
73                      "fromMessage":username
74                  })
75              })
76              const response = await res.json()
77
78              //messagesStore.addMessagesData(response)
79              return response
80          }catch(error){
81
82          }
83      }
84
85      // modificar mensaje
86      async updateMsg(msg){
87          try{
88              await axios
89                      .put('http://127.0.0.1:8000/messages/updatemsg/'
                            ,msg)
90                      .then((response) =>{
91                          console.log(response);
92                      });
93              return true
94          }catch(error)
95          {
96              console.log(error)
97          }
98      }
99
100     // cargar mensajes recibidos
101     async loadMsgRec(username){
102         try{
103             const res = await fetch('http://127.0.0.1:8000/messages/
                    receivemsg/',{
104                 method: 'POST',
105                 headers: {
106                     'Accept':'application/json',
107                     'Content-Type': 'application/json'
108                 },
109                 body: JSON.stringify({
110                     "toMessage":username
111                 })
112             })
113             const response = await res.json()
114             return response
115
116         }catch(error)
117         {
118             console.log(error)
119         }
120     }
121
122
123
```

```
}

export default MessagesService
```

Fichero RateService.js.

```
/* eslint-disable */
import { ref } from 'vue'

import axios from 'axios';

/** Esta clase se encarga de las funciones asociadas a las valoraciones
 * actua de enlace con la API REST
 */
class RateService{
    // enviar valoracion
    async sendRate(rateData){
        try{

            const res = await fetch('http://127.0.0.1:8000/rates/addrate/',{
                method:'POST',
                headers:{
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(rateData)
            })
            const response = await res.json()
            console.log(response)
            return true
        } catch(error){
            return false
        }

    }

    // enviar mensaje al usuario
    async sendMessageToUsers(messageData){
        try{

            const res = await fetch('http://127.0.0.1:8000/messages/sendmsg/',{
                method:'POST',
                headers:{
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    "fromMessage":messageData.value.fromMessage,
                    "toMessage":messageData.value.toMessage,
                    "bodyMessage":messageData.value.bodyMessage,
                    "messageAvoidReply":false
                    })
            })
            const response = await res.json()
            console.log(response)
```

```
            return true
        } catch(error){

        }

    }

    // cargar mis valoraciones
    async loadMyRates(username){
        try{

            const res = await fetch('http://127.0.0.1:8000/rates/
               myrates/',{
                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    "rateUser":username
                })
            })
            const response = await res.json()

            //messagesStore.addMessagesData(response)
            return response

        }catch(error)
        {
            console.log(error)
        }
    }

    // cargar mis mensajes enviados
    async loadMsgSen(username){
        try{
            const res = await fetch('http://127.0.0.1:8000/messages/
               sentmsg/',{
                method: 'POST',
                headers: {
                    'Accept':'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    "fromMessage":username
                })
            })
            const response = await res.json()

            //messagesStore.addMessagesData(response)
            return response
        }catch(error){

        }
    }

        // modificar mensaje
    async updateMsg(msg){
```

```
106         try{
107             await axios
108                     .put('http://127.0.0.1:8000/messages/updatemsg/'
                            ,msg)
109                     .then((response) =>{
110                         console.log(response);
111                     });
112             return true
113         }catch(error)
114         {
115             console.log(error)
116         }
117     }
118 
119 }
120 
121 export default RateService
```

Fichero TripsService.js.

```
1 
2 /* eslint-disable */
3 import { ref } from 'vue'
4 
5 import axios from 'axios';
6 
7 /** Esta clase se encarga de las funciones asociadas a las reservas
8  * actua de enlace con la API REST
9  */
10 class TripsService {
11 
12     /** carga las reservas del conductor */
13     async loadDriverBooks(userIDowner){
14         try{
15             const res = await fetch('http://127.0.0.1:8000/trips/
                   trip/',{
16                 method: 'POST',
17                 headers: {
18                     'Accept':'application/json',
19                     'Content-Type': 'application/json'
20                 },
21                 body: JSON.stringify({
22                     "userIDowner": userIDowner
23                 })
24                 })
25                 const response = await res.json()
26 
27                 return response
28 
29         }catch(error){
30 
31         }
32     }
33 
34     /** carga las reservas del pasajero */
35     async loadPaxBooks(paxTrip){
36         try{
37             const res = await fetch('http://127.0.0.1:8000/trips/
```

```
                booktrip/',{
                    method: 'POST',
                    headers: {
                        'Accept':'application/json',
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({
                        "tripPax": paxTrip
                    })
                })
                const response = await res.json()
                return response
            }catch(error){

            }

        }

        /*** modifica una reserva */
        async updateTrip(trip){
            try{
                await axios
                .put('http://127.0.0.1:8000/trips/update/',trip)
                .then((response) =>{
                    console.log(response);
                });
                return true
            }catch(error){
                return false
            }


        }

        /** busca una reserva */
        async getTrip(trip){
            try{
                const res = await fetch('http://127.0.0.1:8000/trips/
                    tripid/',{
                    method: 'POST',
                    headers: {
                        'Accept':'application/json',
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({
                        "id": trip.id
                    })
                })
                const response = await res.json()
                return response
            }catch(error){

            }
        }

        /** borra una reserva */
        async delTrip(id){
            try{
```

```
94              const res = await fetch('http://127.0.0.1:8000/trips/
                    deltrip/',{
95                  method: 'POST',
96                  headers: {
97                      'Accept':'application/json',
98                      'Content-Type': 'application/json'
99                  },
100                 body: JSON.stringify({
101                     "id": id
102                 })
103             })
104
105             return true
106         }catch(error){
107             return false
108         }
109     }
110
111     /** aumenta el numero de plazas libres, al finalizar una reserva
            */
112     async increaseSeat(userIDowner){
113         try{
114             let tempDriver=[];
115             const res = await fetch('http://127.0.0.1:8000/drivers/
                    detail/',{
116                 method: 'POST',
117                 headers: {
118                     'Accept':'application/json',
119                     'Content-Type': 'application/json'
120                 },
121                 body: JSON.stringify({
122                     "userIDowner": userIDowner
123                 })
124             })
125             const response = await res.json()
126             tempDriver=response;
127
128
129             let tempseat=tempDriver.driverFreeSeats+1;
130             tempDriver.driverFreeSeats=tempseat;
131             await axios
132             .put('http://127.0.0.1:8000/drivers/update/',tempDriver)
133             .then((response) =>{
134                 console.log(response);
135             });
136             return true
137
138         }catch(error){
139
140         }
141     }
142
143     /*** comprueba si el conductor tiene plazas libres */
144     async seatAvailable(userIDowner){
145         try{
146             let tempDriver=[];
147             const res = await fetch('http://127.0.0.1:8000/drivers/
                    detail/',{
```

```
148                 method: 'POST',
149                 headers: {
150                     'Accept':'application/json',
151                     'Content-Type': 'application/json'
152                 },
153                 body: JSON.stringify({
154                     "userIDowner": userIDowner
155                 })
156             })
157             const response = await res.json()
158             tempDriver=response;
159             return tempDriver.driverFreeSeats
160             /* if(tempDriver.driverFreeSeats==0){
161                 return false
162             } else{
163                 return true
164             } */
165         }catch(error){
166 
167         }
168     }
169 
170     /** modifica el numero de plazas libres del conductor */
171     async updateSeats(userIDowner){
172         try{
173             let tempDriver=[];
174             const res = await fetch('http://127.0.0.1:8000/drivers/
                    detail/',{
175                 method: 'POST',
176                 headers: {
177                     'Accept':'application/json',
178                     'Content-Type': 'application/json'
179                 },
180                 body: JSON.stringify({
181                     "userIDowner": userIDowner
182                 })
183             })
184             const response = await res.json()
185             tempDriver=response;
186 
187 
188             let tempseat=tempDriver.driverFreeSeats-1;
189             console.log("veamos si actualiza")
190             console.log(tempseat)
191             if(tempseat<0){
192                 return false
193             } else{
194                 tempDriver.driverFreeSeats=tempseat;
195             await axios
196             .put('http://127.0.0.1:8000/drivers/update/',tempDriver)
197             .then((response) =>{
198                 console.log(response);
199             });
200             return true
201             }
202 
203 
204         } catch(error){
```

```
205
206         }
207     }
208
209     /*** muestra informacion del pasajero en el mapa */
210     async showMapInfoPax(trip, userIDowner){
211         try{
212             let routeOne=[]
213             let routeTwo=[]
214             const res = await fetch('http://127.0.0.1:8000/routes/
                    routenodes/',{
215                 method: 'POST',
216                 headers: {
217                     'Accept':'application/json',
218                     'Content-Type': 'application/json'
219                 },
220                 body: JSON.stringify({
221                     "userIDowner": userIDowner
222                 })
223             })
224             const response = await res.json()
225             routeOne=response
226             const res2 = await fetch('http://127.0.0.1:8000/routes/
                    routenodes/',{
227                 method: 'POST',
228                 headers: {
229                     'Accept':'application/json',
230                     'Content-Type': 'application/json'
231                 },
232                 body: JSON.stringify({
233                     "userIDowner": trip
234                 })
235             })
236             const response2 = await res2.json()
237             routeTwo=response2
238
239             return [routeOne, routeTwo]
240
241
242
243         }catch(error){
244
245         }
246     }
247 }
248
249 export default TripsService
```

Carpeta store

La Figura C.19 muestra los ficheros de la carpeta store.

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          02/09/2023    21:08            462 driverstore.js
-a---          02/09/2023    21:08           1259 loginstore.js
-a---          02/09/2023    21:09            425 messagesstore.js
-a---          02/09/2023    21:09            443 paxstore.js
-a---          02/09/2023    21:09           1327 routestore.js
-a---          02/09/2023    21:09            722 tripstore.js
-a---          02/09/2023    21:09            967 userstore.js
```

Figura C.19 : Ficheros en la carpeta store.

Se muestra el Código Fuente de los ficheros de la Figura C.19 .

Fichero driverstore.js.

```
import { defineStore } from 'pinia'

/*** almacen de datos de conductores */

export const useDriverStore = defineStore('driverstore',{
    state: () => {
        return {
            driverdata: {

            }
        }
    },

    getters: {
        getDriverData: (state) => state.driverdata
    },

    actions: {
        addDriverStoreData (driverdata){
            this.driverdata=driverdata;
        }
    }


})
```

Fichero loginstore.js.

```
import { defineStore } from 'pinia'

/*** almacen con los datos de los usuarios */

export const useLoginStore = defineStore('loginstore',{
    state: ()=>{
        return{
            token:"",
            username:"",
            error:"",
            authLog: false,
            driver:false,
            route:false,
            passsenger:false
        }
```

```
    },

    getters:{
        getToken:(state)=> state.token,
        getUsername:(state)=> state.username,
        getError:(state)=>state.error,
        getAuthLog: (state)=> state.authLog,
        getDriver:(state)=> state.driver,
        getRoute:(state)=> state.route,
        getPassenger:(state)=> state.passsenger
    },

    actions:{
        addToken (token){
            this.token=token
        },
        addUsername (username){
            this.username=username
        },
        addError(error){
            this.error=error
        },
        addAuthLog(authLog){
            this.authLog=authLog
        },
        addDriver(driver){
            this.driver=driver
        },
        addRoute(route){
            this.route=route
        },
        addPassenger(passenger){
            this.passsenger=passenger
        }
    }
})
```

Fichero messagesstore.js.

```
import { defineStore } from 'pinia'

/*** almacen para los mensajes */

export const useMessageStore = defineStore('messagesstore',{
    state:()=>{
        return {
            messages:[

            ]
        }
    },
    getters: {
        getMessagesData:(state) => state.messages
    },
    actions:{
        addMessagesData(messagesData){
            this.messages=messagesData;
        }
```

```
    }
})
```

Fichero paxstore.js.

```
import { defineStore } from 'pinia'


/*** almacen para los datos de los pasajeros */
export const usePaxStore = defineStore('paxstore',{
    state: () => {
        return {
            paxdata: {

            }
        }
    },

    getters: {
        getPaxData: (state) => state.paxdata
    },

    actions: {
        addPaxStoreData (paxdata){
            this.paxdata=paxdata;
        }
    }


})
```

Fichero routestore.js.

```
import { defineStore } from 'pinia'

/** almacen con los datos de las rutas */

export const useRouterStore = defineStore('routerstore',{
    state: () => {
        return {
            lngOrigen:'',
            latOrigen:'',
            lngDestino:'',
            latDestino:'',


            distanceR:0,
            routeData:{

            }


        }
    },

    getters: {
        getLngO: (state) => state.lngOrigen,
        getLatO: (state) => state.latOrigen,
```

```
        getLngD: (state) => state.lngDestino,
        getLatD: (state) => state.latDestino,
        getRouteData:(state)=> state.routeData,
        getDistance: (state)=> state.distanceR
    },

    actions: {
        addRouteData (routeData){
            this.routeData=routeData
        },
        addLngO (lngOrigen){
            this.lngOrigen=lngOrigen

        },
        addLatO (latOrigen){
            this.latOrigen=latOrigen

        },
        addLngD (lngDestino){
            this.lngDestino=lngDestino

        },
        addLatD (latDestino){
            this.latDestino=latDestino

        },
        addDistanceStoreData (distanceR){
            this.distanceR=distanceR;
        }
    }


})
```

Fichero tripstore.js.

```
import { defineStore } from 'pinia'


/*** almacen con los datos de las reservas */
export const useTripStore = defineStore('Tripstore',{
    state: () => {
        return {
            tripdata: '',
            tripPax:'',
            tripFull:[]
        }
    },

    getters: {
        getTripData: (state) => state.tripdata,
        getTripPax: (state) => state.tripPax,
        getTripFull: (state)=> state.tripFull
    },

    actions: {
        addTripStoreData (tripdata){
            this.tripdata=tripdata;
```

```
        },
        addTripPax (tripPax){
            this.tripPax=tripPax
        },
        addTripFull (tripFull){
            this.tripFull=tripFull
        }
    }


})
```

Fichero userstore.js.

```
import { defineStore } from 'pinia'

/*** almacen con datos de usuario. NO funcional */
export const useUserStore = defineStore('userstore',{
    state: () => {
        return {
            userdata: {

            },
            driverready: false,
            paxready: false,
            routeready: false,
        }
    },

    getters: {
        getUserData: (state) => state.userdata,
        getUserReady: (state) => state.driverready,
        getRouteReady: (state) => state.routeready,
        getPaxready: (state) => state.paxready
    },

    actions: {
        addUserStoreData (userdata){
            this.userdata=userdata;
        },
        setDriverReady(driverready){
            this.driverready=driverready;
        },
        setRouteReady(routeready){
            this.routeready=routeready;
        },
        setPaxReady(paxready){
            this.paxready=paxready;
        }
    }


})
```

Carpeta views

La Figura C.20 muestra los ficheros de la carpeta views.

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          02/09/2023    21:10            295 BackGroundView.vue
-a---          02/09/2023    21:10           1753 ConfigUser.vue
-a---          02/09/2023    21:10           2841 HomeViewTemp.vue
-a---          02/09/2023    21:11            407 LiveMapView.vue
-a---          02/09/2023    21:11            586 LoginAccessView.vue
-a---          02/09/2023    21:11            668 LogOutView.vue
-a---          02/09/2023    21:12            202 MessagesPageView.vue
-a---          02/09/2023    21:12           2392 ProfileUserView.vue
-a---          02/09/2023    21:12            677 RegAccessView.vue
-a---          02/09/2023    21:12            173 TripsListView.vue
```

Figura C.20 : Ficheros en la carpeta views.

Se muestra el Código Fuente de los ficheros de la Figura C.20 .

Fichero BackGroundView.vue.

```
<template>
<div class="mainbackgroundview">
<label class="backgndlabel"></label>
</div>
</template>
<script setup>
/** fondo de pantalla */
</script>

<style >
.mainbackgroundview{
    background: url("../assets/pexels-donald-tong-55787.png");

    height: 90vh;
}

</style>
```

Fichero ConfigUser.vue.

```
<!-- Vista de configuracion y sus componentes -->
<template>

    <div class="configboxuser">



        <div class="driverbox" v-if="loginstore.getDriver">

            <DriverChangeConfComp/>
        </div>

        <div class="passengerbox" v-if="loginstore.getPassenger">


```

```
            <PassChangeConfComp/>
        </div>
        <div class="userdatabox">
            <UserDataEditComp/>
        </div>









    </div>



</template>

<script setup>
/* eslint-disable */
import DriverChangeConfComp from '@/components/
   DriverChangeConfComp.vue';
import PassChangeConfComp from '@/components/PassChangeConfComp.vue'
import DriverConfigComp from '@/components/DriverConfigComp.vue';
import PassengerConfigComp from '@/components/
   PassengerConfigComp.vue';
import MapBoxUserMap from '@/components/MapBoxUserMap.vue';
import MessagesBoxComp from '@/components/MessagesBoxComp.vue';
import UserDataEditComp from '@/components/UserDataEditComp.vue'
import { useLoginStore } from '@/store/loginstore';
const loginstore = useLoginStore();
</script>

<style scoped>
.configboxuser{
    border:1px solid lightblue;
    border-radius: 15px;
    height:80vh;
    display: flex;
    flex-direction: column;
}

.driverbox{
    padding: 10px;
}

.passengerbox{
    padding: 10px;

}

.userdatabox{
    padding:10px;
    position: relative;
    top:100px;

}
/* .configboxuser{
    border: 1px solid green;
    display: flex;
```

```
    flex-direction: row;
}
.driverbox{
    border: 1px solid yellow;
    width:100%;
    height: 50vh;
}
 */

</style>
```

Fichero HomeViewTemp.vue.

```
<!-- Vista home principal -->
<template>
    <div class="boxhome">
        <div class="container">
            <label id="brand">Seats2Share</label>
        </div>

        <div class="boxes">
            <div id="carouselExampleIndicators" class="carousel
               slide" data-bs-ride="carousel">
                <div class="carousel-indicators">
                    <button type="button" data-bs-target="#
                       carouselExampleIndicators" data-bs-slide-to="
                       0" class="active" aria-current="true" aria-
                       label="Slide 1"></button>
                    <button type="button" data-bs-target="#
                       carouselExampleIndicators" data-bs-slide-to="
                       1" aria-label="Slide 2"></button>
                    <button type="button" data-bs-target="#
                       carouselExampleIndicators" data-bs-slide-to="
                       2" aria-label="Slide 3"></button>
                </div>
                <div class="carousel-inner">
                    <div class="carousel-item active">
                    <img src="../assets/pexels-life-of-pix-7674.png"
                         class="d-block w-100" >
                    </div>
                    <div class="carousel-item">
                        <img src="../assets/pexels-dario-fernandez-
                           ruz-6440616.png" class="d-block w-100" >

                    </div>
                    <div class="carousel-item">
                        <img src="../assets/pexels-andrea-piacquadio
                           -3783084.png" class="d-block w-100" >
                    </div>
                </div>
                <button class="carousel-control-prev" type="button"
                   data-bs-target="#carouselExampleIndicators" data-
                   bs-slide="prev">
                    <span class="carousel-control-prev-icon" aria-
                       hidden="true"></span>
                    <span class="visually-hidden">Previous</span>
                </button>
                <button class="carousel-control-next" type="button"
```

```
                    data-bs-target="#carouselExampleIndicators" data-
                    bs-slide="next">
                     <span class="carousel-control-next-icon" aria-
                        hidden="true"></span>
                     <span class="visually-hidden">Next</span>
                </button>
                </div>

        </div>

    </div>




</template>

<script setup>
/* eslint-disable */
import { useLoginStore } from '@/store/loginstore';

const loginstore = useLoginStore();
</script>

<style scoped>

.boxhome{
    height: 90vh;
    background: url("../assets/pexels-pixabay-531880.png");
    background-size: auto;
    border-radius: 15px;
    display: flex;
    flex-direction: column;

}

.boxes{

    width: 50%;
    height: 10%;
    position: relative;
    left: 25%;

}

.container{

    justify-content: center;
    text-align: center;


}
#brand{
    font-size: 7em;
    color:rebeccapurple

}
```

```
</style>
```

Fichero LiveMapView.vue.

```
<!-- Vista Mapa con todos los usuarios -->
<template>
    <div class="livemapMB">
        <MapBoxLiveMap/>
    </div>


</template>

<script setup>
 /* eslint-disable */
 import MapBoxLiveMap from "../components/MapBoxLiveMap.vue";
 import { watch, onMounted } from 'vue';
 import axios from "axios";

 onMounted(() => {


 })


</script>
<style scoped>

</style>
```

Fichero LoginAccessView.vue.

```
<!-- Vista Login y acceso -->
<template>
    <BackGroundView/>
    <div class="loginboxmain">
        <LoginFormComp />
    </div>

</template>

<script setup>
/* eslint-disable */
    import LoginFormComp from '@/components/LoginFormComp.vue';
    import BackGroundView from '@/views/BackGroundView.vue'

</script>

<style scoped>


.loginboxmain{
    margin: 0 auto;
    position: absolute;
    width: auto;
    top:50%;
    left:45%;
    transform: translate(-45%,-50%);
```

```
    background-color: whitesmoke;
    border-radius: 25px;

}
</style>
```

Fichero LogOutView.vue.

```
<!-- Vista log out -->
<template>
Log Out

</template>

<script setup>
/* eslint-disable */
import { useLoginStore } from '@/store/loginstore';
import { useRouter } from 'vue-router';
import Swal from 'sweetalert2';
const loginstore = useLoginStore();
const router = useRouter()
loginstore.addToken("");
loginstore.addUsername("");
loginstore.addError("");
loginstore.addAuthLog(false);
Swal.fire({
                  position: 'center',
                  icon: 'success',
                  title: 'Sesion Cerrada',
                  showConfirmButton: false,
                  timer: 1500
                });
router.push("/");

</script>
```

Fichero MessagesPageView.vue.

```
<!-- Vista Mensajes y componentes -->
<template>
<MessagesBoxComp/>
</template>
<script setup>
import MessagesBoxComp from '@/components/MessagesBoxComp.vue';
</script>
<style scoped>

</style>
```

Fichero ProfileUserView.vue.

```
<!-- Vista perfil del usuario y componentes -->
<template>

    <div class="grid-layout">


```

```
8
9           <div class="box">
10
11              <DriverConfigComp/>
12
13              <PassengerConfigComp/>
14
15
16          </div>
17          <div class="boxmap">
18              <MapBoxUserMap/>
19          </div>
20
21
22
23      </div>
24
25
26
27  </template>
28
29  <script setup>
30  /* eslint-disable */
31  import DriverConfigComp from '@/components/DriverConfigComp.vue';
32  import PassengerConfigComp from '@/components/
        PassengerConfigComp.vue';
33  import MapBoxUserMap from '@/components/MapBoxUserMap.vue';
34  import MessagesBoxComp from '@/components/MessagesBoxComp.vue';
35  import { useLoginStore } from '@/store/loginstore';
36  const loginstore = useLoginStore();
37  </script>
38
39  <style>
40
41  @media(max-width: 1300px){
42      .grid-layout{
43
44          grid-template-rows: 100% 50%;
45          display: flex;
46          flex-direction: column-reverse;
47      }
48
49      .box{
50          grid-row: 2;
51
52          margin: 0 auto;
53          min-height: 60vh;
54
55      }
56      .boxmap{
57          grid-row: 1;
58
59
60
61          width: auto;
62
63
64      }
```

```
}

@media(max-width=450px){
    .grid-layout{


        display: flex;
        flex-direction: column-reverse;
    }

    .box{
        position: absolute;

        grid-row: 2;
        width: 320px;
        margin: 0 auto;
        min-height: 30vh;


    }
    .boxmap{
        position: absolute;

        justify-content: center;
        grid-row: 1;
        margin:0 auto;
        width: 350px;
        left:5px;

    }
}
.grid-layout{
    position: relative;
    width: 80%;
    margin: 0 auto;
    display: grid;
    grid-template-columns: 35%  65%;
    justify-content: stretch;
    height: 90vh;
    font-size: 20px;
    background-color: white;
    border-radius: 15px;


}

.box{
    grid-column: 1;
    box-sizing: border-box;
    border-radius: 15px;
    min-width: 300px;
    height: 100vh;
    position: relative;
    top:25px;
    padding:10px;


}
```

```
123
124 .boxmap{
125
126     grid-column: 2;
127     box-sizing: border-box;
128     border-radius: 15px;
129     padding:10px;
130
131 }
132
133 </style>
```

Fichero RegAccessView.vue.

```
1  <!-- Vista registro de usuarios -->
2  <template>
3      <BackGroundView/>
4      <div class="regformboxmain">
5          <RegisterFormComp />
6      </div>
7
8  </template>
9
10 <script setup>
11 /* eslint-disable */
12     import RegisterFormComp from '@/components/RegisterFormComp.vue'
           ;
13     import BackGroundView from '@/views/BackGroundView.vue'
14
15 </script>
16
17 <style scoped>
18 @media(max-width: 1200px){
19     .regformboxmain{
20         width: 250px;
21     }
22 }
23 .regformboxmain {
24     margin:0;
25     position: absolute;
26     width: 320px;
27     top: 50%;
28     left:45%;
29     transform: translate(-45%, -50%);
30     background-color: whitesmoke;
31     border-radius: 25px;
32 }
33 </style>
```

Fichero TripsListView.vue.

```
1 <!-- Vista Mis Viajes con las reservas -->
2 <template>
3 <TripsListComp/>
4 </template>
5 <script setup>
6 import TripsListComp from '@/components/TripsListComp.vue';
7 </script>
```