(SINGLETICKET)
Plataforma para la gestión de billete multimodal

Memoria

Grado en Ingeniería Informática

VNiVERSiDAD D SALAMANCA

Septiembre de 2022

Autor
Álvaro Rosa González

Tutores
Pablo Chamoso Santos
Francisco Pinto Santos
Luis Blázquez Miñambres

Dr. Pablo Chamoso Santos, profesor adscrito al Departamento de Informática y Automática de la Universidad de Salamanca y D. Francisco Pinto Santos y D. Luis Blázquez Miñambres, investigadores de la Universidad de Salamanca, en calidad de co-tutores,

Hacen constar:

Que el trabajo titulado "(SINGLETICKET) Plataforma para la gestión de billete multimodal" ha sido realizado por D. Álvaro Rosa González, con DNI 80092896L y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a miércoles, 7 de Septiembre de 2022

Firmado digitalmente por CHAMOSO SANTOS PABLO - 70888952P
Fecha: 2022.09.07 18:34:26 +02'00'

2022.09.07 17:48:37 +02'00'

BLAZQUEZ MIÑAMBRES LUIS - 70910465Q
Firmado digitalmente por BLAZQUEZ MIÑAMBRES LUIS - 70910465Q
Fecha: 2022.09.07 17:47:34 +02'00'

Dr. Pablo Chamoso Santos | D. Francisco Pinto Santos | Luis Blázquez Miñambres

Resumen

Hoy en día existen múltiples plataformas, tanto analógicas como digitales, dedicadas a la venta de billetes de transporte de todas las modalidades existentes, por lo general propiedad de un operador de transporte o dedicadas a una misma modalidad de transporte. También existen aplicaciones que permiten la compra de billetes de varias modalidades, pero requieren de la aplicación propia del operador para acceder a la información de cada uno de estos.

El proyecto titulado SingleTicket realizado como Trabajo de Fin de Grado de Ingeniería Informática, pretende cubrir esta carencia, ofreciendo una plataforma en la que además de obtener tus viajes, puedas acceder a la información y detalles de todos tus billetes en un mismo lugar.

SingleTicket permite a una agencia de viajes planificar la movilidad de sus clientes a través de un planificador de viajes que, haciendo uso de las funciones ofrecidas por la API Amadeus y la extensión de PostgreSQL pgRouting, brinda las herramientas necesaria a la agencia para organizar un viaje haciendo uso tanto del medio aéreo como de trenes y autobuses dando además a sus clientes la ventaja de poder acceder a todos sus billetes de manera rápida en un mismo lugar.

Para el desarrollo del sistema se ha seguido el patrón arquitectónico MVC (Modelo, Vista, Controlador) ya que se adecua perfectamente a las necesidades del proyecto. La vista estaría situada en el paquete de single-ticket-web, paquete destinado al cliente, donde se encuentran todas las vistas y funcionalidad de interacción con el usuario, mientras que el modelo y controlador corresponderían al paquete node-backend, paquete destinado al servidor, donde se encuentra toda la lógica de la aplicación y el tratamiento de datos.

Tras todo el proceso de desarrollo del proyecto, comenzando con planificación, pasando por análisis y diseño y acabando con implementación y pruebas, se ha obtenido un sistema que cumple con los objetivos propuestos para este Trabajo de Fin de Grado, logrando una plataforma que ofrece a las agencias una planificación de viajes sencilla asistida, haciendo uso de la API de Amadeus y las rutas de autobús y tren que han introducido previamente los operadores en el sistema haciendo uso de la API que se les proporciona y utilizando como nodos de inicio y fin de trayecto los puntos de interés extraídos por el sistema de la base de datos de OpenStreetMaps. Todos los billetes que conforman los viajes planificados por las agencias son accesibles desde la vista detallada de cada viaje por el usuario, cumpliéndose así el objetivo principal que da título a este trabajo.

Palabras clave: transporte multimodal, Amadeus, planificador de viajes, gestión, agencia.

Abstract

Nowadays there are multiple platforms, both analog and digital, dedicated to the sale of transport tickets of all existing modalities, usually owned by a transport operator or dedicated to the same mode of transport. There are also applications that allow the purchase of tickets of various modes, but require the operator's own application to access the information of each of these.

The project named SingleTicket, carried out as a Computer Engineering Final Degree Project, aims to fill this gap by offering a platform where, in addition to obtaining your trips, you can access the information and details of all your tickets in the same place.

SingleTicket allows a travel agency to plan the mobility of its customers through a travel planner that, making use of the functions offered by the Amadeus API and the PostgreSQL extension pgRouting, provides the necessary tools to the agency to organize a trip using both air, trains and buses, also giving its customers the advantage of being able to access all their tickets quickly in one place.

For the development of the system, the MVC (Model, View, Controller) architectural pattern has been followed, since it is perfectly adapted to the needs of the project. The view would be located in the single-ticket-web package, package destined to the client, where all the views and user interaction functionality are located, while the model and controller would correspond to the node-backend package, package destined to the server, where all the application logic and data processing is located

After the whole development process of the project, starting with planning, going through analysis and design and finishing with implementation and testing, we have obtained a system that meets the objectives proposed for this Final Degree Project, achieving a platform that offers agencies a simple assisted travel planning, making use of the Amadeus API and the bus and train routes that have been previously introduced by the operators in the system using the API provided to them and using as start and end nodes the points of interest extracted by the system from the OpenStreetMaps database. All the tickets that make up the trips planned by the agencies are accessible from the detailed view of each trip by the user, thus fulfilling the main objective that gives title to this work.

Keywords: multi-modal transport, Amadeus, trip planner, management, agency.

1. Contenido

Índice de tablas

Índice de ilustraciones

Introducción

Hoy en día existen múltiples plataformas, tanto analógicas como digitales, dedicadas a la venta de billetes de transporte de todas las modalidades existentes, por lo general propiedad de un operador de transporte o dedicadas a una misma modalidad de transporte.

A veces el viaje que deseamos hacer no se puede realizar en un solo trayecto, sino que requiere de varios trayectos parciales, en ocasiones combinando distintos operadores y medios de transporte. Cuando se dan estas situaciones, la gestión de los billetes necesarios para el viaje se vuelve confusa dado que para cada billete de trayecto parcial que adquiramos tendremos que consultarlo a través de una aplicación diferente propia del operador que nos haya vendido el viaje. Es en ese punto donde surge la primera necesidad que pretende cubrir esta plataforma, la posibilidad de tener todos los billetes en un mismo lugar a tan solo unos *clics* de consultarlo.

Una vez cubierta esta necesidad, aparece otra cuestión derivada de la gestión de viajes compuestos por varias etapas proporcionadas por diversos operadores. La planificación de este tipo de viajes puede ser muy tediosa y confusa para el usuario medio, y cada día más teniendo en cuenta el creciente número de plataformas que trabajan en el sector turístico dedicadas al transporte que ofrecen un gran rango de viajes con gran diversidad de precios para el mismo trayecto. Por esto surge la idea de incluir Agencias de viajes en la plataforma para que gestionen y planifiquen los viajes para los clientes quitándoles así esa carga. Para facilitar esta función a las agencias se decide implementar un planificador de viajes que combinando diferentes tecnologías convierta el proceso de la planificación en un proceso semiautomático donde el usuario de la agencia solo necesite escoger las opciones deseadas para crear un viaje con la posibilidad de combinar billetes de múltiples operadores y medios de transporte.

Para esto se plantea este Trabajo de Fin de Grado, para proporcionar una plataforma para que las agencias puedan planificar viajes multimodales de manera sencilla e intuitiva permitiendo a los clientes acceder a sus pasajes todos en un mismo lugar. Con este fin se plantean una serie de requisitos que debe cumplir el sistema para alcanzar cada uno de los objetivos planteados y otros derivados de estos.

A lo largo de esta documentación se expone cada uno de los aspectos clave que conciernen el desarrollo de este proyecto:

- Resumen
- Introducción
- Objetivos del proyecto
- Conceptos teóricos
- Técnicas y herramientas
- Aspectos relevantes del desarrollo
- Conclusiones y líneas de trabajo futuras
- Referencias

1. Objetivos del proyecto

El objetivo principal de este proyecto es la creación de una plataforma para la gestión de billete multimodal, con un planificador de viajes para agencias integrado, intentando desarrollar un sistema escalable, portable, extensible, seguro y usable. Para esto se establecen los siguientes objetivos:

- Gestión de usuarios: el sistema debe gestionar la información de los usuarios que hagan uso de este. Para esto, se desarrolla un sistema de usuarios con tres roles diferentes, *user*, *agency* y *operator*, cada cual con ciertas funciones únicas y ciertas funciones compartidas con respecto al resto de objetivos. Dentro de la gestión de usuarios se dispone de las siguientes funciones:
    - Sistema de *login*: sistema básico de *login* que permite iniciar sesión, cerrar sesión y recuperar contraseña. Estas funciones estarán disponibles para los tres roles por igual.
    - Sistema de registro: permite crear una cuenta tanto con rol de user como de agency para ingresar al sistema.
    - Modificar perfil: se permite a los usuarios de tipo user y agency actualizar los datos de su cuenta.
- Gestión de API de operadores: el sistema debe ofreces a los operadores una interfaz de comunicación a través de la cual puedan añadir y eliminar sus rutas del sistema para ofrecerlas a las agencias.
- Gestión de viajes: el sistema debe permitir la gestión de los viajes que maneja. Dentro de este objetivo también se definen dos subobjetivos:
    - Gestión de billetes: el sistema debe permitir la gestión de los billetes relativos a los viajes gestionados.
    - Gestión de incidencias: el sistema debe permitir la gestión de las incidencias relativas a los viajes gestionados.
- Planificación de viajes: el sistema debe ofrecer a las agencias una solución efectiva y fácil de comprender para planificar los viajes de sus clientes.
- Gestión de puntos de interés: el sistema debe ofrecer una lista de estaciones de transporte a los operadores para que estos basen sus rutas en ellas. Para esto se extraen los datos de *OpenStreetMaps* a través del API de *Overpass*.

2. Conceptos teóricos

En este apartado se describen los distintos conceptos teóricos que son necesarios explicar detalladamente para comprender que aportan al funcionamiento del sistema.

2.1. API

Significa “interfaz de programación de programación de aplicaciones”.

Las API se pueden ver como un contrato que define cómo se comunican entre sí dos aplicaciones. Esto permite usa y agregar una funcionalidad de una aplicación ajena en la propio, u ofrecer funcionalidades de la aplicación propia para su uso por aplicaciones ajenas. [1]

2.2. Punto de interés (POI)

Punto de interés (POI del inglés Point Of Interest) hace referencia a la posibilidad de representar una característica que ocupa un lugar concreto. [2]

Este punto estaría representado por ese lugar y las propias características que lo hacen de interés.

De forma más gráfica en el fragmento del mapa de Salamanca de la Ilustración 1 se pueden ver varios puntos de interés marcados por un icono referente a la característica que los identifica, como podría ser las iglesias, marcadas con una cruz negra sin fondo, o los

hospitales, marcados con una cruz blanca sobre fondo rojo.

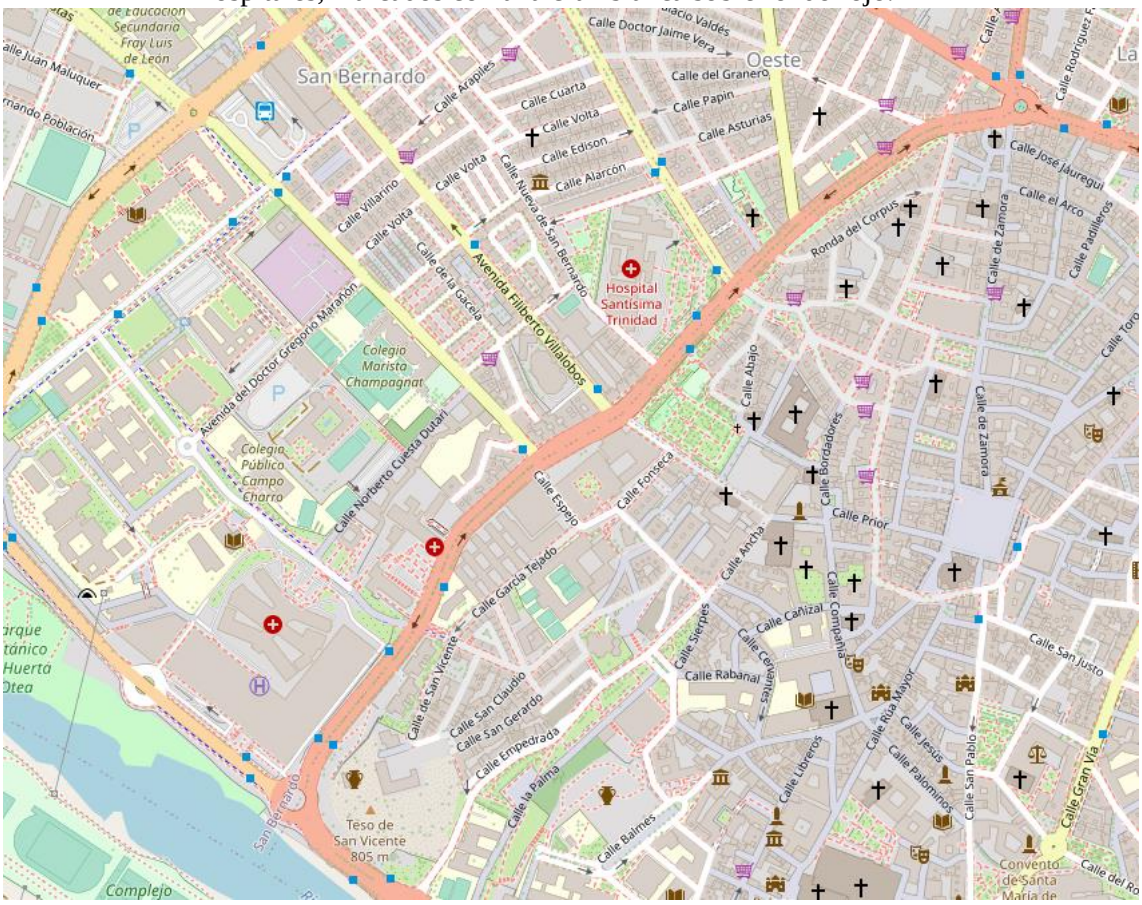

Ilustración 1. Mapa de salamanca

2.3. Algoritmo de Dijkstra

Algoritmo voraz usado para el cálculo del camino más corto entre dos nodos de un grafo con pesos en los arcos. [3]

Este algoritmo consiste en ir explorando todos los caminos más cortos hasta dar con el camino de menor coste desde el nodo origen hasta el nodo destino.

2.4. Framework

Un *Framework* es un entorno de trabajo preconfigurado con unas herramientas y características específicas disponibles para agilizar el proceso de desarrollo de un proyecto dado el enfoque específico para el que el *Framework* ha sido diseñado. [4]

Un ejemplo de *Framework* específico para el desarrollo de un backend con *Node.js* sería *Express.js* el cual ofrece herramientas muy útiles para el desarrollo web como peticiones y respuestas HTTP y el sistema gestor de paquetes NPM. [5]

3. Técnicas y herramientas

En esta apartado se describen las distintas técnicas y herramientas empleadas en el desarrollo del proyecto. Se indica detalladamente cada una de ellas, mostrando el porqué de su elección y que ventajas representas frente a otras.

3.1. Técnicas

3.1.1. MVC

El MVC o modelo vista controlador es un patrón arquitectónico utilizado en sistemas que implementan interfaces gráficas de usuario, como sería una aplicación web. Este patrón separa la aplicación en tres componentes lógicos:

Modelo: alberga la lógica de datos.

Vista: contiene el código referente a todo lo que se muestra de manera visual al usuario.

Controlador: es el enlace entre la vista y el modelo y contiene toda la lógica funcional de la aplicación [6]

Ilustración 2. MVC

3.1.2. Singleton

Singleton es un patrón arquitectónico cuyo objetivo es evitar que un objeto se instancie más de una vez dentro de la misma clase. Esto se consigue a través de un método que

genera una instancia dentro de la clase y estableciendo el constructor de dicho objeto privado. [7]

3.1.3. Método de Durán y Bernárdez

Es una metodología para la elicitación de requisitos descrita por Amador Durán Toro y Beatriz Bernárdez Álvarez. El objetivo de la metodología es definir que tareas deben realizarse, que técnicas se aplicarán y que productos se obtendrán.

Las tareas propuestas en esta metodología según la documentación oficial [8] son:

- Tarea 1: Obtener información sobre el dominio del problema y el sistema actual.
- Tarea 2: Preparar y realizar las reuniones de elicitación/negociación.
- Tarea 3: Identificar/revisar los objetivos del sistema.
- Tarea 4: Identificar/revisar los requisitos de almacenamiento de información.
- Tarea 5: Identificar/revisar los requisitos funcionales.
- Tarea 6: Identificar/revisar los requisitos no funcionales.
- Tarea 7: Priorizar objetivos y requisitos.

3.2.Herramientas

4.2.1. Base de datos PostgreSQL

Sistema de gestión de base de datos relacional de código abierto basado en SQL. Al tratarse de software libre existen numerosas extensiones que pueden ser muy útiles para una amplia diversidad de proyectos, como algunas que se mencionarán a continuación [9]

3.2.1.1. PostGIS

Extensión de PostgreSQL que permite trabajar con la base de datos como una base de datos espacial. Esto habilita a PostgreSQL como base de datos en un sistema que trabaje con grafos, como, por ejemplo, cualquier sistema que trabaje con coordenadas geográficas. [10]

3.2.1.2. pgRouting

Extensión de PostgreSQL que extiende PostGIS para añadir la posibilidad del cálculo de rutas a través de una consulta a la base de datos. [11]

3.2.2. Amadeus API

API ofrecida por la empresa del sector turístico Amadeus, la cual ofrece todo tipo de funcionalidades necesarias para desarrollar cualquier sistema relacionado con el sector, desde un motor de reservas de hoteles hasta un motor de reservas para vuelos pasando por otros. [12]

3.2.3. Overpass API

API proporcionada por *OpenStreetMaps* para la realización de mapas. Permite realizar consultas a su base de datos utilizando su propio lenguaje, Overpass QL. [13]

Overpass QL

```
(
   node(51.249,7.148,51.251,7.152);
   <;
);
out meta;
```

Ilustración 3. Overpass QL

3.2.4. Lenguajes de programación

3.2.4.1. Python

Lenguaje de programación de código abierto de tipado dinámico y multiplataforma. Soporta varios paradigmas de programación y se utiliza para desarrollar aplicaciones de cualquier tipo, aunque su uso principal sea en el campo de la inteligencia artificial [14]

Bibliotecas de Python empleadas:

- JSON: biblioteca empleada para dar formato a los datos que se utilizan en el script de extracción de los puntos de interés.
- Requests: biblioteca para realizar peticiones HTTP a las páginas web necesarias para la extracción.
- Psycopg2: biblioteca empleada para realizar la conexión con la base de datos PostgreSQL.
- Geocoder: biblioteca empleada para la geo codificación, utilizada en la obtención de la información de la ciudad a través de las coordenadas, en concreto, latitud y longitud.

3.2.4.2. JavaScript

Lenguaje de programación débilmente tipado y orientado a objetos enfocado principalmente al desarrollo web, aunque su uso para otro tipo de aplicaciones también es notable. Al igual que Python es un lenguaje de tipado dinámico, por lo que una misma variable puede admitir valores de distintos tipos a lo largo de una misma ejecución del programa. [15]

En el terreno de las aplicaciones web se utiliza tanto en el lado del frontend como en el backend.

3.2.5. Herramientas Case

3.2.5.1. REM

Requirements Management es una herramienta gratuita de gestión de requisitos. Permite definir los requisitos de un proyecto software según la metodología de Durán y Bernárdez. [16]

En este proyecto se ha utilizado esta herramienta para definir los actores, objetivos y los requisitos de información, funcionales y no funcionales en la fase de ingeniería del software.

3.2.5.2. EZEstimate

Herramienta CASE simple diseñada para la estimación de costes y esfuerzo de desarrollo de un proyecto software teniendo en cuenta tanto factores técnicos como

de entorno además del número de actores y casos de uso involucrados en el sistema y su complejidad. [17]

3.2.5.3. Microsoft Project

Herramienta CASE propiedad de Microsoft que destinada a la gestión de proyectos. Esta herramienta ofrece funcionalidades muy útiles para el proceso de gestión como son los diagramas de *Gantt*, calendario laborar, repartición de recursos y planificación temporal, definición de hitos en el proceso de desarrollo, entre otras. [18]

3.2.5.4. Visual Paradigm for UML

Herramienta Case que ofrece todas las funcionalidades necesarias para el proceso de Ingeniería de Software haciendo uso del modelo UML. Algunos de los artefactos que permite elaborar son diagramas de clases, diagramas de paquetes, diagramas de secuencia… entre otras muchas funciones que ofrece como la generación de código fuente y documentación. [19]

3.2.6. Visual Studio Code

Editor de texto de código abierto desarrollado por Microsoft. Cuenta con una amplia comunidad y gran variedad de herramientas que facilitan el desarrollo de aplicaciones en diversos lenguajes, incluida la pila escogida para el desarrollo de SingleTicket. [20]

4. Aspectos relevantes del desarrollo

En este apartado se recogen las partes más significativas de las distintas fases que forman parte de este proyecto. Cada una de estas fases se incluyen de manera más detallada en sus anexos correspondientes.

4.1. Marco de trabajo

El marco de trabajo utilizado en el desarrollo del proyecto es el Proceso Unificado.

El Proceso unificado se adapta a todo tipo de proyectos y es extensible. Este marco de trabajo define un ciclo de vida iterativo e incremental, lo que quiere decir que el ciclo de vida se extenderá a lo largo de diversas iteraciones [21]. En cada una de estas iteraciones, aunque no en la misma medida, se desarrollará cada una de las fases del proceso unificado. Estas fases son:

- Inicio
- Elaboración
- Construcción
- Transición

Las fases del proceso unificado se extienden a lo largo del flujo de trabajo de este marco. Como se puede observar en la Ilustración 4, el flujo de trabajo del Proceso Unificado consiste en:

- Modelado del negocio
- Requisitos
- Análisis
- Diseño
- Implementación
- Pruebas

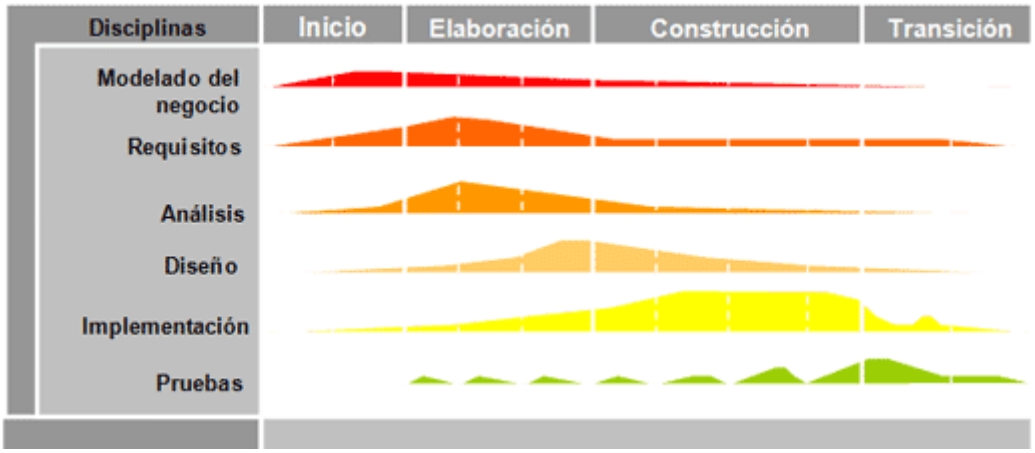

Ilustración 4. Fases del Proceso Unificado

Se escoge este marco de trabajo para el desarrollo de SingleTicket ya que facilita el proceso de estimación temporal permitiendo determinar el tiempo de realización de cada fase a través del seguimiento de hitos.

Para más detalle sobre la aplicación del proceso unificado se puede leer el Anexo I - Planificación temporal del proyecto software.

4.2. Estimación de coste y esfuerzo

En esta sección se analiza el método utilizado para obtener las estimaciones de esfuerzo, así como la duración y el coste del proceso de desarrollo del proyecto. Estas estimaciones se realizan a partir de los factores definidos a continuación.

Para más información sobre las estimaciones de coste y esfuerzo se puede consultar el Anexo I - Planificación temporal del proyecto software.

4.2.1. Actores

Se identifican los actores involucrados en el sistema y se les atribuye una complejidad siguiendo el siguiente criterio:

- Simple: sistema que se comunica mediante una API.
- Medio: sistema que se comunica mediante un protocolo.
- Complejo: persona con interfaz gráfica.

4.2.2. Casos de uso

Se identifican los casos de uso correspondientes a cada funcionalidad y se les asigna una complejidad según el criterio a continuación:

- Simple: 3 o menos pasos.
- Medio: entre 4 y 7 pasos
- Complejo: más de 7 pasos.

4.2.3. Factores

También se identifica la complejidad de los factores técnicos y de entorno dándoles un valor entre 0 y 5 según la importancia del del factor según se muestra en la Tabla 1.

Importancia	Valor
No influye	0
Incidental	1
Moderado	2
Medio	3
Significativo	4
Esencial	5

Tabla 1. Importancia de los factores

4.2.3.1. Factores técnicos

Factores que influyen directamente a la complejidad técnica de la aplicación en diversos aspectos. En la Tabla 2 se puede observar los valores asignados.

Nombre del factor	Valor
Sistemas distribuidos	2
Rendimiento	3
Eficiencia del usuario final	1
Procesamiento Interno Complejo	2
Reusabilidad	2
Facilidad de instalación	1
Facilidad de uso	5
Portabilidad	4
Facilidad de cambio	3
Concurrencia	0
Características especiales de seguridad	2
Acceso directo a terceras partes	0
Se requiere entrenamiento especial del usuario	0

Tabla 2. Factores técnicos

4.2.3.2. Factores de entorno

Factores derivados de la experiencia y conocimiento del equipo de trabajo. En la Tabla 3 se pueden observar los valores asignados.

Nombre del factor	Valor
Familiaridad con UML	3
Trabajadores a tiempo parcial	0
Capacidad de los analistas	3
Experiencia en la aplicación	3
Experiencia en la orientación a objetos	4
Motivación	5
Dificultad del lenguaje de programación	3
Estabilidad de los requisitos	3

Tabla 3. Factores de entorno

4.2.4. Resultado obtenido

El resultado de la estimación de coste y esfuerzo elaborada por EZEstimate se puede ver en la Tabla 4

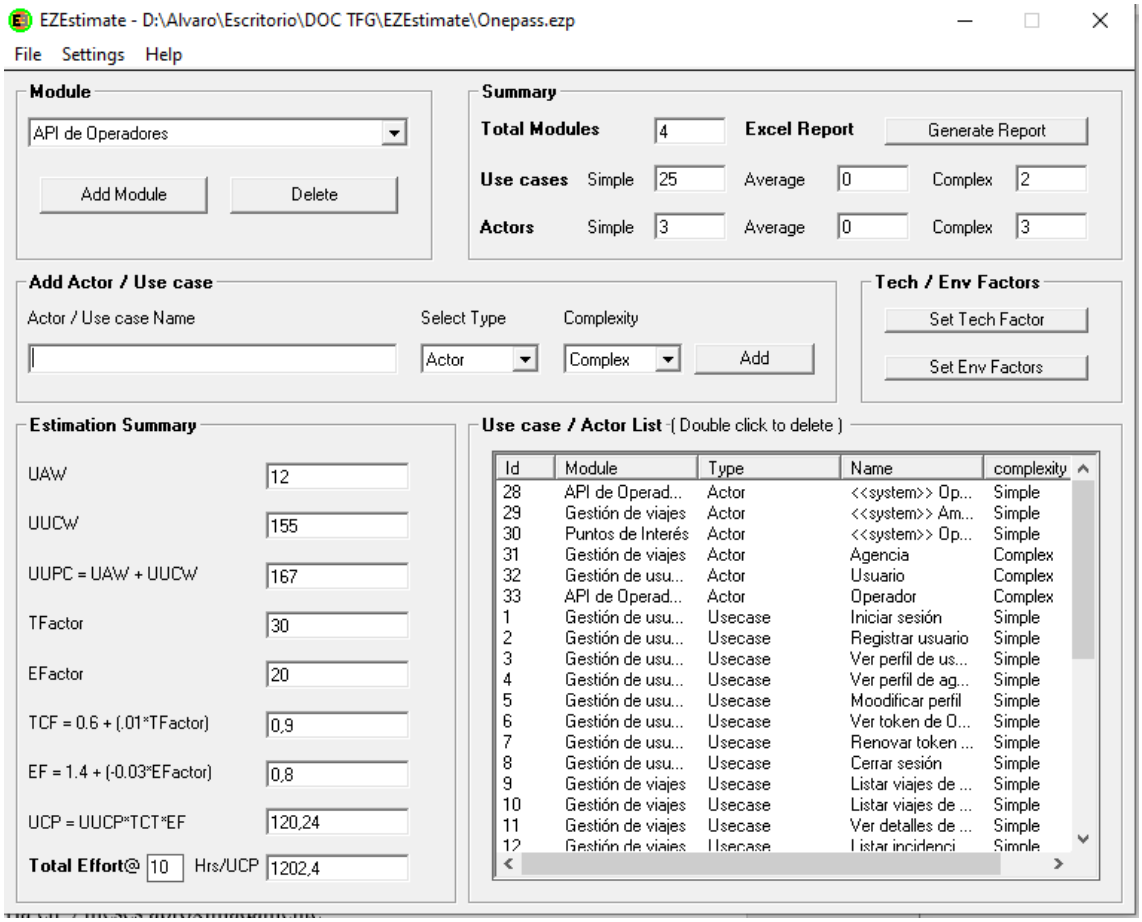

Tabla 4. Estimación de coste y esfuerzo

El esfuerzo en horas empleadas por cada caso de uso se ha fijado en 10 debido a que todos los casos de uso exceptuando dos son simples y comparten similitudes que permiten reciclar parte del código.

4.3. Planificación Temporal

Se identifican las tareas a realizar y se les asigna tiempo y recursos organizándolas a su vez cronológicamente con el fin de minimizar el tiempo de desarrollo.

4.3.1. Planteamiento cronológico

Se definen una serie de iteraciones incrementales a través de las cuales se organiza y optimiza el proceso de desarrollo, dividiéndolo por módulos de funcionalidad.

Las iteraciones planteadas son:

- **Iteración inicial:** se investigará sobre el estado del arte de cada funcionalidad principal del proyecto buscando obtener unos requisitos, análisis, diseño y arquitectura iniciales. Toda especificación inicial será susceptible a cambios durante las siguientes iteraciones, en las cuales se refinarán estas.
  Además, se preparará el entorno de desarrollo y realizarán los desarrollos iniciales relacionados con la arquitectura del sistema y la gestión de usuarios para en siguientes iteraciones desplegar el resto de los componentes según vayan estando listos.
- **Iteración obtención de puntos de interés:** se investigará sobre cómo realizar la obtención de puntos de interés
  También se investigará sobre las técnicas más actuales y qué tecnologías utilizar para esta funcionalidad.
  Además, se pulirán los requisitos, análisis y diseño de esta funcionalidad del sistema. Después, se realizará la implementación de los componentes relacionados con esta funcionalidad y se integrarán en el entorno de desarrollo.
- **Iteración API de Operadores:** se investigará sobre cómo realizar la API de operadores.
  También se investigará sobre las técnicas más actuales y qué tecnologías utilizar para esta funcionalidad.
  Además, se pulirán los requisitos, análisis y diseño de esta funcionalidad del sistema. Después, se realizará la implementación de los componentes relacionados con esta funcionalidad y se integrarán en el entorno de desarrollo.
- **Iteración gestión de billetes de transporte multimodales:** se investigará sobre cómo realizar la gestión de billetes de transporte multimodales.
  También se investigará sobre las técnicas más actuales y qué tecnologías utilizar para esta funcionalidad.
  Además, se pulirán los requisitos, análisis y diseño de esta funcionalidad del sistema. Después, se realizará la implementación de los componentes relacionados con esta funcionalidad y se integrarán en el entorno de desarrollo.
- **Iteración planificación de viajes:** se investigará sobre cómo realizar la planificación de viajes.
  También se investigará sobre las técnicas más actuales y qué tecnologías utilizar para esta funcionalidad.
  Además, se pulirán los requisitos, análisis y diseño de esta funcionalidad del sistema. Después, se realizará la implementación de los componentes relacionados con esta funcionalidad y se integrarán en el entorno de desarrollo.

4.3.2. Asignación temporal

Se asigna una cantidad estimada de días de duración a cada tarea a llevar a cabo, así como los recursos que se le van a dedicar (En este casi siempre el 100% de Álvaro, que es el único desarrollador del proyecto)

De la Ilustración 5 a la Ilustración 15se puede observar está asignación utilizando Microsoft Project.

Inicio del proyecto	0 días	mar 01/03/2	mar 01/03/2		
Iteración inicial	**50 días**	**mar 01/03/2**	**mié 27/04/2**		
Modelado de negocio	**17 días**	**mar 01/03/22**	**sáb 19/03/22**		
Investigación sobre el estado del arte del problema	3 días	mar 01/03/22	jue 03/03/22	1	Álvaro
Investigación sobre arquitecturas y soluciones existentes	11 días	vie 04/03/22	mié 16/03/22	4	Álvaro
Invesitgacion de las tecnologías más apropiadas para el desarrollo del sistema	3 días	jue 17/03/22	sáb 19/03/22	5	Álvaro
Requisitos	**1 día**	**lun 21/03/22**	**lun 21/03/22**		
Modelo de requisitos inicial	1 día	lun 21/03/22	lun 21/03/22	6	Álvaro
Análisis	**6 días**	**mar 22/03/2**	**lun 28/03/22**		
Análisis del subsistema de gestión de usuarios	1 día	mar 22/03/22	mar 22/03/22	8	Álvaro
Análisis del subsistema de gestión de billeres de transporte multimodales	2 días	mié 23/03/22	jue 24/03/22	10	Álvaro
Análisis del subsistema de la API de Operadores	1 día	vie 25/03/22	vie 25/03/22	11	Álvaro

Ilustración 5. Iteración inicial 1

Análisis del subsistema de puntos de interés	1 día	sáb 26/03/22	sáb 26/03/22	12	Álvaro
Análisis del subsistema de planificación de viajes	1 día	lun 28/03/22	lun 28/03/22	13	Álvaro
Diseño	**12 días**	**mar 29/03/2**	**lun 11/04/22**		
Diseño de la arquitectura inicial	1,5 días	mar 29/03/22	mié 30/03/22	14	Álvaro
Diseño inicial del subsistema de gestión de usuarios	1,5 días	mié 30/03/22	jue 31/03/22	16	Álvaro
Diseño inicial del subsistema de gestión de billetes de transporte multimodal	3 días	vie 01/04/22	lun 04/04/22	17	Álvaro
Diseño inicial del subsistema de la API de Operadores	1,5 días	mar 05/04/22	mié 06/04/22	18	Álvaro
Diseño inicial del subsistema de puntos de interés	3 días	mié 06/04/22	sáb 09/04/22	19	Álvaro
Diseño del subsistema de planificación de viajes	1,5 días	sáb 09/04/22	lun 11/04/22	20	Álvaro
Implementación	**12 días**	**mar 12/04/2**	**lun 25/04/22**		
Preparación del entorno de desarrollo	1 día	mar 12/04/22	mar 12/04/22	21	Álvaro
Despliegue del sistema de comunicación	4 días	mié 13/04/22	sáb 16/04/22	23	Álvaro

Ilustración 6. Iteración inicial 2

Implementació de interfaz de comunicación de subsistemas inicial	3 días	lun 18/04/22	mié 20/04/22	24	Álvaro
Implementació del subsistema de gestión de usuarios	4 días	jue 21/04/22	lun 25/04/22	25	Álvaro
Pruebas	**2 días**	**mar 26/04/2**	**mié 27/04/2**		
Pruebas del subsistema de gestión de usuarios	1 día	mar 26/04/22	mar 26/04/22	26	Álvaro
Pruebas del sistema de comunicación entre subsistemas	1 día	mié 27/04/22	mié 27/04/22	28	Álvaro
Hito fin fe iteración inicial	0 días	mié 27/04/22	mié 27/04/22	29	

Ilustración 7. Iteración inicial 3

Iteración obtención de puntos de interés	**26,5 días**	**jue 28/04/22**	**sáb 28/05/22**		
Modelado de negocio	**2 días**	**jue 28/04/22**	**vie 29/04/22**		
Investigación sobre las fuentes para la estracción de puntos de interés	2 días	jue 28/04/22	vie 29/04/22	30	Álvaro
Requisitos	**1 día**	**sáb 30/04/22**	**sáb 30/04/22**		
Refinamiento de los requisitos del subsistema de puntos de interés	1 día	sáb 30/04/22	sáb 30/04/22	33	Álvaro
Análisis	**1 día**	**lun 02/05/22**	**lun 02/05/22**		
Refinamiento del análisis del subsistema de puntos de interés	1 día	lun 02/05/22	lun 02/05/22	35	Álvaro
Diseño	**0,5 días**	**mar 03/05/2**	**mar 03/05/2**		
Refinamiento del diseño edl subsistema de puntos de interés	0,5 días	mar 03/05/22	mar 03/05/22	37	Álvaro
Implementación	**19 días**	**mar 03/05/2**	**mié 25/05/2**		
Implementació del subsistema de puntos de interés	18 días	mar 03/05/22	mar 24/05/22	39	Álvaro
Integración y despliegue de subsistemas en entorno de desarrollo	1 día	mar 24/05/22	mié 25/05/22	41	Álvaro

Ilustración 8. Iteración de obtención de puntos de interés 1

Pruebas	**3 días**	**mié 25/05/2**	**sáb 28/05/22**		
Pruebas del subsistema de puntos de interés	2 días	mié 25/05/22	vie 27/05/22	42	Álvaro
Pruebas de integración	1 día	vie 27/05/22	sáb 28/05/22	44	Álvaro
Hito fin de iteración puntos de interés	0 días	sáb 28/05/22	sáb 28/05/22	45	

Ilustración 9. Iteración de obtención de puntos de interés 2

Iteración API de Operadores	**15 días**	**sáb 28/05/22**	**mié 15/06/22**		
Modelado de negocio	**3 días**	**sáb 28/05/22**	**mié 01/06/22**		
Investigación sobre técnicas para realizar una API	3 días	sáb 28/05/22	mié 01/06/22	46	Álvaro
Requisitos	**1 día**	**mié 01/06/2**	**jue 02/06/22**		
Refinamiento de los requisitos del subsistema de API de operadores	1 día	mié 01/06/22	jue 02/06/22	49	Álvaro
Análisis	**1 día**	**jue 02/06/22**	**vie 03/06/22**		
Refinamiento del análisis del subsistema de API de operadores	1 día	jue 02/06/22	vie 03/06/22	51	Álvaro
Diseño	**1 día**	**vie 03/06/22**	**sáb 04/06/22**		
Refinamiento del diseño del subsistema de API de operadores	1 día	vie 03/06/22	sáb 04/06/22	53	Álvaro
Implementación	**7 días**	**sáb 04/06/22**	**lun 13/06/22**		
Implementació del subsistema de API de operadores	6 días	sáb 04/06/22	sáb 11/06/22	55	Álvaro
Integración y despliegue de subsistemas en entorno de desarrollo	1 día	sáb 11/06/22	lun 13/06/22	57	Álvaro

Ilustración 10. Iteración API de Operadores

Pruebas	**2 días**	**lun 13/06/22**	**mié 15/06/2**		
Pruebas del subsistema de API de operadores	1 día	lun 13/06/22	mar 14/06/22	58	Álvaro
Pruebas de integración	1 día	mar 14/06/22	mié 15/06/22	60	Álvaro
Hito fin de iteración API de OPERADORES	0 días	mié 15/06/22	mié 15/06/22	61	

Ilustración 11. Iteración API de Operadores 2

Iteración planificación de viajes	**24 días**	**mié 15/06/22**	**mié 13/07/22**			
Modelado de negocio	**7 días**	**mié 15/06/22**	**jue 23/06/22**			
Investigación sobre técnicas y soluciones existentes para la planificación de viajes	7 días	mié 15/06/22	jue 23/06/22	62	Álvaro	
Requisitos	**1 día**	**jue 23/06/22**	**vie 24/06/22**			
Refinamiento de los requisitos del subsistema de planificación de viajes	1 día	jue 23/06/22	vie 24/06/22	65	Álvaro	
Análisis	**1 día**	**vie 24/06/22**	**sáb 25/06/22**			
Refinamiento del análisis del subsistema de planificación de viajes	1 día	vie 24/06/22	sáb 25/06/22	67	Álvaro	
Diseño	**1 día**	**sáb 25/06/22**	**lun 27/06/22**			
Refinamiento del diseño del subsistema de planificación de viajes	1 día	sáb 25/06/22	lun 27/06/22	69	Álvaro	
Implementación	**9 días**	**lun 27/06/22**	**jue 07/07/22**			
Implementació del subsistema de planificación	8 días	lun 27/06/22	mié 06/07/22	71	Álvaro	
Integración y despliegue de subsistemas en entorno de desarrollo	1 día	mié 06/07/22	jue 07/07/22	73	Álvaro	

Ilustración 12. Iteración de planificación de viajes 1

Pruebas	**5 días**	**jue 07/07/22**	**mié 13/07/2**		
Pruebas del subsistema de planificación de viajes	4 días	jue 07/07/22	mar 12/07/22	74	Álvaro
Pruebas de integración	1 día	mar 12/07/22	mié 13/07/22	76	Álvaro
Hito fin de iteración de planificación de viajes	0 días	mié 13/07/22	mié 13/07/22	77	

Ilustración 13. Iteración de gestión de billetes de panificación de viajes 2

Iteración de gestión de billetes de transporte multimodales	**32 días**	**mié 13/07/22**	**vie 19/08/22**		
Modelado de negocio	**4 días**	**mié 13/07/22**	**lun 18/07/22**		
Investigación sobre técnicas y soluciones existentes para la gestión de billetes multimodales	4 días	mié 13/07/22	lun 18/07/22	78	Álvaro
Requisitos	**1 día**	**lun 18/07/22**	**mar 19/07/2**		
Refinamiento de los requisitos del subsistema de gestión de billetes multimodales	1 día	lun 18/07/22	mar 19/07/22	81	Álvaro
Análisis	**1 día**	**mar 19/07/2**	**mié 20/07/2**		
Refinamiento del análisis del subsistema de gestión de billetes multimodales	1 día	mar 19/07/22	mié 20/07/22	83	Álvaro
Diseño	**1 día**	**mié 20/07/2**	**jue 21/07/22**		
Refinamiento del diseño edl subsistema de gestión de billetes multimodales	1 día	mié 20/07/22	jue 21/07/22	85	Álvaro

Ilustración 14. Iteración de gestión de billetes de transporte multimodales 1

Implementación	**21 días**	**jue 21/07/22**	**lun 15/08/22**		
Implementació del subsistema de gestión de billetes multimodales	20 días	jue 21/07/22	sáb 13/08/22	87	Álvaro
Integración y despliegue de subsistemas en entorno de desarrollo	1 día	sáb 13/08/22	lun 15/08/22	89	Álvaro
Pruebas	**4 días**	**lun 15/08/22**	**vie 19/08/22**		
Pruebas del subsistema de gestión de billetes multimodales	3 días	lun 15/08/22	jue 18/08/22	90	Álvaro
Pruebas de integración	1 día	jue 18/08/22	vie 19/08/22	92	Álvaro
Hito fin de iteración gestión de billetes multimodales	0 días	vie 19/08/22	vie 19/08/22	93	

Ilustración 15. Iteración de gestión de billetes de transporte multimodales

4.3.3. Resultado obtenido

Tras la asignación temporal se obtuvo el diagrama de Gantt del proyecto usando la misma herramienta, Microsoft Project. En este diagrama se puede observar de forma gráfica la extensión temporal del proyecto desglosada por tareas e hitos.

Como se puede ver de la a la se obtiene una duración del proyecto similar a la obtenida con la herramienta EZEstimate, lo cual es un buen indicador de que la planificación es coherente y correcta.

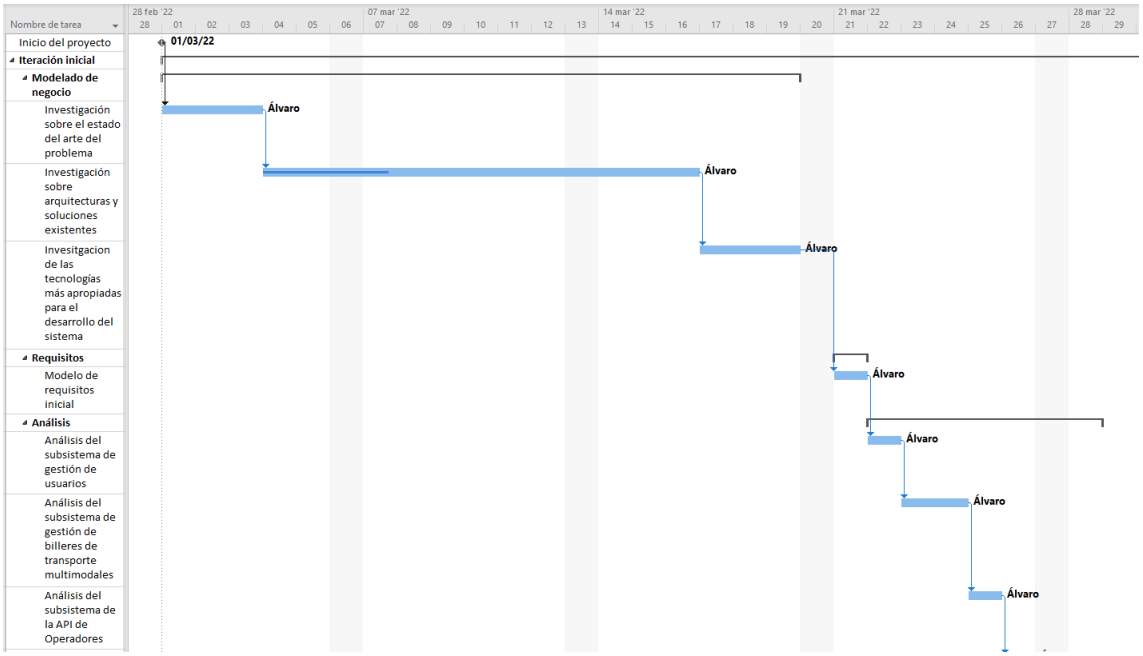

Ilustración 16. Diagrama de Gantt 1

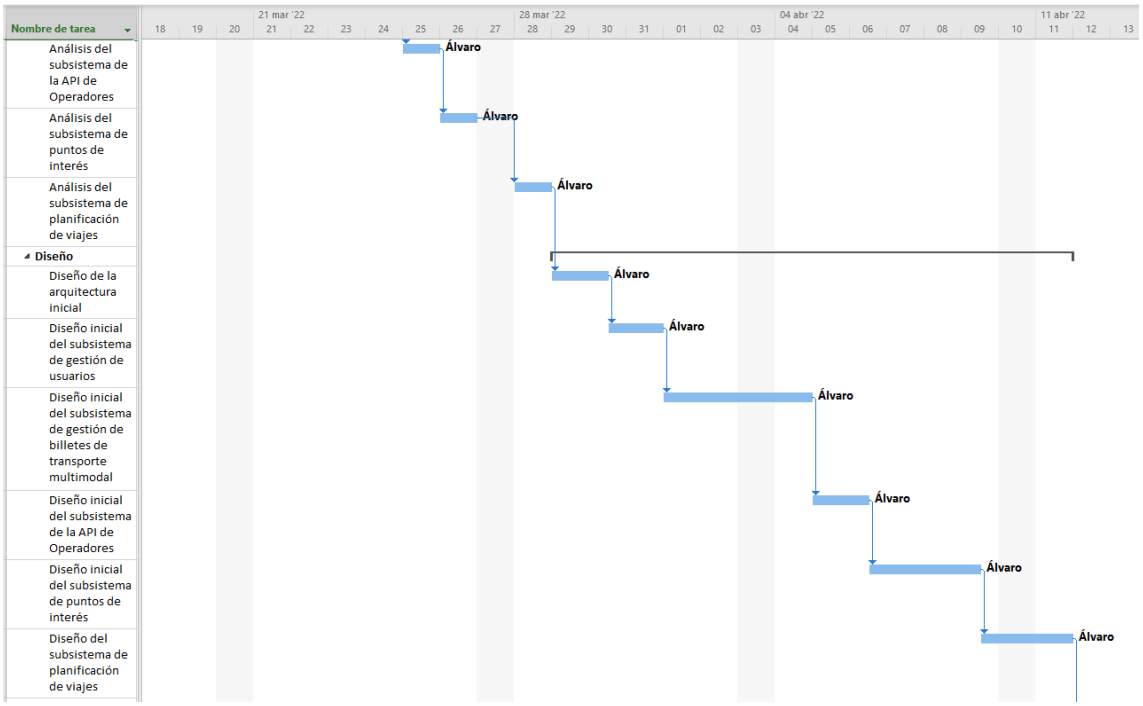

Ilustración 17. Diagrama de Gantt 2

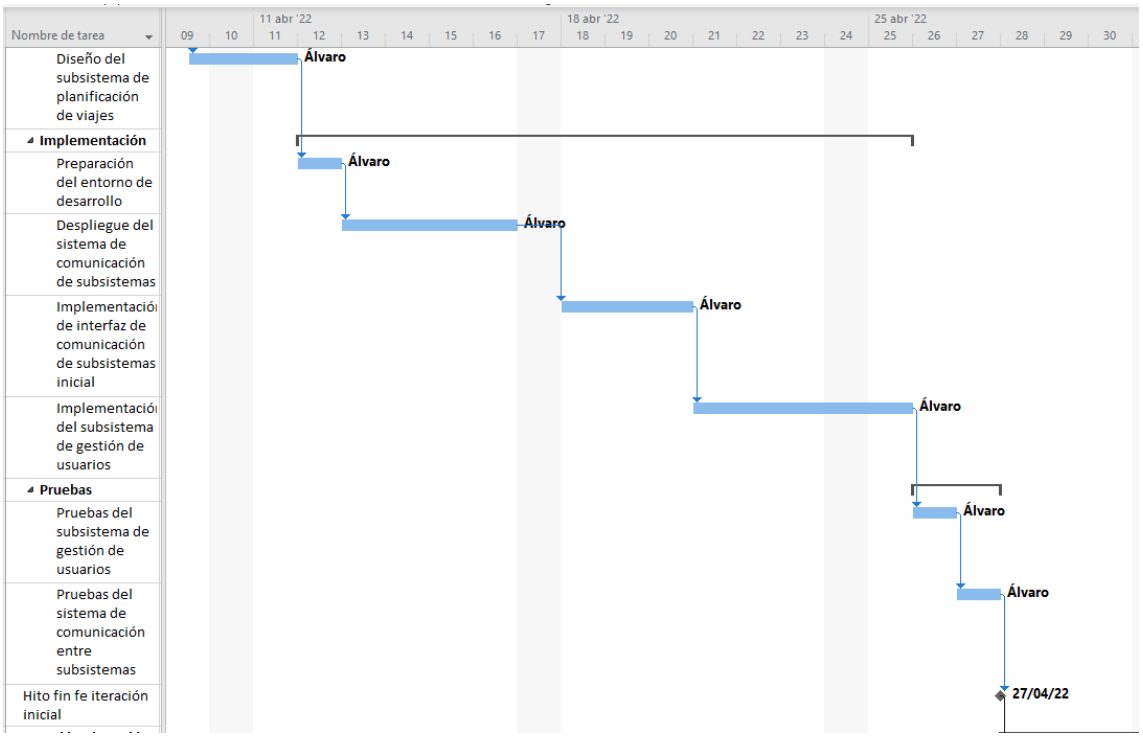

Ilustración 18. Diagrama de Gantt 3

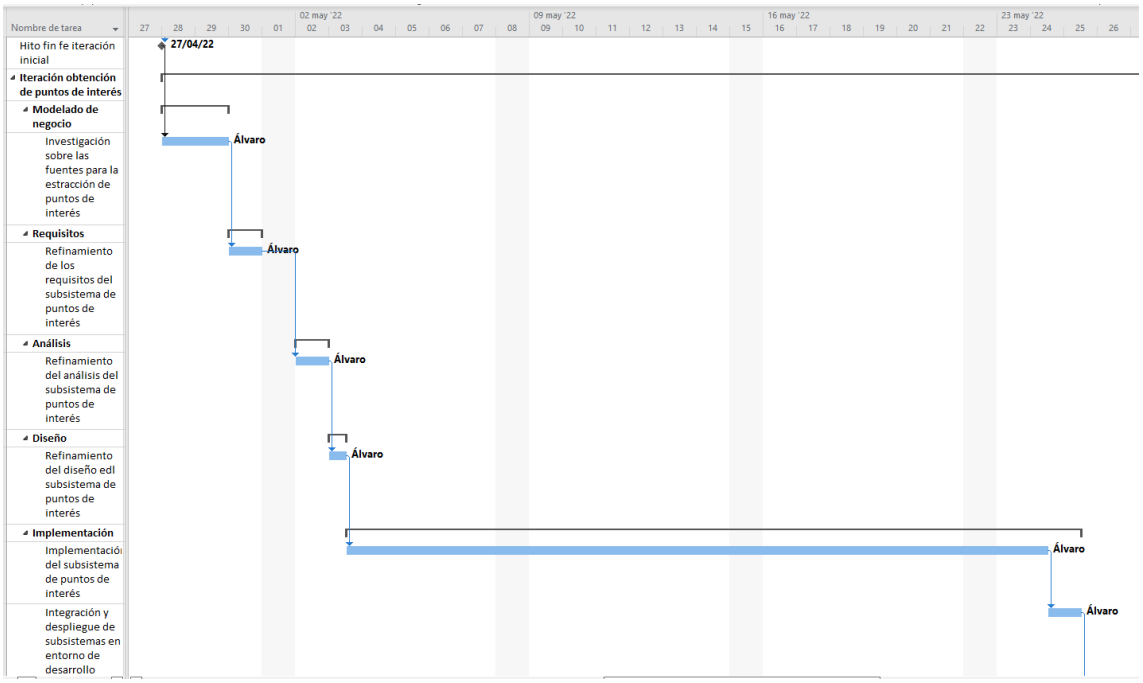

Ilustración 19. Diagrama de Gantt 4

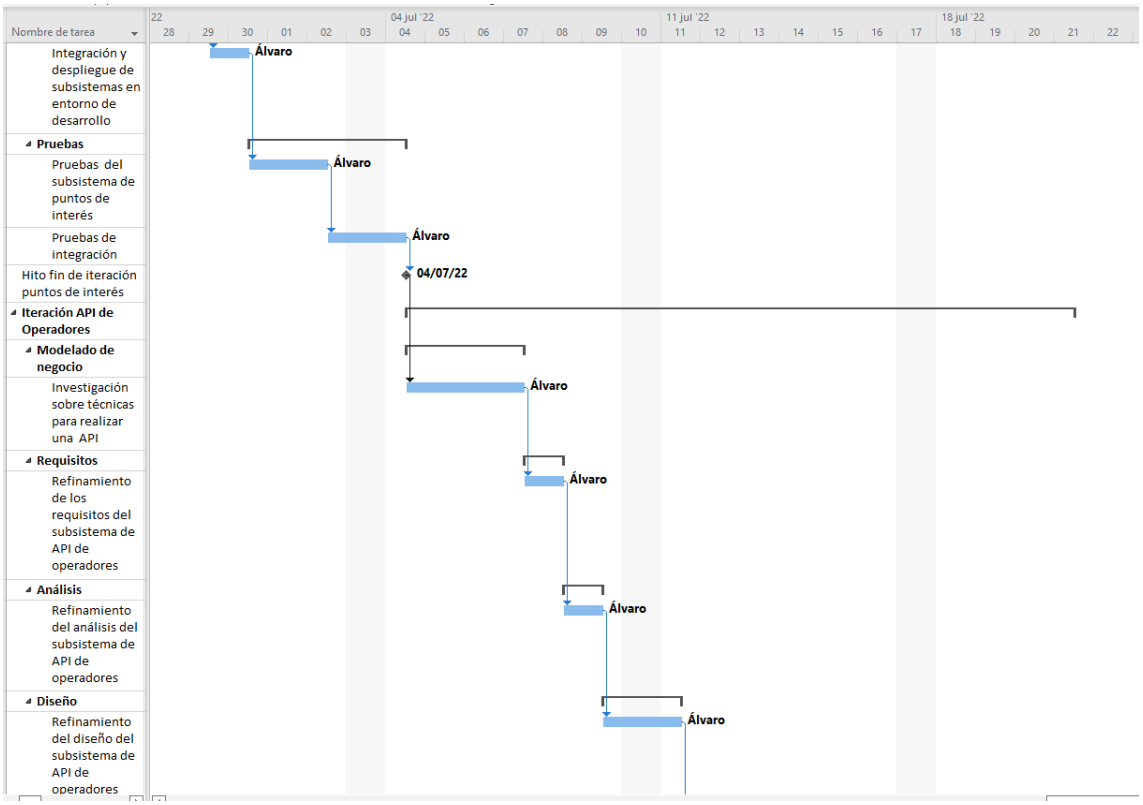

Ilustración 20. Diagrama de Gantt 5

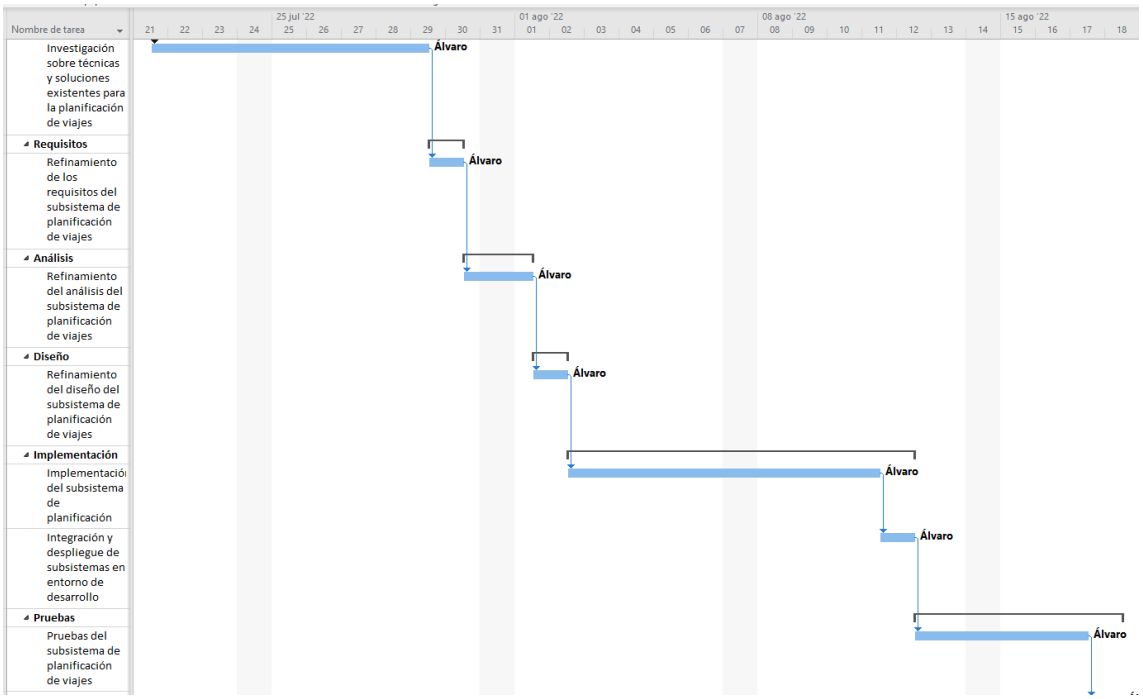

Ilustración 21. Diagrama de Gantt 6

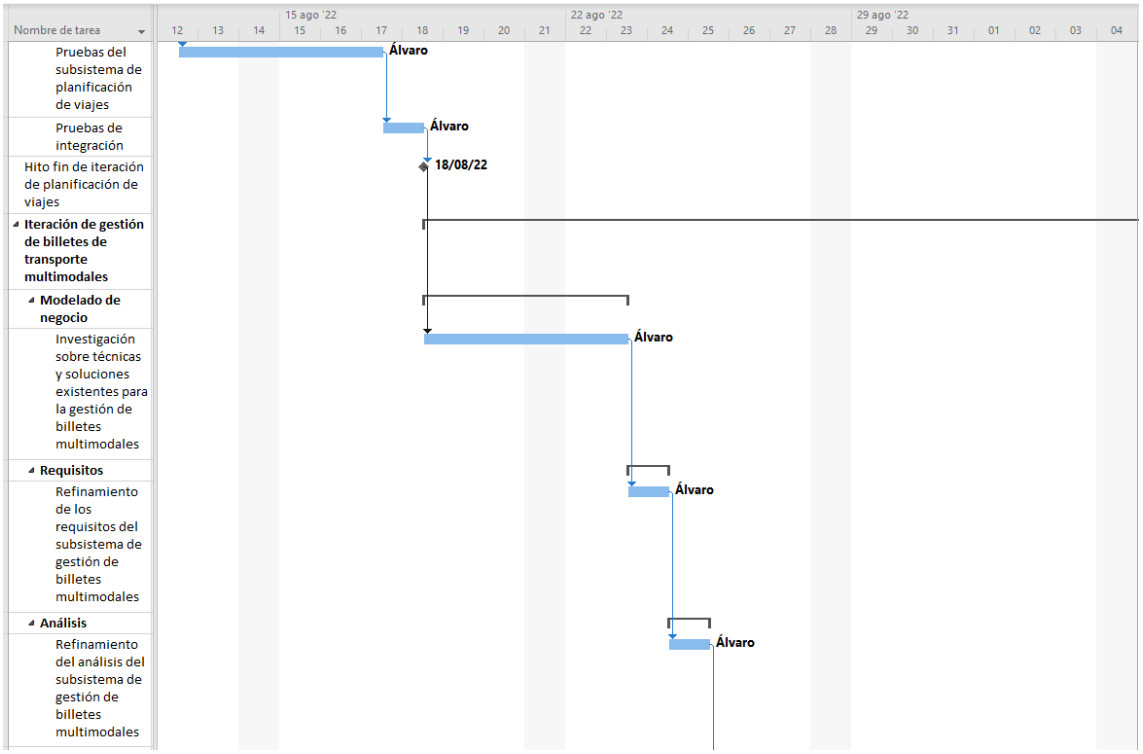

Ilustración 22. Diagrama de Gantt 7

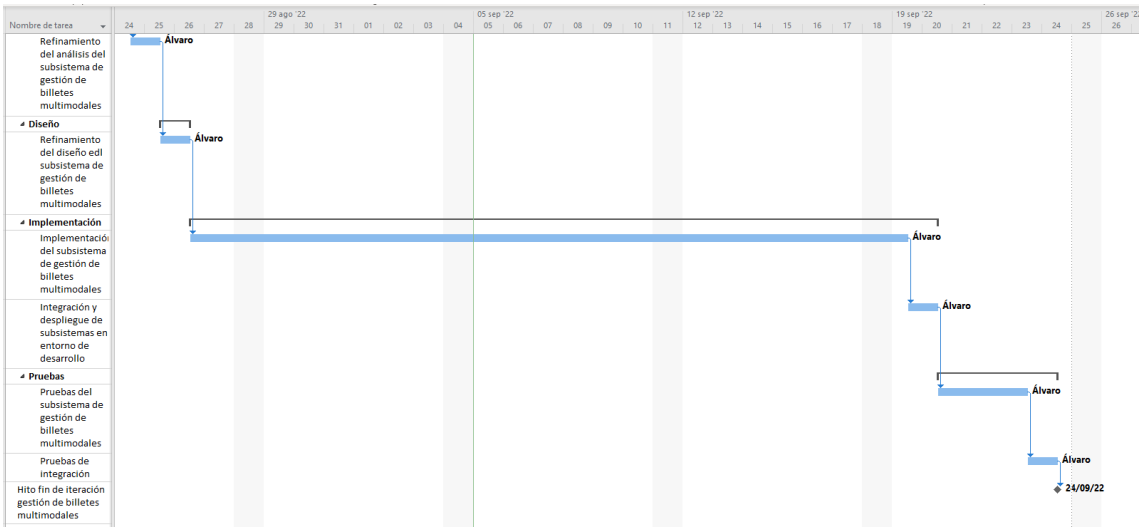

Ilustración 23. Diagrama de Gantt 7

Para más detalles sobre la planificación temporal consultar el Anexo I.

4.4. Especificación de los requisitos

La especificación de requisitos se basa en definir los actores, objetivos, requisitos no funcionales, requisitos de información y requisitos funcionales que el sistema debe cumplir.

El sistema por definir es una plataforma de gestión de billetes de transporte multimodal comunes a un mismo viaje.

Cada cliente puede consultar una lista con sus viajes. Puede también ver los detalles de estos viajes, donde se pueden ver los datos de la agencia, usuario, lista de billetes que conforman el trayecto del viaje e incidencias sobre el viaje. Todo esto también lo puede consultar una agencia sobre su viaje.

El usuario puede abrir una nueva incidencia sobre un viaje si lo desea siempre y cuando este viaje no tenga ninguna incidencia abierta sin resolver previamente.

En los detalles de una incidencia tanto el usuario como la agencia pueden intercambiar mensajes hasta resolverla. Una vez resuelta la agencia cierra la incidencia pulsando un botón. Las incidencias se pueden ver también a través de un listado de todas ellas.

El sistema cuenta con un planificador de viajes, donde la agencia puede buscas y contratar vuelos haciendo el sistema uso de *APIs* externas. Además, el sistema ofrecerá transporte desde las ciudades origen y destino y la localidad de los aeropuertos basándose en las rutas previamente añadidas al sistema por parte de los operadores desde el api que se les ofrece.

Para más detalles sobre la especificación de los requisitos, consultar el Anexo II- Especificación de los requisitos del software.

4.4.1. Actores del sistema

Muestra información sobre los actores que participan en las funcionalidades del sistema.

En el sistema participan un total de 6 actores:

- Usuario: actor que representa al cliente que interactúa con el sistema.
- Agencia: actor que representa a la agencia que interactúa con el sistema.
- Operador: actor que al operador que interactúa con el sistema.
- <<*system*>> Operador: actor que representa la *API* de operadores.
- <<*system*>> OpenStreetMaps: actor que representa la *API* de OpenStreetMaps.
- <<*system*>> Amadeus: actor que representa la *API* de gestión de vuelos de la empresa Amadeus.

4.4.2. Objetivos del sistema

Objetivos que se desea alcanzar en el desarrollo del sistema.

Los objetivos por alcanzar en este proyecto son:

- Gestión de usuarios: se gestionará la información de los usuarios que hagan uso del sistema.
- Gestión de API de operadores: el sistema deberá permitir a el actor operador añadir y eliminar las rutas de sus propiedad a través de una *API*

- Gestión de viajes: será necesario la gestión de los viajes añadidos por los actores agencia.
    - Gestión de billetes: el sistema deberá gestionar todos los billetes correspondientes a cada viaje.
    - Gestión de incidencias: el sistema deberá gestionar las incidencias correspondientes a cada viaje
- Planificación de viajes: el sistema deberá aportar las herramientas necesarias a el actor agencia para planificar un viaje de origen a destino.
- Gestión de puntos de interés: el sistema deberá permitir la extracción de la información sobre estaciones de transporte de medios externos al propio sistema

4.4.3. Requisitos de información

Se define la información que se debe almacenar en el sistema para su correcto funcionamiento.

La información necesaria en este proyecto es:

- Información de usuarios: información perteneciente a los usuarios en el sistema
    - Nombre
    - Apellidos
    - Fecha de nacimiento
    - Email
    - Contraseña
    - Número de teléfono
    - Documento de identidad
    - Dirección
    - Género
    - Tipo
- Información de agencia: información correspondiente a los actores agencias registrados en el sistema.
    - Nombre
    - Descripción
    - Token
- Información de operador: información correspondiente a los actores operadores registrados en el sistema.
    - Nombre
    - Descripción
    - Token
- Información de puntos de interés: información correspondiente a las estaciones de tren y bus existentes en las distintas ciudades.
    - Nombre
    - Ciudad
    - Calle
    - Coordenadas
    - Descripción
    - Tipo de transporte
- Información de rutas: información correspondiente a las rutas de transporte de los operadores.

    - Operador
    - Origen
    - Destino
    - Salida
    - Llegada
    - Duración
    - Tipo de transporte
    - Precio
    - Enlace de compra
- Información de viajes: información correspondiente a los viajes planificados por la agencia.
    - Origen
    - Destino
    - Salida
    - Llegada
    - Agencia
    - Usuario
    - Precio
- Información de billetes: información correspondiente a los billetes asociados a cada viaje.
    - Nombre del pasajero
    - Origen
    - Destino
    - Salida
    - Llegado
    - Viaje
    - Asiento
    - Localizador del billete
    - Clase
    - Precio
    - Tipo de transporte
- Información de incidencias: información correspondiente a las incidencias asociadas a un viaje.
    - Fecha
    - Viaje
    - Usuario
    - Agencia
    - Asunto
    - Descripción
    - Estado
    - Mensajes
- Información de ciudades: información correspondiente a las ciudades relativas a los puntos de interés almacenados.
    - Nombre
    - País

4.4.4. Requisitos no funcionales

Los requisitos no funcionales son propiedades que un sistema debe tener.

Los requisitos no funcionales del sistema son:

- Usabilidad:
- Control de errores: se mostrará mensajes de error compresibles para el usuario.
- Escalabilidad: el sistema deberá ser escalable ante un incremento en el número de agencias, operadores o usuarios.
- Portabilidad: se deberá portar el sistema a otras plataformas de forma parcial o total sin requerir mucho esfuerzo.
- Extensibilidad: el sistema deberá ser extensible en cuanto a medios de transportes con los que se trabaja.
- Seguridad: se debe garantizar seguridad en las distintas conexiones del sistema.

Los requisitos no funcionales se realizan como muestra la **¡Error! No se encuentra el o rigen de la referencia.**

4.4.5. Requisitos funcionales

Los requisitos funcionales describen cada función que el sistema debe ofrecer.

Requisitos funcionales obtenidos como solución para cada módulo planteado en el sistema:

- Gestión de usuarios:
  - Iniciar sesión
  - Registrar usuario
  - Ver perfil de usuario
  - Ver perfil de agencia
  - Ver token de operador
  - Modificar perfil
  - Renovar token del operador
  - Cerrar sesión
- Gestión de API de operadores:
  - Extraer rutas
  - Extraer puntos de interés
  - Extraer ciudades
  - Añadir ruta
  - Eliminar ruta
- Gestión de viajes:
  - Listar viajes de usuario
  - Listar viajes de agencia
  - Ver detalles de viaje
  - Añadir billete a viaje
  - Ver detalles de incidencia
  - Añadir incidencia a viaje
  - Enviar mensaje sobre incidencia
  - Cerrar incidencia
- Planificación de viajes:
  - Planificar viaje
- Gestión de puntos de interés:
  - Obtener puntos de interés

Se puede ver la totalidad de los requisitos funcionales del sistema en el Anexo II – Especificación de requisitos del software.

4.5. Diseño y almacenamiento de datos.

Basándose en los módulos funcionales utilizados anteriormente para la definición de las iteraciones se debe diseñar un modelo para el almacenamiento de datos. Para esto primero se plantean una serie de clases requeridas para el almacenamiento de los datos necesarios para el correcto y completo funcionamiento de cada funcionalidad del sistema.

Para comenzar, en referencia a la gestión de usuarios, se necesita una clase *Users* donde se almacenan los datos de cada usuario, nombre, apellidos, email, fecha de nacimiento, dirección…

Una vez cubierta esa necesidad funcional de la gestión de usuarios, la siguiente necesidad a cubrir es la obtención de los puntos de interés. Se desea almacenar una serie de puntos de interés que posteriormente accederán los operadores a través de la *API* que se les proporciona para introducir sus rutas al sistema. Para esto será necesaria una clase POI donde se almacenen todos los atributos relevantes que estos puntos de interés puedan contener. Además, harán falta una clase *Streets*, una clase *Cities*, *y una clase Countries*, para almacenar todos los posibles registros de estos una sola vez y compartir esa información entre todos los puntos de interés en los que coincidan, ahorrando así espacio en memoria.

Como se menciona en el párrafo anterior, se le proporciona una *API* a los operadores para que introduzcan accedan a los datos necesarios para dar a sus rutas un formato compatible con el sistema y tras esto introducirlas en el para que las agencias puedan hacer uso de ellas. Para esto lo primero que se necesitará es una clase *Operators* que almacene los datos del operados, así como un token generado aleatoriamente para proporcionarles acceso al sistema a través de la *API.* También se necesita una clase *Routes* para almacenar todos los datos referentes a las rutas de los operadores.

En cuando a la gestión de billetes de transporte multimodales, lo primero que se necesitan son tanto una clase *Tickets*, como una clase *Travels* para almacenar los datos de los viajes y sus billetes. Como los viajes y billetes se los proporciona al usuario una agencia se necesita una clase *Agencies* para identificar las diferentes agencias y sus usuarios. Por último, para implementar el sistema de incidencias a través del cual se pueda resolver cualquier duda o problema con el viaje, se añadirán una clase *Compalints* y una clase *Complaint_messages* para los mensajes intercambiados entre usuario y agencia para resolver la incidencia.

En cuanto al módulo de planificación de viajes, sus necesidades en cuanto al almacenamiento de datos quedan cubiertas con las clases definidas para el resto de los módulos, por lo que no se necesita añadir ninguna clase adicional.

Todas estas clases se pueden ver representadas en la Ilustración 24.

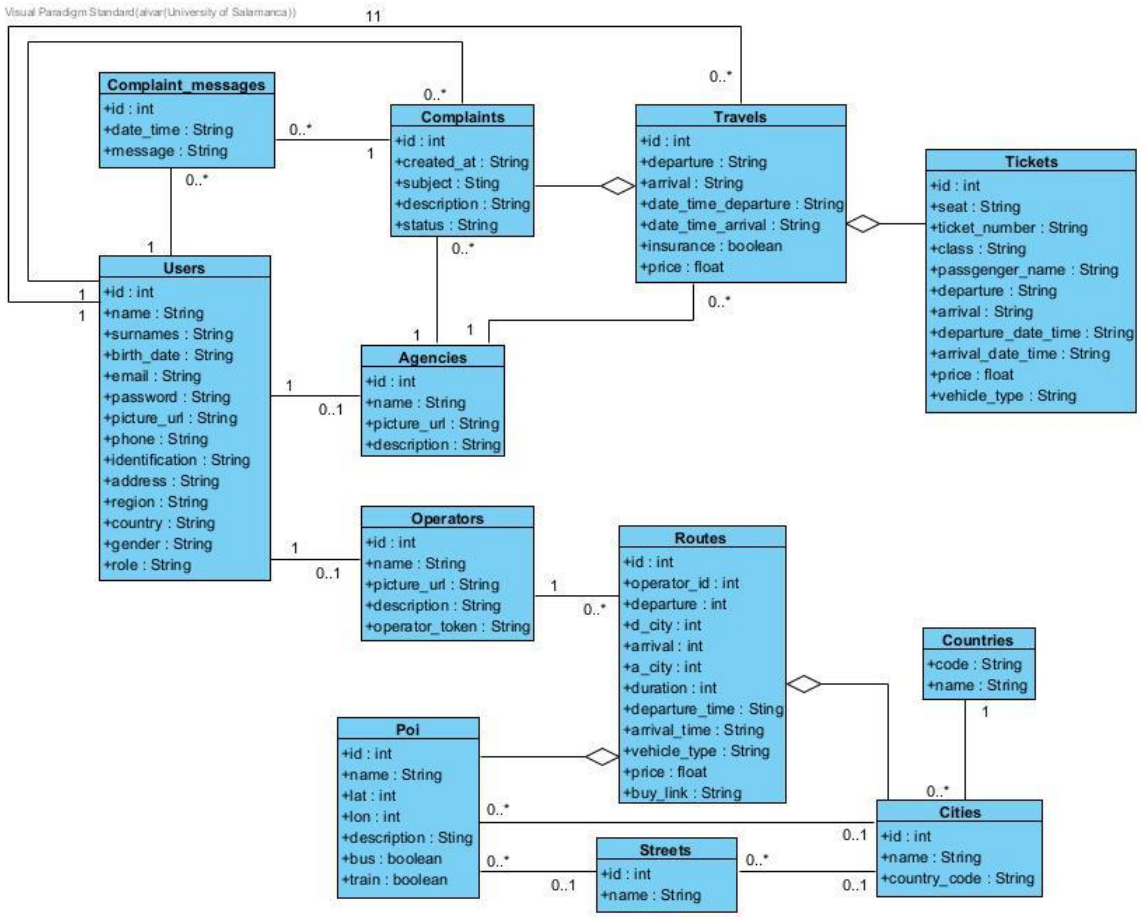

Ilustración 24. Diagrama de clases de análisis

Una vez llevado este concepto a una base de datos relacional SQL como la empleada por el sistema, se obtienen las tablas y relaciones observables en la Ilustración 25 descritas anteriormente.

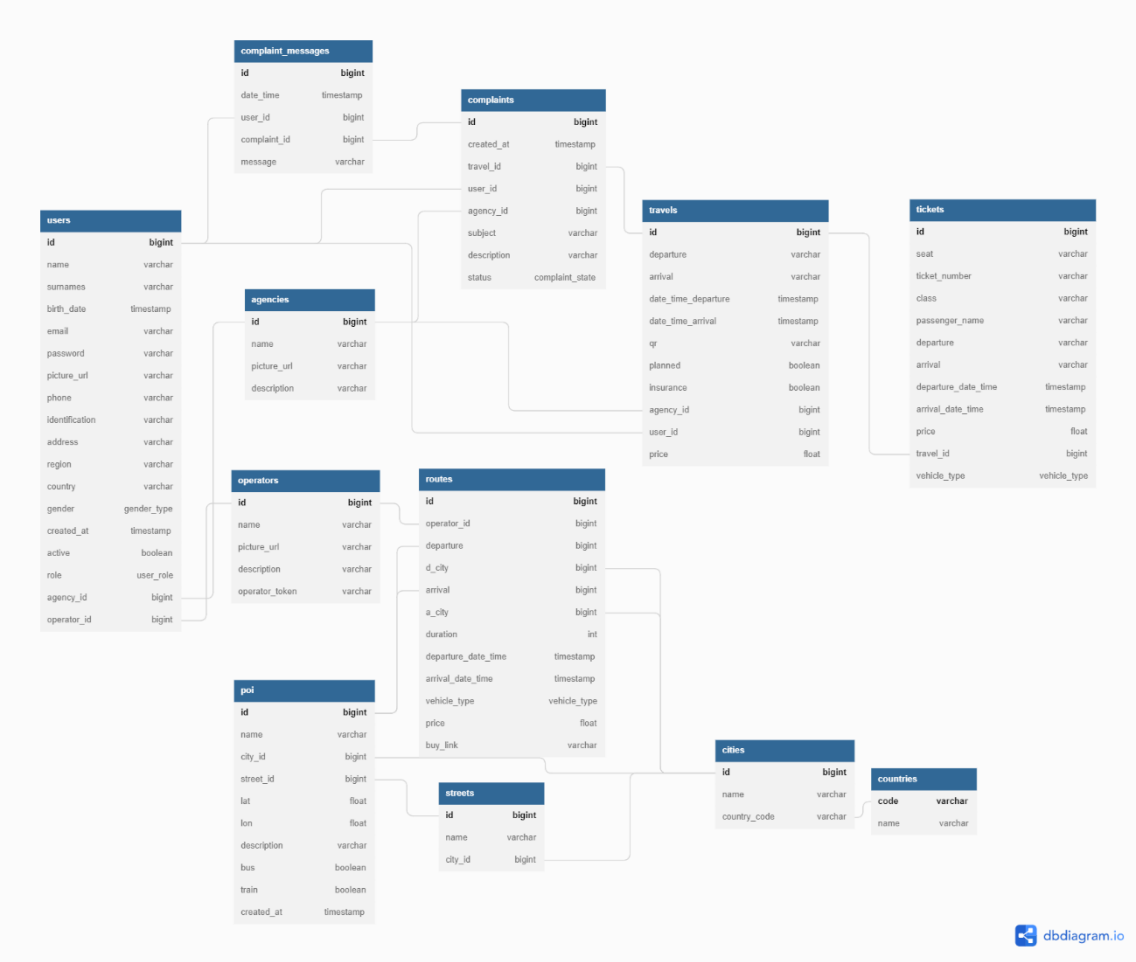

Ilustración 25. Diagrama de base de datos realizado en dbdiagram.io

4.6. Análisis y diseño de procedimientos

Para las fases de análisis y diseño de los procedimientos del sistema se han realizado los casos de uso definidos en la fase de especificación de requisitos en modo de diagramas de secuencia de análisis y diagramas de secuencia de diseño. En este punto se presentan los diagramas de los casos de uso más relevantes de cada uno de los módulos funcionales presentados anteriormente en el documento. Para visualizar todos los diagramas elaborados en la fase de análisis y la fase de diseño consular el Anexo III – Especificación de diseño

4.6.1. Gestión de usuarios

En relación con el módulo Gestión de usuario cuyo objetivo es desarrollar un sistema de usuarios donde estos puedan acceder a diferentes funcionalidades de la plataforma según su rol en las ilustraciones Ilustración 26 e Ilustración 27 se muestran los diagramas de la realización del caso de uso de Iniciar sesión.

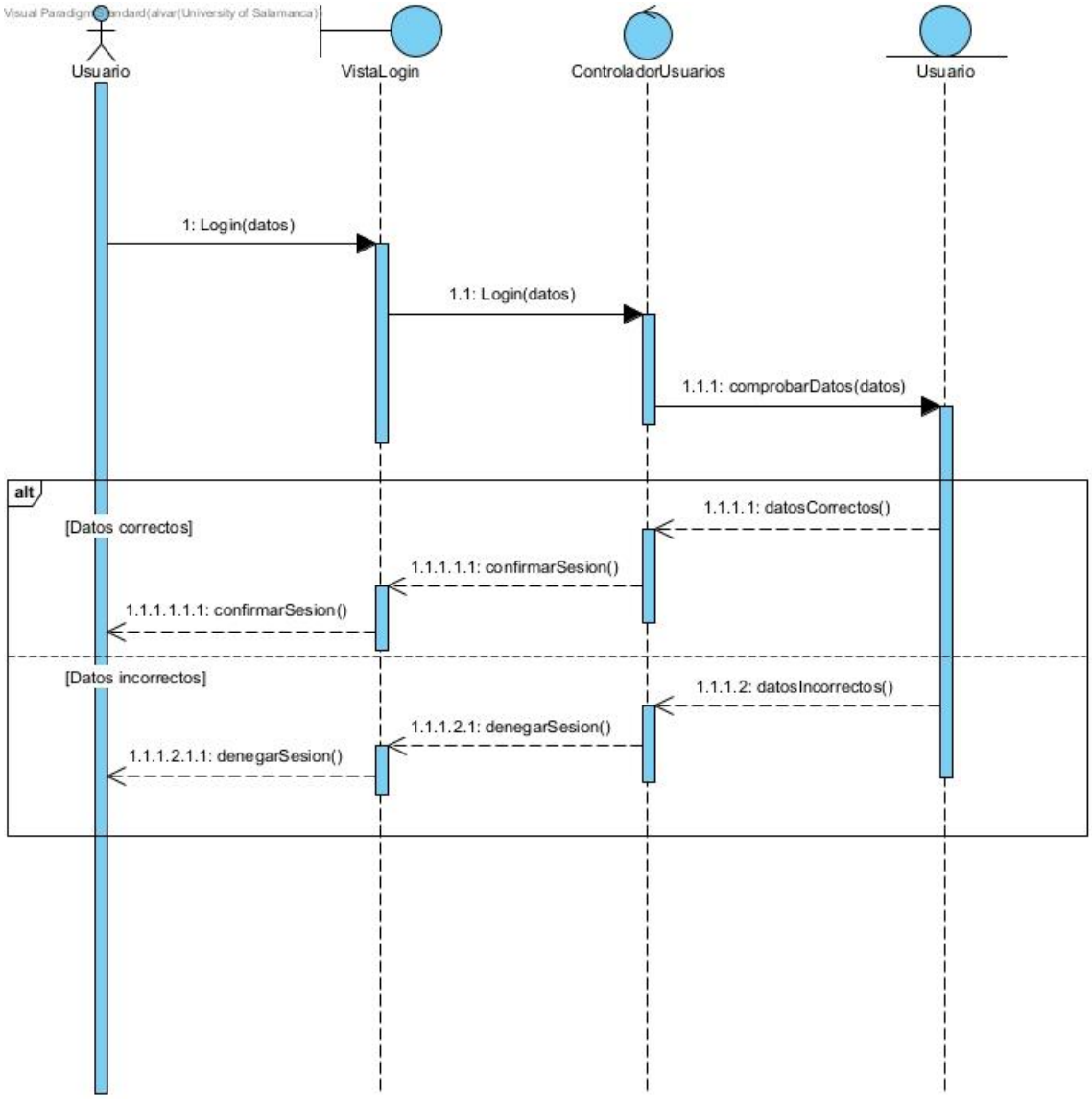

Ilustración 26. Diagrama de secuencia de análisis de Iniciar sesión

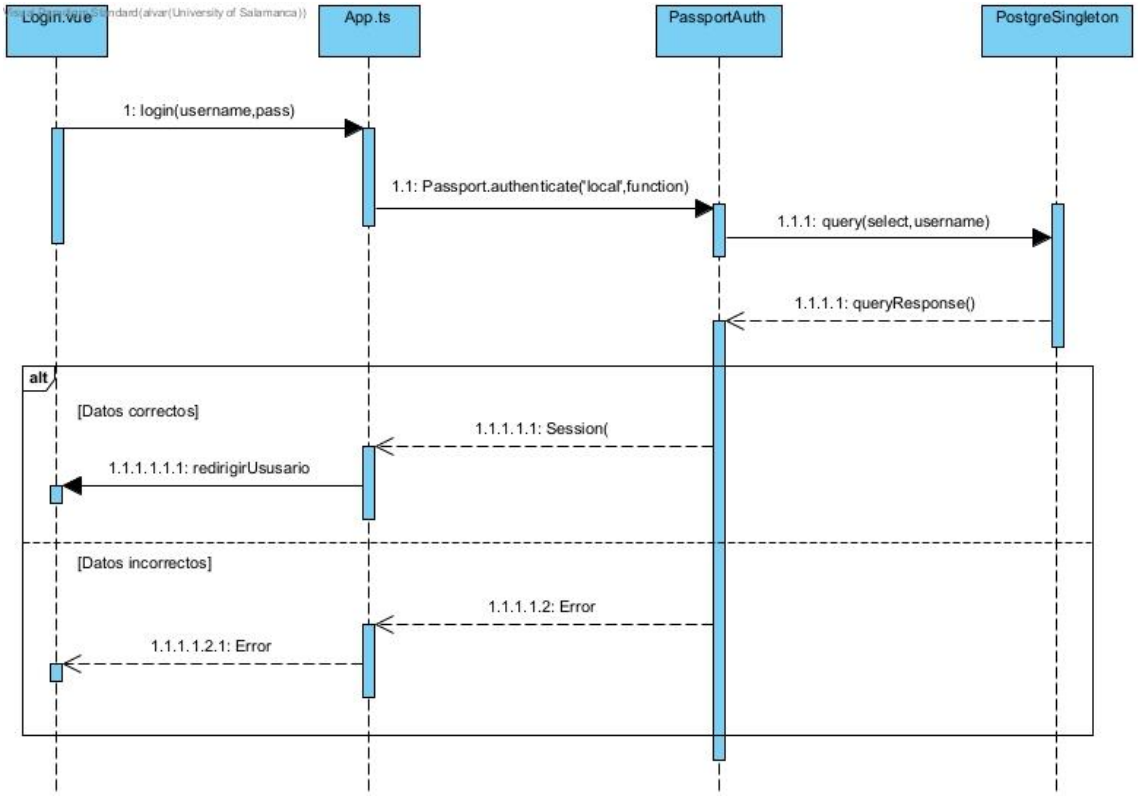

Ilustración 27. Diagrama de secuencia de diseño de iniciar sesión

4.6.2. Obtención de puntos de interés

En relación con el módulo Obtención de puntos de interés cuyo objetivo es la extracción de la información sobre estaciones de transporte de medios externos al propio sistema, en las ilustraciones Ilustración 28 e Ilustración 29 se muestran los diagramas de la realización del caso de uso único de este módulo, Obtener puntos de interés.

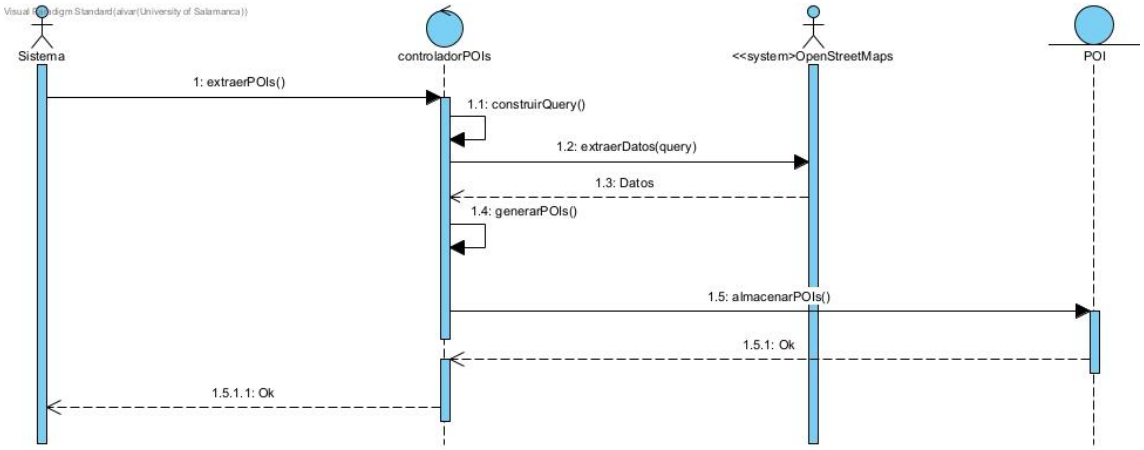

Ilustración 28. Diagrama de secuencia de análisis de Obtener puntos de interés

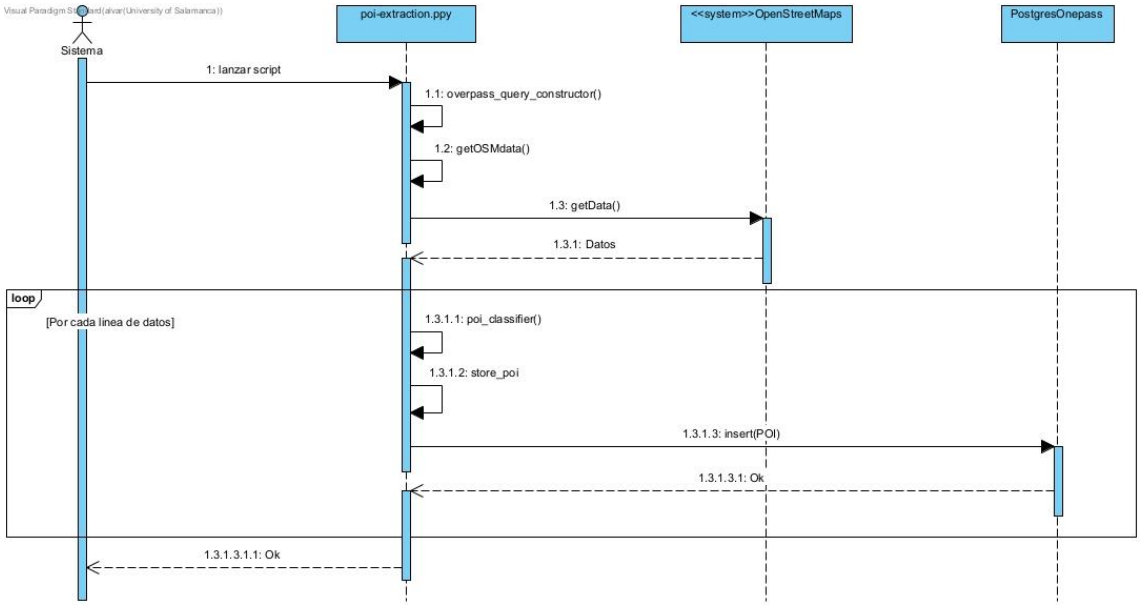

Ilustración 29. Diagrama de secuencia de diseño de Obtener puntos de interés

4.6.3. API de Operadores

En relación con el módulo de la API de Operadores cuyo objetivo es permitir a los operadores de transporte añadir y eliminar sus rutas a través de una API, en las ilustraciones Ilustración 30 e Ilustración 31 se muestran los diagramas de la realización del caso de uso más esencial de este módulo, Añadir ruta.

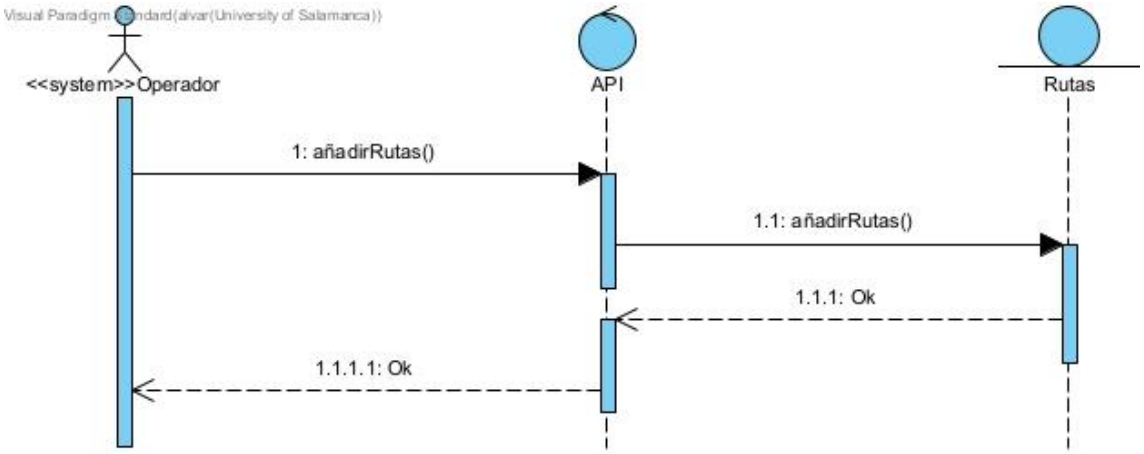

Ilustración 30. Diagrama de secuencia de análisis de Añadir ruta

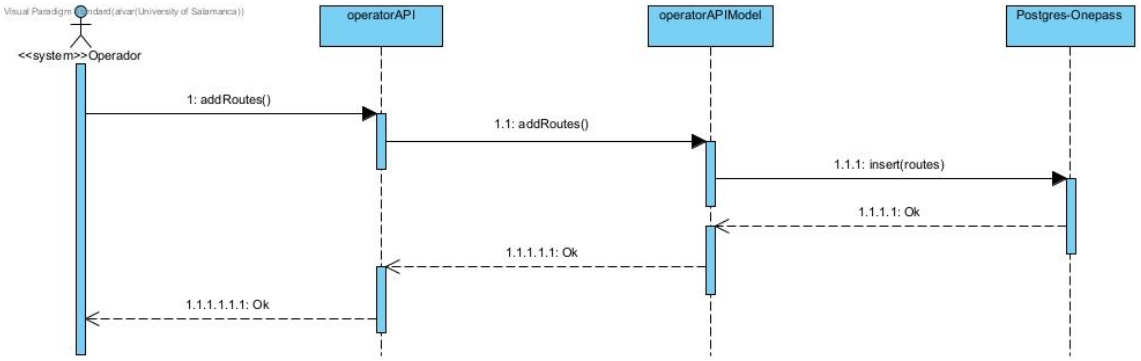

Ilustración 31. Diagrama de secuencia de diseño de Añadir rutas

4.6.4. Gestión de billetes de transporte multimodales

En relación con el módulo de Gestión de billetes de transporte multimodales cuyo objetivo gestionar los billetes de transporte correspondientes a diversos medios de transporte y operadores pertenecientes a un mismo viaje, en las ilustraciones Ilustración 32 e Ilustración 33 se muestran los diagramas de la realización del caso de uso más esencial de este módulo, Añadir billete a viaje.

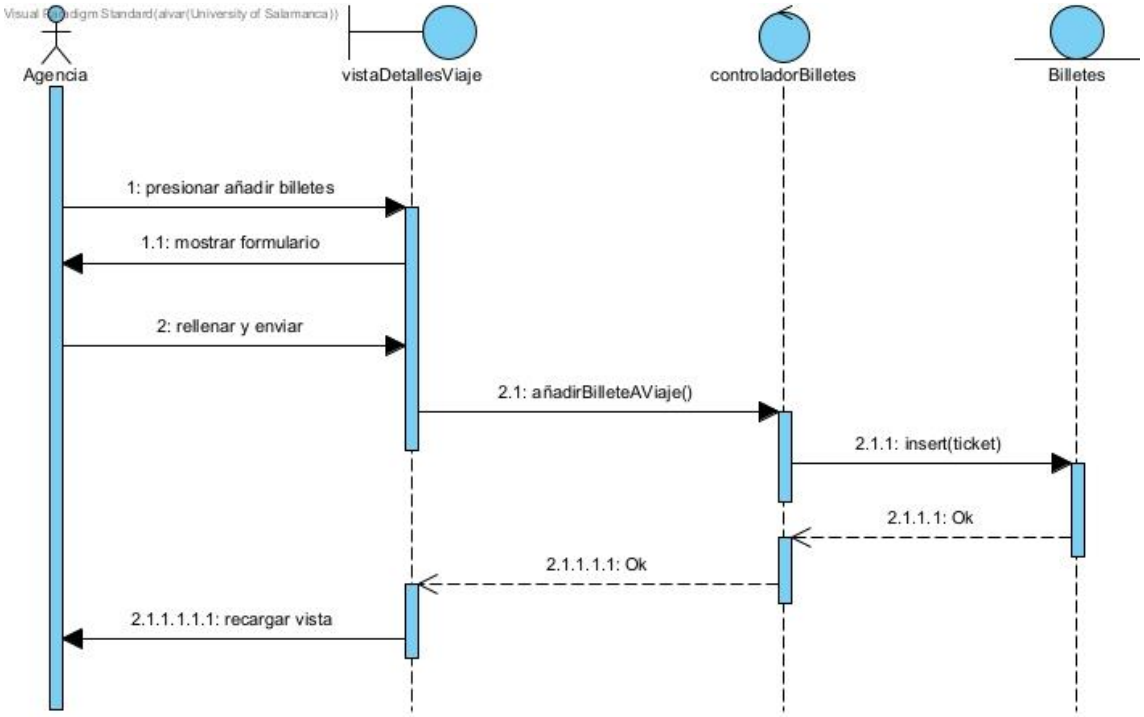

Ilustración 32. Diagrama de secuencia de análisis de Añadir billete a viaje

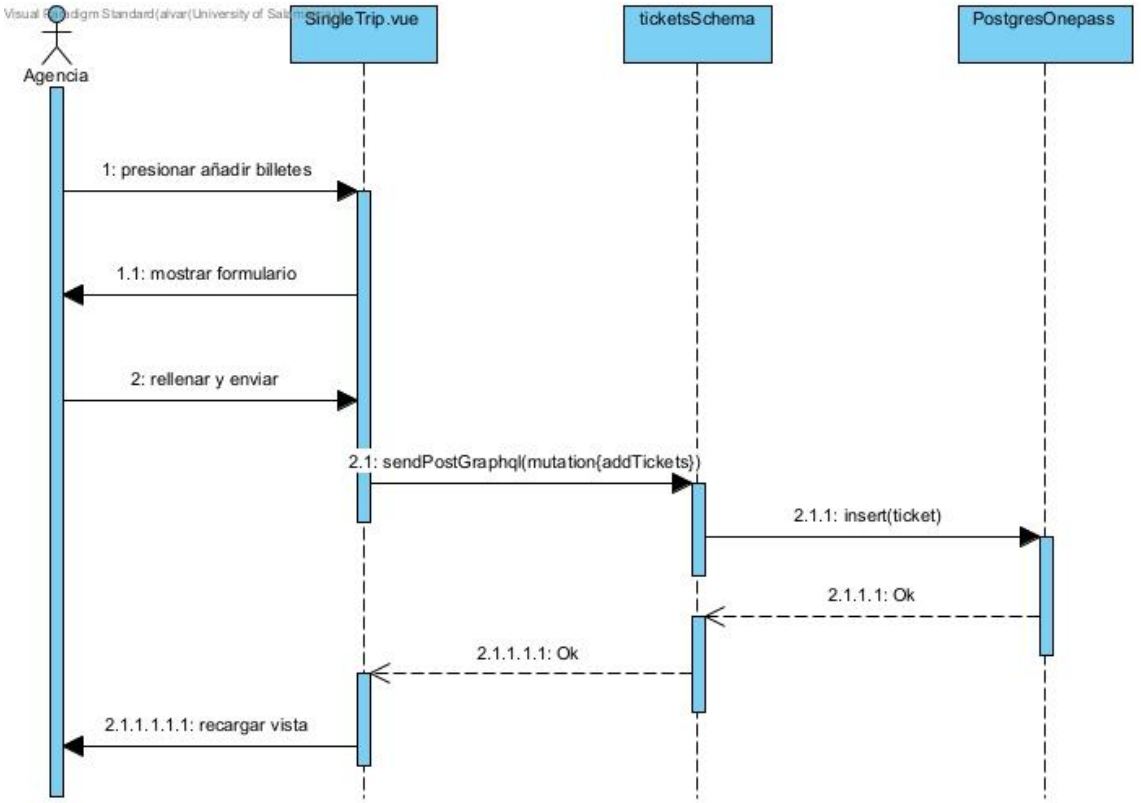

Ilustración 33. Diagrama de secuencia de diseño de Añadir billete a viaje

4.6.5. Planificación de viajes

En relación con el módulo de Planificación de viajes cuyo objetivo aportar las herramientas necesarias al usuario tipo agencia para planificar un viaje de origen a destino, en las ilustraciones Ilustración 34 e Ilustración 35 se muestran los diagramas de la realización del caso de uso único más esencial de este módulo, Planificar viaje.

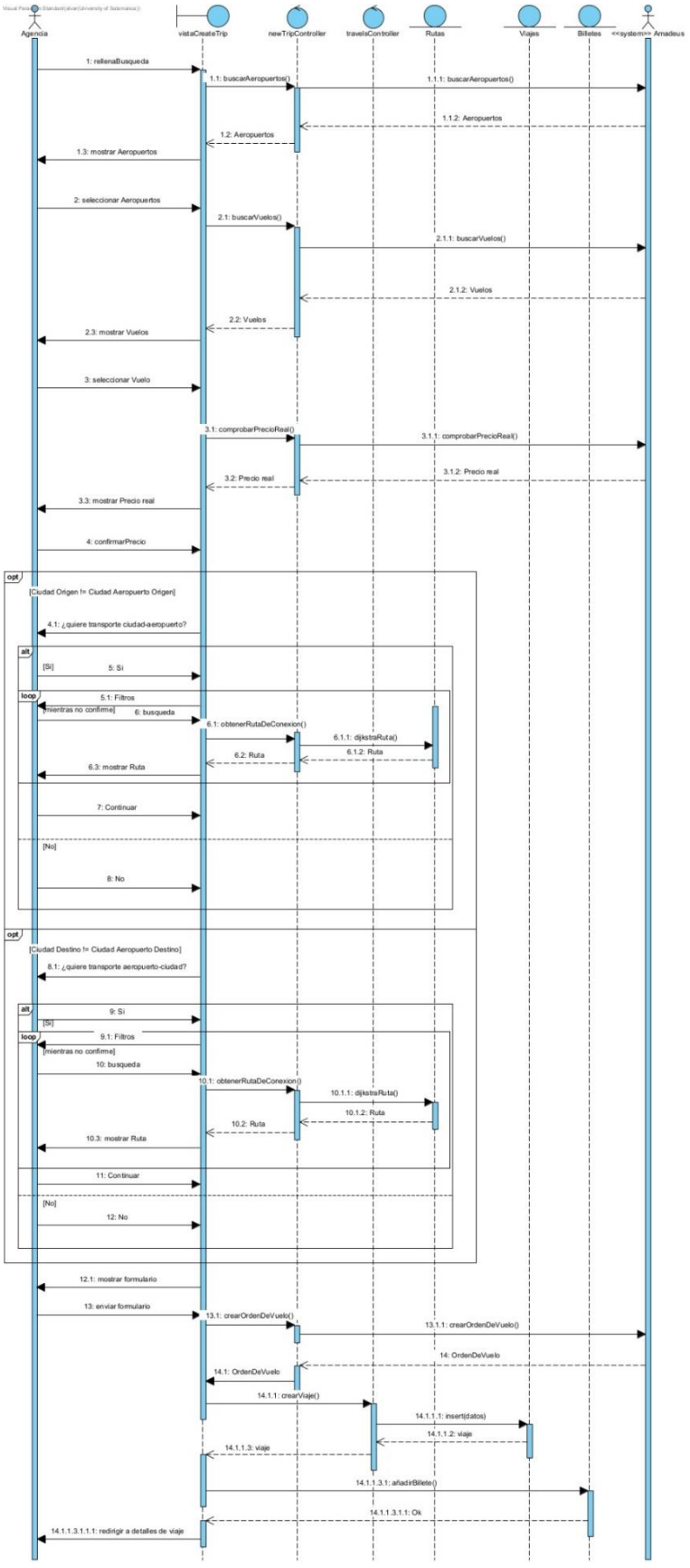

Ilustración 34. Diagrama de secuencia de análisis de Planificar viaje

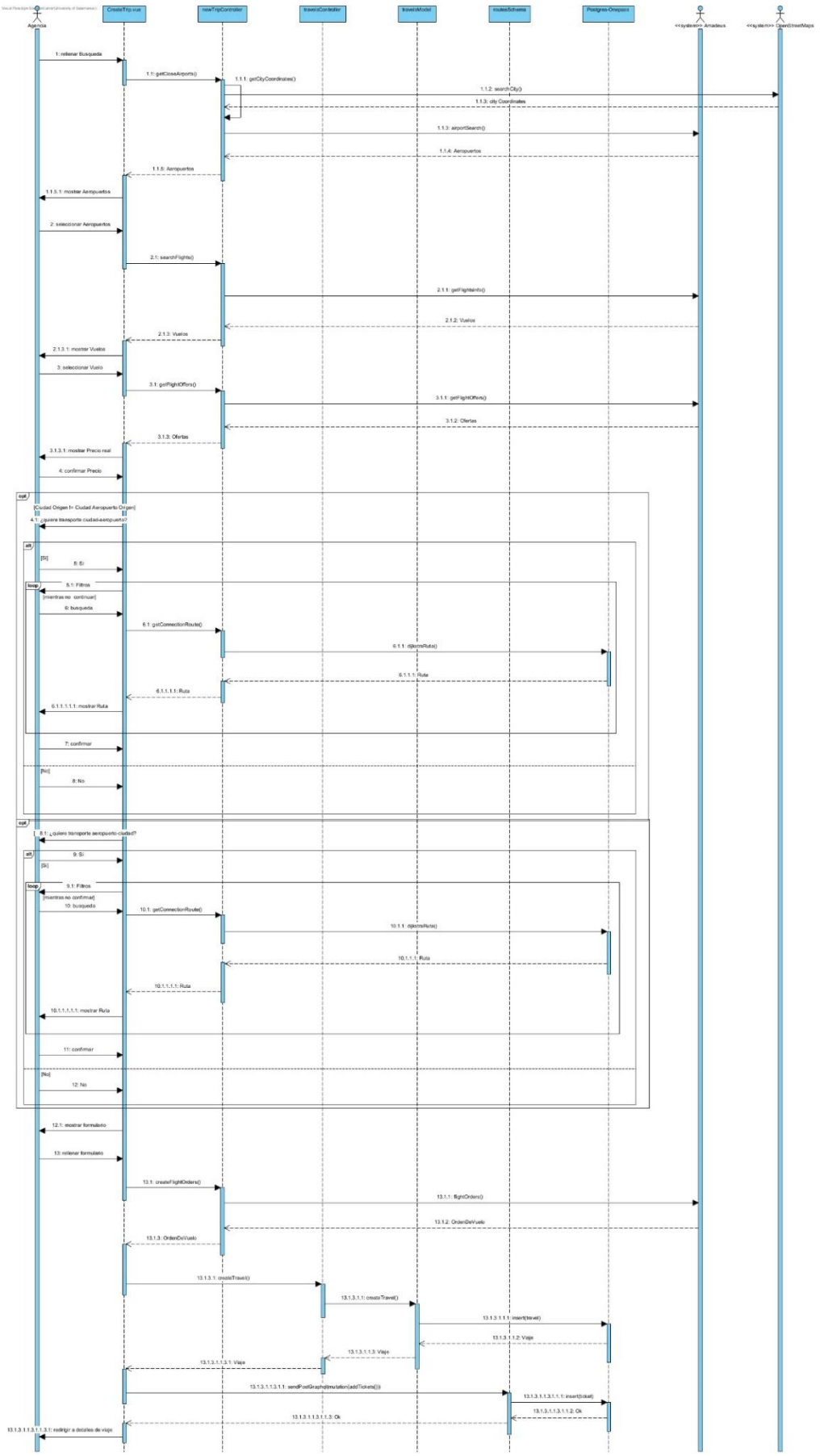
Agencia
CreateTrip.vue
newTripController
travelsController
travelsModel
routesSchema
Postgres-Onepass
<<system>> Amadeus
<<system>> OpenStreetMaps
1: rellenar Busqueda
1.1: getCloseAirports()
1.1.1: getCityCoordinates()
1.1.2: searchCity()
1.1.3: city Coordinates
1.1.3: airportSearch()
1.1.4: Aeropuertos
1.1.5: Aeropuertos
1.1.5.1: mostar Aeropuertos
2: seleccionar Aeropuertos
2.1: searchFlights()
2.1.1: getFlightsInfo()
2.1.2: Vuelos
2.1.3: Vuelos
2.1.3.1: mostrar Vuelos
3: seleccionar Vuelo
3.1: getFlightOffers()
3.1.1: getFlightOffers()
3.1.2: Ofertas
3.1.3: Ofertas
3.1.3.1: mostrar Precio real
4: confirmar Precio
opt
[Ciudad Origen != Ciudad Aeropuerto Origen]
4.1: ¿quiere transporte ciudad-aeropuerto?
alt
[Si]
5: Si
loop
5.1: Filtros
[mientras no confirmar]
6: busqueda
6.1: getConnectionRoute()
6.1.1: dijkstraRuta()
6.1.1.1: Ruta
6.1.1.1.1: Ruta
6.1.1.1.1.1: mostrar Ruta
7: confirmar
[No]
8: No
opt
8.1: ¿quiere transporte aeropuerto-ciudad?
alt
9: Si
[Si]
loop
9.1: Filtros
[mientras no confirmar]
10: busqueda
10.1: getConnectionRoute()
10.1.1: dijkstraRuta()
10.1.1.1: Ruta
10.1.1.1.1: Ruta
10.1.1.1.1.1: mostrar Ruta
11: confirmar
[No]
12: No
12.1: mostrar formulario
13: rellenar formulario
13.1: createFlightOrders()
13.1.1: flightOrders()
13.1.2: OrdenDeVuelo
13.1.3: OrdenDeVuelo
13.1.3.1: createTravel()
13.1.3.1.1: createTravel()
13.1.3.1.1.1: insert(travel)
13.1.3.1.1.2: Viaje
13.1.3.1.1.3: Viaje
13.1.3.1.1.3.1: Viaje
13.1.3.1.1.3.1.1: sendPostGraph(simulationAddTickets())
13.1.3.1.1.3.1.1.1: insert(ticket)
13.1.3.1.1.3.1.1.2: Ok
13.1.3.1.1.3.1.1.3: Ok
13.1.3.1.1.3.1.1.3.1: redirigir a detalles de viaje

Ilustración 35. Diagrama de secuencia de diseño de Planificar viaje

4.7. Arquitectura del sistema

En este apartado, se presenta la arquitectura del sistema a partir de los patrones arquitectónicos y las decisiones de diseño tomadas, junto a sus respectivas justificaciones. Comenzando por una breve introducción a las partes principales que componen el sistema, para posteriormente exponer una explicación de cada una de estas por separado.

Debido a las características del proyecto, en todo momento se ha buscado una arquitectura modular, que permita generar un sistema extensible, mantenible y que permita la replicación total o parcial de este.

Como se observa en la Ilustración 36 Arquitectura del sistema se divide en distintos componentes para formar el sistema, entre ellos se encuentran:

- Aplicación web:
    - SingleTicket Frontend: una de las dos partes de la aplicación web encargada de la visualización del sistema. Contempla cada una de las vistas necesaria para la interacción con el usuario.
    - SingleTicket Backend: segunda parte de la aplicación web conectada al Frontend, en esta se realizan las tareas de la lógica y conexiones con el resto de los componentes. Se emplea para poder realizar las distintas funcionalidades del sistema relacionadas con la lógica de este.
- Amadeus *API*: *API* ofrecida por la empresa del sector turístico Amadeus. Descrita en detalle en la sección 17.
- PostgreSQL: Extensión de PostgreSQL que permite trabajar con la base de datos como una base de datos espacial. Descrita en detalle en la sección 17.
- POI-Extraction: Script escrito en el lenguaje de programación Python utiliza OverPassAPI para extraer los puntos de interés empleados en el sistema que serán almacenados posteriormente en la base de datos PostgreSQL.
- OverPassAPI: *API* proporcionada por *OpenStreetMaps* para la realización de mapas. Descrita en detalle en la sección 17.

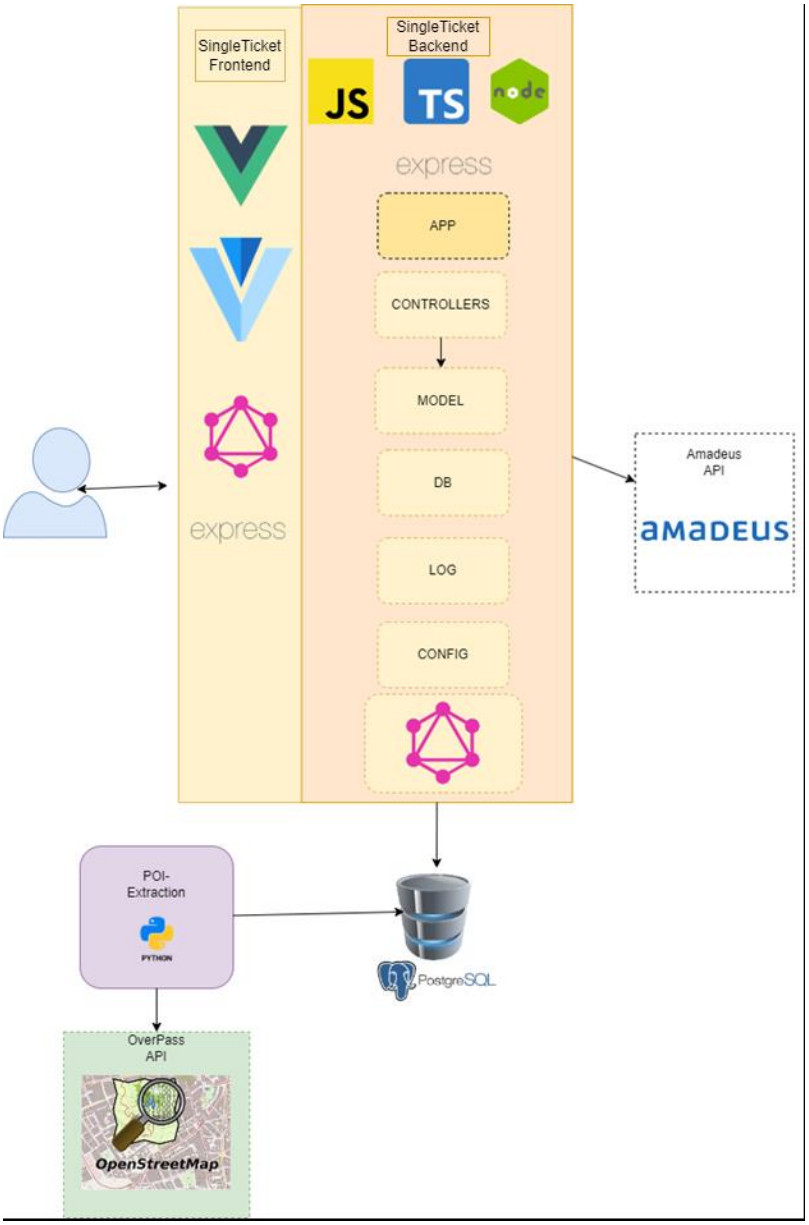

Ilustración 36 Arquitectura del sistema

4.8. Diseño de la interfaz

La totalidad de las interfaces ofrecidas por el sistema se sirven a través del protocolo HTTPS, el cual es la forma más usada hoy en día para la interacción entre sistemas a través de internet.

Con el fin de aportar una representación más visual del diseño de interfaces se ha plasmado en un diagrama de interfaces que se puede observar en las ilustraciones Ilustración 37, Ilustración 38 e Ilustración 39. Se ha dividido el diagrama en 3 imágenes para favorecer la visibilidad y comprensión de este.

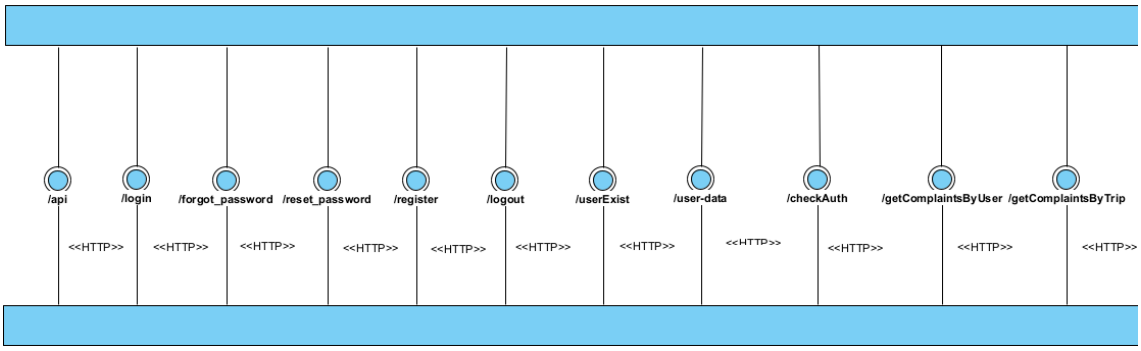

Ilustración 37. Diseño de interfaces 1

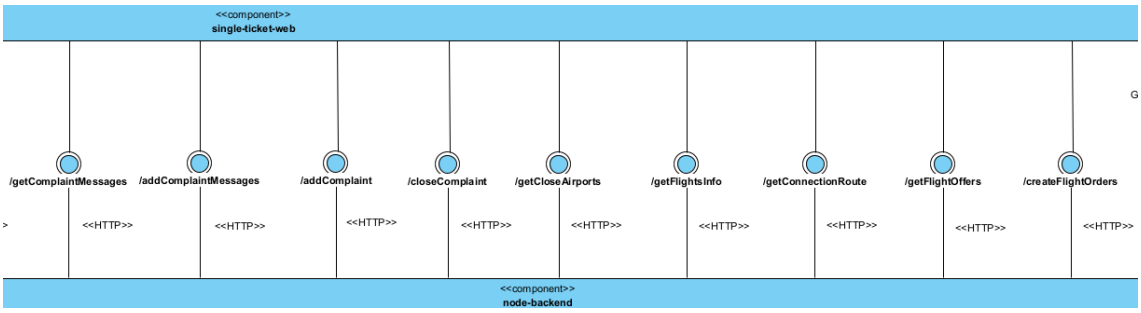

Ilustración 38. Diseño de interfaces 2

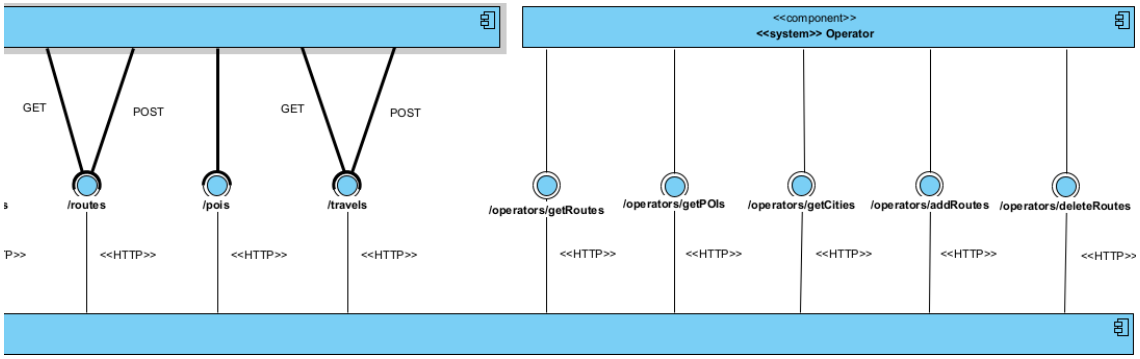

Ilustración 39. Diseño de interfaces 3

4.8.1. Especificación de interfaces HTTP

En este apartado, se presentan en detalle las interfaces servidas por medio del protocolo HTTPS a modo de tablas.

Endpoint	**/forgot_password**
Modulo	*Node-backend*
Descripción	Envía un email al usuario con un enlace para restaurar la contraseña
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 5 Descripción de la interfaz ofrecida por HTTPS del endpoint /forgot_password

Endpoint	**/reset_password**
Modulo	*Node-backend*
Descripción	Actualiza la contraseña del usuario
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 6 Descripción de la interfaz ofrecida por HTTPS del endpoint /reset_password

Endpoint	**/register**
Modulo	*Node-backend*
Descripción	Registra un nuevo usuario en el sistema
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 7 Descripción de la interfaz ofrecida por HTTPS del endpoint /register

Endpoint	**/login**
Modulo	*Node-backend*
Descripción	Permite al usuario iniciar sesión
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 8 Descripción de la interfaz ofrecida por HTTPS del endpoint /login

Endpoint	**/logout**
Modulo	*Node-backend*
Descripción	Permite al usuario cerrar sesion
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-

Parámetros	-
Retorno	-

Tabla 9 Descripción de la interfaz ofrecida por HTTPS del endpoint /logout

Endpoint	**/userExist**
Modulo	*Node-backend*
Descripción	Comprueba sin un usuario existe
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 10 Descripción de la interfaz ofrecida por HTTPS del endpoint /userExist

Endpoint	**/user-data**
Modulo	*Node-backend*
Descripción	Devuelve los datos del usuario
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 11 Descripción de la interfaz ofrecida por HTTPS del endpoint /user-data

Endpoint	**/checkAuth**
Modulo	*Node-backend*
Descripción	Comprueba que el usuario ha iniciado sesion y devuelve el tipo de usuario al que pertenece
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 12 Descripción de la interfaz ofrecida por HTTPS del endpoint /checkAuth

Endpoint	**/getComplaintsByUser**
Modulo	*Node-backend*
Descripción	Devuelve el listado de incidencias de un usuario en especifico
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 13 Descripción de la interfaz ofrecida por HTTPS del endpoint /getComplaintsByUser

Endpoint	**/getComplaintsByTrip**
Modulo	*Node-backend*
Descripción	Devuelve el listado de incidencias de un viaje en especifico
Operación	*GET*
Ubicación de los parámetros	-

Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 14 Descripción de la interfaz ofrecida por HTTPS del endpoint /getComplaintsByTrip

Endpoint	**/getComplaintsMessages**
Modulo	*Node-backend*
Descripción	Devuelve el listado de mensajes de una incidencia en especifico
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 15 Descripción de la interfaz ofrecida por HTTPS del endpoint /getComplaintsMessages

Endpoint	**/addComplaintMessage**
Modulo	*Node-backend*
Descripción	Añade un mensaje a la incidencia
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 16 Descripción de la interfaz ofrecida por HTTPS del endpoint /addComplaintMessage

Endpoint	**/addComplaint**
Modulo	*Node-backend*
Descripción	Añade una incidencia a un viaje
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 17 Descripción de la interfaz ofrecida por HTTPS del endpoint /addComplaint

Endpoint	**/closeComplaint**
Modulo	*Node-backend*
Descripción	Cambia el estado de una incidencia a RESOLVED
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 18 Descripción de la interfaz ofrecida por HTTPS del endpoint /closeComplaint

Endpoint	**/getCloseAirports**
Modulo	*Node-backend*
Descripción	Devuelve los aeropuertos cercanos a una ciudad
Operación	*POST*
Ubicación de los parámetros	-

Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 19 Descripción de la interfaz ofrecida por HTTPS del endpoint /getCloseAirports

Endpoint	**/getFligthsInfo**
Modulo	*Node-backend*
Descripción	Devuelve los información de vuelos programados entre dos aeropuertos
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 20 Descripción de la interfaz ofrecida por HTTPS del endpoint /getFlightsInfo

Endpoint	**/getConnectionRoute**
Modulo	*Node-backend*
Descripción	Devuelve una ruta filtrada por el usuario de tipo agencia
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 21 Descripción de la interfaz ofrecida por HTTPS del endpoint /getConnectionRoute

Endpoint	**/getFlightOffer**
Modulo	*Node-backend*
Descripción	Devuelve la oferta actualizada para un viaje
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 22 Descripción de la interfaz ofrecida por HTTPS del endpoint /getFlightOffer

Endpoint	**/createFlightOrders**
Modulo	*Node-backend*
Descripción	Devuelve la orden de reserva de un vuelo, creada para un pasajero
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 23 Descripción de la interfaz ofrecida por HTTPS del endpoint /createFlightOrders

Endpoint	**/operators/getRoutes**
Modulo	*Node-backend*

Descripción	Devuelve los rutas que tiene un usuario de tipo operador registradas en el sistema
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 24 Descripción de la interfaz ofrecida por HTTPS del endpoint /operators/getRoutes

Endpoint	**/operators/getPOIs**
Modulo	*Node-backend*
Descripción	Devuelve los puntos de interés almacenados en el sistema
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 25 Descripción de la interfaz ofrecida por HTTPS del endpoint /operators/getPOIs

Endpoint	**/operators/getCities**
Modulo	*Node-backend*
Descripción	Devuelve los ciudades almacenadas en el sistema
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 26 Descripción de la interfaz ofrecida por HTTPS del endpoint /operators/getCities

Endpoint	**/operators/addRoutes**
Modulo	*Node-backend*
Descripción	Añade las rutas deseadas por usuario de tipo operador.
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 27 Descripción de la interfaz ofrecida por HTTPS del endpoint /operators/addRoutes

Endpoint	**/operators/deleteRoutes**
Modulo	*Node-backend*
Descripción	Elimina las rutas deseadas por usuario de tipo operador.
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 28 Descripción de la interfaz ofrecida por HTTPS del endpoint /operators/deleteRoutes

Endpoint	**/routes**
Modulo	*Node-backend*

Descripción	Devuelve las rutas registradas en el sistema
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 29 Descripción de la interfaz ofrecida por HTTPS del endpoint /Routes

Endpoint	**/routes**
Modulo	*Node-backend*
Descripción	Añade las rutas en el sistema
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 30 Descripción de la interfaz ofrecida por HTTPS del endpoint /Routes

Endpoint	**/pois**
Modulo	*Node-backend*
Descripción	Devuelve las puntos de interés registrados en el sistema
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 31 Descripción de la interfaz ofrecida por HTTPS del endpoint /pois

Endpoint	**/travels**
Modulo	*Node-backend*
Descripción	Devuelve las viajes registrados en el sistema
Operación	*GET*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 32 Descripción de la interfaz ofrecida por HTTPS del endpoint /travels

Endpoint	**/travels**
Modulo	*Node-backend*
Descripción	Añade viajes en el sistema
Operación	*POST*
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 33 Descripción de la interfaz ofrecida por HTTPS del endpoint /travels

Endpoint	**/api**

Modulo	*Node-backend*
Descripción	Permite la conexión por parte del servidor con GraphQL
Operación	-
Ubicación de los parámetros	-
Formato de *marshaling*	-
Parámetros	-
Retorno	-

Tabla 34 Descripción de la interfaz ofrecida por HTTPS del endpoint /api

4.8.2 Especificación inicial de interfaz hombre-máquina ofrecida por el sistema

En este apartado se muestra una fase inicial de los posibles diseños del sistema, incluyendo, Tipografía, color y estructuración de cada una de las vistas que se presentan para la interacción con el usuario en las distintas funcionalidades del sistema.

4.8.2.1 Diseño *color y tipografía*

Como se observa en Ilustración 40 Diseño color y tipografía se muestra el color y tipografía que han sido seleccionado para el sistema.

Ilustración 40 Diseño color y tipografía

4.8.2.2 Diseño inicial de las interfaces graficas

Durante este apartado se mostrarán el diseño inicial de las vistas que conformarán el sistema.

Para el registro el usuario empleará la Ilustración 41 Wireframe Registro usuario donde introducir los datos necesarios para poder realizar el registro y que posteriormente pueda iniciar sesion con su usuario y contraseña en el sistema.

Para el inicio de sesion el usuario empleará la vista Ilustración 42 Wireframe Inicio de sesión donde introducirá el usuario y contraseña para iniciar sesión, si es correcto entrará en el sistema.

Para el dashboard del cliente una vez ha iniciado sesion se empleará la vista Ilustración 43 Wireframe dashboard cliente, en donde se muestran los datos personales y un código QR que devuelve su información personal.

Para que el cliente pueda ver los viajes que ha realizado se empleara la vista Ilustración 44 Wireframe My trips. Donde se muestra un listado con todas los viajes y una información general.

En caso de que desee visualizar el viaje detalladamente deberá pulsar el botón “More details” y se mostrara la vista Ilustración 45 Wireframe More details trip cliente que contiene toda la información en detalle con respecto al viaje, y que permite añadir nuevas incidencias.

Para que el cliente pueda ver las incidencias que ha realizado se empleará la vista Ilustración 46 Wireframe My incidents. Donde se muestra un listado de incidencias con su información general.

En caso de que el usuario desee ver la incidencia en detalle y añadir información en ella deberá pulsa el botón “More details” para que se abra la

vista Ilustración 47 Wireframe more details incident cliente y pueda observar toda la información en detalle e incluir más mensajes a ella.

Para el inicio de sesión y registro de agencias se emplearán al igual que el usuario las mismas vistas Ilustración 41 Wireframe Registro usuario, Ilustración 42 Wireframe Inicio de sesión, solo que estará identificado como agencia en la base de datos dado a que en la vista de registro indicará que es una agencia.

Para la vista principal de agencia se empleará la vista Ilustración 48 New trip agencia donde se indica el registro de un nuevo viaje por parte de esa agencia, introduciendo los datos necesario en el sistema.

Para la vista de los viajes perteneciente a la agencia se empleará la vista Ilustración 44 Wireframe My trips al igual que en el cliente donde se muestra un listado con los viajes al igual que los usuario. Para mostrar en detalle cada viaje se pulsa el botón “More details” y se abrirá la vista Ilustración 49 More details trip agencia que muestra toda la información perteneciente al viaje y que permite añadir nuevos tickets.

Para las incidencias en las agencia se emplea al igual que en el usuario, la vista Ilustración 46 Wireframe My incidents donde se muestra un listado de incidencias para la agencia con la información general. En caso de que se desee ver una información más detallada deberá pulsar en el botón “More details” y se abrirá la vista Ilustración 50 More details incident agencia con toda la información de la incidencia en detalle y un botón para poder cerrarla.

Para el operador su registro se realiza de manera manual por parte de un administrador por lo tanto solo emplea la vista Ilustración 42 Wireframe Inicio de sesión común al resto de usuarios.

El operador empleara la vista Ilustración 51 Wireframe dashboard operador una vez iniciada la sesión, en donde se muestran sus datos junto al token que le permite realizar acciones en sistema mediante la API disponible para ello.

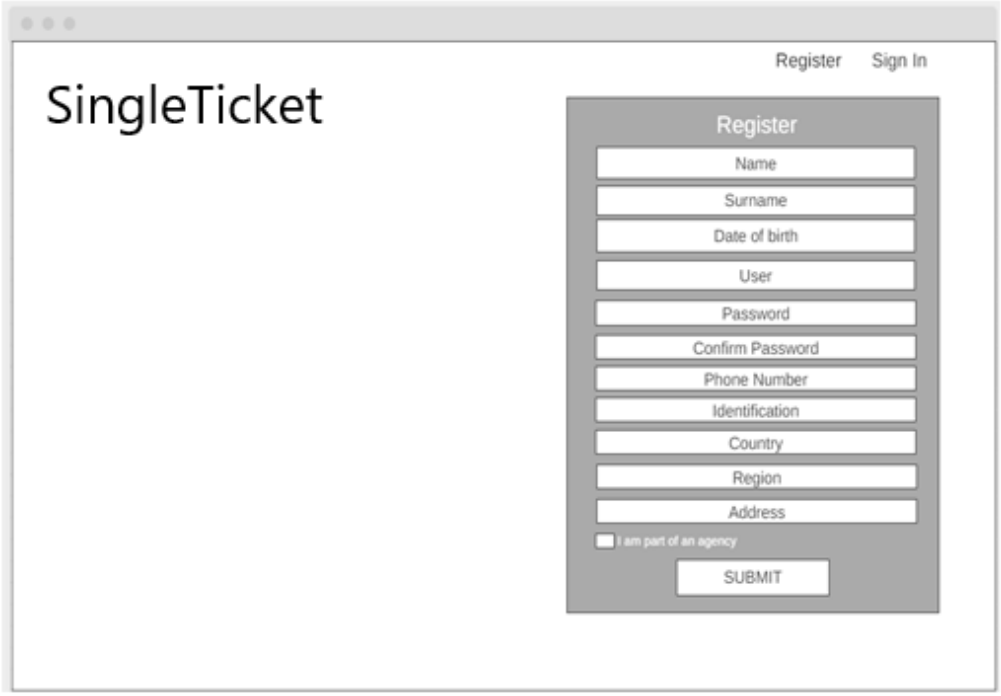

Ilustración 41 Wireframe Registro usuario

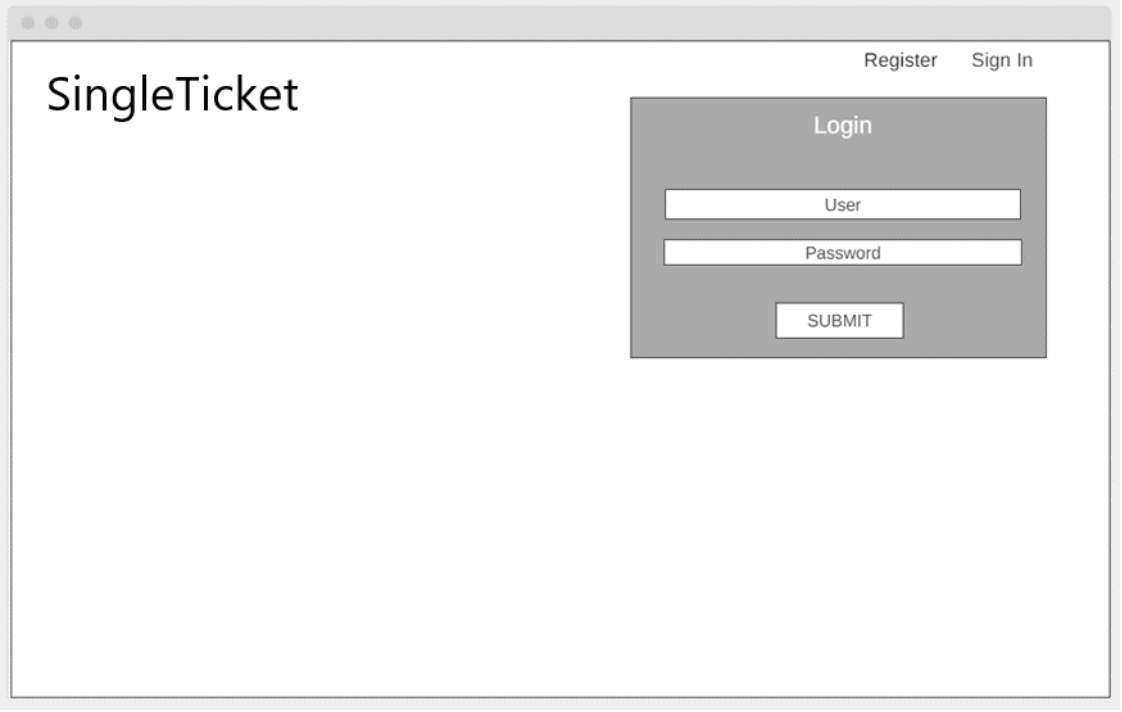

Ilustración 42 Wireframe Inicio de sesión

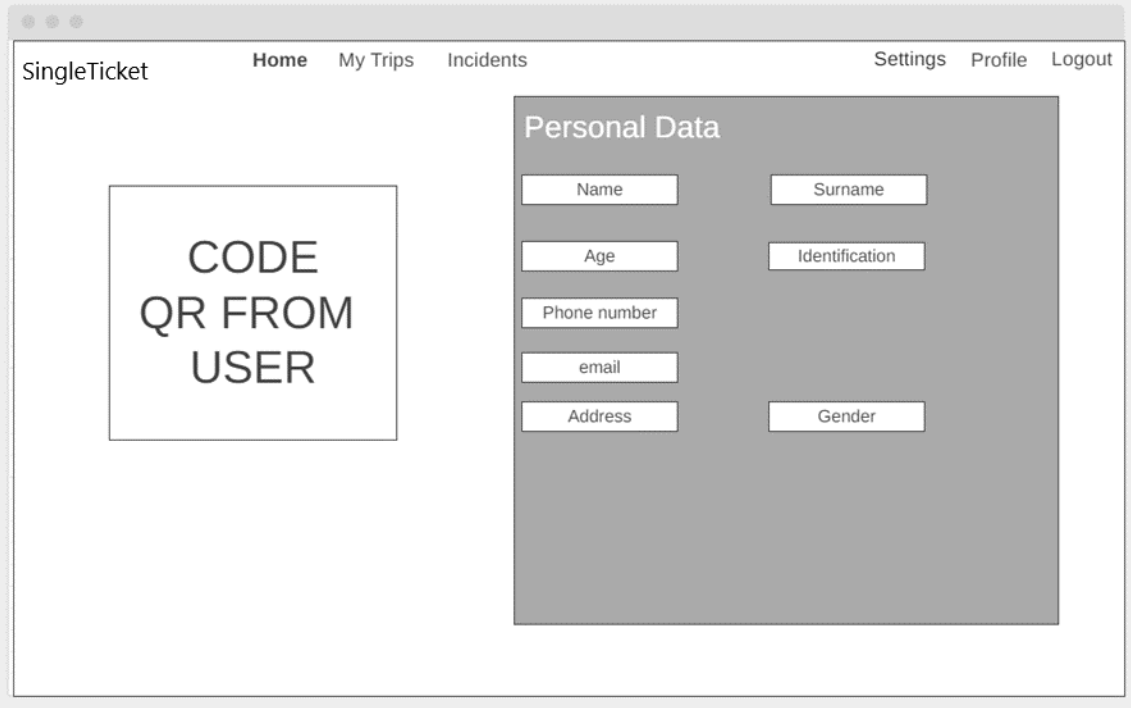

Ilustración 43 Wireframe dashboard cliente

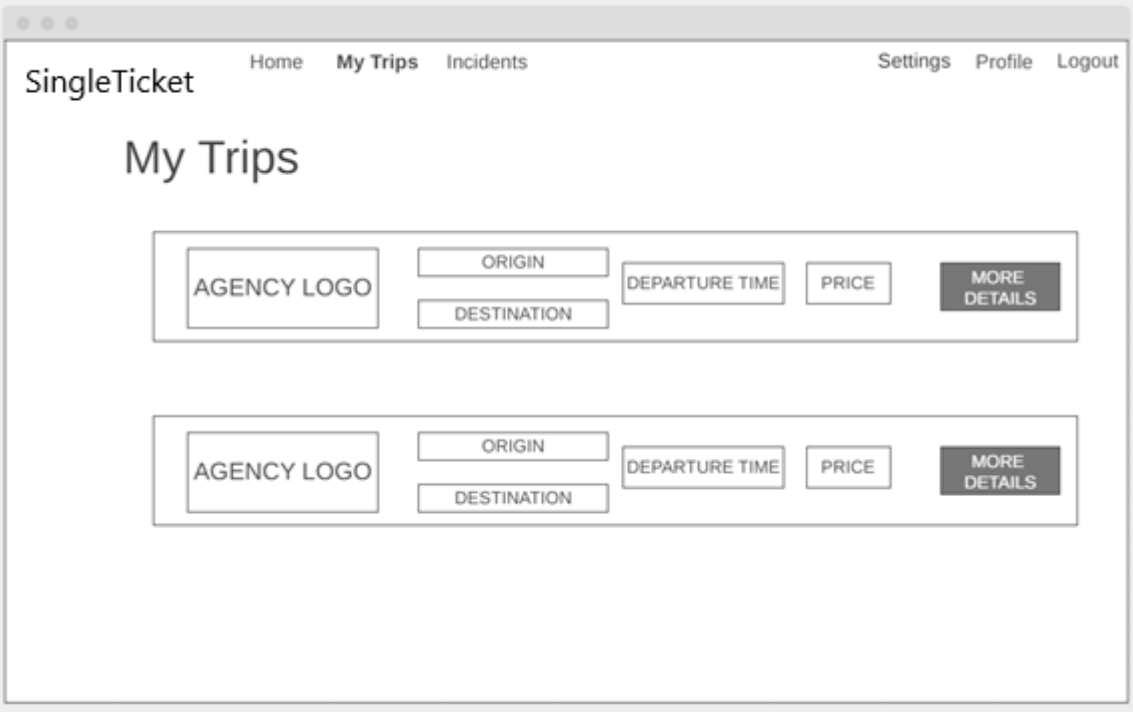

Ilustración 44 Wireframe My trips

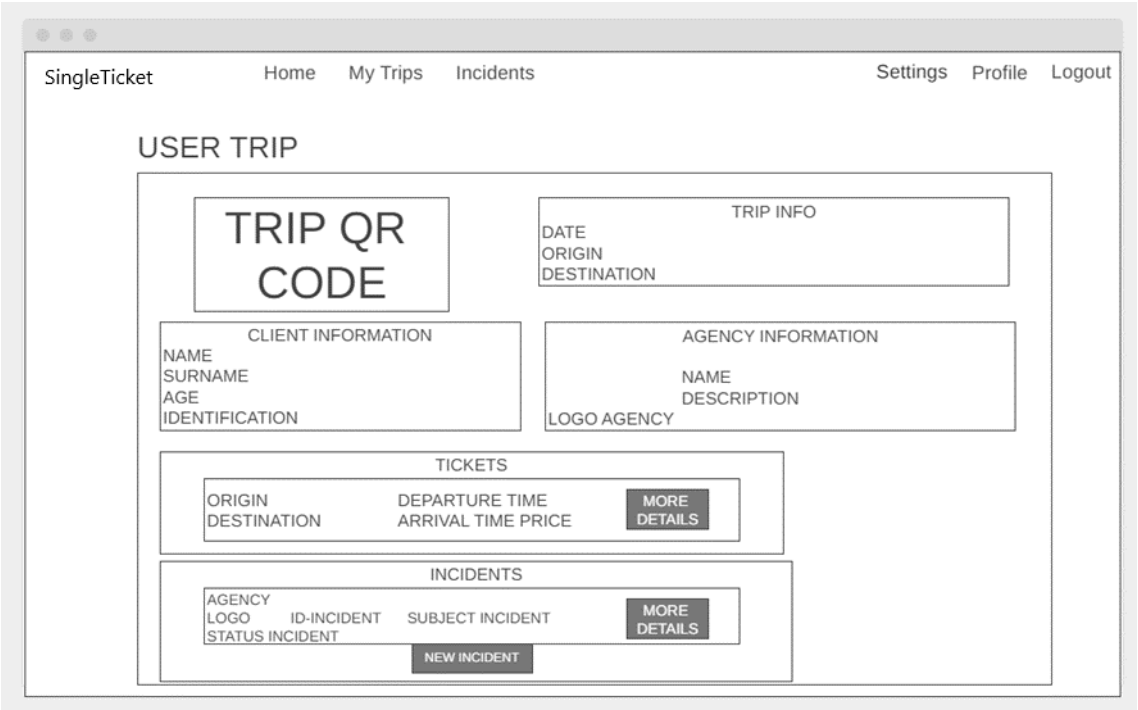

Ilustración 45 Wireframe More details trip cliente

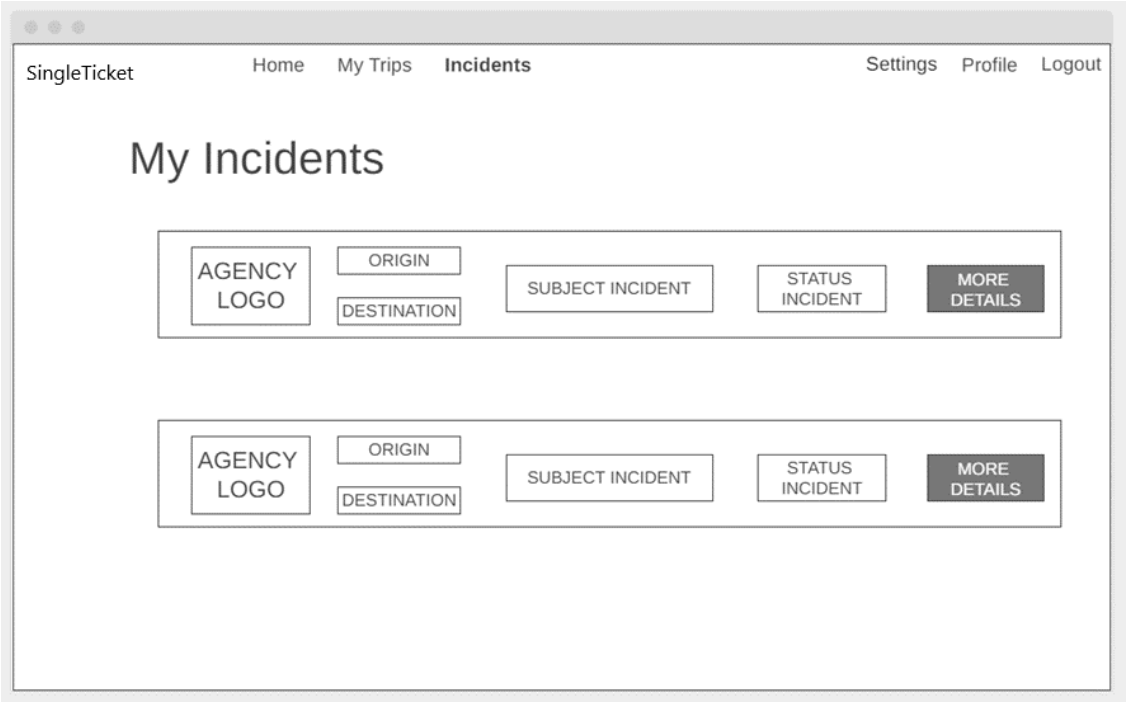

Ilustración 46 Wireframe My incidents

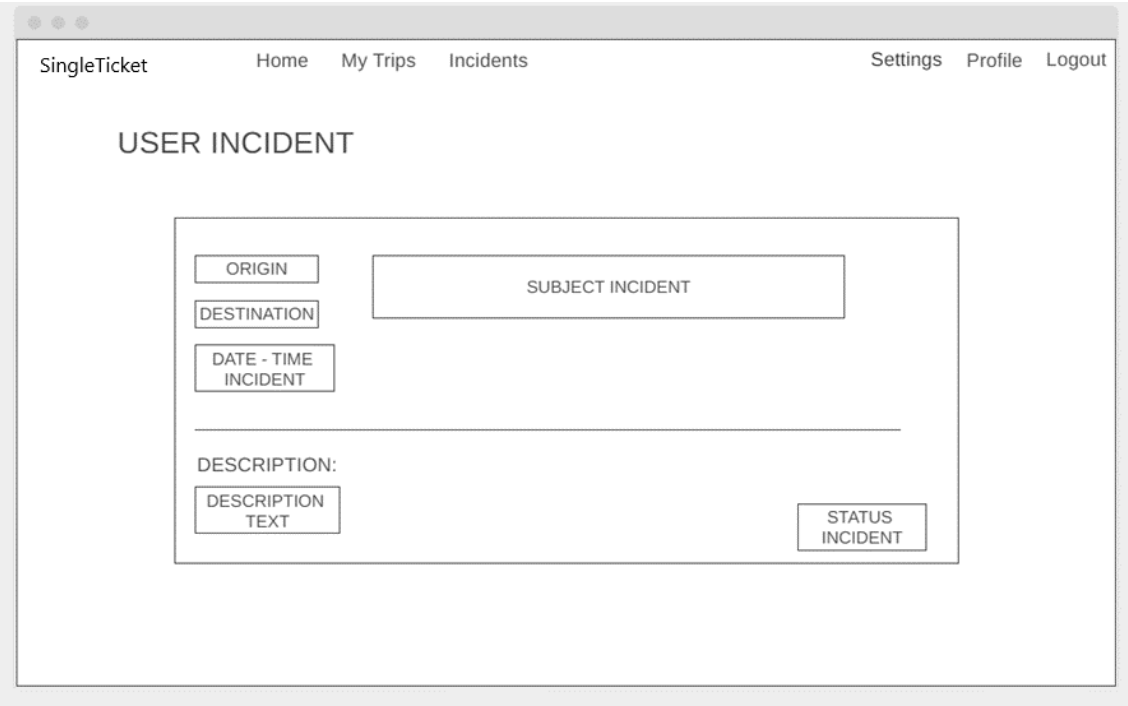

Ilustración 47 Wireframe more details incident cliente

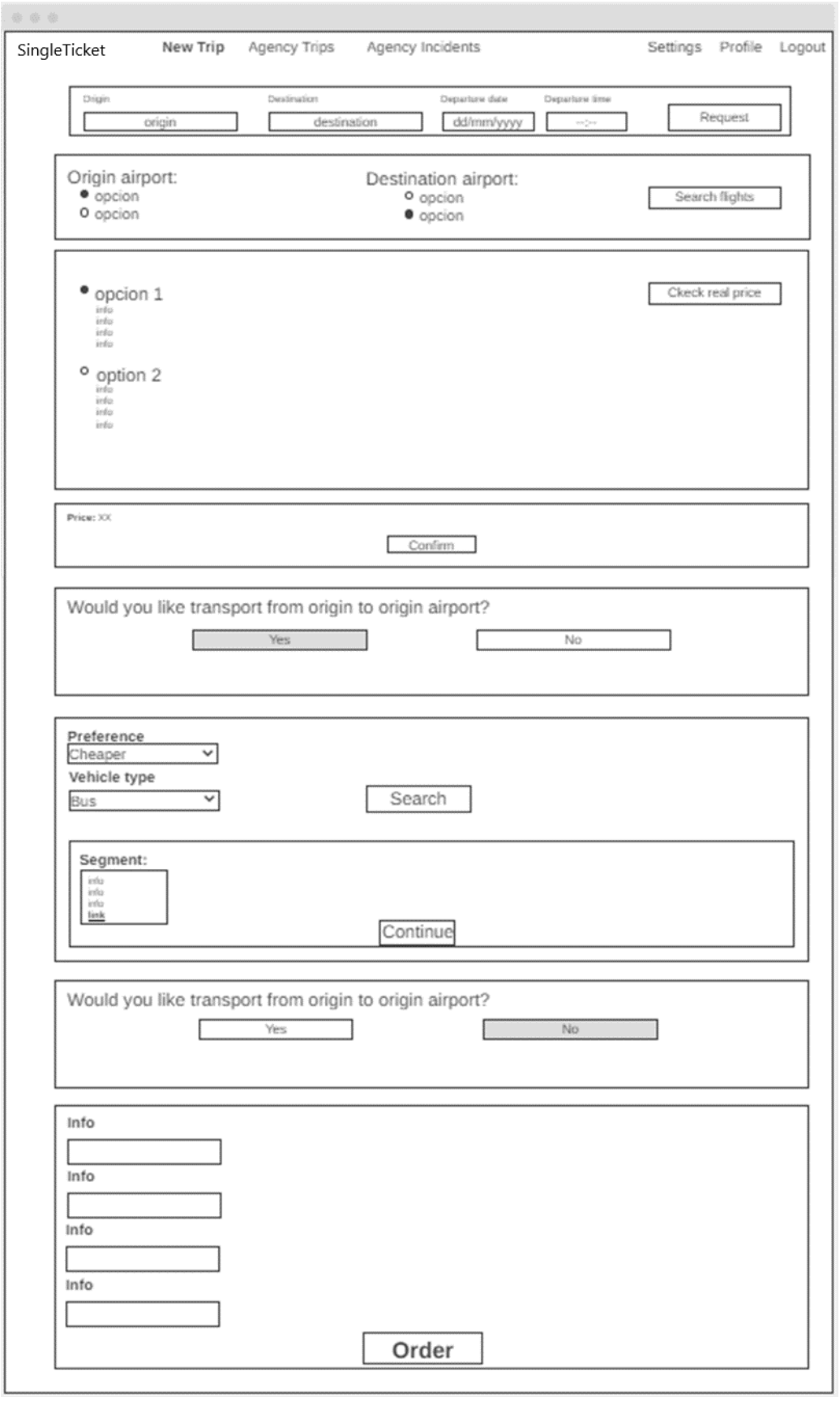

Ilustración 48 New trip agencia

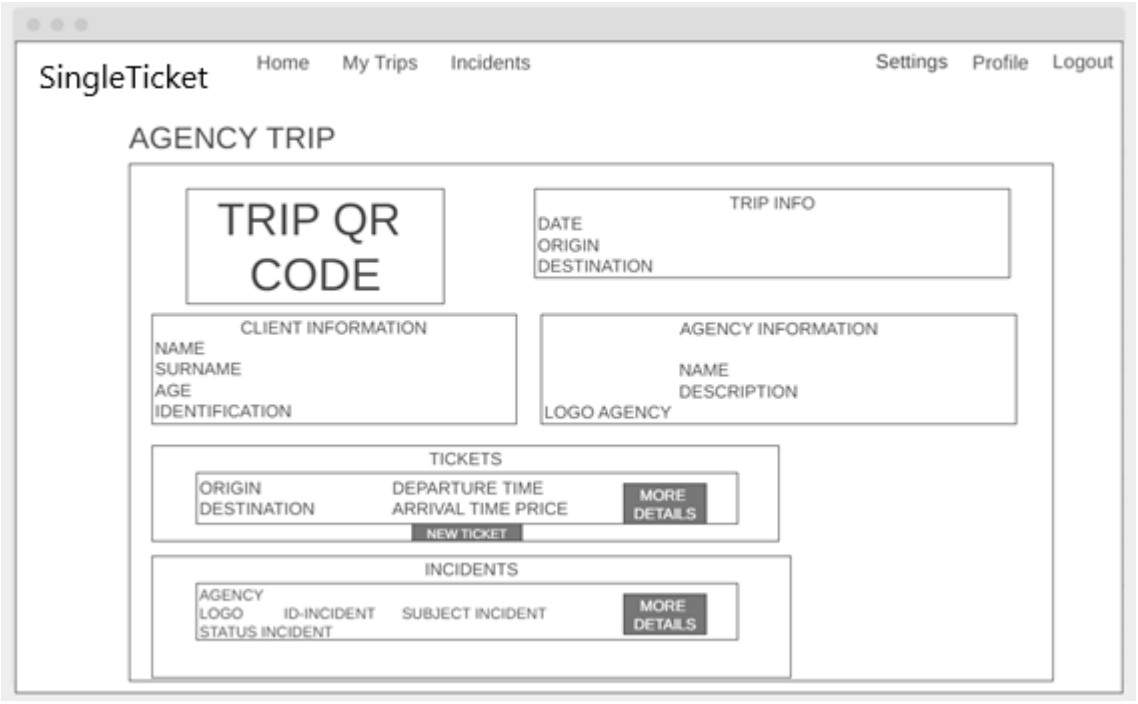

Ilustración 49 More details trip agencia

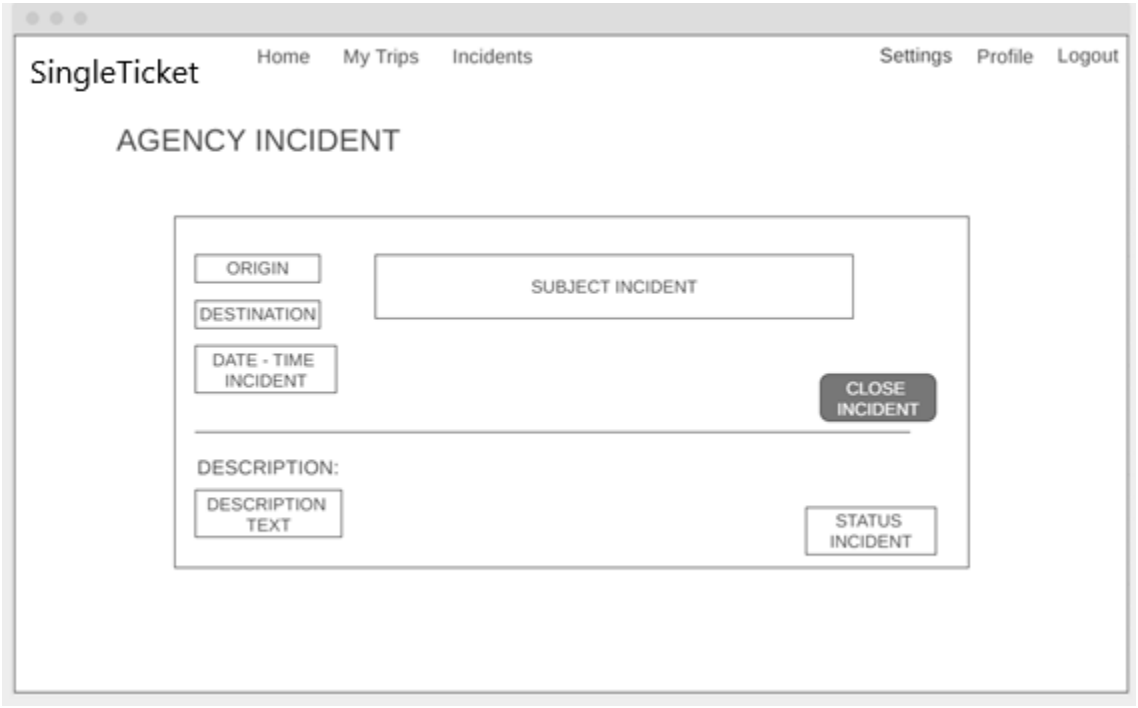

Ilustración 50 More details incident agencia

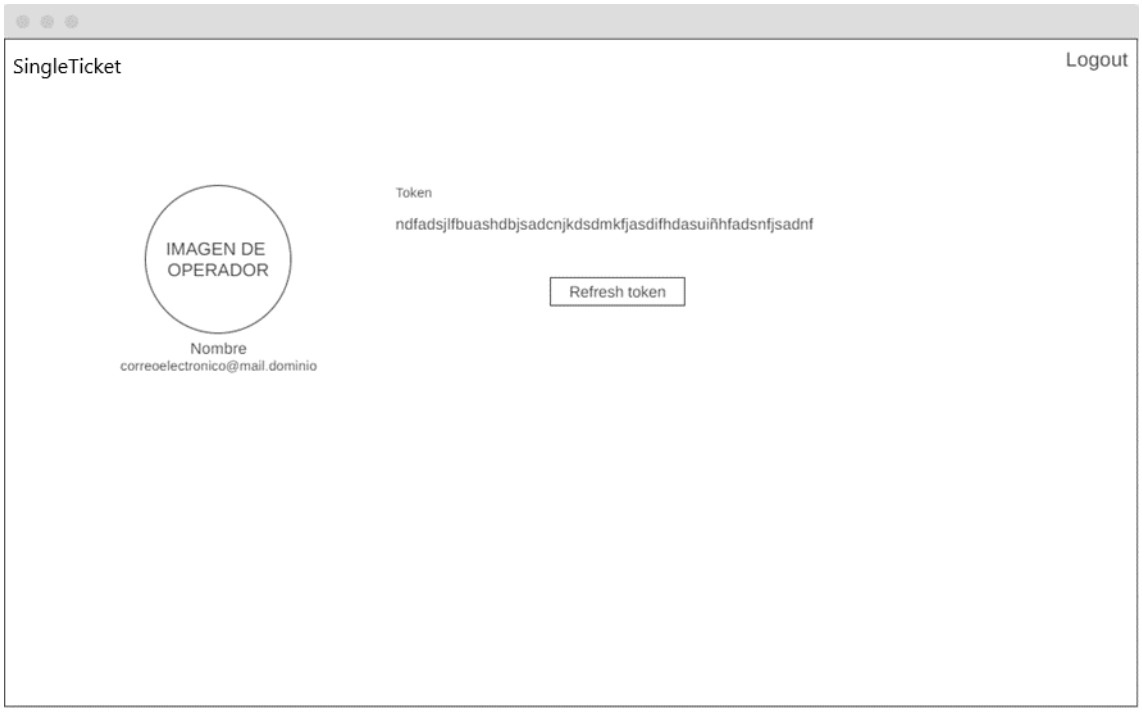

Ilustración 51 Wireframe dashboard operador

4.9. Implementación

El sistema se ha implementado a partir de los diagramas de la fase de diseño.

Se han usado las técnicas y herramientas descritas en el apartado Técnicas y herramientas.

Para la gestión de usuarios se ha implementado un sistema tradicional de cuentas de usuario con tres distintos tipos de rol, *User, Agency, Operator,* cada uno con una serie de funciones propias dentro de la plataforma. Para el manejo de las sesiones, *login y logout*, se ha hecho uso de *Passport.js* haciendo uso de una estrategia local clásica basada en usuario/contraseña.

Para la extracción de los puntos de interés requeridos por el sistema se ha desarrollado un script en Python que a través de *Overpass api* extrae la información necesaria para almacenar los puntos de interés en el formato necesario para posteriormente almacenarlos en la base de datos.

En cuanto al api de operadores, se han implementado una serie de *endpoints* dedicados en el backend de la aplicación a modo de controlador para satisfacer cada una de las necesidades plasmadas en los diagramas de diseño. Las rutas para estos endpoints se han configurado de manera distintiva como "*SingleTicket.ddnt.net/operator/endpointName*". Estos endpoints se han implementado haciendo uso de *Typescript*.

En cuanto a la planificación de viajes, para la reserva del trayecto principal, el vuelo, se ha hecho uso de la API que ofrece *Amadeus* para integrar en tu aplicación todas las funcionalidades necesarias para desarrollar un motor de reserva de vuelos. Para el cálculo de la ruta óptima entre las ciudades de origen y destino y las ciudades donde se encuentran los aeropuertos de origen y destino se ha optado por el uso de una función proporcionada por la extensión de *PostgeSQL*, *PGRouting*, que hace uso del Algoritmo de Dijkstra para el

cálculo del camino más corto en un grafo dirigido con pesos, para calcular la ruta de menor coste, según la opción seleccionada, precio o duración del trayecto, entre un nodo origen y un nodo destino, siendo en este caso las ciudades mencionadas los nodos del grafo construido a partir de las rutas aportadas por los operadores que serían consideradas los arcos.

4.10. Pruebas

A medida que se han ido implementando las distintas funcionalidades implicadas en cada iteración, como indica el Proceso Unificado, se han realizado pruebas unitarias sobre cada componentes implementado, al terminar su desarrollo, para cerciorarse de un funcionamiento correcto y estable.

Además, se han realizado pruebas de integración para asegurar que el componente se ha integrado con el resto sistema de manera correcta, en especial los componentes más delicados como pueden ser los que conectan con sistemas externos como es el caso del planificador de viajes o la API de operadores.

Finalmente se han llevado a cabo una serie de pruebas del sistema en las que se comprueba el funcionamiento y la integración de este a nivel global, como un todo.

5. Conclusiones y líneas de trabajo futuras

En este apartado se expondrán las conclusiones y las líneas de trabajo futuras que se proponen como continuación a este proyecto.

Se realizó un proceso de investigación previo al comienzo de este proyecto enfocado en el sector del transporte con el fin de identificar las necesidades que actualmente destacan dentro de este ámbito. Tras esta investigación se identificó la necesidad de una plataforma donde el cliente pudiese acceder a todos sus billetes de viaje sin importar el medio de transporte, y con esta la conveniencia de facilitar la herramienta a los profesionales del sector para así simplificar y reducir al máximo el esfuerzo necesario por el lado del cliente.

Así surge el planteamiento del Trabajo de Fin de Grado titulado "(SingleTicket) Plataforma para la gestión de billete multimodal", un sistema cuyos objetivos principales son la gestión de billetes multimodales (tren, bus y avión) y la planificación de viajes a través de la combinación de estos medios. Tras definir estos objetivos, se infirió el conjunto total de objetivos a cumplir para la realización de esta idea, gestión de usuarios, gestión de una API para operadores, gestión de viajes, planificación de viajes y gestión de puntos de interés.

Con estos objetivos definidos, el siguiente paso a dar fue realizar el análisis y diseño del sistema. Se decidió desarrollar el sistema como una aplicación web por la oportunidad que aportaba esto de hacer uso de la plataforma desde cualquier dispositivo que disponga de un navegador web cumpliendo con esto objetivos secundarios como la portabilidad y escalabilidad.

Para la gestión de viajes, y con esto la gestión de billetes e incidencias, así como la gestión de usuarios solo hubo que desarrollar un CRUD (*Créate, Reed, Update, Delete*) para cada componente relacionado con estos objetivos utilizando la pila elegida para el desarrollo de este proyecto, Vue.js para el *frontend*, node.js para el *backend* y una base de datos PostgreSQL para el almacenamiento de todos los datos estáticos necesarios para el funcionamiento de la aplicación. Se implementaron todas las vistas, controladores y modelos necesarios para el cumplimiento de todos los requisitos funcionales planteados para estos objetivos.

Para cubrir el objetivo del api de operadores se facilitaron una serie de interfaces HTTP a través de las cuales los operadores, identificándose a través de un token de operador enviado en el *body* de la petición por seguridad, pueden realizar todas las funciones definidas en los requisitos de la aplicación.

En cuanto a la planificación de viajes se desarrolló en varias partes, primero se definió como medio de transporte principal y prioritario el avión. Para planificar los vuelos se hizo uso del API de "*Flight booking"* proporcionada por la empresa del sector turismo/transportes Amadeus. A través de esta API se pueden consultar los aeropuertos cercanos a las ciudades desde y hasta las que se desea viajar, consultar los vuelos disponibles y sus precios entre esos aeropuertos y realizar la reserva de estos. En cuanto a los otros dos medios de transporte con los que trabaja la plataforma, se les ofrece a las agencias la posibilidad de calcular las rutas más rápidas o baratas de tren o bus desde las ciudades de origen o destino hasta las ciudades de los aeropuertos de origen o destino en el caso de que estas no sean las mismas, haciendo uso de las rutas almacenadas en el sistema por parte de los operadores de transporte. Para el cálculo de estas rutas de utiliza la extensión de PostgreSQL PGRouting.

Por último, para la extracción de puntos de interés, se ha implementado un script en Python que extrae los datos deseados de la API Overpass de OpenStreetMaps. Al ser un proceso que lleva

su tiempo y puede llegar a consumir bastantes recursos, se programará el CRON del servidor para que se ejecute una vez al mes con el motivo de mantener los datos de puntos de interés actualizados.

Tras las pruebas realizadas a la finalización del desarrollo, se puede afirmar que cada componente cumple su función, con la restricción del planificador de viajes, donde los datos devueltos son un *set* de datos de pruebas, ya que para trabajar con la API Amadeus se necesita de un token proporcionado después de pagar una cuota que no se ha abonado para el desarrollo de este proyecto. Aun así, los *endpoints* están preparados para obtener y devolver datos reales en el momento que se disponga de dicho token.

Con esto se concluye que se han cumplido los objetivos principales como secundarios planteados para el desarrollo de este proyecto. Además, se ha cumplido mi objetivo personal de desarrollarme como programador y me ha aportado una valiosa primera experiencia real con el uso de numerosos métodos y herramientas aprendidas durante mis estudios de grado, como son patrones de arquitectura, herramientas y metodologías tanto de diseño y análisis de requisitos como de planificación temporal y estimación de costes.

En cuanto a las líneas propuestas para trabajo futuro, sería deseable facilitar la compra de los billetes a través de la plataforma, facilitando una información más detallada de estos y ofreciendo una pasarela de pago entre agencias y operadores.

Otra línea interesante que seguir sería permitir a los usuarios solicitar los viajes y comprarlos a las agencias a través de la plataforma. Esto podría hacerse implementando una herramienta de solicitud de viajes que de manera sencilla permitiese al usuario detallar a la agencia los requisitos deseados para la panificación de su viaje y haciendo llegar estos requisitos a la agencia. En cuanto al pago, al igual que en la línea anterior se solucionaría con la implementación de una pasarela de pago entre usuario y agencia.

Por último, otra línea más sencilla pero igual o más importante sería aumentar la funcionalidad en cuanto a la gestión de los viajes, por ejemplo, facilitando cancelaciones y modificaciones de estos una vez creados.

6. Referencias

[1] Amazon Web Services, Inc., «¿Qué es una API? - Guía sobre las API para principiantes - AWS,» 2022. [En línea]. Available: https://aws.amazon.com/es/what-is/api/#:~:text=API%20significa%20%E2%80%9Cinterfaz%20de%20programaci%C3%B3n,de%20servicio%20entre%20dos%20aplicaciones.. [Último acceso: 04 09 2022].

[2] OpenStreetMap, «Puntos de interés - OpenStreetMap Wiki,» 12 12 2020. [En línea]. Available: https://wiki.openstreetmap.org/wiki/ES:Puntos_de_inter%C3%A9s. [Último acceso: 04 09 2022].

[3] Wikipedia, «Algoritmo de Dijkstra - Wikipedia, la enciclopedia libre,» 06 Enero 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra. [Último acceso: 04 Septiembre 2022].

[4] F. Cristancho, «¿Qué es un framework en programación? - Talently,» 8 Febrero 2022. [En línea]. Available: https://talently.tech/blog/que-es-un-framework-en-programacion/. [Último acceso: 04 09 2022].

[5] S. Mohanadasan, «¿Qué es Express.js? Todo lo que Debes Saber - Kinsta,» 25 Julio 2022. [En línea]. Available: https://kinsta.com/es/base-de-conocimiento/que-es-express/#qu-es-expressjs. [Último acceso: 4 Septiembre 2022].

[6] M. A. Alvarez, «Qué es MVC - Desarrollo Web,» 28 7 2020. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html. [Último acceso: 04 Septiembre 2022].

[7] IONOS, «te explicamos cómo funciona el patrón singleton - IONOS,» 19 Febrero 2021. [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/patron-singleton/. [Último acceso: 4 Septiembre 2022].

[8] A. Durán Toro y B. Bernárdez Jiménez, «http://www.lsi.us.es/,» Octubre 2022. [En línea]. Available: http://www.lsi.us.es/docs/informes/lsi-2000-10.pdf. [Último acceso: 04 09 2022].

[9] S. Borges, «¿Qué es PostgreSQL? - Para qué sirve, Características e ...,» 19 Noviembre 2019. [En línea]. Available: https://blog.infranetworking.com/servidor-postgresql/. [Último acceso: 5 Septiembre 2022].

[10] N. Losada, «PostgreSQL y PostGIS: Qué son y cómo se relacionan,» 25 Mayo 2022. [En línea]. Available: https://geoinnova.org/postgresql-ypostgis-que-son-y-como-se-relacionan/. [Último acceso: 5 Septiembre 2022].

[11] D. Alonso, «Cómo configurar una base de datos PostGIS para pgRouting ...,» 26 Enero 2015. [En línea]. Available: https://mappinggis.com/2015/01/como-configurar-una-base-de-datos-postgis-para-pgrouting/. [Último acceso: 5 Septiembre 2022].

[12] E. Symington, «Amadeus lanza una nueva API con funciones NDC,» 22 Julio 2019. [En línea]. Available: https://amadeus.com/es/articulos/noticias/amadeus-lanza-una-nueva-api-con-funciones-

ndc#:~:text=Amadeus%20Travel%20API%20es%20una,gradualmente%20en%20todo%20el%20mundo.. [Último acceso: 5 Septiembre 2022].

[13] S. Higuera, «6. Overpass API — DownloadOSMData 1.0 documentation,» 2016. [En línea]. Available: https://osmdatause.readthedocs.io/en/latest/overpassapi.html. [Último acceso: 5 Septiembre 2022].

[14] Santander Universidades, «Python: qué es y por qué deberías aprender a utilizarlo,» 20 Abril 2021. [En línea]. Available: https://www.becas-santander.com/es/blog/python-que-es.html. [Último acceso: 5 Septiembre 2022].

[15] R. Ramos, «¿Qué es JavaScript y para qué sirve? - Rafa Ramos,» [En línea]. Available: https://soyrafaramos.com/que-es-javascript-para-que-sirve/. [Último acceso: 5 Septiembre 2022].

[16] A. D. Toro, «Herramienta REM,» 25 Noviembre 2004. [En línea]. Available: http://www.lsi.us.es/descargas/descarga_programas.php?id=3. [Último acceso: 6 Sepriembre 2022].

[17] «ezestimation - Sitios de Google,» [En línea]. Available: https://sites.google.com/site/ezestimation/. [Último acceso: 5 Sepriembre 2022].

[18] S. O. Lugo, «¿Qué es Microsoft Project y para qué sirve? - Alpha Consultoría,» [En línea]. Available: https://www.alpha-consultoria.com/que-es-microsoft-project-y-para-que-sirve/. [Último acceso: 6 Septiembre 2022].

[19] [En línea]. Available: https://www-visual--paradigm-com.translate.goog/solution/freeumltool/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc.

[20] F. Flores, «Qué es Visual Studio Code y qué ventajas ofrece,» 22 Julio 2022. [En línea]. Available: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/. [Último acceso: 5 Septiembre 2022].

[21] J. González, «LiderDeproyecto,» [En línea]. Available: http://www.liderdeproyecto.com/articulos/el_papel_central_de_los_procesos_ing_de_negocios.html. [Último acceso: 04 Septiembre 2022].

[22] Fundación Wikimedia, Inc., «Wikipedia,» 6 Enero 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra.

[23] E. I. G. Pérez, «¿Qué es Vue.JS?,» 1 Abril 2019. [En línea]. Available: https://codigofacilito.com/articulos/que-es-vue. [Último acceso: 4 Septiembre 2022].

[24] J. Lucas, «Qué es NodeJS y para qué sirve - OpenWebinars,» 4 Septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-nodejs/. [Último acceso: 5 Septiembre 2022].

[25] H. M. Fernandes, «¿Qué es TypeScript y para qué sirve?,» 9 Julio 2020. [En línea]. Available: https://marquesfernandes.com/es/tecnologia-es/what-and-typescript-and-for-that-serves/. [Último acceso: 5 Septiembre 2022].

[2 Vue.js, «Introducción - Vue.js,» [En línea]. Available: https://es.vuejs.org/v2/guide/.
6] [Último acceso: 5 Septiembre 2022].