My Itinerary: Aplicación web para la creación de itinerarios de viaje

Trabajo de Fin de Grado
GRADO EN INGENIERÍA INFORMÁTICA

Septiembre de 2023

Autor
María González García

Tutores
Álvaro Lozano Murciego
María N. Moreno García

Certificado de los tutores

D. Álvaro Lozano Murciego y Dña. María N. Moreno García, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

HACEN CONSTAR

Que el trabajo titulado “My Itinerary: Aplicación web para la creación de itinerarios de viaje”, que se presenta, ha sido realizado por Dña. María González García con DNI 70905948F y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática en esta Universidad.

Salamanca, a 7 de septiembre de 2023

Fecha: 2023.09.07 20:57:27 +02'00'

MORENO GARCIA MARIA NAVELONGA - 07843490F

Firmado digitalmente por MORENO GARCIA MARIA NAVELONGA - 07843490F
Fecha: 2023.09.07 20:35:02 +02'00'

Fdo.:_____

Resumen

La pandemia causada por la COVID–19 hizo que el estilo de vida y los hábitos de la población cambiaran drásticamente. Uno de los ámbitos más afectados fue el del turismo. La situación producida hizo que los turistas se plantearan nuevas formas de viajar y de disfrutar de su tiempo libre.

Uno de los sectores que más creció fueron los viajes en *campers* o autocaravanas, tipos de viajes que se centran más en realizar otro tipo de vacaciones que dotan de más independencia y flexibilidad al turista, aunque estas signifiquen en algunos casos un mayor nivel de polución y un mayor impacto medioambiental en comparación con las vacaciones tradicionales.

Sin embargo, este tipo de turismo tiene una problemática asociada que está relacionada con el diseño de un itinerario de viaje. El perfil de turista actual es más crítico y selectivo con la experiencia, visitando solo aquellos lugares que se adapten a sus requisitos y evitando otros que, por muy turísticos que sean, puedan no satisfacer sus expectativas de viaje. Además, el turista actual y los operadores turísticos están más concienciados que nunca con el respeto al medio ambiente y la realización de lo que se denomina un turismo sostenible.

Esto supone que, cuando se planee el itinerario a seguir en el viaje, se requiera invertir mucho tiempo previamente explorando e investigando el destino para poder extraer una cantidad de información suficiente sobre él y, a partir de esta, organizar los días de viaje, lo que no suele ser una tarea fácil y no siempre el resultado es el más respetuoso con el medio ambiente.

Para dar respuesta a la problemática anterior, se propone en este Trabajo Fin de Grado un sistema capaz de asistir a los usuarios en la creación de itinerarios de viaje, especialmente viajes realizados en vehículos como autocaravanas o similares, donde se realiza una recomendación personalizada (basada en los propios gustos del usuario) y, además, organiza los días en base a la ruta más óptima a seguir para ayudar a la reducción, en la medida de lo posible, del impacto medioambiental de los trayectos. Para ello, el sistema se apoya en dos subsistemas externos: un sistema de recomendación de puntos de interés (POIs) para el usuario basado en sus gustos o el propósito del viaje, y un sistema de optimización de la ruta de acuerdo con los POIs a visitar y los lugares de pernoctación para encontrar la mejor ruta entre ellos.

En este documento, se detalla la investigación realizada, los requisitos del sistema, las técnicas y herramientas empleadas, las distintas etapas de desarrollo e implementación del proyecto, así como las conclusiones tras su realización

Palabras clave: creación de itinerarios, turismo, sistema de recomendación de POIs, sistema de optimización de rutas.

Abstract

The COVID-19 pandemic changed drastically the lifestyle and habits of the population. One of the most affected areas was tourism. Due to this emergency situation, tourists were obliged to consider new ways of travelling and enjoying their leisure time.

In this scenario, campervan and motorhome travel was one of the sectors that grew the most, thanks to it being a type of travel that gives more independence and flexibility to the tourist. Even if this means, in some cases, a higher level of pollution and a greater environmental impact compared to traditional holidays.

However, this new type of tourism presents a problem related to the design of a travel itinerary. Today's tourist profile is more critical and selective with the experience, visiting only those places that really suit their requirements and avoiding others that, however touristy they may be, may not meet their travel expectations. Moreover, today's tourists and tour operators are more aware than ever of the need to respect the environment and to carry out what is known as sustainable tourism.

This means that, when planning the trip itinerary, it is necessary to invest a lot of time exploring and researching the destination beforehand to get enough information about the place and to organise the whole trip, which is not usually an easy task, and the result may not always be the most environmentally friendly possible.

In response to the problems mentioned above, this Final Degree Project proposes a system capable of assisting users in the creation of travel itineraries, especially trips made in vehicles such as motorhomes or similar, where a personalised recommendation is made regarding the user's own tastes and the days are organised based on the most optimal route to follow in order to reduce the environmental impact of the journeys as much as possible. To do this, the system relies on two external subsystems: a POIs (points of interest) recommender system to the user, based on their tastes or on the purpose of the trip, and a route optimisation system according to the POIs to be visited and the places to spend the night in order to find the best route between them.

This paper contains the research carried out, the requirements of the system, the techniques and tools used, the different stages of development and implementation of the project, as well as the conclusions after its completion.

Keywords: itinerary creation, tourism, POI recommender system, route optimisation system.

ÍNDICE

1. INTRODUCCIÓN 1

1.1. PRESENTACIÓN DE LA TEMÁTICA DEL PROYECTO 1
1.2. ESTRUCTURA DE LA MEMORIA 2

2. OBJETIVOS DEL PROYECTO 3

2.1. OBJETIVOS DEL PROYECTO 3
2.2. OBJETIVOS PERSONALES 4

3. CONCEPTOS TEÓRICOS 5

3.1. DISEÑO DE ITINERARIOS TURÍSTICOS 5
3.2. SISTEMAS DE RECOMENDACIÓN 6
3.3. SISTEMAS DE OPTIMIZACIÓN DE RUTAS 7
3.4. *PROGRESSIVE WEB APP (PWA)* 8

4. TÉCNICAS Y HERRAMIENTAS 9

4.1. *FIREBASE* 9
4.2. *VUE.JS* 9
4.3. PYTHON Y FAST API 10
4.4. POETRY 10
4.5. DOCKER 11
4.6. ARQUITECTURA *SERVERLESS* 11
4.7. HERRAMIENTAS DE DISEÑO DE LA INTERFAZ GRÁFICA 12
4.8. HERRAMIENTAS CASE 13
4.8.1. MICROSOFT PROJECT 13
4.8.2. EZESTIMATE 13
4.8.3. VISUAL PARADIGM 13
4.8.4. DRAW.IO 13
4.8.5. PLANTUML 14

5. ASPECTOS RELEVANTES DEL DESARROLLO 15

5.1. CICLO DE VIDA 15
5.2. ESPECIFICACIÓN DE REQUISITOS 18
5.3. ESPECIFICACIÓN DE ANÁLISIS 20
5.4. ESPECIFICACIÓN DE DISEÑO 23
5.4.1. DIAGRAMA DE CLASES 23
5.4.2. SUBSISTEMAS DE DISEÑO 24
5.4.3. VISTA DE ARQUITECTURA 26
5.4.4. BASE DE DATOS 28
5.4.5. MODELO DE COMPONENTES Y MODELO DE DESPLIEGUE 29
5.4.6. PATRONES DE DISEÑO SOFTWARE 30

5.4.7. DISEÑO DEL SISTEMA DE RECOMENDACIÓN 32
5.4.8. DISEÑO DEL SISTEMA DE OPTIMIZACIÓN DE RUTAS 32
5.4.9. DISEÑO DE LA IDENTIDAD GRÁFICA 33
5.4.10. DISEÑO DE LA INTERFAZ 34
5.5. IMPLEMENTACIÓN 36
5.5.1. VUE.JS 36
5.5.2. BASE DE DATOS 36
5.5.3. ENTORNO DE DESARROLLO 37
5.5.3.2. INSTALACIÓN DE FIREBASE 37
5.6. PRESENTACIÓN DE LA APLICACIÓN 39
5.6.1. GESTIÓN DE USUARIOS 39
5.6.2. GESTIÓN DE RUTAS 42
5.6.3. GESTIÓN DE ITINERARIOS 46

6. CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS 49

6.1. CONCLUSIONES 49
6.2. LÍNEAS DE TRABAJO FUTURAS 50

7. REFERENCIAS 51

ILUSTRACIONES

Ilustración 1 Diagrama de las fases de la planificación temporal del PU [52]..... 15
Ilustración 2 Vista de la primera iteración (Fase de inicio)..... 17
Ilustración 3 Diagrama de paquetes..... 18
Ilustración 4 Paquete de Gestión de usuarios..... 18
Ilustración 5 Paquete de Gestión de rutas..... 19
Ilustración 6 Paquete de Gestión de itinerarios..... 19
Ilustración 7 Modelo del dominio del análisis..... 20
Ilustración 8 Arquitectura (análisis). Relaciones entre paquetes..... 21
Ilustración 9 Diagrama de secuencia del CU-0009. Seleccionar puntos de interés..... 22
Ilustración 10 Diagrama de clases del diseño..... 23
Ilustración 11 Subsistema: paquete *View*..... 24
Ilustración 12 Subsistema: paquete *Stores*..... 25
Ilustración 13 Subsistema: paquete *Services*..... 25
Ilustración 14 Subsistema: API de recomendación..... 26
Ilustración 15 Arquitectura del sistema..... 27
Ilustración 16 Diagrama de realización de CU-Diseño del CU - 0009. Seleccionar puntos de interés..... 28
Ilustración 17 Estructura de un documento de la colección Itinerarios de Firebase..... 29
Ilustración 18 Modelo de componentes..... 29
Ilustración 19 Relaciones entre componentes y artefactos..... 30
Ilustración 20 Modelo de despliegue..... 30
Ilustración 21 Esquema del patrón de diseño MVVM [57]..... 31
Ilustración 22 Esquema del patrón de diseño *Facade* [58]..... 32
Ilustración 23 Logotipo y paleta de colores..... 33
Ilustración 24 Vista del prototipo inicial para la creación de rutas..... 34
Ilustración 25 Vista de introducción de datos de la creación de rutas en el diseño final 35
Ilustración 26 Vista de la selección de POIs en el diseño final..... 35
Ilustración 27 Estructura de la colección Itinerarios de la base de datos..... 36
Ilustración 28 Cabecera de la página de *landing*..... 39
Ilustración 29 Formulario de registro..... 40
Ilustración 30 Formulario de inicio de sesión..... 40
Ilustración 31 Página de restauración de contraseña..... 41
Ilustración 32 Perfil del usuario..... 41
Ilustración 33 Vista de modificación de datos..... 42
Ilustración 34 Vista de bienvenida de la creación de itinerarios..... 42
Ilustración 35 Vista de introducción de datos del viaje..... 43
Ilustración 36 Vista de la elección de POIs..... 43
Ilustración 37 Vista de la elección de campings..... 44
Ilustración 38 Vista de la personalización de la ruta..... 45
Ilustración 39 Vista del resumen de la información de la ruta (parte 1)..... 45
Ilustración 40 Vista del resumen de la información de la ruta (parte 2)..... 46
Ilustración 41 Vista del itinerario final (parte 1)..... 46
Ilustración 42 Vista del itinerario final (parte 2)..... 47
Ilustración 43 Vista del historial de itinerarios..... 47
Ilustración 44 Vista del mapa de exploración..... 48

1. Introducción

1.1. Presentación de la temática del proyecto

El año 2020 estuvo marcado por la recesión causada por la pandemia debida a la COVID–19 y sus consecuencias, que repercutieron prácticamente en todos los ámbitos (economía, sociedad, educación, etc.). Esta pandemia afectó a toda la población en mayor o menor medida, haciendo que cambiaran sus hábitos diarios (interacciones sociales [1][2] o actividades rutinarias como simplemente ir al supermercado [3]), pero también afectó a los ámbitos más relacionados con el ocio, en especial al turismo [4]. Este sector ha sufrido uno de los cambios más notables, uno de ellos relacionado con el turismo en autocaravana o "camper". Esto se puede constatar en el estudio hecho por la *European Caravan Federation*, en el que se afirma que alrededor de 235.000 vehículos recreaciones fueron matriculados en Europa en 2020 [5], siendo este el mejor resultado desde 1980, dato que puede tomarse como indicativo de cómo la población ha preferido adaptarse y optar por estas alternativas turísticas más independientes antes que seguir con las estancias tradicionales en hoteles u otro tipo de lugares de pernoctación estáticos.

Debido a este cambio, se pueden clasificar los veraneantes en dos grupos según la planificación de sus vacaciones: por un lado, es posible diferenciar a aquellos que prefieren delegar el proceso de creación de un itinerario a las agencias de viajes tradicionales; y, por otro lado, están los más innovadores e independientes que decidieron ser ellos mismos los planificadores del itinerario [6]. Es cierto que ambos grupos presentan aspectos positivos claros. Por ejemplo, en el caso de las agencias, el usuario solo tendría que preocuparse de comunicar al trabajador el propósito de su viaje y la idea de vacaciones que tiene en mente, siendo el resto del trabajo responsabilidad de su interlocutor; mientras tanto, en el otro grupo, al ser el usuario el que planifica todo el itinerario, se podría llegar a un resultado más acertado, que estuviese formado por esos puntos de interés que son realmente más afines a sus gustos. Sin embargo, esto supone un esfuerzo importarte por parte del usuario, ya que pasará muchas horas explorando los diferentes destinos e indagando para encontrar lugares que se ajusten a sus preferencias, lo cual puede ser una tarea bastante tediosa.

Como se espera que este cambio perdure en el tiempo, ya que es un estilo vacacional que está encajando muy bien con algunos usuarios (todos esos compradores de vehículos como autocaravanas tomaron esa decisión pensando en un largo plazo), se han intentado diseñar sistemas [7]–[10] que faciliten esta tarea lo máximo posible, haciendo así que el usuario esté más satisfecho con el proceso de creación, aunque es cierto que todavía hay un gran margen de mejora en este campo incorporando nuevas tecnologías.

Por otro lado, tanto el turista como los operadores turísticos buscan hoy más que nunca que el impacto del turismo sobre el medio ambiente sea el mínimo posible. Este turismo se denomina turismo sostenible y pretende que el consumidor (el turista) esté informado de la huella que su acción turística provoca en el medio ambiente y también busca que los operadores turísticos ofrezcan alternativas respetuosas con el medio ambiente para un consumo informado y responsable. Sin embargo, este factor no es algo que el turista pueda tener en cuenta de forma sencilla en el diseño de su itinerario.

Para dar respuesta a las problemáticas expuestas anteriormente, en este Trabajo de Fin de Grado se plantea, como objetivo principal, el desarrollo de un sistema que facilite la labor de creación de itinerarios de viaje de manera sencilla y enfocada en el usuario, es decir, teniendo en cuenta los intereses y los gustos del usuario para ofrecer aquellos puntos que se estima que van a encajar con él, incrementando así la satisfacción del viajante al final de este periodo vacacional. Además, permitirá al usuario consultar estos itinerarios creados cuando él desee (gracias a la implementación de un historial). Por último, el sistema proporcionará al usuario la mejor ruta para cada día del itinerario diseñado, buscando reducir la huella de CO2 de dicho itinerario.

1.2. Estructura de la memoria

El resto de este documento está estructurado como en los siguientes apartados:

- **Objetivos del proyecto**: en este apartado se definen y explicas estos objetivos, tanto los del software como aquellos personales.
- **Conceptos teóricos**: se explican algunos conceptos relativos al dominio del problema que aborda la aplicación y que permiten entender el proyecto de manera completa.
- **Técnicas y herramientas**: se indican las herramientas empleadas durante las distintas fases de la realización del proyecto.
- **Aspectos relevantes del desarrollo del proyecto**: se explican las partes más importantes del proceso de desarrollo y se destacan las características esenciales del proceso seguido.
- **Conclusiones y líneas de trabajo futuras**: se discuten y analizan los resultados finales tras a ejecución del proyecto y se explican las posibles mejoras y líneas de trabajo futuro.

Además, se adjuntan cinco anexos técnicos para ampliar la información proporcionada sobre el proyecto:

- **Anexo I. Planificación temporal**: se especifica el plan a seguir a lo largo del tiempo para la realización del proyecto.
- **Anexo II. Especificación de requisitos del software**: donde se definen los requisitos del proyecto.
- **Anexo III. Especificación de análisis y diseño**: contiene el análisis y el diseño del proyecto.
- **Anexo IV. Manual del programador**: detalla el contenido más técnico en cuanto al desarrollo del proyecto.
- **Anexo V. Manual de usuario**: que constituye una guía de usuario para instalar y usar la plataforma desarrollada.

2. Objetivos del proyecto

En este apartado, se describe el objetivo principal del proyecto, así como los objetivos específicos que se deben cumplir para llevarlo a cabo.

2.1. Objetivos del proyecto

El **objetivo principal** del proyecto desarrollado es el diseño e implementación de una aplicación web para la creación de itinerarios de viaje basados en los gustos personales del usuario, para lo que se utilizará un sistema de recomendación de puntos de interés o *Points of Interests* (POIs) afines a este, y, además, enfocado sobre todo a los viajes en autocaravana para ayudar a reducir el consumo y la huella de carbono que se produce durante este tipo de viajes, para lo que se empleará un sistema de optimización de rutas.

Para conseguir este objetivo principal, se plantean se describen unos objetivos específicos:

- **Desarrollar un módulo de gestión de usuarios**: el sistema deberá ser capaz de gestionar correctamente los usuarios que se registren en él, permitiendo el alta, baja, modificación y consulta de sus datos, así como todo el apartado de autenticación en el sistema.

- **Desarrollar un sistema de recomendación de POIS**: se desarrollar un sistema capaz de recomendar POIs relativos a un *feedback* en forma de lenguaje natural proporcionado por el usuario y en una localización geográfica concreta.

- **Desarrollar un sistema de optimización de rutas**: la aplicación deberá ofrecer una ruta óptima o subóptima al usuario con el objetivo de producir la menor huella de carbono en el desarrollo de su viaje. Por ello uno de los subsistemas del sistema a desarrollar deberá encargarse de optimizar las rutas entre los POIs elegidos por el usuario para cada día del itinerario.

- **Desarrollar un módulo de gestión de rutas**: el sistema deberá permitir la creación de las rutas que componen cada uno de los días del itinerario de viaje. Para ello, el sistema manejará información como los lugares a visitar (POIs), los emplazamientos de pernoctación, y los puntos de inicio y fin de itinerario completo. Además, deberá interactuar con el sistema externo de optimización para poder crear la ruta a seguir, ordenando esos datos mencionados anteriormente para crear así el objeto fundamental y central del itinerario.

- **Desarrollar un módulo de gestión de itinerarios**: el sistema, una vez haya creado las rutas que formarán el viaje, deberá ser capaz de incluirlas en un itinerario, formando así un resumen detallado del viaje completo, con información sobre el destino, los lugares a visitar, y otros aspectos más técnicos que indicarán la cantidad de CO2 emitido, etc.

2.2. Objetivos personales

Este proyecto tiene una motivación personal por varias razones. A la hora de elegir un tema para este proyecto, quería enfocarme en la parte más relacionada con el desarrollo *frontend*, ya que es la que menos había tratado y la que más llamaba mi atención. Además, viajar es una de mis aficiones y algo que hago con bastante frecuencia, y, en concreto, este tipo de viajes por carretera que cuentan con múltiples paradas son comunes para mí. Por ello, al estar familiarizada con esto y ser consciente de la problemática que suponen con frecuencia este tipo de viajes, cuando los tutores me presentaron la idea de crear este sistema, me pareció una buena oportunidad para intentar darles una solución y ayudar a otros viajeros que estén en la misma circunstancia.

Además, también se tuvieron en cuenta una serie de objetivos más técnicos que ayudaran a conocer más tecnologías en uso actualmente:

- Obtener nuevos conocimientos sobre el desarrollo *frontend* con el aprendizaje de *frameworks* como *VueJS* [11], descubriendo cómo se emplea en la actualidad, sus detalles de implementación, etc.
- Investigar de forma introductoria los sistemas de recomendación y de optimización, desconocidos hasta la fecha.
- Conocer el funcionamiento de bases de datos NoSQL como las que ofrece *Firebase Cloud Firestore* [12], aprendiendo a manejar los servicios que ofrece.
- Poder aplicar los conocimientos adquiridos durante el desarrollo del Grado de manera más práctica en un proyecto completo que resuelva un problema real.

3. Conceptos teóricos

3.1. Diseño de itinerarios turísticos

En la actualidad, se ha puesto el foco en la sostenibilidad como uno de los valores fundamentales para garantizar la conservación del medio ambiente. Este valor trasciende múltiples aspectos de la vida cotidiana, siendo el turismo también uno de los ámbitos donde este aspecto ha cobrado gran importancia. La planificación de los viajes turísticos y el turismo sostenible son dos conceptos entrelazados que buscan aunar el disfrute y la satisfacción del viajero con la exploración del entorno desde la responsabilidad y el respeto hacia el medio ambiente y las comunidades locales.

A pesar de todas estas características positivas, el turismo también puede tener un impacto negativo y muy significativo en el ecosistema, sobre todo en los viajes realizados por vehículos recreacionales, donde el automóvil realiza grandes trayectos. Esta es una de las causas de la presencia del turismo sostenible como enfoque imprescindible para garantizar que estas experiencias no comprometan el entorno. Por ello, la planificación de los itinerarios desde la sostenibilidad implica adoptar medidas que promuevan prácticas responsables y conscientes, siendo una de ellas el uso eficiente de recursos, que en este caso se traduce en la búsqueda de la ruta óptima a seguir por el viajero y en ahorro del gasto de combustible.

Para poder hallar la ruta, hay que tener en cuenta una gran cantidad de factores, (complejidad del terreno, estado de las carreteras, recepción de señal, etc.). Por ello, el método tradicional basado en acudir a una agencia de viaje o una empresa de alquiler de estos automóviles y tener en cuenta los factores mencionados previamente para calcular la mejor ruta a seguir puede no resultar factible.

El principal problema de este método deriva de que el trabajador puede no tener en cuenta algún aspecto o usar alguna herramienta poco refinada que haga que la ruta no sea óptima, en cuyo caso el viaje contará con un mayor impacto medioambiental. Esto también conlleva problemas desde el punto de vista más humano: si se desea realizar un viaje personalizado, el trabajador deberá pasar mucho tiempo tanto conociendo todos los detalles del viaje a realizar, como buscando información sobre el destino, para poder encontrar los lugares afines al viajero. Además, al calcular la ruta, tendrá que introducir todos los datos (paradas, lugares de pernoctación, etc.) en dicho sistema. Esto hace que el trabajador dedique mucho tiempo de su jornada laboral en esta tarea, lo que impide que se centre en otras actividades, viéndose así incrementada su carga de trabajo.

Por todas estas razones, es necesario encontrar una solución que no solo cualquier usuario de internet pueda utilizar, sino que también pueda ayudar a las empresas a liberar la carga de trabajo de sus empleados. Esto hace que en este proyecto se busque resolver la problemática expuesta mediante varios sistemas externos al desarrollado, los cuales darán información sobre los puntos de interés (POIs) afines al usuario y sobre la ruta óptima que une todos ellos, para así poder llegar a un itinerario final que reúna toda la información sobre nuestro viaje. A continuación, se explican algunos conceptos necesarios para el buen entendimiento del proyecto:

- **POI (Punto de interés, PDI o *Point of Interest*)**: se refiere a un punto de ubicación específica que el usuario al que se le presenta pueda encontrar interesante en algún aspecto. En este caso, se consideran POIs aquellos lugares turísticos que encajan con las características y preferencias especificadas por el usuario que está haciendo uso de la plataforma.

- **Itinerario**: es el resultado final. Incluye las rutas a seguir (con los POIs a visitar, etc.), las fechas de realización del viaje y otros datos más técnicos (como el CO2 producido) o de interés sobre los puntos a visitar (descripciones de los lugares que se visitan, del destino concreto al que se viaja, etc.). Una vez creado, se podrá consultar y descargar siempre que el usuario lo desee.

3.2. Sistemas de recomendación

Los sistemas de recomendación son herramientas que permiten proveer al usuario información que le resultará de interés entre una vasta cantidad de información. Estos sistemas a menudo recogen previamente datos sobre dicho usuario, pudiendo así personalizar su experiencia. En este caso, el concepto clave del sistema de recomendación de puntos de interés utilizado es sugerir nuevos lugares a visitar, como museos, restaurantes, etc., basándose en los gustos del usuario y estando limitado por la región geográfica indicada.

Estos gustos podrían ser recogidos de varias maneras: gracias a un sistema de *ranking* entre algunos elementos; o bien de manera más implícita mediante interacciones con el sistema (clicks, me gusta, visitas, etc.). Tradicionalmente, se han empleado algoritmos como Filtrado Colaborativo [13] para implementar este tipo de sistemas.

A pesar de esto, en muchas ocasiones, la información de la que se dispone inicialmente de un usuario en el sistema es bastante limitada, por lo que el sistema de recomendación no tiene suficiente información con la que trabajar para realizar la recomendación. A esta problemática, común a muchas técnicas de recomendación, se la conoce en la literatura como el problema del arranque en frío o *cold start* [14], [15].

Esto provoca que sea habitual en muchas redes sociales o software con sistemas de recomendación, obtener de forma preliminar información explicita por parte del usuario que indique dichos gustos y oriente de manera inicial al sistema. En el caso del turismo, para resolver este problema del arranque en frío, los sistemas tienden a mostrar información sobre los cinco lugares más populares o mejor valorados del sitio indicado, aunque esto no es una recomendación personalizada. Una segunda solución sería hacer que el usuario indique las categorías que más se adaptan al viaje deseado de entre un grupo de ellas (como lo que se puede observar en algunas aplicaciones de música al crear un nuevo perfil de usuario). De esta manera, el sistema puede recoger preferencias del usuario que el usuario haya tenido que interactuar con él mediante clics, visitas, etc.

Por otro lado, las técnicas de recomendación se basan habitualmente en el historial del usuario (viajes que ha realizado, lugares que ha clicado “me gusta”). En ciertas ocasiones, en este dominio en concreto, el historial puede no ser representativo del viaje que se busca realizar en un futuro, por ejemplo, si el usuario siempre ha realizado viajes de ocio y

diversión y ahora desea disfrutar de uno relacionado con la cultura y la historia del destino, los datos que se tienen de ese usuario hasta ese momento no servirán de mucho en este caso.

Por estos motivos, en este trabajo se plantea la implementación del sistema de recomendación como un sistema que resuelva el arranque en frío y que además permita realizar recomendaciones. Para ello se utilizará una descripción en lenguaje natural que deberá proporcionar el usuario para cada viaje y, empleado técnicas de *Natural Language Processing* (NLP) y los *Large Language Models* (LLM) extraer POIs que estén relacionados con el motivo del viaje.

El diseño e implementación del sistema se detallarán en los aspectos relevantes del desarrollo del proyecto.

3.3. Sistemas de optimización de rutas

En este proyecto se define ruta como el camino que conecta un numero finito de puntos geográficos determinados. Es decir, en el caso de esta aplicación, se refiere a la conexión que se hace, por un lado, entre los distintos POIs y el punto de origen y de pernoctación para cada día; y, por otro lado, entre el punto de origen del viaje, los lugares de pernoctación de cada día y el punto final del viaje. De esta forma, se consiguen estos dos tipos de rutas: una por cada día (más detallada al incluir los POIs) y una de todo el viaje (menos detallada) de varios días. Ambas formarán parte del itinerario final.

Si se desea un sistema enfocado en un turismo sostenible, es necesario que el coste de realizar esa ruta sea el mínimo posible en términos de CO2. Por tanto, las rutas generadas entre los POIs seleccionados deben intentar ser la ruta óptima o al menos una subóptima cercana a ella. La creación de estas rutas turísticas optimizadas es un proceso complejo y estudiado ampliamente en la literatura científica donde de forma general se consideran los siguientes enfoques para formular el problema de optimización:

- **Problema del vendedor viajero (*Traveling Salesman Problem* o TSP)**: es un problema de optimización combinatoria que implica encontrar la ruta más corta que visita un conjunto de ubicaciones una sola vez, siendo el punto de inicio y de fin el mismo.

- **Problema de enrutamiento de vehículos (*Vehicle Routing Problem* o VRP)**: es una generalización del problema anterior donde el objetivo es encontrar rutas óptimas para varios vehículos que visitan un conjunto de ubicaciones y parten desde el mismo sitio.

- **Problema de orientación (*Orienteering Problem* u OP)**: se considera un problema de enrutamiento selectivo para determinar el orden y los nodos a visitar, siendo el objetivo la maximización del beneficio sin excedernos de un tiempo determinado. Este enfoque es más flexible ya que no obliga a visitar todos los puntos indicados. Se pueden aplicar también otras variantes de este problema que pueden ser útiles debido a que permiten la construcción de varias rutas al mismo tiempo [16].

Estos algoritmos son empleados por diversos servicios de enrutamiento y navegación como es el caso de Open Route Service [17] o Google Maps Directions [18].

En el ámbito de trabajo de este proyecto no se pretende el desarrollo de algoritmos de esta índole, delegando el computo de este proceso a esta clase de servicios externos que lo ofrecen a través de una *Application Programming Interface* (API). De este modo el sistema desarrollado en este proyecto se comunicará con servicios externos para completar este proceso de optimización cuando sea necesario.

3.4. *Progressive Web App (PWA)*

Una aplicación web progresiva o *Progressive Web App* (PWA) [19], es una solución basada en la web tradicional que incorpora una serie de particularidades que la hacen parecerse a una aplicación nativa para móviles o dispositivos similares. Estas particularidades van desde la posibilidad de utilizarlas de forma offline a recibir notificaciones *push*.

Estas aplicaciones web son usadas en cualquier navegador web, pero también permiten ser instaladas como una aplicación nativa, lo que hace que no sea necesario depender de tiendas externas como *Google Play* o *App Store* para su distribución y publicación, simplemente es necesaria conexión a Internet en un primer momento para su instalación ya que posteriormente algunas de ellas podrían incluso ser usadas sin conexión.

Finalmente, cabe destacar que, este tipo de aplicaciones, por norma general, suelen ser más rápidas que otras web ya que a menudo se valen del uso de la caché para almacenar la información que permanece estática en la aplicación.

Por todo ello, las PWA se presentan como una gran alternativa a tener en cuenta en el desarrollo de aplicaciones.

4. Técnicas y herramientas

En este apartado, se detallan las técnicas y herramientas utilizadas para el desarrollo del proyecto.

4.1. *Firebase*

Firebase [20] ha sido empleada como plataforma *serverless* en este proyecto. Se ha empleado para el almacenamiento de la información de la aplicación. Gracias a esta plataforma, se han podido almacenar los usuarios que forman parte del sistema y su autenticación y toda la información relacionada con los itinerarios. Se han utilizado las siguientes funcionalidades dentro del amplio catálogo que ofrece *Firebase*:

- ***Firebase Cloud Firestore*** [12] es una base de datos de documentos NoSQL que permite almacenar, sincronizar y consultar de manera sencilla los datos de las aplicaciones web. Es flexible, escalable y mantiene los datos sincronizados mediante agentes de escucha en tiempo real. Gracias a todas estas características, ha sido la opción perfecta para convertirse en la base de datos de la plataforma, tanto para los usuarios como para los itinerarios.
- ***Firebase Authentication*** [21] es una solución de identidad de extremo a extremo, compatible con correo electrónico y contraseña, Google y Facebook, entre otras plataformas. Además, proporciona servicios *backend*, SDKs sencillos de usar y bibliotecas que permiten la autenticación de los usuarios.
- ***Firebase Hosting*** [22] es un servicio de alojamiento de sitios web que ha permitido alojar la aplicación web progresiva de manera muy sencilla.

Se ha contemplado la utilización de plataformas como Supabase [23], que se presentan como alternativa *open source* a *Firebase* y que permite un despliegue *on premise* propio. Sin embargo, todavía se encuentra en una etapa temprana de desarrollo y finalmente se descartó.

La principal razón por la que se ha elegido Firebase como plataforma *serverless* de desarrollo es la gran cantidad de servicios que ofrece de manera gratuita y ampliamente documentados, haciendo que el primer contacto con una plataforma de este tipo haya sido tenido una curva de aprendizaje muy suave.

4.2. *Vue.js*

Para el desarrollo del *front-end* de la aplicación, se ha utilizado el marco de trabajo *Vue.js* [11]. Este popular marco de trabajo [24] se caracteriza por su "*accesibilidad, eficacia y versatilidad a la hora de crear interfaces de usuario web*". El uso de este *framework* permite tener todo el código de una vista en el mismo fichero (incluyendo la parte de *HTML*, *CSS* y *JavaScript*). A nivel arquitectónico, *Vue.js* utiliza el patrón MVVM (Model – View – ViewModel) [25]. El lenguaje usado en este proyecto ha sido *JavaScript* [26], junto con *HTML* [27] y *CSS* [28].

El principal motivo de esta elección ha sido debido a la sencillez de aprendizaje e implementación del *framework* en comparación con otras soluciones, como Angular [29], además del útil desacoplamiento que ofrece el patrón MVVM entre la lógica de negocio y la lógica de interfaz de las aplicaciones.

Para el desarrollo de la aplicación, se ha usado la plantilla *sakai vue cli* [30] de *PrimeFaces* [31], una biblioteca de componentes de código abierto que facilitan la creación de aplicaciones web. Para poder manejar estos componentes, se ha utilizado la biblioteca gratuita orientada a *Vue.js*, llamada *PrimeVue* [32]. Esta plantilla ha sido empleada debido a su amplia gama de componentes, que permiten dotar a la aplicación de controles sencillos y visualmente agradables de manera fácil y rápida, sin tener que diseñarlos de manera individual.

El uso de *PrimeVue* junto con *PrimeFlex* [33], una biblioteca *responsive* de CSS, ha permitido modelar la interfaz de manera rápida y sencilla mediante clases predefinidas.

4.3. Python y Fast API

Para el desarrollo de la parte *back-end* de la aplicación, en particular para el desarrollo de la API del sistema de recomendación, se ha empleado Python [34]. La filosofía de este lenguaje de programación hace hincapié en una sintaxis limpia y un código legible. También es compatible con varias plataformas y sistemas operativos, lo que facilita la creación de APIs que puedan funcionar en diferentes entornos.

FastAPI [35] es un *framework* de desarrollo web para Python empleado para crear aplicaciones web y APIs *RESTful* de manera rápida. Es muy sencilla de usar, con una sintaxis intuitiva que agiliza el desarrollo de principiantes. FastAPI cuenta con diversas características que la hacen una buena opción de uso, como es la generación automática de documentación (más información en el Anexo IV de esta documentación) o la validación de datos y el manejo de errores, entre otras.

Todas estas características y su relación con FastAPI, así como la sencillez del lenguaje que, como bien indican ellos mismos, tiene "*una curva de aprendizaje buena*", hacen de *Python* una buena elección para el desarrollo de este servicio externo.

4.4. Poetry

Poetry [36] es una herramienta y un administrador de paquetes para Python que facilita la gestión de las dependencias de los proyectos, la distribución de paquetes y la administración del entorno virtual.

La gestión de las dependencias la lleva a cabo mediante un archivo llamado *pyproject.toml* que se crea cuando se configura un entorno, incluyendo también las bibliotecas externas a usar por el proyecto. Este método hace que se puedan definir las versiones de las dependencias y las restricciones de compatibilidad de manera sencilla y clara. Además, tiene otras ventajas como es la posibilidad de instalar paquetes de Python con ella, gestionando automáticamente las dependencias de desarrollo y de producción.

Estas y otras características han hecho que *Poetry* haya sido una buena opción de uso y una gran ayuda para la gestión de estos elementos en un entorno donde no se contaba con casi nula experiencia.

4.5. Docker

Docker [37] es una plataforma de código abierto que facilita la creación, implementación y administración de aplicaciones y entornos de desarrollo en contenedores. Los contenedores son unidades estándares de software que empaquetan el código y todas sus dependencias, de forma que la aplicación pueda ejecutarse de manera rápida y fiable de un entorno informático a otro.

Toda la información sobre dependencias, código a ejecutar, etc., se indica en el conocido como *Dockerfile* [38], un fichero de texto que contiene todos los comandos que un usuario tendría que introducir para poder crear una imagen de *Docker* (un paquete de software ligero, independiente y ejecutable que incluye lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas y bibliotecas del sistema y configuraciones).

Este método, que ha sido empleado para poder desplegar la API gracias a la plataforma *Fly.io* [39], hace que cada contenedor actúe de forma aislada (por lo que no es afectado ni afecta a los otros contenedores de la misma máquina). Además, ofrece otras ventajas, como es la eliminación de la necesidad de instalar y administrar dependencias en el sistema anfitrión, o la rápida implementación y despliegue de aplicaciones y servicios, entre otras. Todo ello hace de esta herramienta la adecuada para el objetivo propuesto: el despliegue de la API del sistema de recomendación.

4.6. Arquitectura *serverless*

La aplicación desarrollada sigue una arquitectura *serverless*, que es aquella en la que se pueden crear y ejecutar aplicaciones y servicios sin administrar la infraestructura del lado del servidor, como explica AWS [40]. Esto no significa que la aplicación no se ejecute en un servidor, sino que existe una entidad que es la que gestiona la administración de dichos servidores, siendo esta entidad *Firebase* en este caso.

Usando una arquitectura *serverless*, el desarrollador de la aplicación puede dedicar más tiempo y recursos al servicio principal a ofrecer, quedando libre de la carga de gestión y administración mencionada anteriormente. Según afirma la plataforma AWS, esta reducción de gastos hace que los productos y servicios desarrollados sean de mejor calidad, más confiables, etc., al poner el profesional toda su dedicación en su mantenimiento y no en otras cuestiones.

El principal motivo de la elección de esta arquitectura ha sido la ventaja que supone no tener que realizar el diseño de toda la parte *back-end* de la aplicación, lo cual no solo ha supuesto un ahorro de trabajo y esfuerzo de desarrollo, sino también de tiempo de desarrollo. Esto hace que haya sido posible aplicar este esfuerzo en tareas relacionadas con el *front-end* y con la funcionalidad principal del sistema.

4.7. Herramientas de diseño de la interfaz gráfica

Los *wireframes* expuestos en el Anexo III han sido realizados a mano en una Tablet, en la aplicación *Goodnotes* [41].

El logotipo expuesto también en dicho anexo se creó gracias a la plataforma *Brandmark* [42]. Una vez obtenido el logo, se extrajo la paleta de colores que se observa en toda la aplicación (también recogida en el documento mencionado) gracias una plataforma generadora de paletas de colores a partir de una imagen [43].

Además, para la obtención de algunas de las imágenes que se ven en la plataforma, como es la del encabezado de la *landing page*, se utilizó el generador de imágenes de *Microsoft Bing* [44]. Esta web permite que, mediante el registro en ella, se escriba una solicitud de imagen en lenguaje natural, como un *prompt*, con aquella imagen que se quiera crear y que será interpretado por la IA para mostrar unos resultados.

Para la detección de errores de contraste entre los elementos que forman las vistas de la aplicación, se ha empleado la extensión *WAVE Evaluation Tool* [45], que muestra un informe detallado de los distintos elementos que contiene la web analizada.

4.8. Herramientas CASE

4.8.1. Microsoft Project

Microsoft Project [46] es el software de administración de proyectos empleado para la planificación temporal expuesta en el Anexo I. Ha sido elegida gracias a la amplia funcionalidad que presenta, como la asignación de recursos a tareas, la administración del presupuesto y los detallados informes tanto de actividades como de recursos, entre otros servicios. Además, como ya había sido empleado anteriormente durante el Grado, se ha reducido el tiempo de aprendizaje y toma de contacto con la aplicación.

4.8.2. EZEstimate

Para la estimación del esfuerzo en el Anexo I de esta documentación, fue necesario el uso del software EZEstimate, que permite calcular el esfuerzo que supone el desarrollo de una aplicación, teniendo en cuenta no solo los actores y los casos de uso del sistema, sino también algunos factores de complejidad técnica que tienen peso en el sistema. Es decir, siguiendo el modelo de coste basado en los puntos de caso de uso (UCP – *Use Case Points*) [47]

Su elección se hizo basándose en la funcionalidad ofrecida, así como también en el anterior uso de este software.

4.8.3. Visual Paradigm

Para algunos de los diagramas (como los de casos de uso del Anexo II o los de diseño del Anexo III), se empleó Visual Paradigm [48]. Esta herramienta ofrece una amplia funcionalidad al permitir realizar diversos tipos de diagramas, desde secuencia hasta arquitectura, entre otros. Debido a esto y a su anterior utilización, fue una buena elección para poder realizar la tarea expuesta de manera sencilla.

4.8.4. Draw.io

Como alternativa a la herramienta anterior cuando esta presentaba algún tipo de problema, se empleó la página web *draw.io* [49]. Esta herramienta permite también el diseño de diferentes diagramas, aunque su funcionalidad es más limitada ya que no ofrece la realización de diagramas tan diferentes y su acabado no es tan profesional como el que ofrece Visual Paradigm.

4.8.5. PlantUML

En la misma línea que los dos anteriores, *PlantUML* [50] ha sido otra alternativa a la hora de crear algunos diagramas. En este caso, se basa en una herramienta que, proporcionándole un texto en formato JSON, por ejemplo, es capaz de crear un diagrama que represente lo explicado por el usuario prácticamente en lenguaje natural. En este contexto, se utilizó para la creación de los diagramas de las estructuras internas de los documentos almacenados en la base de datos, los cuales se pueden apreciar en el Anexo III. A pesar de que no es tan gráfica como las anteriores dos herramientas, es muy sencilla de usar ya que no requiere casi tiempo de aprendizaje, razón por la cual fue una buena opción a la hora de realizar dichos diagramas.

5. Aspectos relevantes del desarrollo

En este apartado se presentan los aspectos a destacar del desarrollo. En los distintos apartados, se irán exponiendo aquellos conceptos clave de los distintos anexos que acompañan este documento.

5.1. Ciclo de vida

Para el desarrollo de este proyecto se ha seguido el modelo del Proceso Unificado [51], un marco de trabajo basado en la Ingeniería del Software que permite guiar y estructurar el ciclo de vida del desarrollo de proyectos software. Además, se caracteriza por la utilización de casos de uso, estar centrado en la arquitectura y ser incremental e iterativo. Estas características, destacando la última, hace que esta sea una metodología más flexible, ha permitido que la adaptación a modificaciones en los requisitos u objetivos, o la aparición de nuevos desafíos, haya sido mucho más sencilla gracias los ciclos sucesivos que se han realizado.

La planificación propuesta se compone de cuatro fases de desarrollo: Inicio, Elaboración, Construcción y Transición. Todas ellas se forman, a su vez, por varias iteraciones, dentro de las cuales se encuentran varias disciplinas: Modelado de negocio, Requisitos, Análisis, Diseño, Implementación y Pruebas.

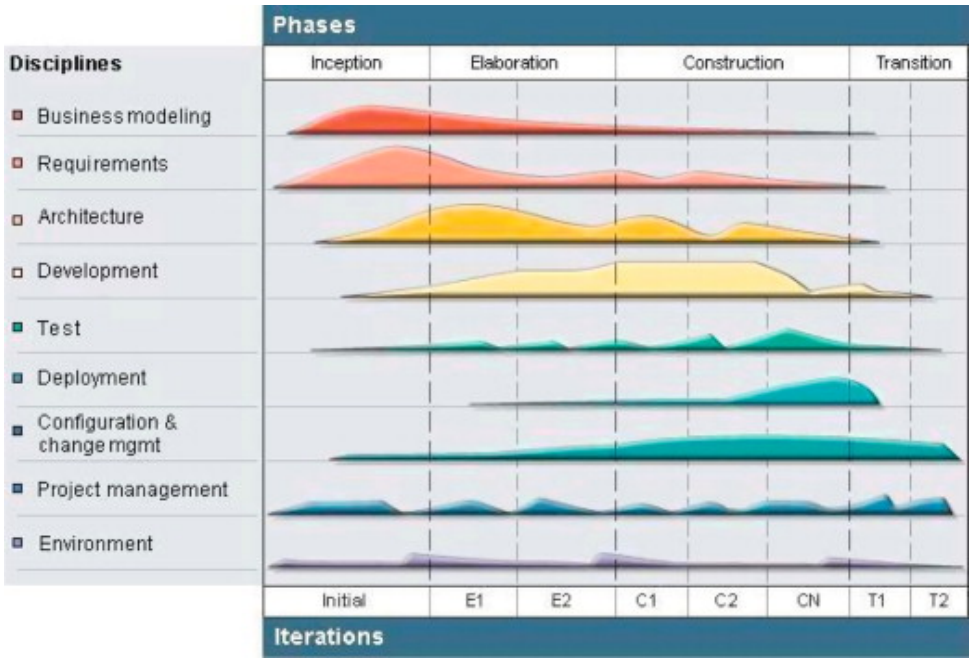


Ilustración 1 Diagrama de las fases de la planificación temporal del PU [52]

Para empezar, se estableció un calendario de trabajo (con las horas que se esperaba dedicar cada día a este desarrollo, que obtuvo un valor de, aproximadamente 40 horas semanales) y se realizó una estimación del esfuerzo para calcular el coste de la construcción del sistema. Para ello, se siguió el modelo basado en los Puntos de Caso de Uso (UCP) [47] el cual, tras asignar una serie de pesos a actores, casos de uso, etc., se

realizan ciertos cálculos que, combinados, permiten obtener el esfuerzo total (ver el Anexo I para una información más detallada sobre las fórmulas a seguir)

En líneas generales, el proceso fue como sigue: una vez determinados los actores y los casos de uso (fase de Elaboración que se explica más adelante), se les asignó un peso a cada uno de ellos, para determinar su esfuerzo de desarrollo. Posteriormente, se establecieron los valores de los factores de complejidad técnica y del entorno atendiendo a las necesidades y requerimientos del sistema a desarrollar. Una vez que se habían propuesto todos estos datos, se introdujeron en la herramienta *EZEstimate*, a la que se le indicó que el factor de conversión F tenía un valor de 6 horas de persona por caso de uso. De esta forma, el programa calculó las horas de persona necesarias para desarrollar la totalidad del sistema, siendo el resultado de 739 horas. Si ese dato se relaciona con las horas semanales que se esperaba dedicar al proyecto, salía un total de 5 meses de trabajo.

El siguiente paso fue realizar una planificación temporal para poder llevar un seguimiento del proceso de desarrollo, para lo que se empleó la herramienta de *Microsoft Project*. En ella, se fueron estableciendo las distintas iteraciones y fases de desarrollo del Proceso Unificado que se han mencionado anteriormente. A continuación, se detallan brevemente las actividades que tuvieron lugar en cada fase de desarrollo:

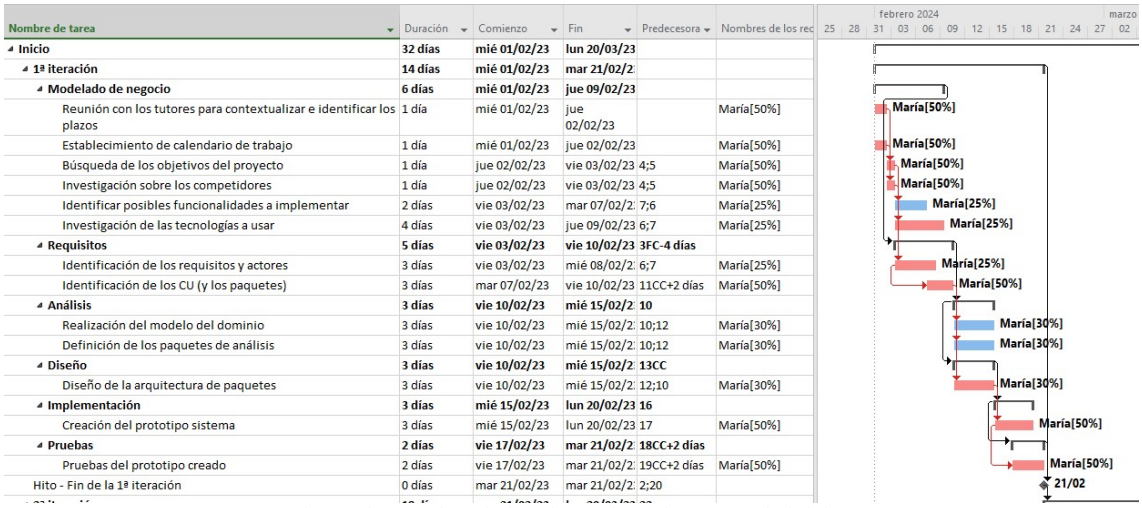
- **Inicio**: consta de dos iteraciones, en las cuales se identificaron los objetivos del proyecto y se determina su alcance. Estas iteraciones, por tanto, tienen una gran carga en la disciplina de Modelado de negocio, ya que es donde se dieron los primeros pasos en el desarrollo. También se incluyeron tareas de aprendizaje de tecnologías a emplear, ya que la mayoría eran desconocidas hasta la fecha, y, una vez definida la visión general del proyecto, se pasó a la siguiente fase.

- **Elaboración**: cuenta con dos iteraciones, donde la mayor carga recae en los Requisitos y el Análisis. Es decir, en esta fase se realizó un análisis detallado de los requisitos del sistema, estableciendo los casos de uso, actores, modelo del dominio, etc. También se determinó la arquitectura y se llevaron a cabo algunas actividades de implementación sencillas que dotaron a la aplicación de una funcionalidad básica, como son las funciones de registro o inicio de sesión, pero que establecieron las bases para la implementación posterior.

- **Construcción**: en esta fase destacaron las tareas de implementación del sistema, llevadas a cabo de manera incremental, para ir añadiendo funcionalidad a la plataforma y refinando aquellos aspectos que suponían errores. En ella se desarrollaron los componentes y vistas del sistema según la arquitectura que ya se había establecido en la fase anterior. Para comprobar que lo implementado era llevado a cabo correctamente, se fueron realizando pruebas de funcionamiento que permitieron detectar fallos y puntos de mejora. Al final de esta fase, se consiguió una versión funcional y probada del sistema.

- **Transición:** es la fase final, y en ella se preparó el sistema para poder ser desplegado. Se llevaron a cabo unas últimas tareas de corrección y refinamiento de errores, así como pruebas de la totalidad del sistema para asegurarse de su correcto funcionamiento. Estas actividades condujeron a la obtención de un sistema completamente implementado y funcional.

A continuación, se observa, en la ilustración 2, una iteración de la planificación realizada con la herramienta *Microsoft Project*.



Nombre de tarea	Duración	Comienzo	Fin	Predecesora	Nombres de los rec
Inicio	**32 días**	**mié 01/02/23**	**lun 20/03/23**		
1ª iteración	**14 días**	**mié 01/02/23**	**mar 21/02/2**		
Modelado de negocio	**6 días**	**mié 01/02/23**	**jue 09/02/23**		
Reunión con los tutores para contextualizar e identificar los plazos	1 día	mié 01/02/23	jue 02/02/23		María[50%]
Establecimiento de calendario de trabajo	1 día	mié 01/02/23	jue 02/02/23		María[50%]
Búsqueda de los objetivos del proyecto	1 día	jue 02/02/23	vie 03/02/23	4;5	María[50%]
Investigación sobre los competidores	1 día	jue 02/02/23	vie 03/02/23	4;5	María[50%]
Identificar posibles funcionalidades a implementar	2 días	vie 03/02/23	mar 07/02/2	7;6	María[25%]
Investigación de las tecnologías a usar	4 días	vie 03/02/23	jue 09/02/23	6;7	María[25%]
Requisitos	**5 días**	**vie 03/02/23**	**vie 10/02/23**	**3FC-4 días**	
Identificación de los requisitos y actores	3 días	vie 03/02/23	mié 08/02/2	6;7	María[25%]
Identificación de los CU (y los paquetes)	3 días	mar 07/02/23	vie 10/02/23	11CC+2 días	María[50%]
Análisis	**3 días**	**vie 10/02/23**	**mié 15/02/2**	**10**	
Realización del modelo del dominio	3 días	vie 10/02/23	mié 15/02/2	10;12	María[30%]
Definición de los paquetes de análisis	3 días	vie 10/02/23	mié 15/02/2	10;12	María[30%]
Diseño	**3 días**	**vie 10/02/23**	**mié 15/02/2**	**13CC**	
Diseño de la arquitectura de paquetes	3 días	vie 10/02/23	mié 15/02/2	12;10	María[30%]
Implementación	**3 días**	**mié 15/02/23**	**lun 20/02/23**	**16**	
Creación del prototipo sistema	3 días	mié 15/02/23	lun 20/02/23	17	María[50%]
Pruebas	**2 días**	**vie 17/02/23**	**mar 21/02/2**	**18CC+2 días**	
Pruebas del prototipo creado	2 días	vie 17/02/23	mar 21/02/2	19CC+2 días	María[50%]
Hito - Fin de la 1ª iteración	0 días	mar 21/02/23	mar 21/02/2	2;20	

Ilustración 2 Vista de la primera iteración (Fase de inicio)

En el diagrama de Gantt anterior, se aprecia de manera visual el tiempo de realización de cada tarea. A simple vista, puede parecer que se ha seguido una metodología en cascada, pero sí que se encuentran paralelismos entre las tareas cuando se observa más detalladamente. Este hecho se da porque el desarrollo ha sido llevado a cabo por una sola persona, haciendo más difícil la paralelización de múltiples actividades.

Además, algunas de estas tareas (la mayoría, marcadas en rojo) son críticas, lo que supone que, en caso de que sean retrasadas, conllevarán un retraso del resto de tareas del proceso. Por ello, a pesar de que la cantidad de horas estimadas que planteaba *Microsoft Project* eran 771 (32 horas de diferencia con respecto al cálculo de la herramienta *EZEstimate*), era muy probable que aumentase dicha cantidad, ya que es prácticamente imposible que alguna de todas esas actividades no fuera retrasada.

Para obtener una versión más detallada del trabajo realizado en este apartado, se recomienda la consulta del Anexo I que acompaña a este documento.

5.2. Especificación de requisitos

Se describe brevemente a continuación el comportamiento del software mediante la especificación de requisitos, aunque la explicación detallada se puede consultar en el Anexo II que acompaña a este documento. Para la especificación de dichos requisitos y de los objetivos, se ha seguido la Metodología de Elicitación de Requisitos de Durán y Bernárdez [53].

El sistema implementado se ha estructurado en tres paquetes, los cuales se explican en los párrafos siguientes.

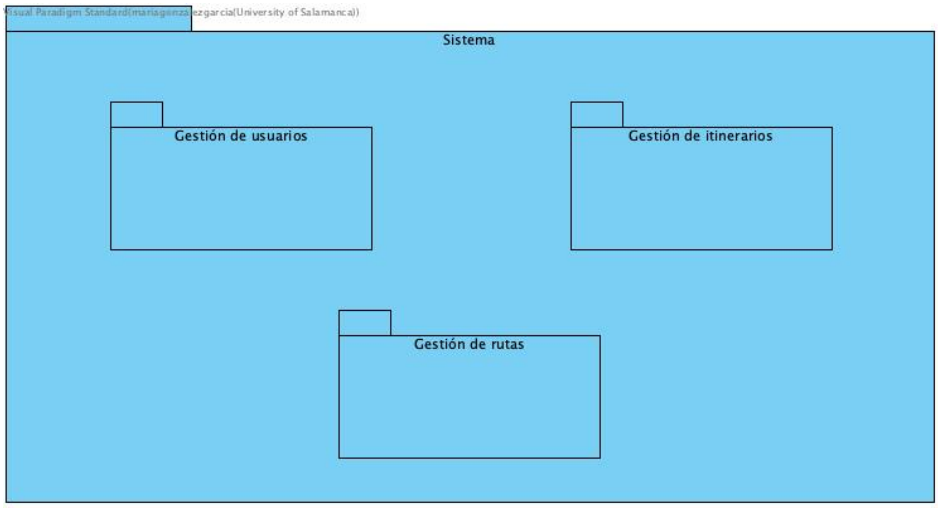


Ilustración 3 Diagrama de paquetes

- **Gestión de usuarios**: se encarga de las tareas relacionadas con los usuarios del sistema (dar de alta, dar de baja, modificación de datos, etc.). Su vista en detalle se puede observar en la ilustración 4.

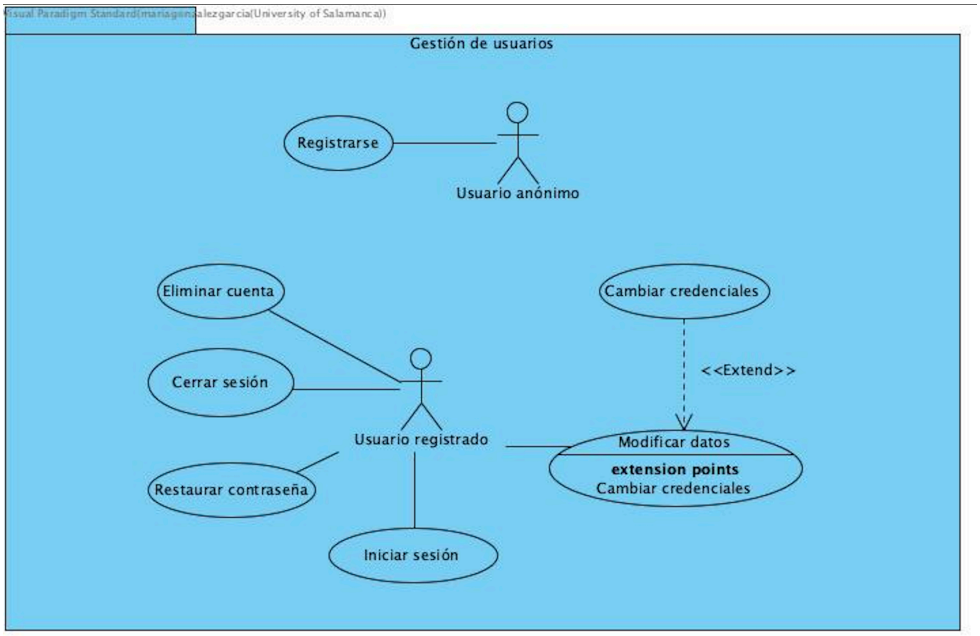


Ilustración 4 Paquete de Gestión de usuarios

- **Gestión de rutas**: gestiona datos como los puntos a visitar o los lugares en los que pernoctar para poder crear la ruta que se adapte a dichos datos. Su detalle se puede observar en la ilustración 5.

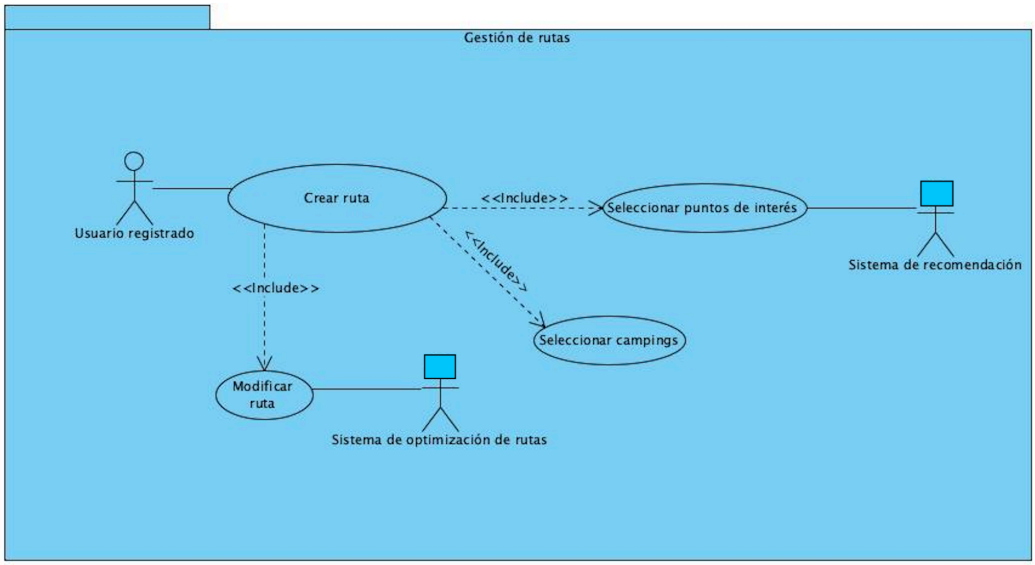


Ilustración 5 Paquete de Gestión de rutas

- **Gestión de itinerarios:** se encarga de gestionar los itinerarios diseñados en su totalidad. Debe contar también con los datos de las rutas y del usuario. Se detalla en la ilustración 6.

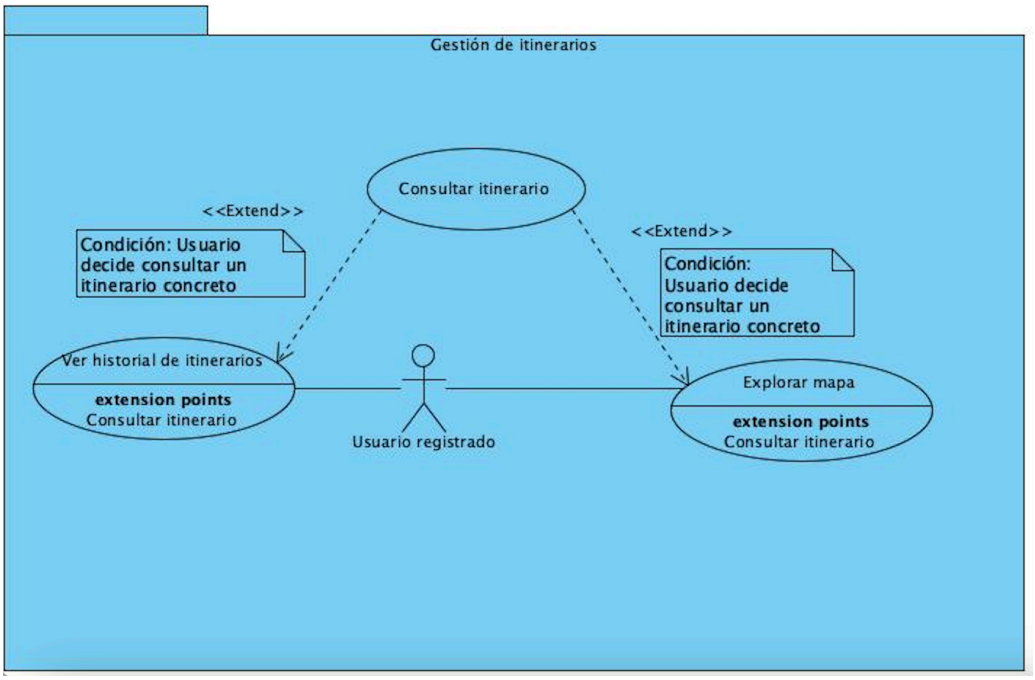


Ilustración 6 Paquete de Gestión de itinerarios

5.3. Especificación de análisis

Se describe a continuación del modelo del dominio del sistema mediante el diagrama de clases observado en la ilustración 7. Este diagrama permite observar las distintas clases entidad del sistema y sus atributos.

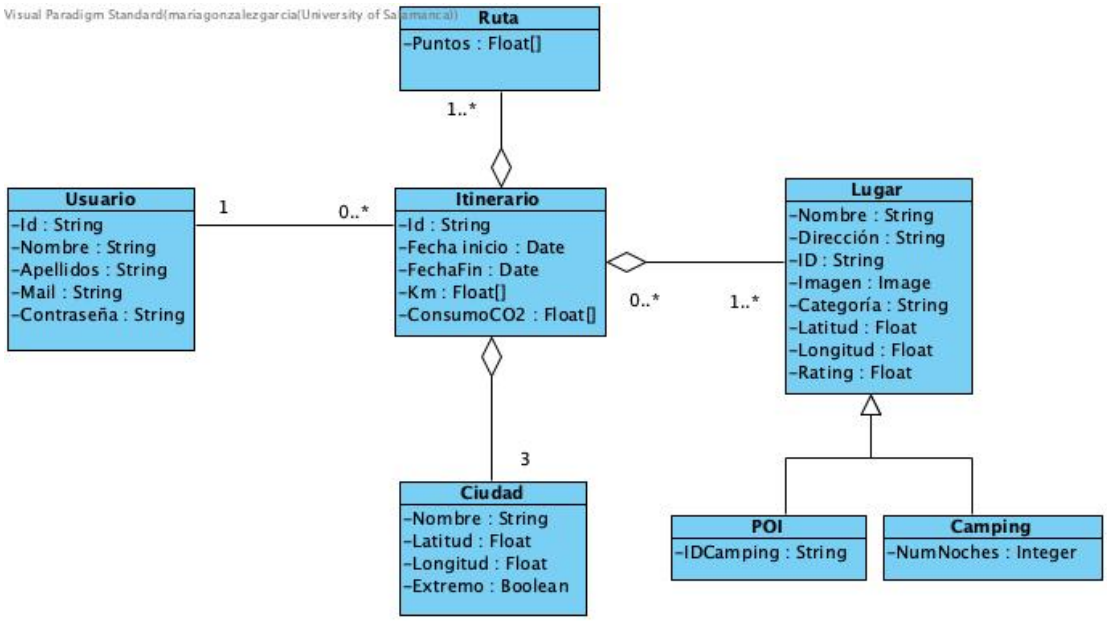


Ilustración 7 Modelo del dominio del análisis

Las clases del diagrama anterior representan los datos que el sistema debe manejar para poder funcionar correctamente. Estas clases entidad se relacionarán con los distintos elementos de control que se encuentran en los paquetes especificados en el apartado anterior. Así, la clase "Usuario" se relacionaría simplemente con el "ControlUsuario", al tratar con información más personal. Lo mismo pasaría con "Itinerario", que solo se comunica con "ControlItinerarios", esta vez debido a que toda la lógica de los datos de las rutas se gestiona por separado con el "ControlRuta", quien se comunicaría con todas las demás entidades que se representan el resto de clases de la ilustración 7. La asignación de estas clases de control con los paquetes en los que se divide el sistema (y las relaciones entre dichos paquetes) se encuentra en la ilustración 8.

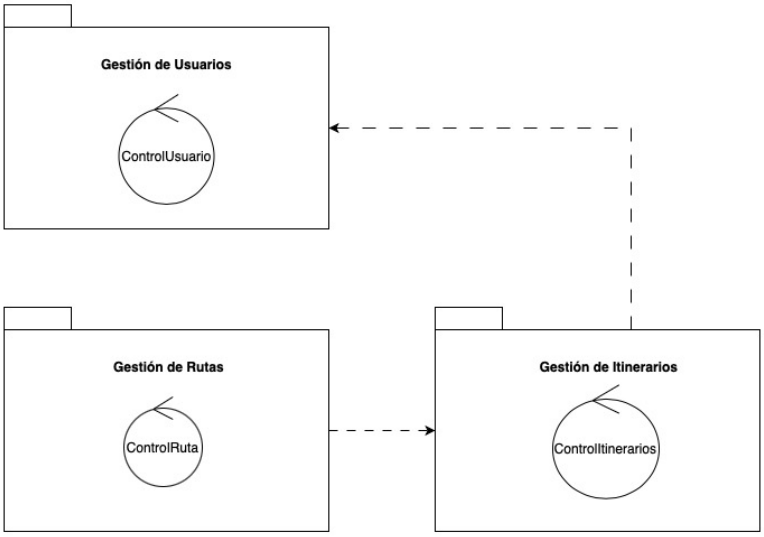


Ilustración 8 Arquitectura (análisis). Relaciones entre paquetes.

Para poder entender de mejor manera la comunicación entre todos estos elementos, se presenta como ejemplo, en la ilustración 9, el diagrama de secuencia del caso de uso de seleccionar puntos de interés.

En este ejemplo, el actor usuario registrado desea seleccionar unos POIs para su ruta. Para poder obtener unos datos iniciales, el control de la ruta se comunicará con el sistema de recomendación externo para preguntarle por información en una zona especificada. Una vez que se obtienen datos, se presentan al usuario, quien irá informándose de cada uno y eligiendo aquellos que desee.

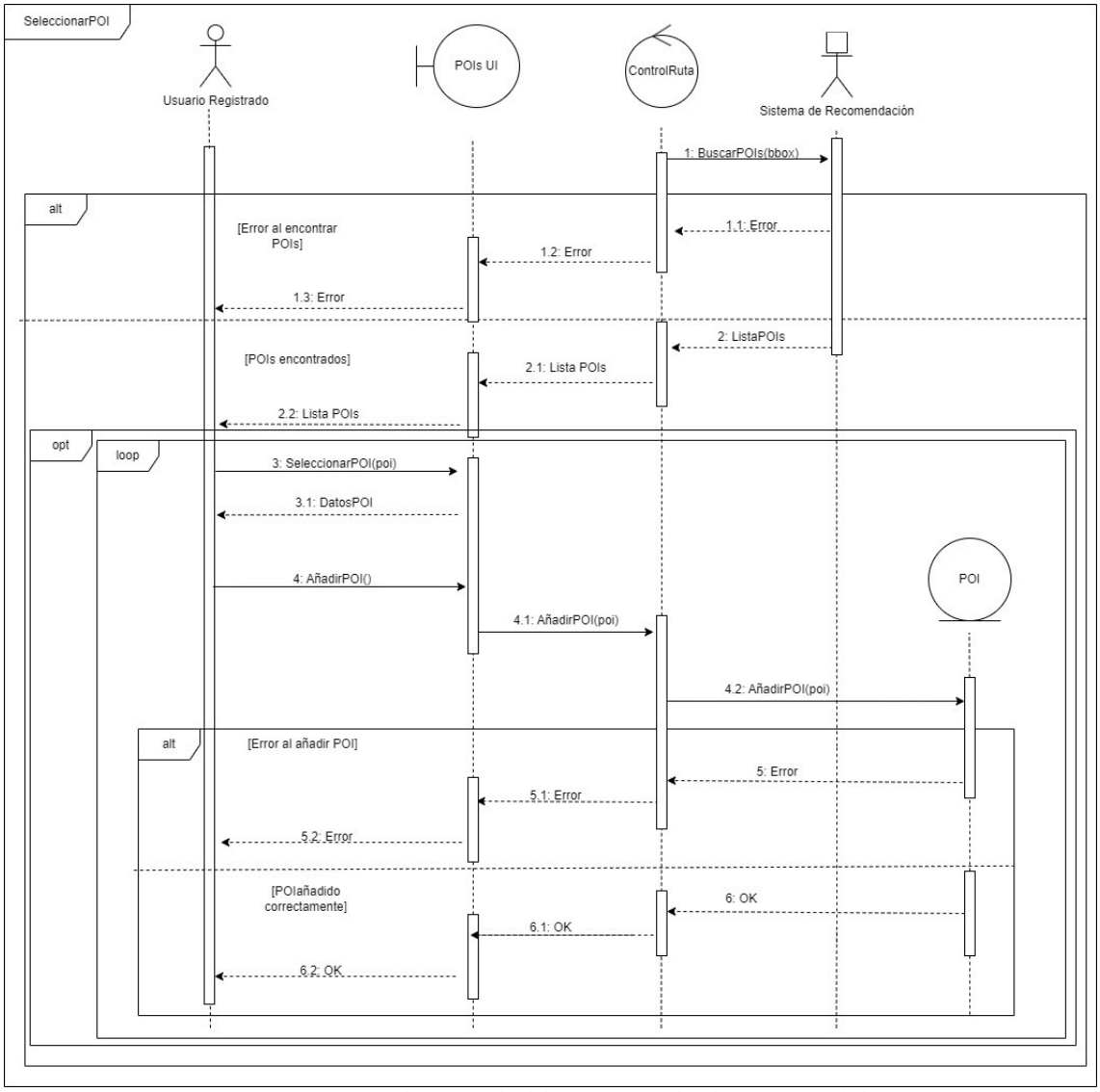


Ilustración 9 Diagrama de secuencia del CU-0009. Seleccionar puntos de interés

5.4. Especificación de diseño

En este apartado se expone el diseño del sistema, cuya explicación detallada se encuentra en el Anexo III. En esta parte se conocerá la arquitectura del sistema, el diseño final del diagrama de clases, los subsistemas del diseño, los componentes del sistema y los diagramas de secuencia con las interfaces diseñadas.

5.4.1. Diagrama de clases

El diagrama observado en la ilustración 10 posee la misma estructura que el modelo del dominio (ilustración 7), pero ha sido ampliado para recoger las operaciones necesarias para almacenar la información de cada atributo de las clases.

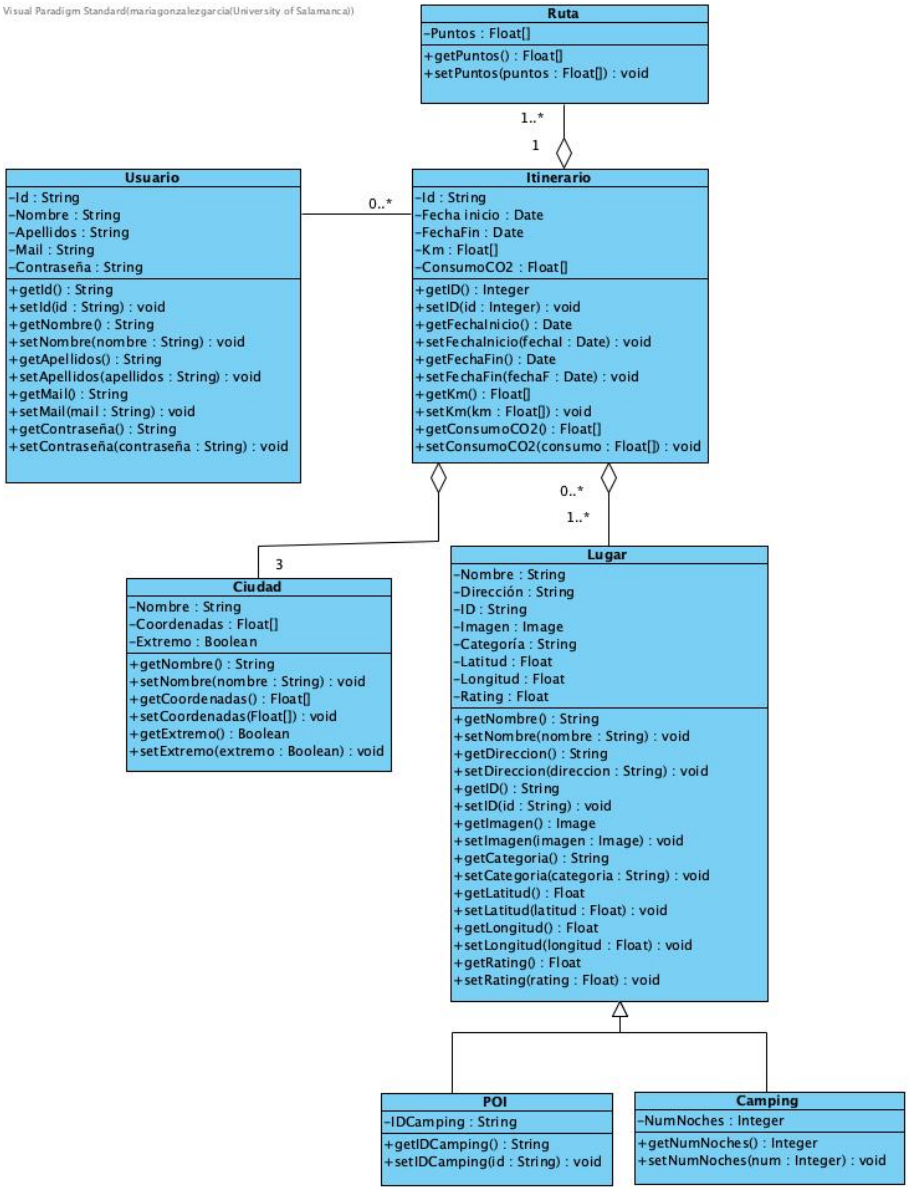


Ilustración 10 Diagrama de clases del diseño

5.4.2. Subsistemas de diseño

Debido a la tecnología elegida para la implementación del sistema y a los patrones utilizados, los subsistemas de diseño, derivados de los paquetes de análisis, se han organizado en diferentes paquetes que permiten desacoplar la parte de presentación de la parte del modelo y la lógica de negocio de la aplicación. A continuación se muestran estos paquetes y las clases que contienen.

El paquete *View* observado en la ilustración 11 almacena las clases que representan los archivos de *Vue.js*, que modelan las vistas de la aplicación, por lo que este paquete no trata con la parte lógica del sistema (según el patrón MVVM seguido, que será explicado posteriormente).

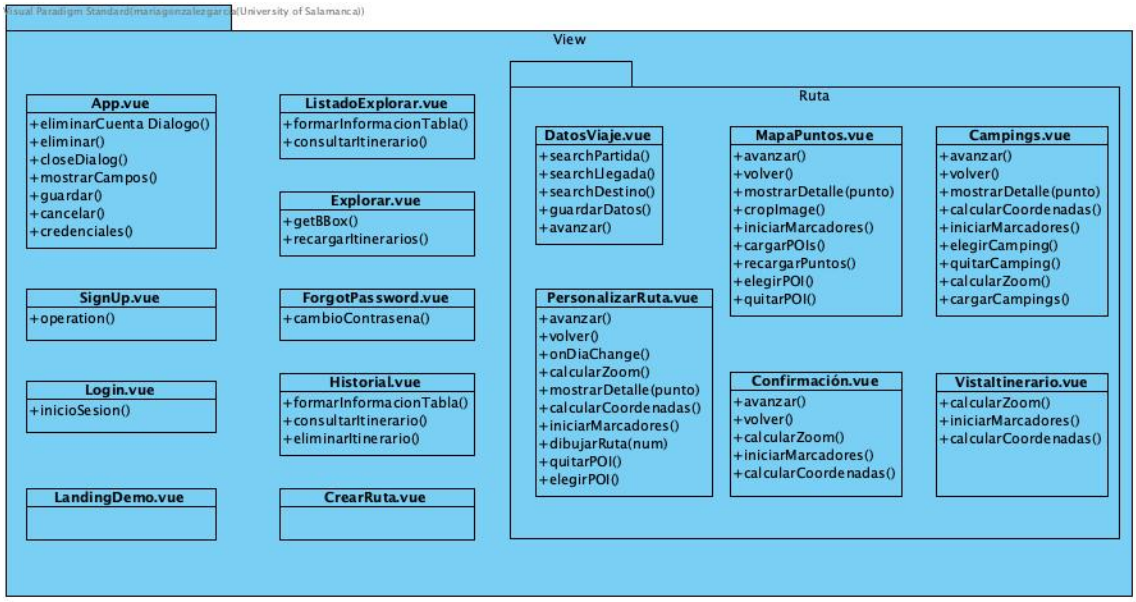


Ilustración 11 Paquete *View*

El paquete *Stores* (ilustración 12) almacena diversas clases. Entre ellas, las que trabajan con la lógica del sistema. Estas clases representan las clases de control expuestas en la ilustración 8. Por otro lado, se observan otras dos clases que representan estructuras de datos, que hacen referencia a algunas de las clases entidad que forman el sistema (las clases que se observan en el modelo del dominio, ilustración 7).

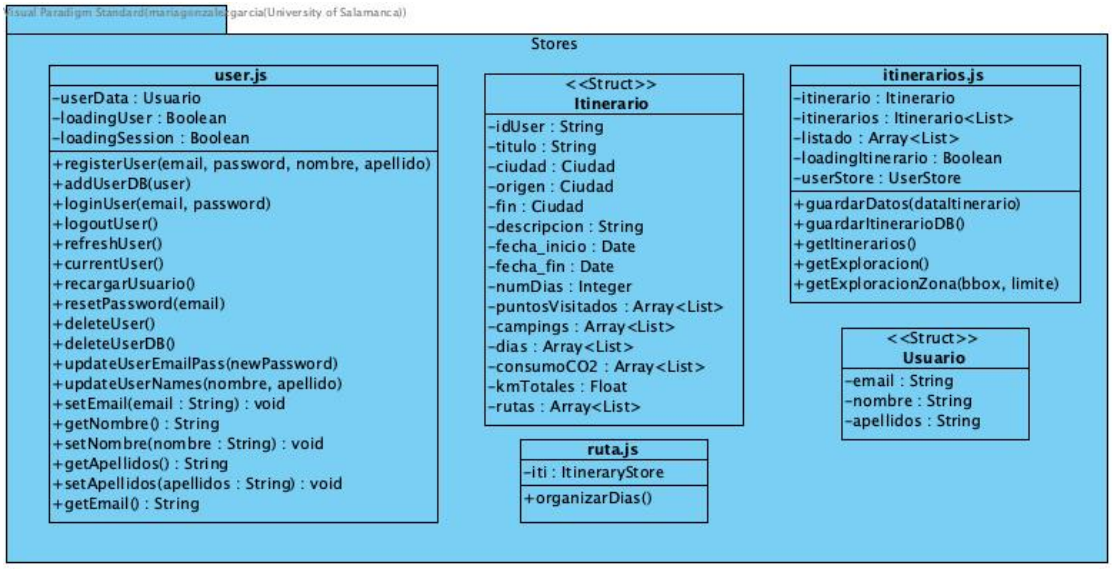


Ilustración 12 Paquete *Stores*

El paquete *Services*, que se encuentra representado en la ilustración 13, incluye tres clases. Estas contienen métodos que acceden a servicios externos (que no han sido implementados en este proceso de desarrollo) y procesan los datos que se les devuelve para poder presentarlos al usuario.

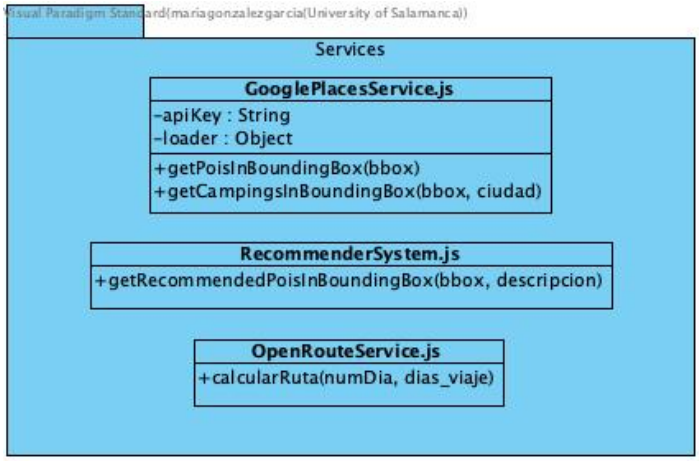


Ilustración 13 Paquete *Services*

Finalmente, el último subsistema (ilustración 14) hace referencia a la *API de recomendación*, que permite obtener POIs en un lugar de búsqueda. A pesar de ser un subsistema, no acoge a ningún elemento de control o de entidad ya que se trata de una API.

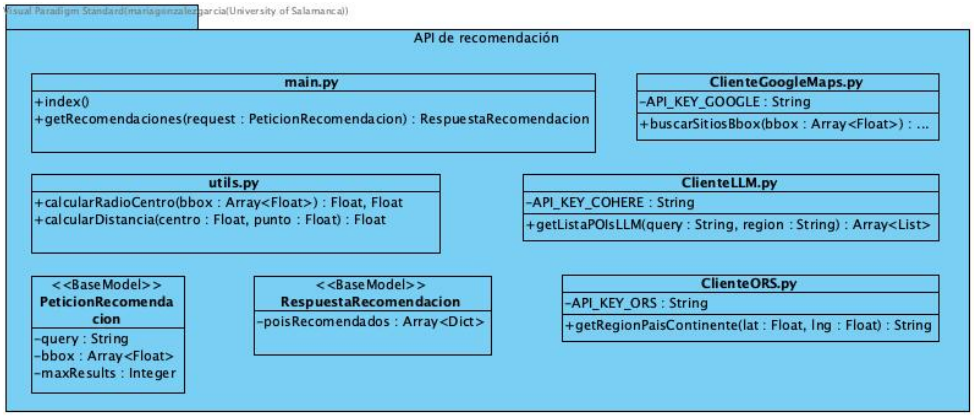


Ilustración 14 Subsistema: API de recomendación

La comunicación entre estos subsistemas hace posible el correcto funcionamiento de la aplicación desarrollada.

5.4.3. Vista de arquitectura

A continuación, en la ilustración 15, se presenta la vista de arquitectura del sistema implementado.

Se puede diferenciar la parte *front-end*, paquete *MyItineraryWebPWA*, de la parte *back-end*, representada por la API de recomendación. La parte del *front-end* representa el cliente desarrollado, ya que, como se comentó en el apartado 4.6. (Arquitectura *serverless*), este tipo de arquitecturas permiten centrarse en la parte cliente sin implementar nada en la parte del servidor. En esta parte superior de la imagen, por tanto, se encuentra el paquete *View* como representación de la capa de presentación. Este paquete se comunica con *Store* y *Service*, que representan el modelo y la lógica de negocio y, por tanto, capas más inferiores.

Finalmente, en la parte más inferior de la imagen, se observa el paquete *API de recomendación*, que es la única parte del *back-end* implementada.

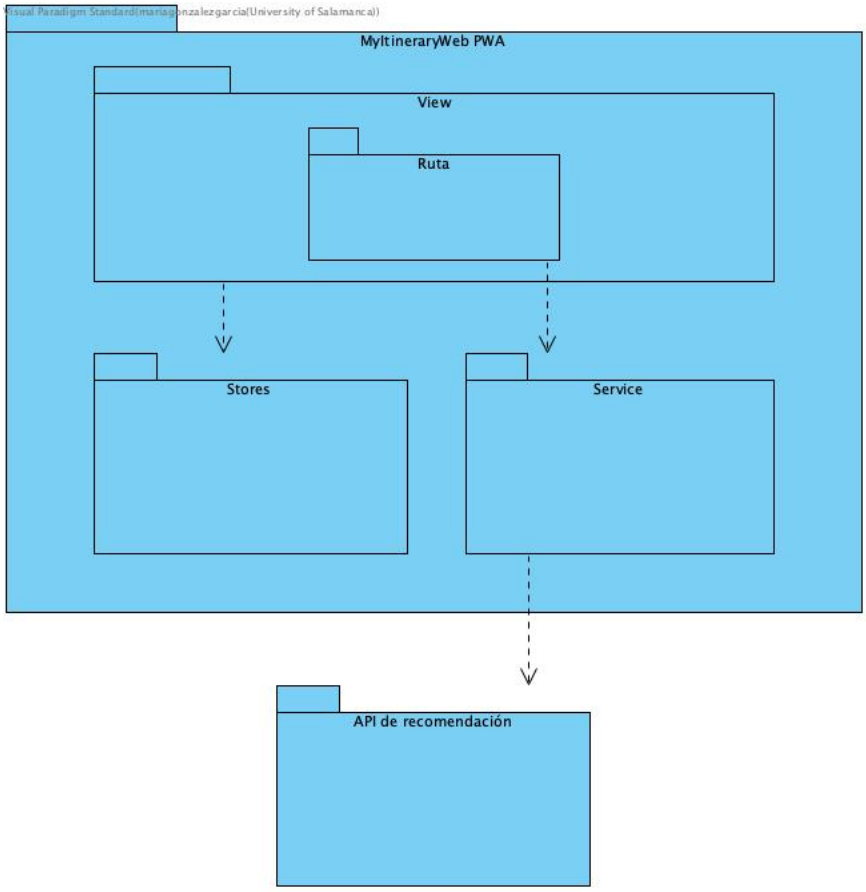


Ilustración 15 Arquitectura del sistema

Para ver más claramente la relación entre estas capas, se expone a continuación el diagrama de secuencia de diseño del caso de uso de selección de puntos de interés (mismo que en la ilustración 9).

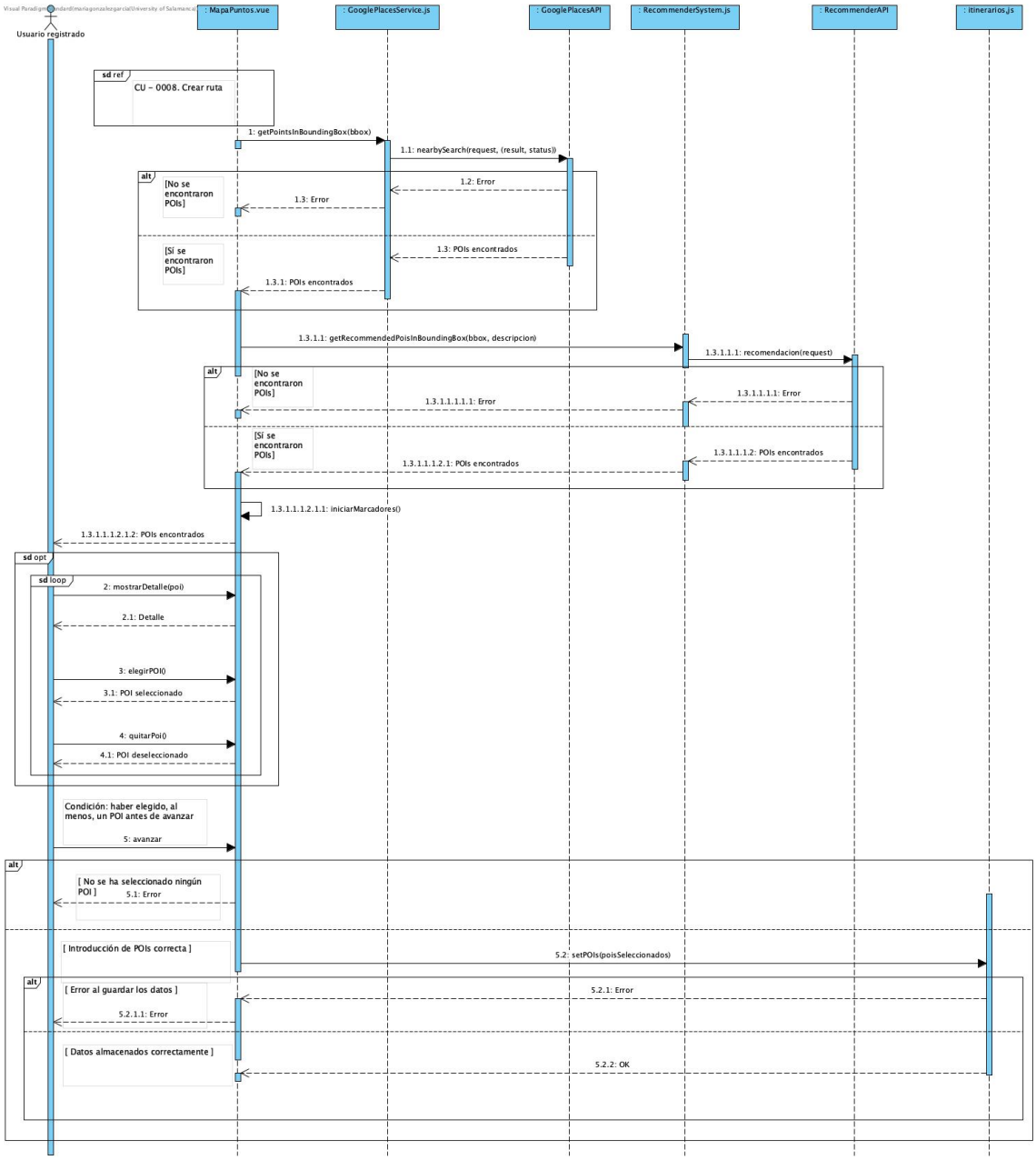


Ilustración 16 Diagrama de realización de CU-Diseño del CU - 0009. Seleccionar puntos de interés

5.4.4. Base de datos

Para la gestión de los datos se empleó la plataforma *Firebase Cloud Firestore* [12]. En esta base de datos se alojan 3 colecciones: una para los usuarios, que almacena datos como su nombre, apellidos, etc.; otra para los itinerarios, que almacena todos aquellos itinerarios diseñados por los usuarios; y, la última, de itinerarios públicos, que contiene, duplicados, aquellos itinerarios que, a la hora de crearse, se establecieron como públicos para que los demás usuarios pudieran tener acceso a él. En la ilustración 16 se aprecia la estructura de un documento de la colección “Itinerarios”.

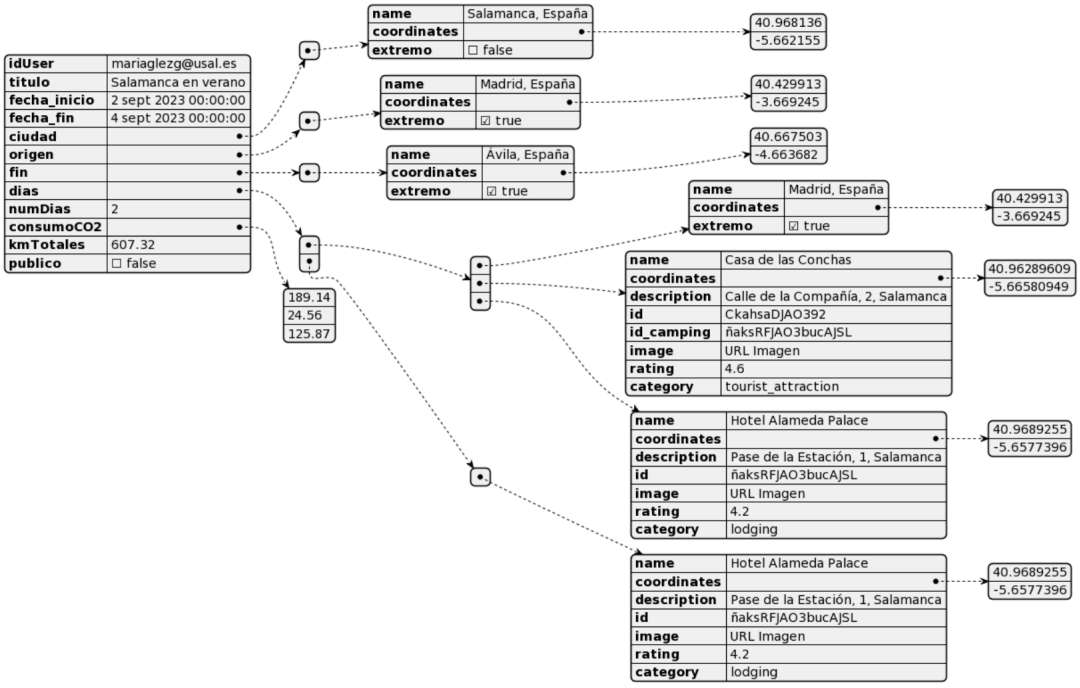


Ilustración 17 Estructura de un documento de la colección Itinerarios de Firebase

5.4.5. Modelo de componentes y modelo de despliegue

Este modelo, visto en la ilustración 18, permite modelar los aspectos físicos de los sistemas orientados a objetos, empleados para el diseño y desarrollo de aplicaciones mediante la combinación de componentes reutilizables [54].

El modelo de componentes del sistema se forma por 6 componentes, cada uno correspondiente a los paquetes comentados en el apartado 5.4.2. Subsistemas de diseño.

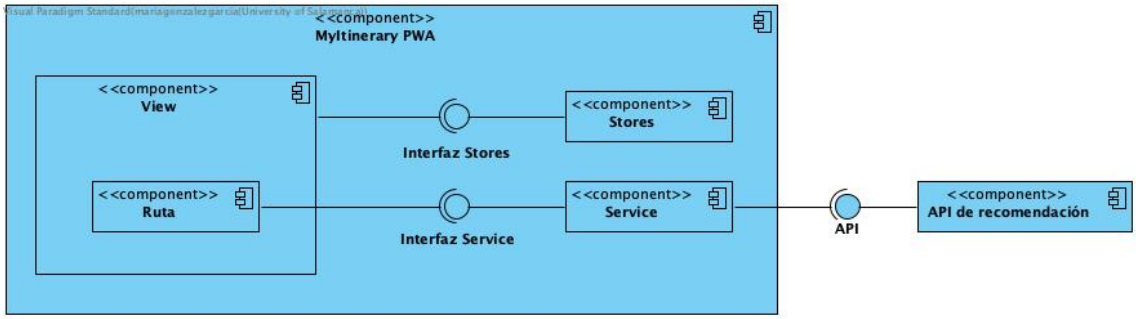


Ilustración 18 Modelo de componentes

Cada componente se despliega en forma de artefacto (ilustración 19) para poder desplegarse en los nodos disponibles del sistema. El modelo de despliegue se muestra en la ilustración 20, donde se pueden apreciar todos los nodos que forman parte del sistema, incluyendo también los correspondientes a servicios externos que utiliza la aplicación desarrollada. La explicación detallada de este apartado corresponde al Anexo III.

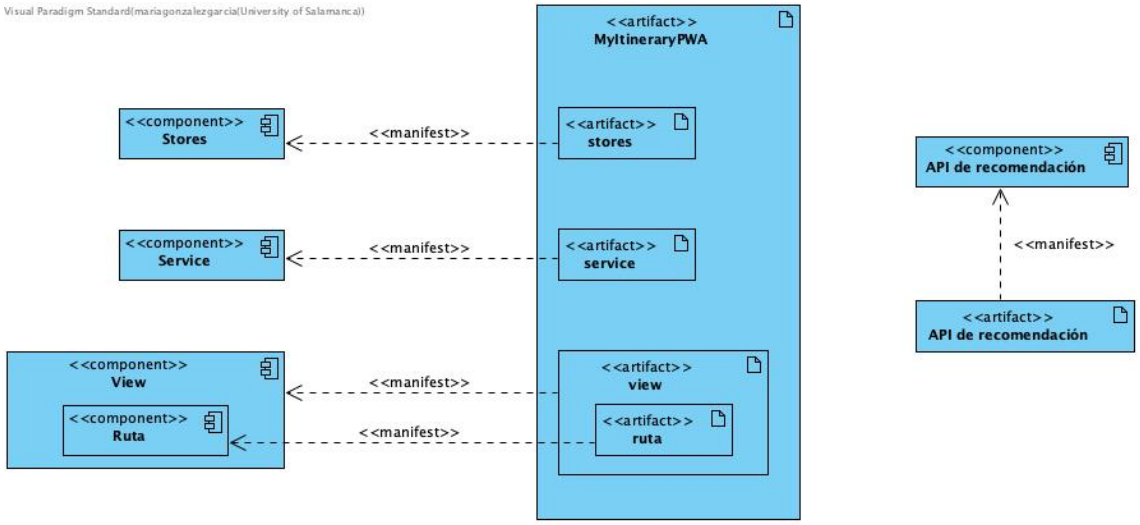


Ilustración 19 Relaciones entre componentes y artefactos

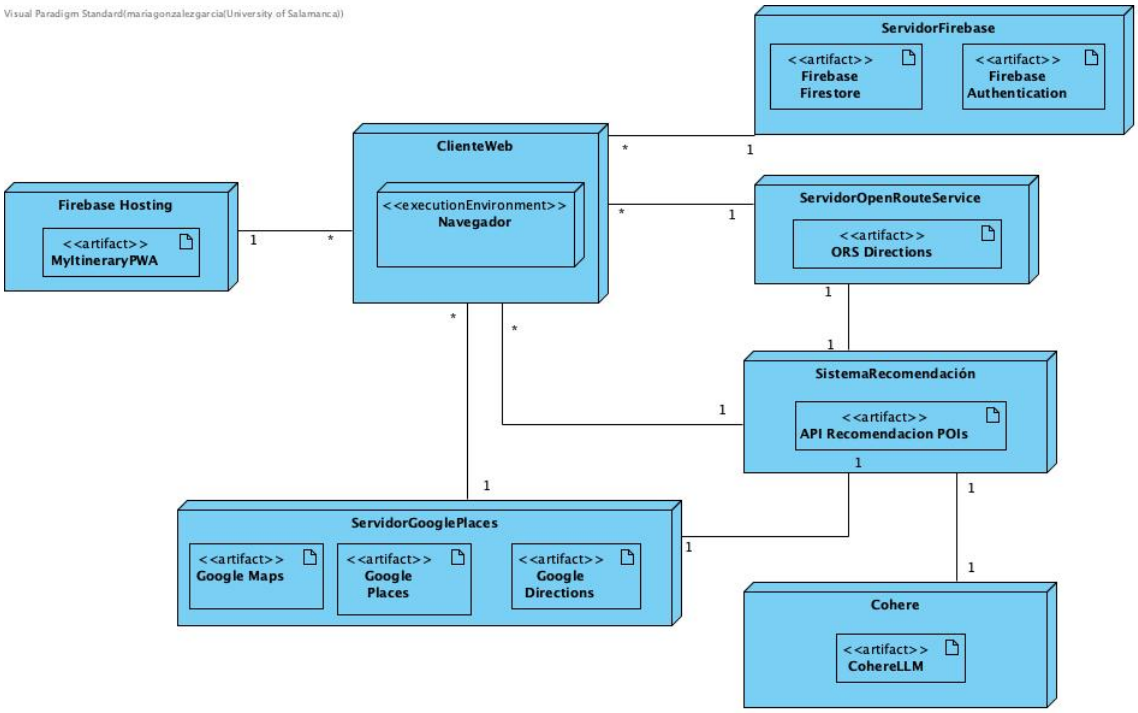


Ilustración 20 Modelo de despliegue

5.4.6. Patrones de diseño software

Una vez presentada brevemente la arquitectura del sistema, se explican a continuación los patrones de diseño empleados.

Para la lógica de negocio, se empleó *Pinia* [55], la versión actualizada que propone *Vue3* frente a *Vuex* [56] de *Vue2*. *Pinia* ofrece muchas ventajas, siendo una de ellas el uso de tantos *stores* (entidades que almacenan la lógica de negocio) como se desee, siendo cada uno independiente y estando formado por su estado, sus acciones, etc. En el caso de *Vuex*, solo era posible contar con un *store*, que podía formarse por varios módulos.

En la parte *front-end* se ha empleado el patrón Modelo–Vista–VistaModelo (*Model–View–ViewModel)* o MVVM, que permite desacoplar la parte lógica de la aplicación de aquella de interfaz, pudiendo así desarrollar proyectos escalables y modularizados. Se puede observar un esquema de su funcionamiento para *Vue.js* en la ilustración 21.

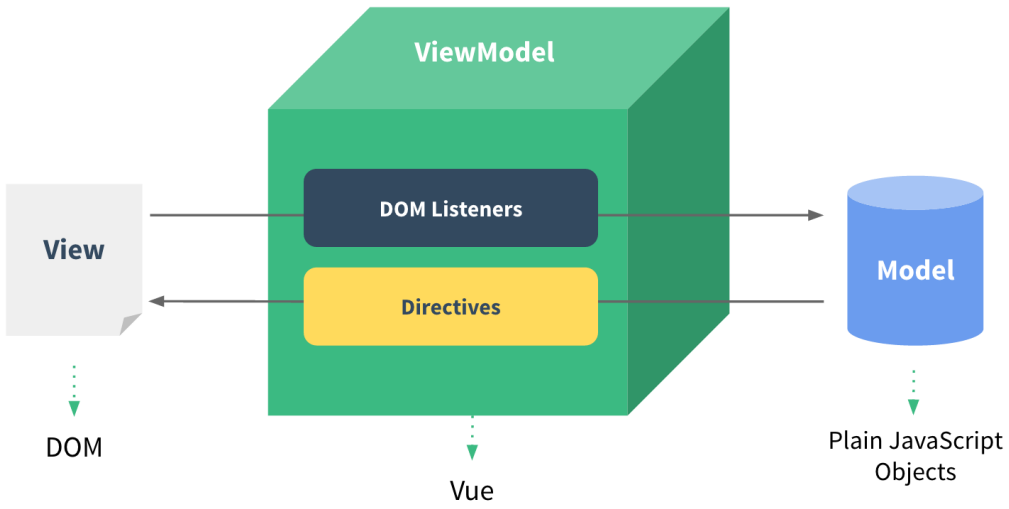


Ilustración 21 Esquema del patrón de diseño MVVM [57]

Las tres partes diferenciadas son las siguientes:

- **Modelo**: representa la capa de datos, aunque no contiene las acciones que los modifican y no se relaciona con la vista de manera directa.
- **Vista**: hace referencia a la información de la aplicación mostrada visualmente, mediante diversos elementos que permiten su representación. En ella se encuentran comportamientos, eventos, etc., que conocen el modelo para poder actuar correctamente.
- **VistaModelo**: intermediario entre la vista y el modelo que contiene toda la lógica de presentación.

Para la parte *back-end* del sistema de recomendación se ha empleado el patrón *facade* (fachada), que permite ocultar la complejidad de un sistema mediante la abstracción de este y otros subsistemas, proporcionando una interfaz de alto nivel que se encarga de la comunicación con todos los demás subsistemas necesarios. En la ilustración 22 se presenta un esquema de dicho patrón.

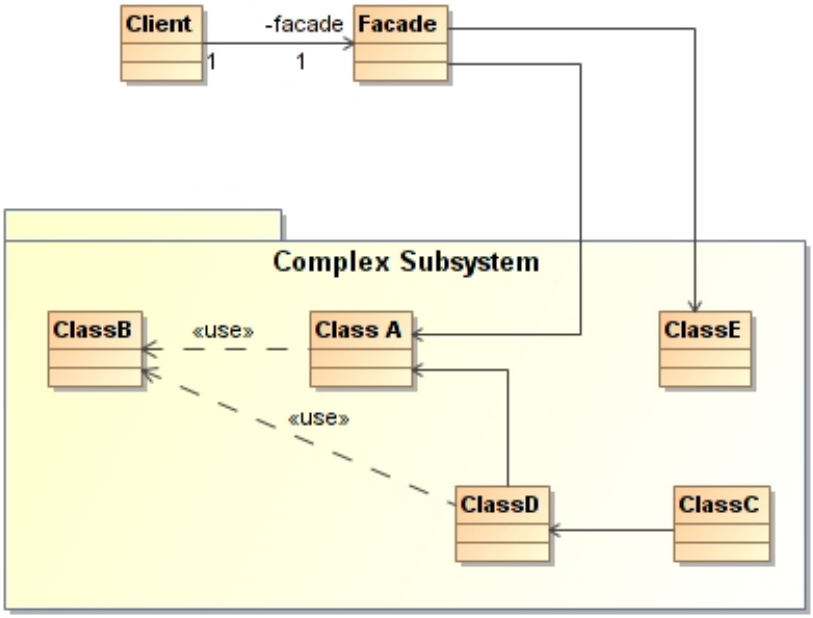


Ilustración 22 Esquema del patrón de diseño *Facade* [58]

5.4.7. Diseño del sistema de recomendación

El sistema de recomendación desarrollado tiene como objetivo clave la obtención de puntos de interés en la ciudad de destino del viaje, ajustándose a las preferencias especificadas por el usuario en texto plano en el momento de la introducción de datos.

En líneas generales, cuando el usuario indica sus preferencias en lenguaje natural, se forma una solicitud con esa descripción, una zona geográfica en la que realizar la búsqueda y un número máximo de resultados que se desea obtener. Esta solicitud es enviada al sistema de recomendación de POIs que, mediante su conexión a otros servicios externos, consigue elaborar una lista de lugares que se adaptan a esas mismas especificaciones.

El desarrollo de este sistema ha sido posible llevarlo a cabo gracias a herramientas como FastAPI [35] y al lenguaje de programación Python [34] (apartado 4.3. Python y FastAPI). Para realizar las operaciones necesarias, se realiza una conexión con servidores externos de OpenRouteService [17], GoogleMaps [59]y Cohere [60]. Para su despliegue, se ha empleado la plataforma Fly.io [39] mediante un Dockerfile [38].

La información detallada sobre todos los servicios externos empleados, su diseño y el despliegue se encuentra en el Anexo III.

5.4.8. Diseño del sistema de optimización de rutas

Los sistemas de optimización de rutas resuelven problemas relaciones con la determinación de la mejor ruta a seguir para viajar entre múltiples POIs. En líneas generales, a este tipo de sistemas se les proporcionan unos datos de entrada con

información sobre el origen, el destino y todos aquellos puntos intermedios que deben visitarse, así como restricciones si las hubiese. Así, el sistema, mediante un modelo matemático o computacional creado, elige el algoritmo de optimización que permita su resolución, tratando de encontrar así la ruta óptima.

En el contexto de esta aplicación, se ha empleado OpenRouteService [17] para la optimización. Gracias a que ofrece servicios como el de *Directions* u *Optimization* de manera gratuita (hasta cubrir un límite de solicitudes), ha sido posible emplear sus funcionalidades para poder obtener aquellas rutas más adecuadas al viaje a tratar.

En cuanto al cálculo del consumo del CO2, se ha empleado la API de Google Maps Directions [18], al cual se le indica el camino a seguir en una solicitud y, proporcionando también el tipo de vehículo o combustible utilizado, permite obtener el cálculo que indique cuánto CO2 ha supuesto la ruta realizada.

La información detallada sobre todo el proceso se encuentra en el Anexo III.

5.4.9. Diseño de la identidad gráfica

Al desarrollar una plataforma, es fundamental que cuente con un diseño que se convierta en la identidad de la marca o aplicación. Para ello, se pensó en el propósito de la aplicación: el diseño de itinerarios de viaje, sobre todo enfocados en autocaravanas debido al alto coste contaminante de estos viajes. Como estos viajes suelen estar relacionados con la naturaleza, la presencia de los llamados colores orgánicos debía estar presente. De la misma forma, este tipo de viajeros suele recurrir a campings como lugares de pernoctación, por lo que las tiendas de campaña son elementos habituales para ellos.

Recurriendo a la herramienta *BrandMark* [42] se consiguió obtener un logo que reuniera todas las características anteriores, como se puede apreciar en la ilustración 23, donde también se observa la paleta de colores extraída de él.

Ilustración 23 Logotipo y paleta de colores

5.4.10. Diseño de la interfaz

Para la creación del diseño de la interfaz se siguieron guías de diseño como la de W3C [61] o PrimeVue [32] (por haber usado la plantulla *Sakai Vue* [30]), entre otras [62], [63]. Todas ellas marcaban unas pautas comunes que conforman unos aspectos claves a la hora de realizar el diseño de una plataforma, siendo algunas de ellas la accesibilidad o la interoperabilidad.

A partir de ellas, se desarrolló un primer prototipo mediante la aplicación *Goodnotes* [41] y se puso a prueba con 6 voluntarios de diversas características, para poder recibir *feedback* de este primer contacto con el diseño y poder empezar la implementación.

Las opiniones y retroalimentación de este grupo de voluntarios fueron muy importante por varios motivos: primero, porque los usuarios de prueba suelen enfocar el problema que se les presenta de una manera que el desarrollador, por conocer el prototipo y su funcionamiento, puede obviar en ocasiones, permitiendo descubrir dificultades en el prototipo a las que el desarrollador era ajeno. También fueron importantes sus observaciones debido a las características de los voluntarios. Tres de ellos son usuarios habituales de este tipo de plataformas, por lo que están acostumbrados a tratar con ellas y sugirieron algunas ideas de implementación en las que no se había pensado. En el caso de los otros tres, no comparte esa característica con los anteriores, pero son personas que no están acostumbradas al uso de las nuevas tecnologías: dos de ellas por ser de edad avanzada y, la última, porque afirmaba tener que consultar con frecuencia los manuales de ayuda de las páginas webs cuando accedía a alguna por primera vez. Este último grupo fue clave para asegurar algunos aspectos claves que indican las guías de diseño, como son el hacer una plataforma accesible e intuitiva.

De esta forma, partiendo del primer prototipo, gracias a las pruebas, se fueron realizando modificaciones sobre la idea inicial. En las ilustraciones siguientes, se puede apreciar la comparación del diseño del primer prototipo (ilustración 24) respecto al diseño por el que se optó posteriormente (ilustraciones 25 y 26).

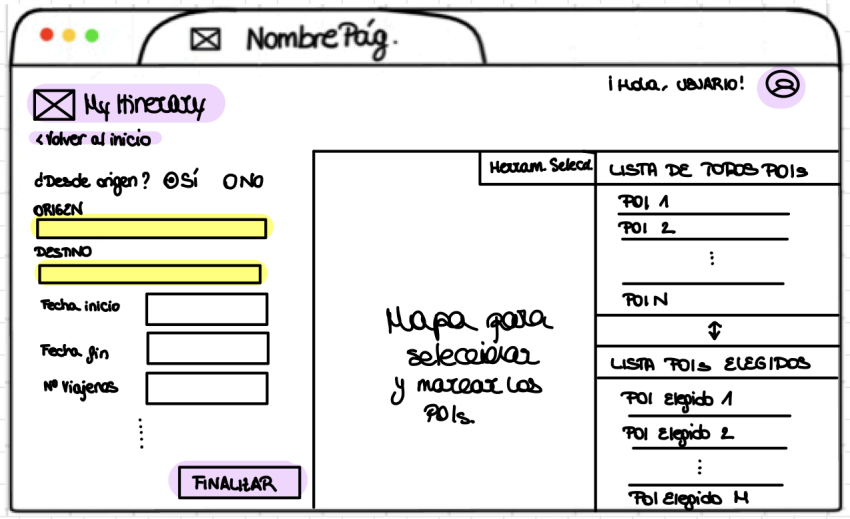


Ilustración 24 Vista del prototipo inicial para la creación de rutas

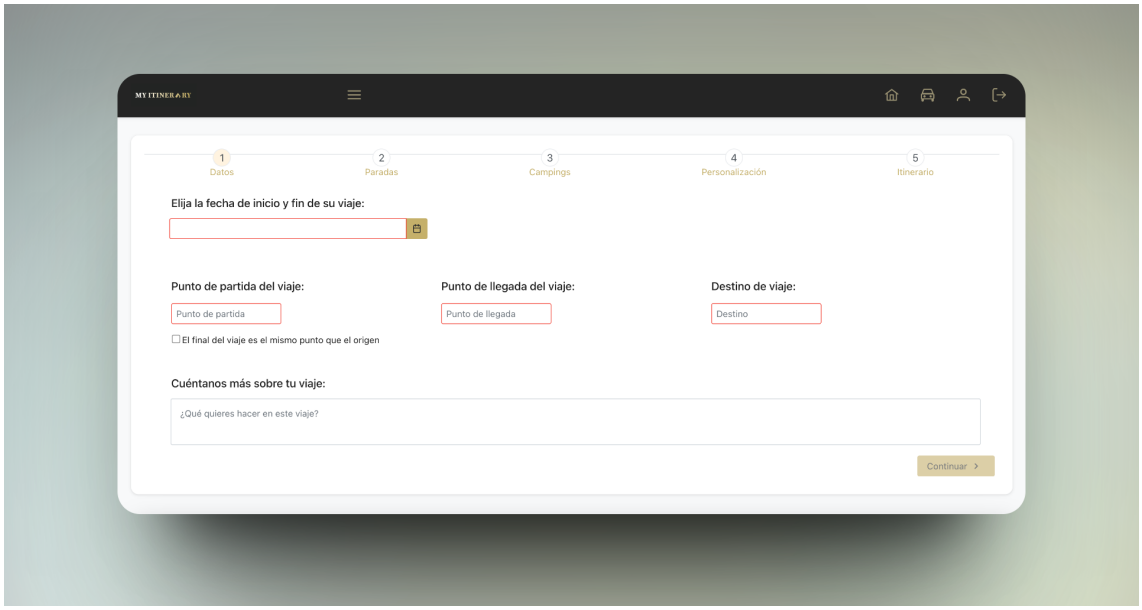


Ilustración 25 Vista de introducción de datos de la creación de rutas en el diseño final

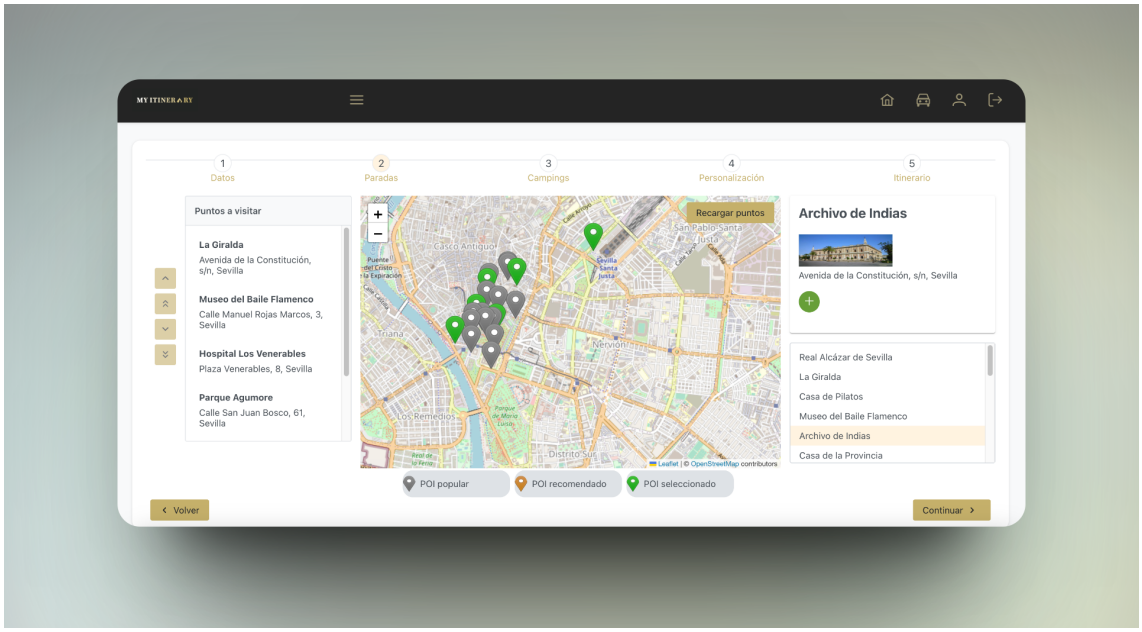


Ilustración 26 Vista de la selección de POIs en el diseño final

Además, durante el desarrollo, se empleó la extensión de Chrome *WAVE Evaluation Tool* para detectar errores de contraste, etc.

5.5. Implementación

En este apartado se explica brevemente la implementación de los componentes del sistema (ilustración 18). Su explicación más detallada puede verse en el Anexo III.

5.5.1. Vue.js

Para el *front-end* de la aplicación (todos los componentes menos el de *API de recomendación*), se ha empleado el *framework* Vue.js [11] (apartado 4.2. Vue.js), el cual permite la creación de ficheros que combinen la parte HTML, CSS y JavaScript. Este marco de trabajo se caracteriza por ser accesible, eficaz y versátil a la hora de crear interfaces de usuario web. Estas características hacen que, a pesar de no manejarlo con anterioridad al desarrollo de este proyecto, sea una tecnología con gran facilidad de uso y de aprendizaje. Además, al combinarla con el *framework* de PrimeVue [32] y con la biblioteca de CSS PrimeFlex [33], ha permitido la implementación de un mejor diseño gracias a sus elementos preconfigurados y listos para usar.

5.5.2. Base de datos

Para la gestión de la base de datos se ha empleado Firebase Cloud Firestore [12] (apartado 4.1. Firebase), una base de datos NoSQL orientada a documentos (se organiza en colecciones que almacenan dichos documentos) y permite la actualización de los datos en tiempo real.

En la ilustración 27 se puede apreciar cómo está estructurada la base, representando la imagen un documento concreto dentro de la colección “Itinerarios”.

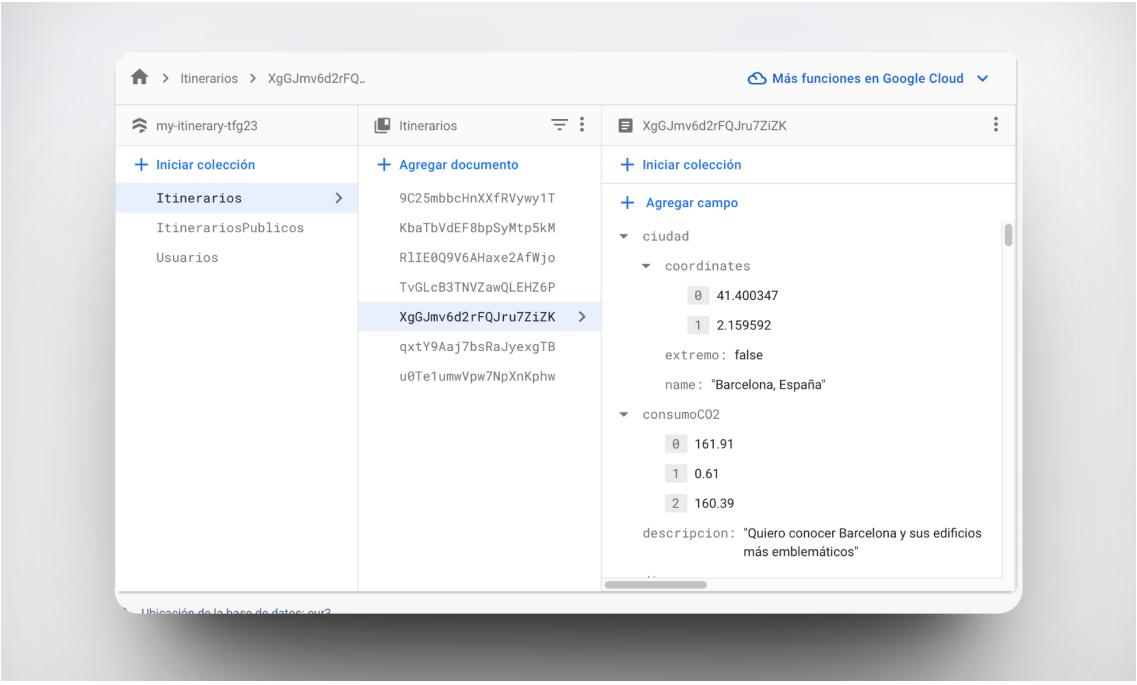


Ilustración 27 Estructura de la colección Itinerarios de la base de datos

5.5.3. Entorno de desarrollo

Para poder llevar a cabo un desarrollo en local como programador, es necesaria la introducción de algunas órdenes en la línea de comandos. Para una explicación más detallada sobre este proceso, así como otros datos necesarios para crear una aplicación de este tipo, se recomienda la consulta del Anexo IV.

5.5.3.1. Instalación de Vue.js

Lo primero es instalar Node.js [64], un gestor de paquetes para la instalación de dependencias del proceso, lo que se consigue con la orden:

```
$ sudo apt install npm
```

Para poder trabajar en el *front*-end, es necesario instalar el CLI (*Command Line Interface*) de Vue.js se emplea la orden:

```
$ npm install-g @vue/cli
```

Posteriormente, se emplea dicho gestor para instalar todas las dependencias necesarias para empezar el desarrollo:

```
$ npm install
```

Con esta orden, se podrá llevar a cabo diversos "modos" de trabajo, siendo los más comunes:

- **`npm run serve`**: para la ejecución de la aplicación en modo de desarrollo de forma local.
- **`npm run build`**: para la construcción del proyecto para producción.

5.5.3.2. Instalación de Firebase

Para añadir Firebase, se debe instalar su herramienta Firebase CLI:

```
$ npm install-g firebase-tools
```

Para el inicio de sesión en la plataforma:

```
$ firebase login
```

Para desplegar la aplicación:

```
$ firebase deploy
```

5.5.3.3. Entorno de desarrollo del sistema de recomendación

Para poder empezar a desarrollar el sistema de recomendación en local, es necesario crear un directorio y un entorno virtual de Python (mediante el módulo *venv*).

```
$ cd carpetaContenedora
$ mkdir nuevoDirectorio
  $ cd nuevoDirectorio
$ python-3 -m venv nombreEntorno-env
```

Para activar el entorno virtual que permita modificar temporalemente el entorno de Python de la sesión actual de la terminal, se utiliza la orden siguiente:

```
$ source nombreEntorno-env/bin/activate
```

A continuación, se instalarían las dependencias, como la de FastAPI (ver en el Anexo IV), y, finalmente, se lanzaría de forma local con:

```
$ uvicorn app.main:app --reload
```

Una vez desarrollada la API, se despliega con Fly.io, por lo que hay que instalar esta herramienta (ver Anexo IV para indicaciones de varios sistemas operativos y siguientes pasos):

```
curl -L https://fly.io/install.sh | sh
```

Para iniciar sesión, se emplea una orden similar a la de Firebase:

```
$ flyctl auth login
```

Y, finalmente, para crear la aplicación y poder desplegarla, se introducen las órdenes (respectivamente):

```
$ flyctl launch
$ flyctl deploy
```

5.6. Presentación de la aplicación

Esta aplicación es accesible desde cualquier navegador moderno [65], como lo es Chrome. Está pensada sobre todo para su uso en dispositivos como ordenadores o dispositivos de gran pantalla, puesto que, al estar en tratamiento constante con elementos como mapas, las pantallas de los móviles podrían no ser las adecuadas y crear una sensación de molestia en el usuario.

El enlace de acceso a la aplicación es el siguiente: https://my-itinerary-tfg23.web.app/ Y, en el siguiente enlace, se puede consultar un vídeo tutorial de la plataforma: https://youtu.be/iadsmbuYAJY Las distintas vistas de la aplicación se encuentran más detalladas en el Anexo V, pero a continuación se muestran algunas de esas vistas en relación con los paquetes que agrupaban la funcionalidad del sistema (ilustración 8, apartado 5.3. Especificación del análisis).

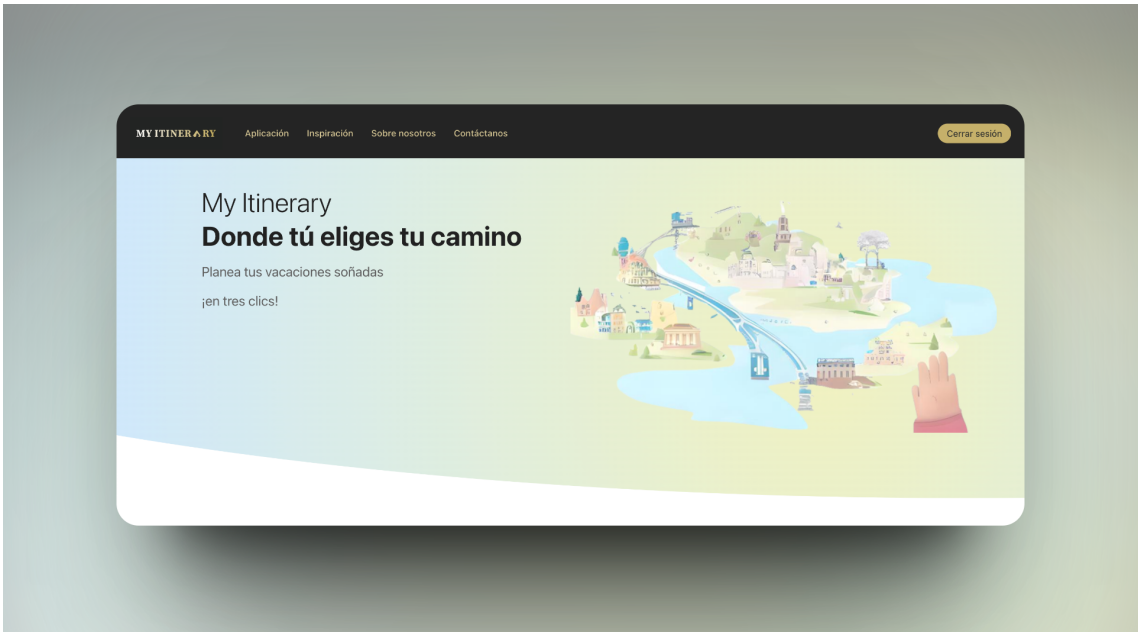


Ilustración 28 Cabecera de la página de *landing*

5.6.1. Gestión de usuarios

Para poder llevar a cabo las funcionalidades básicas de inicio de sesión, registro y restauración de contraseña, se han diseñado las siguientes vistas, las cuales son autoexplicativas.

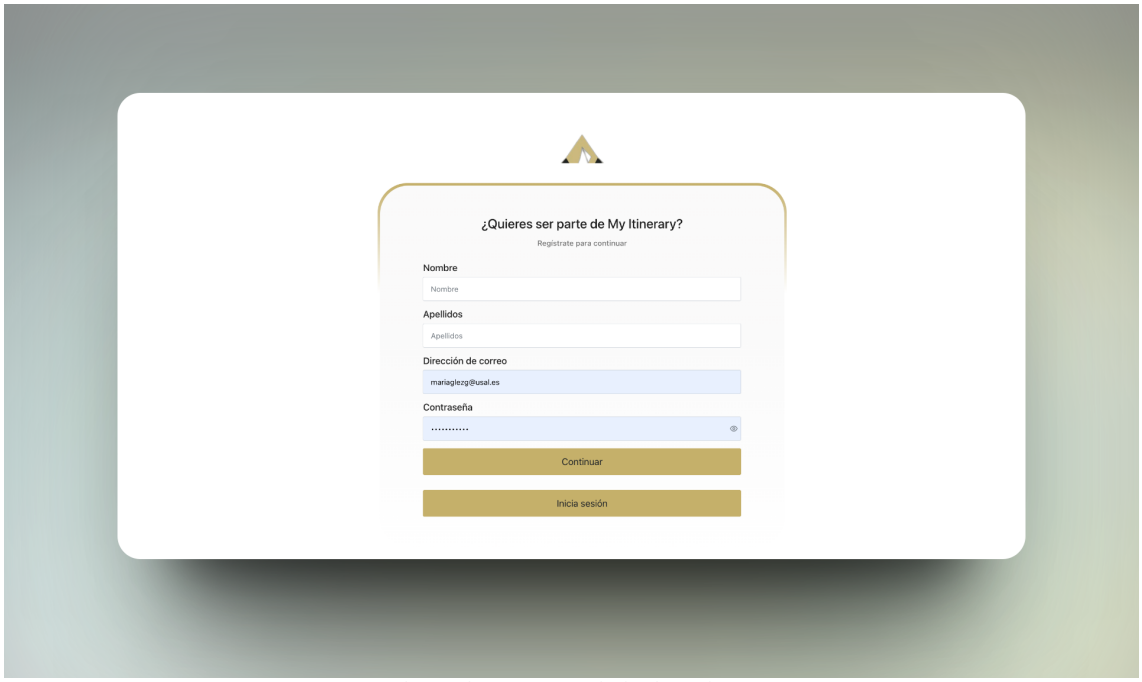


Ilustración 29 Formulario de registro

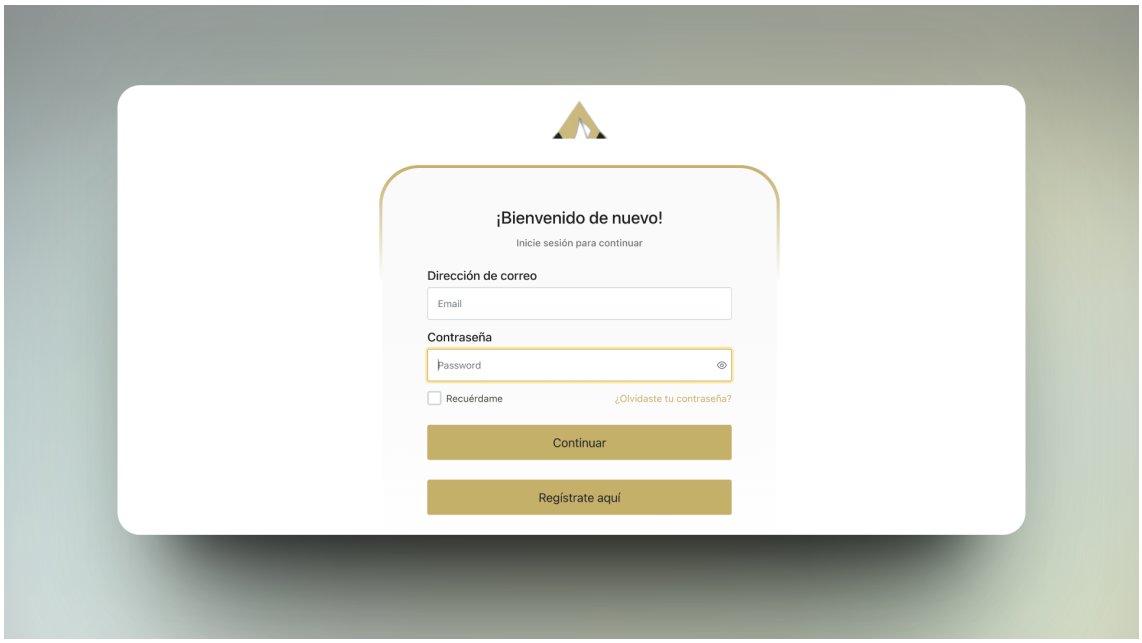


Ilustración 30 Formulario de inicio de sesión

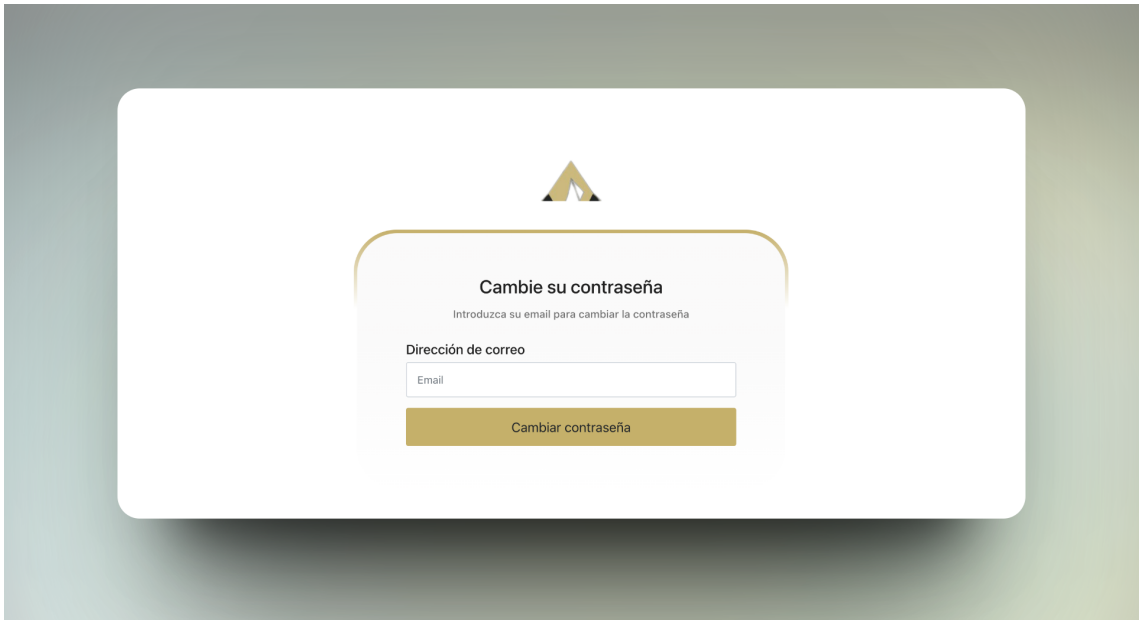


Ilustración 31 Página de restauración de contraseña

El único elemento que cabe destacar de las tres ilustraciones anteriores es el logotipo en la parte superior de todas las vistas. Si se clica dicho elemento, el usuario será redirigido a la página de *landing*.

En cuanto al perfil del usuario y la modificación de datos, se ha reutilizado la misma vista, observada en la ilustración 32. Como se explica en el apartado 3.2.10. Diseño de la interfaz del Anexo III, se decidió hacer lo más minimalista posible para que no abrumara al usuario con información innecesaria.

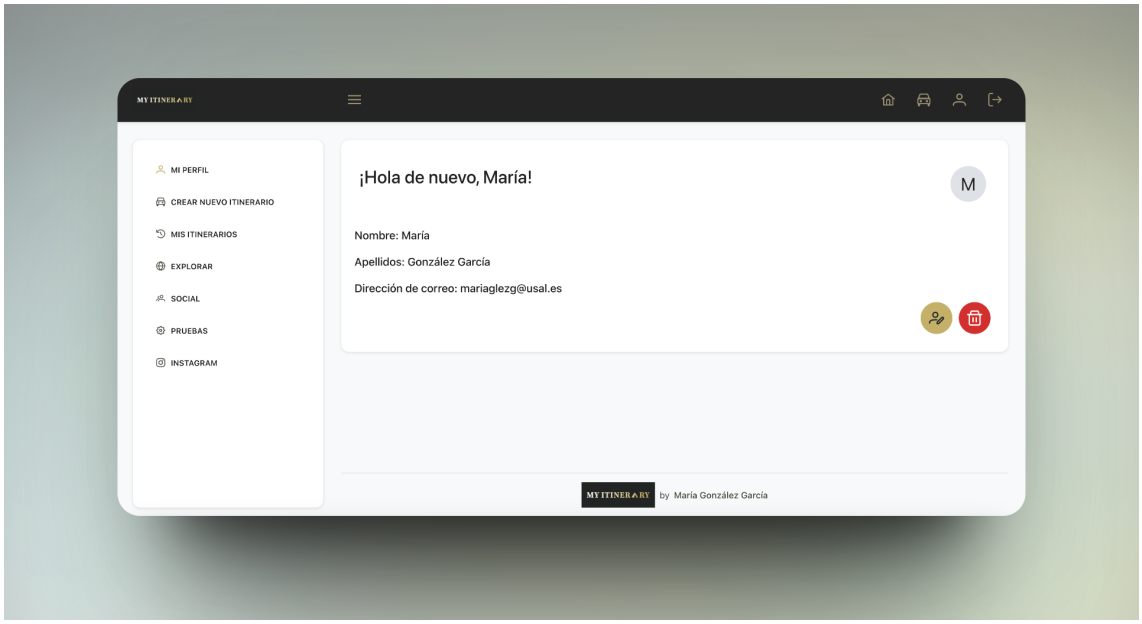


Ilustración 32 Perfil del usuario

Cuando se deseen modificar los datos, se clicará en el botón amarillo de la parte inferior derecha y se mostrarán los campos susceptibles de ser modificados (ilustración 33).

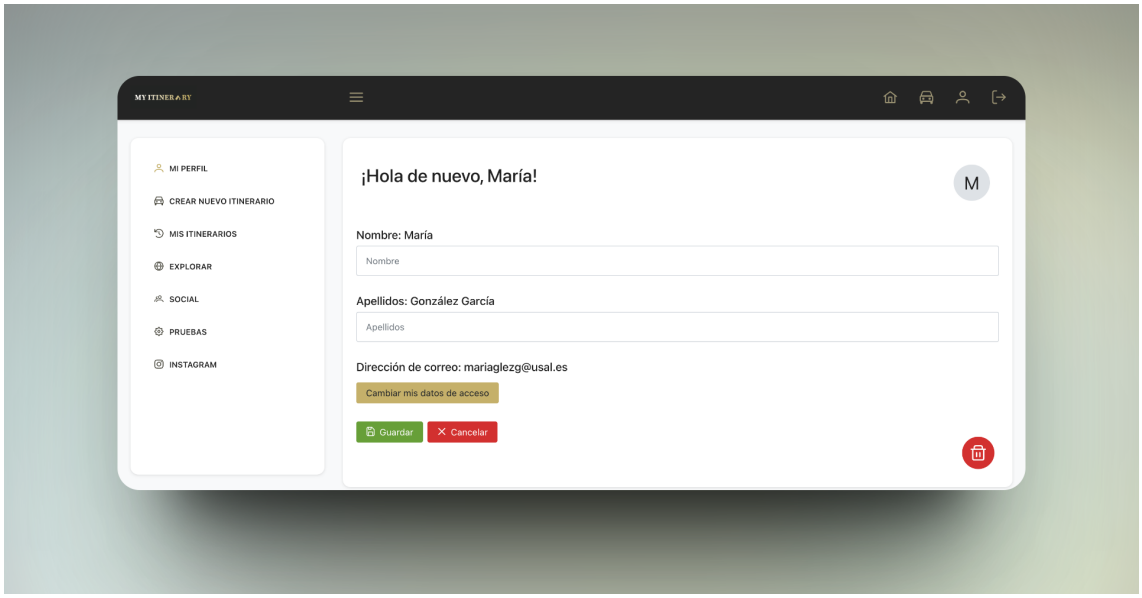


Ilustración 33 Vista de modificación de datos

Para cambiar las credenciales de acceso, se pulsará en el botón de "Cambiar mis datos de acceso" visto en la ilustración 33 y se repetirá el proceso de muestra de campos a modificar (ver la vista en detalle en el Anexo V).

5.6.2. Gestión de rutas

Las siguientes vistas incluyen el proceso de creación de una ruta, desde el momento de introducción de los datos de viaje hasta el resultado final, que sería el itinerario finalizado. Este proceso cuenta con una pantalla de bienvenida (ilustración 34) que recuerda al usuario de manera gráfica aquellos pasos que debe seguir para llegar a obtener su itinerario.

Ilustración 34 Vista de bienvenida de la creación de itinerarios

El primer paso sería la introducción de datos como fechas, destino, puntos de salida del viaje y puntos de llegada y, si el usuario lo desea, una pequeña descripción del propósito de dicho viaje, que será empleado para obtener las recomendaciones. Esta vista se puede ver en la ilustración 35.

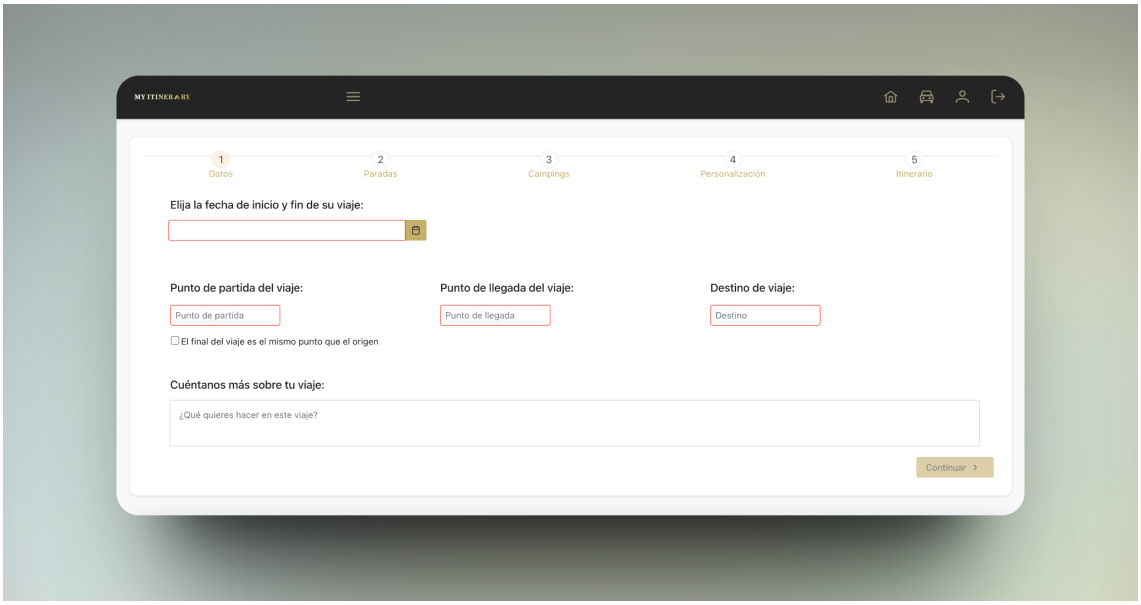


Ilustración 35 Vista de introducción de datos del viaje

Una vez completados los datos necesarios (todos los mostrados menos la descripción), se podrá acceder al siguiente paso, la selección de puntos de interés. En este caso, se le presenta al usuario un mapa interactivo, como se ve en la ilustración 36, con una serie de marcadores pulsables, los cuales revelan información sobre los destinos que indican. De esta forma, el usuario podrá ir seleccionando aquellos que desee añadir a su ruta, pasando al siguiente paso cuando haya terminado:

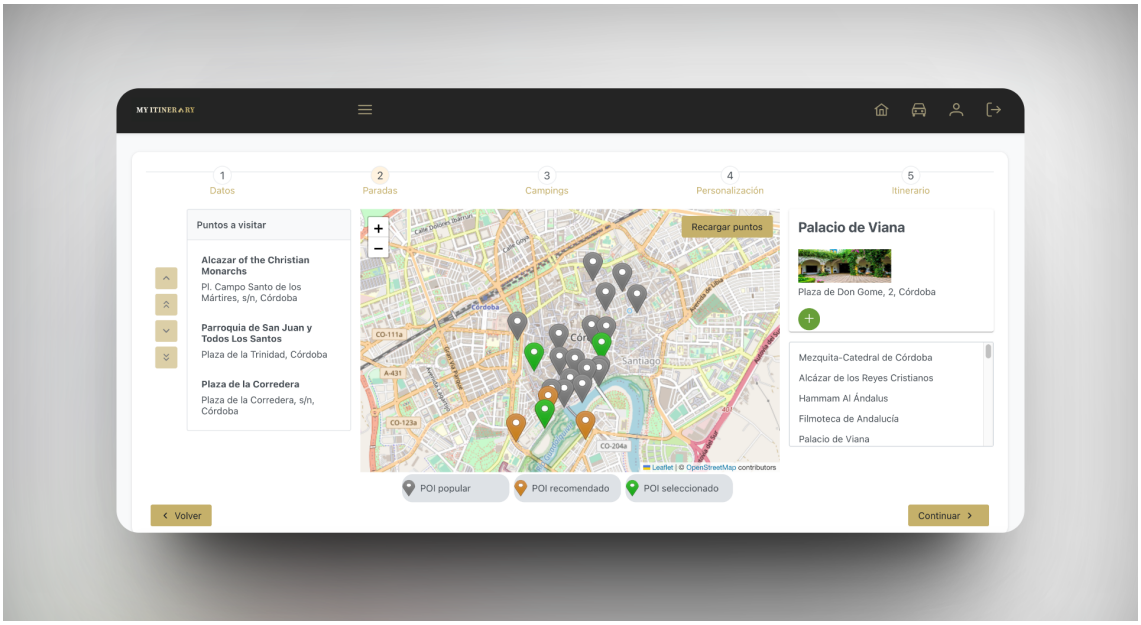


Ilustración 36 Vista de la elección de POIs

Cuando se haya elegido al menos un POI, se podrá continuar al siguiente paso, la elección de lugares de pernoctación (ilustración 37). Esta vista ofrece una funcionalidad muy similar a la anterior, teniendo que realizar el mismo proceso para seleccionar los hoteles o campings. La única diferencia es que, en esta parte, al elegir un lugar, el usuario debe indicar el número de noches que desea alojarse en el mismo, ya que es un dato fundamental a la hora de realizar la organización de los días y, con ello, el cálculo de las rutas.

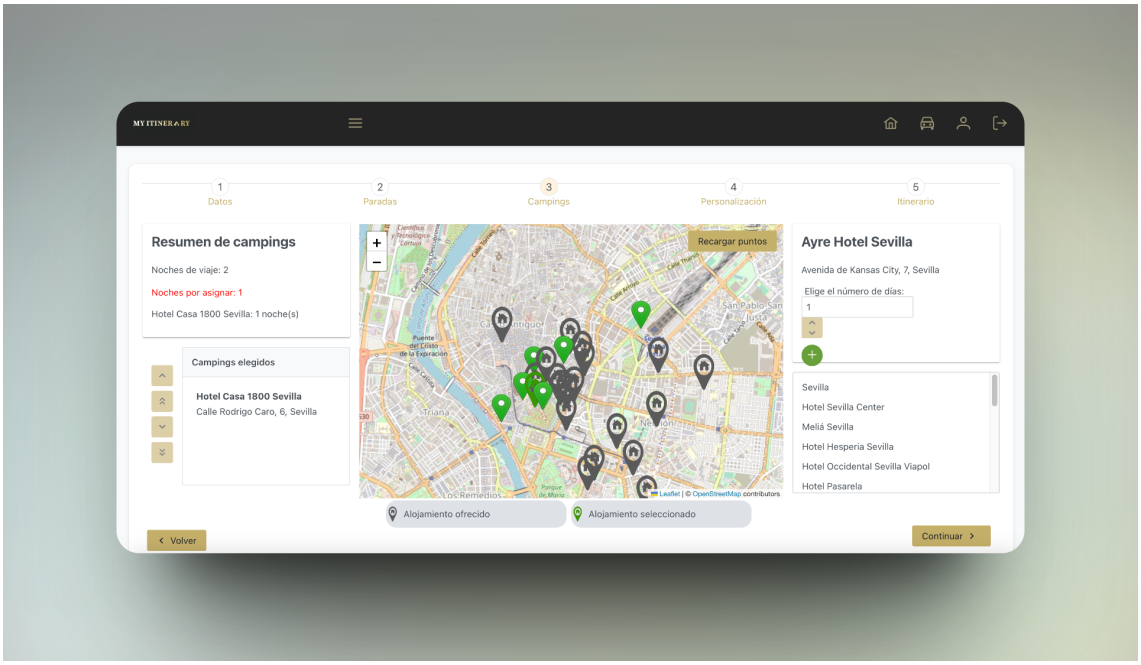


Ilustración 37 Vista de la elección de campings

El siguiente paso del proceso es la personalización de la ruta (ilustración 38). En este caso, ya se muestra la ruta óptima a seguir, así como la organización de los POIs y los campings en días. En esta vista, el usuario puede consultar la ruta para cada día, así como realizar modificaciones en los POIs que ha seleccionado previamente, pudiendo eliminar algunos de su ruta.

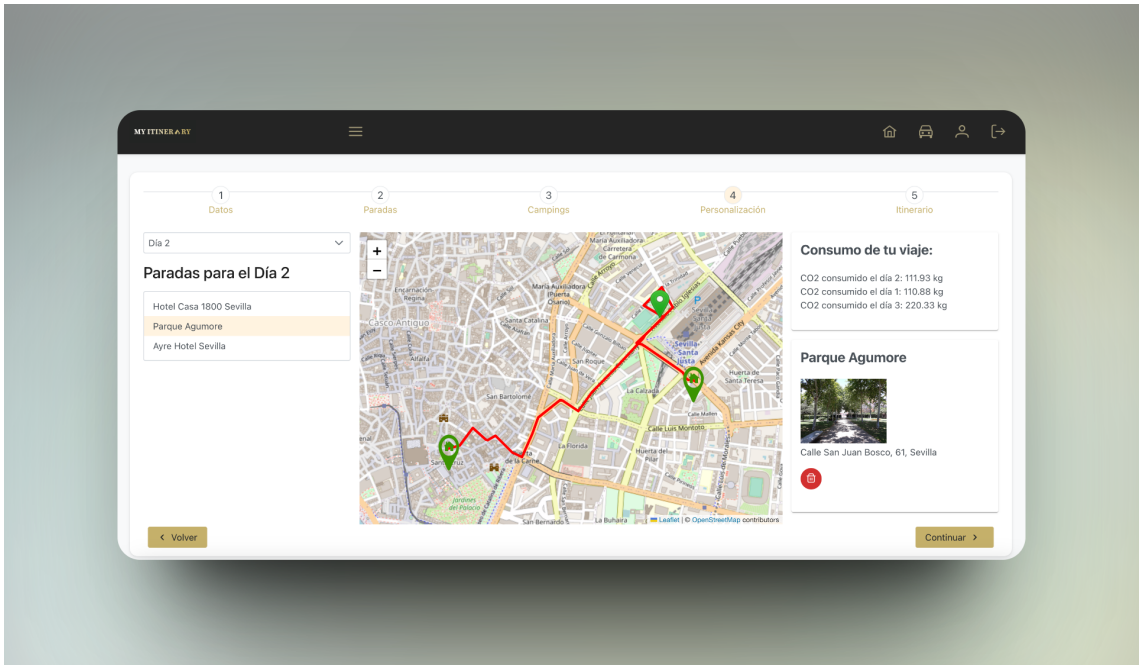


Ilustración 38 Vista de la personalización de la ruta

Finalmente, cuando la personalización haya finalizado, se mostrará una última vista que resuma el viaje (ilustraciones 39 y 40). En la parte superior, se mostrará un cuadro de texto para añadir un título al itinerario creado, así como un *checkbox* para indicar la visibilidad del mismo. Justo debajo se mostrará un mapa con la ruta general a seguir, cada día separado por una línea de distinto color. Finalmente, se encuentra un listado con la información de todos los días de viaje.

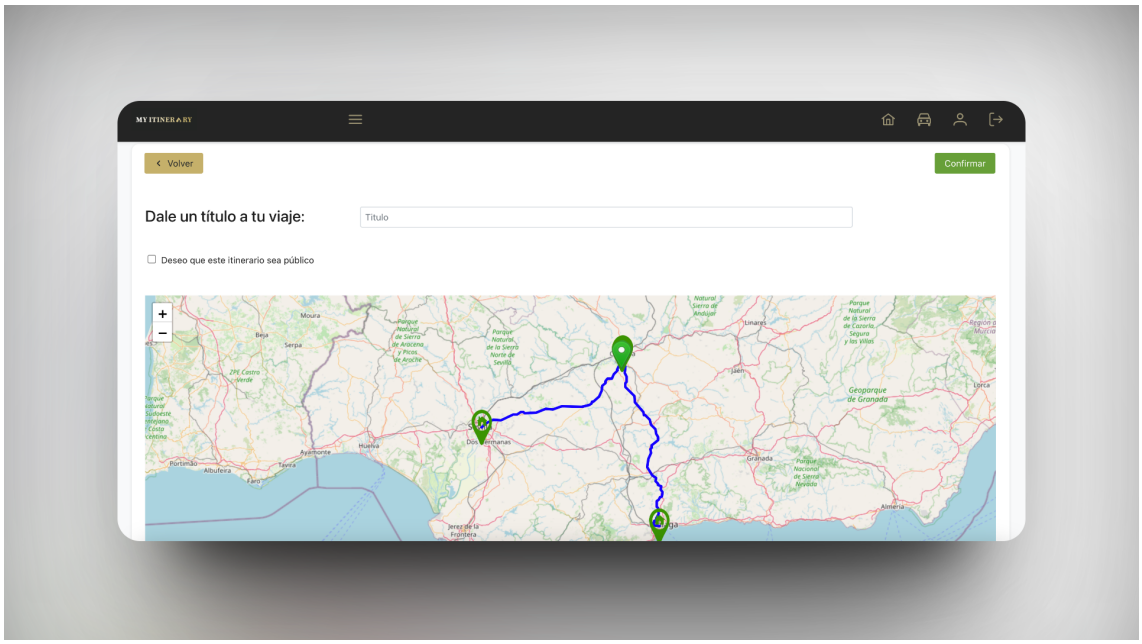


Ilustración 39 Vista del resumen de la información de la ruta (parte 1)

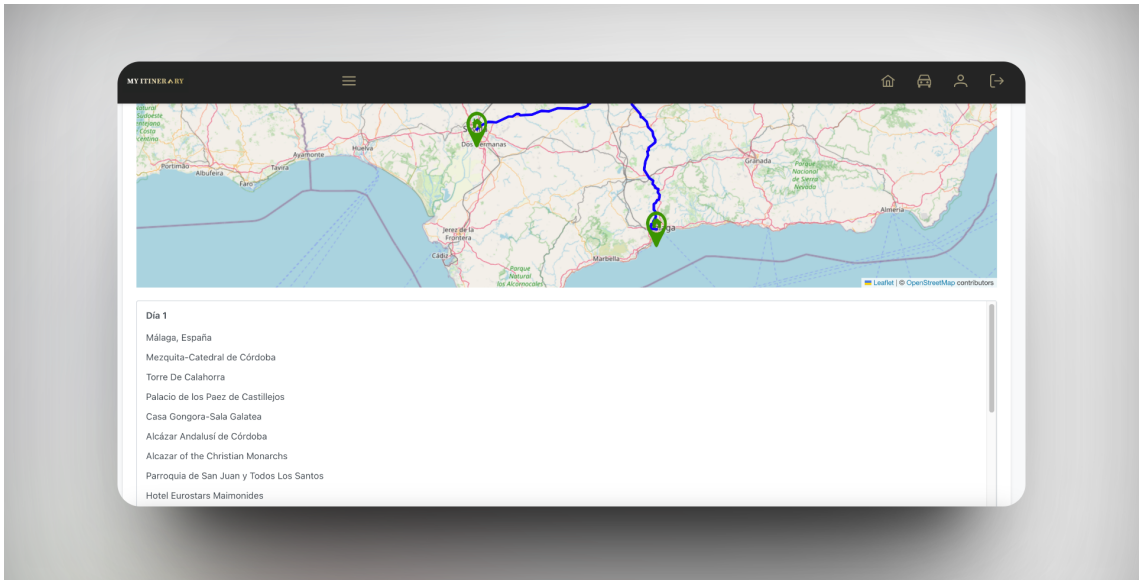


Ilustración 40 Vista del resumen de la información de la ruta (parte 2)

Si se decide confirmar el viaje, se pasaría a la vista de itinerario final, la cual se observará en el apartado siguiente.

5.6.3. Gestión de itinerarios

Una vez se ha creado un itinerario, al finalizar el proceso, el usuario es redirigido a la vista de su itinerario creado (ilustraciones 41 y 42). En ellas, se puede ver de manera más detallada el consumo de CO2, el total de kilómetros recorridos, la organización de la ruta en días, etc.

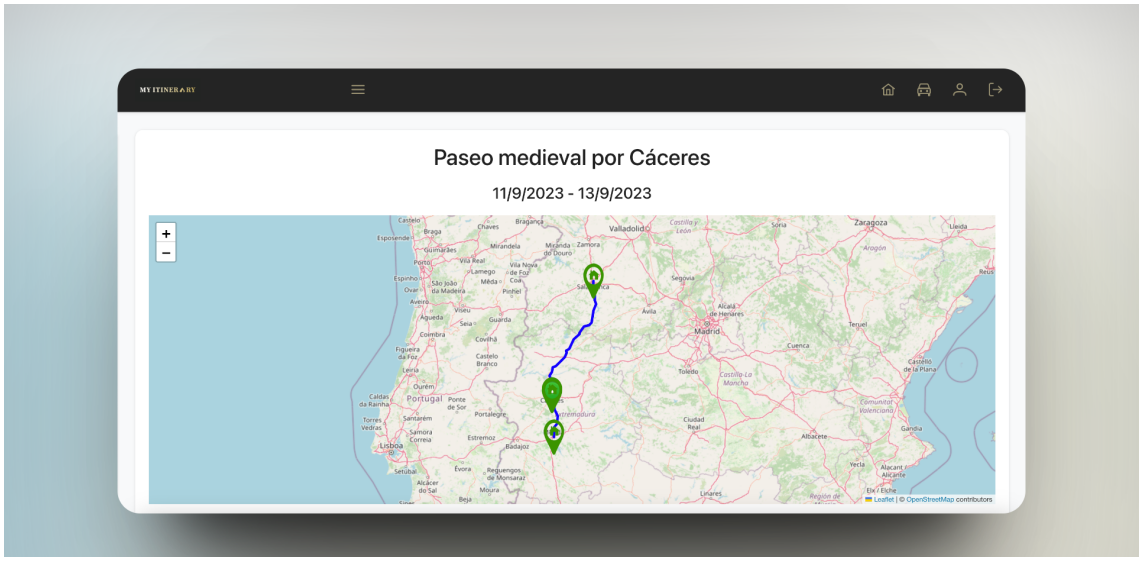


Ilustración 41 Vista del itinerario final (parte 1)

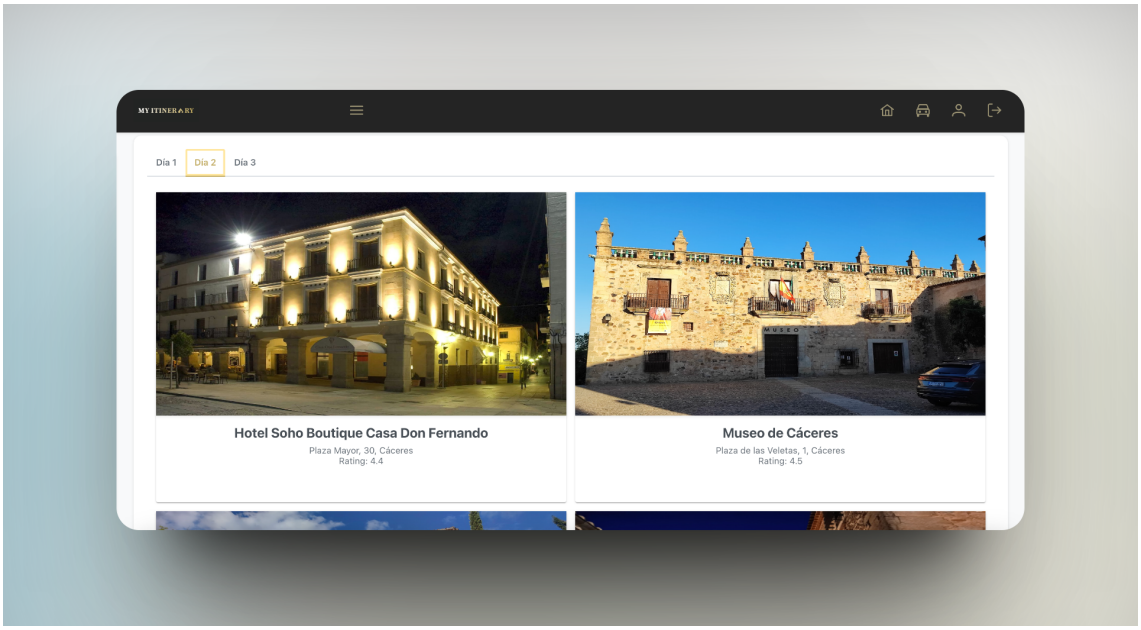


Ilustración 42 Vista del itinerario final (parte 2)

Si el usuario ha diseñado alguna vez algún itinerario, podrá consultarlo en su historial (ilustración 43). Esta vista se basa en una tabla que da unos datos mínimos sobre todos aquellos itinerarios que se el usuario ha ido diseñando, permitiendo realizar acciones como la visualización (será redirigido a la vista de itinerario, ilustraciones 41 y 42) o el borrado, donde se eliminará de la base de datos permanentemente.

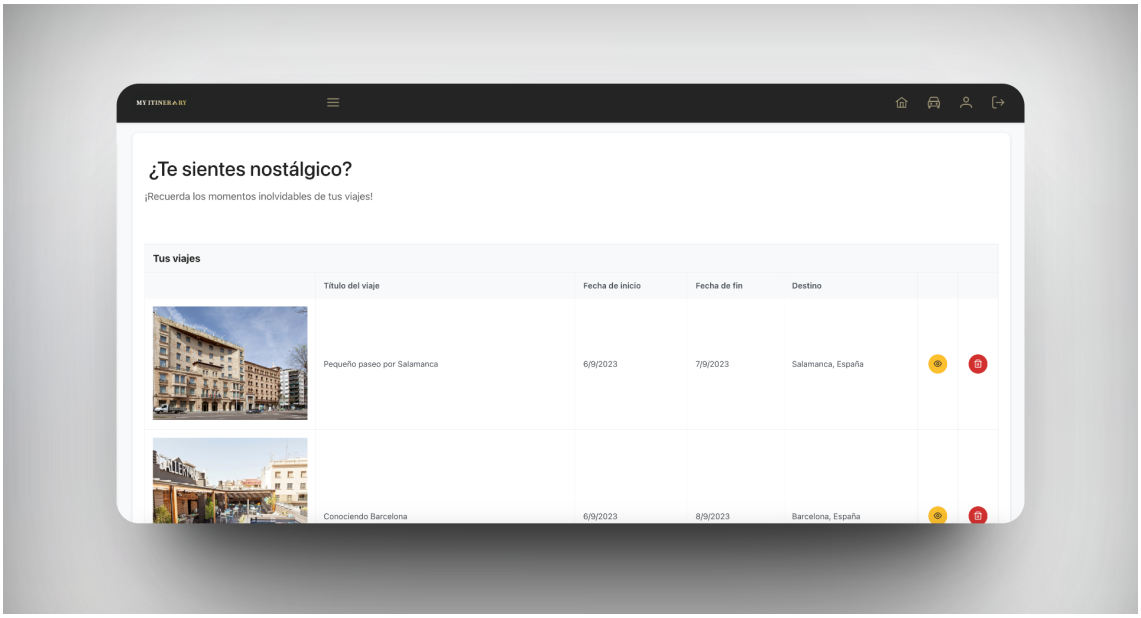


Ilustración 43 Vista del historial de itinerarios

Finalmente, se presenta a continuación el apartado de exploración, en el cual los usuarios pueden descubrir itinerarios creados (que hayan sido marcados como públicos) por otros miembros de la plataforma. Para empezar, se les presenta con un mapa interactivo (ilustración 44) en el cual podrán seleccionar la zona de búsqueda. Cuando deseen explorar dicha zona, pulsarán el botón adecuado para ello y se cargarán las ciudades donde existan itinerarios que correspondan a la búsqueda.

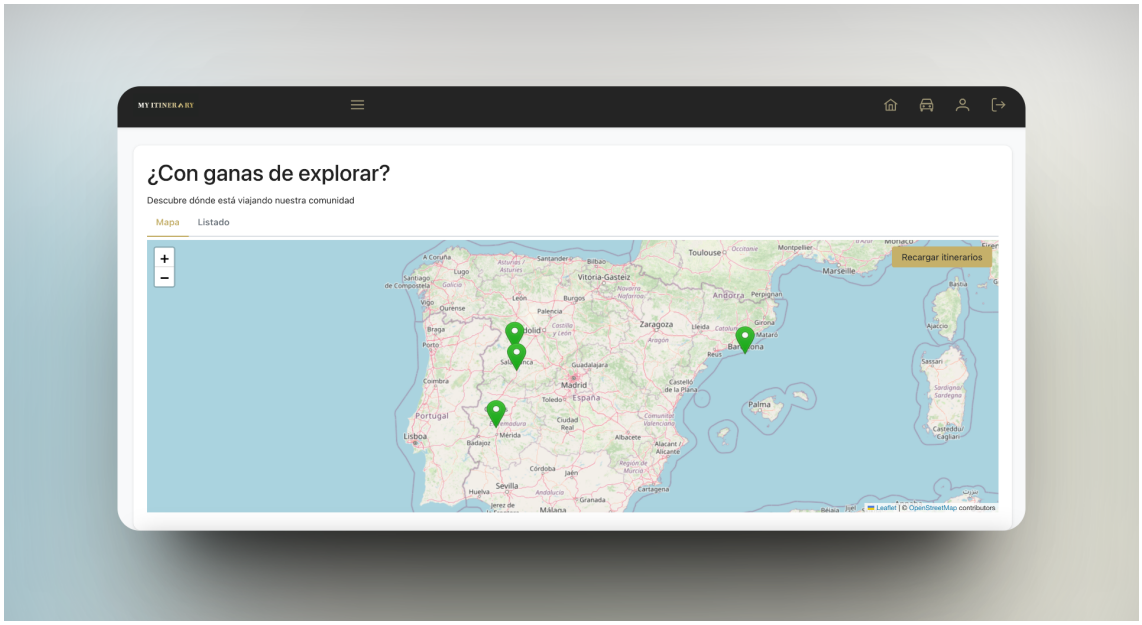
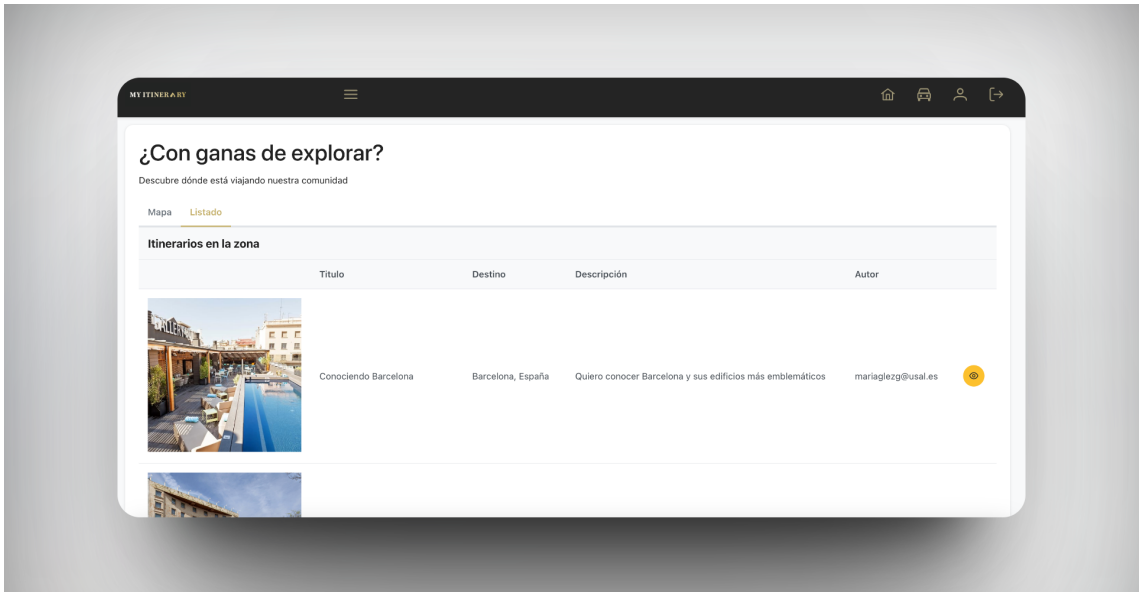


Ilustración 44 Vista del mapa de exploración

Si desean consultar alguno en detalle, seleccionarán la vista de “Listado” y podrán consultar los resultados obtenidos (ilustración 45). Si se desea consultar alguno en detalle, basta con pulsar el botón para ello y se le redirigirá a la vista de itinerario (ilustraciones41 y 42).



6. Conclusiones y líneas de trabajo futuras

6.1. Conclusiones

Tras el desarrollo de este proyecto, las primeras conclusiones son el cumplimiento de los objetivos establecidos:

- **Desarrollo de un módulo de gestión de usuarios**: se ha conseguido la gestión de los datos de usuarios como la baja, el alta o la modificación de los datos.
- **Desarrollo de un sistema de recomendación de POIs**: se ha conseguido implementar un sistema que, tomando un texto en lenguaje natural, pueda interpretarlo y devolver unos resultados acordes a dichos requerimientos.
- **Desarrollo de un sistema de optimización de rutas**: si bien no es una parte implementada, se ha conseguido conocer más a fondo cómo trabajan estos sistemas, así como aprender a utilizar sus servicios.
- **Desarrollo de un módulo de gestión de rutas**: se ha conseguido gestionar todos los datos que forman las rutas de manera conjunta, así como tratar con esos sistemas de optimización mencionados anteriormente para obtener resultados.
- **Desarrollo de un módulo de gestión de itinerarios**: se ha conseguido unir toda la información anterior para poder crear un resultado final que pueda servir al usuario para una tarea concreta, que es la realización de un viaje.

Aunque todavía no se hayan realizado las pruebas pertinentes para su explotación en un entorno real, como sería su uso individual o incluso en empresas como agencias de viaje, estos pequeños logros hacen que el sistema pueda ser capaz de desarrollarse en dicho entorno con las pruebas necesarias y las mejoras que ahí se descubriría. De esta forma, se espera que, cuando esté completamente probado, pueda ser una herramienta útil tanto para esos usuarios individuales como para los trabajadores de las empresas, que puedan usarla para aliviar su carga de trabajo, como ya se mencionaba en los apartados introductorios de este documento.

A nivel personal, este proyecto tiene gran importancia para mí porque ha supuesto alcanzar una serie de objetivos de manera conjunta que antes no me consideraba capaz de lograr. Para lograrlos, me he tenido que apoyar en todo lo aprendido estos cuatro años en diversas asignaturas. Haber sido capaz de recoger todos esos conocimientos, así como los nuevos aprendidos y las explicaciones de los tutores, para poder crear un sistema completo que podría emplearse en distintos contextos y que no fuese una simple idea teórica, ha supuesto una satisfacción personal.

Además, todo el empeño puesto y las indicaciones de los tutores me sirvieron para poder presentar este proyecto al plan de prototipos TCUE [66], en el cual fue seleccionado, brindándome nuevos conocimientos sobre áreas más desconocidas para mí, como es el plano empresarial y económico.

6.2. Líneas de trabajo futuras

Este proyecto constituye solo una parte del gran potencial que podrían tener las plataformas de este estilo, cada vez más usadas debido al gran incremento de este tipo de viajes como se mencionaba en la introducción de este documento. Entre otras posibles líneas, se mencionan algunas de las que se han planteado durante el desarrollo:

- Añadir un sistema de recomendación más experto que pueda obtener información más detallada sobre el usuario. Esto se podría conseguir si, por ejemplo, el sistema fuese capaz de conectarse con otras plataformas, por ejemplo, con redes sociales como Twitter o Instagram, y estudiar aquellas publicaciones con las que interactúa el usuario. De esta forma, se podría añadir un apartado de recomendaciones para cada miembro de la plataforma.

- Cambiar el enfoque de la plataforma y, en lugar de que sea para un uso privado, que adopte unas características de red social. En este contexto, se podría permitir al usuario que tuviese una comunidad de amigos con la que interactuara, pudiendo llevar a cabo acciones como la recomendación de los itinerarios de sus amigos o la compartición de estos a otros miembros.

Además de estas, podrían darse más líneas de mejora que podrían convertirse incluso en buenas oportunidades de negocio y monetización de la plataforma como, por ejemplo, la conexión con plataformas como *Booking* para la oferta de hostales o campings promocionados.

7. Referencias

[1] T. Noguchi, T. Hayashi, Y. Kubo, N. Tomiyama, A. Ochi, and H. Hayashi, "Living Alone and Depressive Symptoms among Older Adults in the COVID-19 Pandemic: Role of Non–Face-to-Face Social Interactions," *J Am Med Dir Assoc*, vol. 24, no. 1, pp. 17-21.e4, Jan. 2023, doi: 10.1016/J.JAMDA.2022.10.014.

[2] R. Wei, Y. Zhang, S. Gao, B. J. Brown, S. Hu, and B. G. Link, "Health disparity in the spread of COVID-19: Evidence from social distancing, risk of interactions, and access to testing," *Health Place*, vol. 82, p. 103031, Jul. 2023, doi: 10.1016/J.HEALTHPLACE.2023.103031.

[3] L. Laguna, S. Fiszman, P. Puerta, C. Chaya, and A. Tárrega, "The impact of COVID-19 lockdown on food priorities. Results from a preliminary study using social media and an online survey with Spanish consumers," *Food Qual Prefer*, vol. 86, p. 104028, Dec. 2020, doi: 10.1016/J.FOODQUAL.2020.104028.

[4] F. González-Reverté, J. M. Gomis-López, and P. Díaz-Luque, "Reset or temporary break? Attitudinal change, risk perception and future travel intention in tourists experiencing the COVID-19 pandemic," *Journal of Tourism Futures*, vol. ahead-of-print, no. ahead-of-print, 2022, doi: 10.1108/JTF-03-2021-0079/FULL/PDF.

[5] "Statistics - European Caravan Federation (ECF)." https://www.e-c-f.com/exposee/statistics/ (accessed Apr. 26, 2023).

[6] Б. Ас and Р. Лe, "The problem of decline in popularity of travel agencies", Accessed: May 12, 2023. [Online]. Available: http://edoc.bseu.by

[7] "Caramaps - Anuario de los campings, áreas de servicio, anfitrión." https://www.caramaps.com/ (accessed Apr. 25, 2023).

[8] "Tripnotes.ai - Intelligent Travel Planner." https://tripnotes.ai/app (accessed Apr. 25, 2023).

[9] J. L. Sarkar and A. Majumder, "gTour: Multiple itinerary recommendation engine for group of tourists," *Expert Syst Appl*, vol. 191, 2022, doi: 10.1016/j.eswa.2021.116190.

[10] L. Chen, L. Zhang, S. Cao, Z. Wu, and J. Cao, "Personalized itinerary recommendation: Deep and collaborative learning with textual information," *Expert Syst Appl*, vol. 144, p. 113070, Apr. 2020, doi: 10.1016/J.ESWA.2019.113070.

[11] "Vue.js - The Progressive JavaScript Framework | Vue.js." https://vuejs.org/ (accessed May 22, 2023).

[12] "Firestore | Firebase." https://firebase.google.com/docs/firestore?hl=es-419 (accessed May 22, 2023).

[13] J. A. Konstan, M. D. Ekstrand, and J. T. Riedl, "Collaborative Filtering Recommender Systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011, doi: 10.1561/1100000009.

[14] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Syst Appl*, vol. 41, no. 4, pp. 2065–2073, Mar. 2014, doi: 10.1016/J.ESWA.2013.09.005.

[15] J. Gope and S. K. Jain, "A survey on solving cold start problem in recommender systems," *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, vol. 2017-January, pp. 133–138, Dec. 2017, doi: 10.1109/CCAA.2017.8229786.

[16] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and Y. Tasoulas, "Cluster-based heuristics for the team orienteering problem with time windows,"

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7933 LNCS, pp. 390–401, 2013, doi: 10.1007/978-3-642-38527-8_34/COVER.
[17] "Openrouteservice." https://openrouteservice.org/ (accessed Aug. 27, 2023).
[18] "Servicio Directions | API de Maps JavaScript | Google for Developers." https://developers.google.com/maps/documentation/javascript/directions?hl=es-419 (accessed Aug. 27, 2023).
[19] M. Vidal, "¿Qué son las Progressive Web Apps? ¿Por qué son tan importantes?," *Thinking for Innovation*, Nov. 2022, Accessed: May 22, 2023. [Online]. Available: https://www.iebschool.com/blog/progressive-web-apps-analitica-usabilidad/
[20] "Documentación de Firebase." https://firebase.google.com/docs?hl=es-419 (accessed May 22, 2023).
[21] "Firebase Authentication | Firebase Authentication." https://firebase.google.com/docs/auth?hl=es-419 (accessed May 22, 2023).
[22] "Firebase Hosting | Firebase Hosting." https://firebase.google.com/docs/hosting?hl=es-419 (accessed May 22, 2023).
[23] "Supabase | The Open Source Firebase Alternative." https://supabase.com/ (accessed Sep. 03, 2023).
[24] S. bin Uzayr, N. Cloud, and T. Ambler, "Vue.js," *JavaScript Frameworks for Modern Web Development*, pp. 523–539, 2019, doi: 10.1007/978-1-4842-4995-6_14.
[25] "Getting Started - vue.js." https://012.vuejs.org/guide/ (accessed May 22, 2023).
[26] "JavaScript | MDN." https://developer.mozilla.org/es/docs/Web/JavaScript (accessed May 22, 2023).
[27] "HTML: Lenguaje de etiquetas de hipertexto | MDN." https://developer.mozilla.org/es/docs/Web/HTML (accessed May 22, 2023).
[28] "CSS | MDN." https://developer.mozilla.org/es/docs/Web/CSS (accessed May 22, 2023).
[29] "Angular." https://angular.io/docs (accessed Sep. 01, 2023).
[30] "primefaces/sakai-vue-cli: Free Vue Admin Template by PrimeVue." https://github.com/primefaces/sakai-vue-cli (accessed Sep. 03, 2023).
[31] "PrimeFaces – Ultimate UI Framework." https://www.primefaces.org/ (accessed Sep. 03, 2023).
[32] "Getting Started - PrimeVue." https://primevue.org/installation (accessed Aug. 20, 2023).
[33] "PrimeFlex." https://www.primefaces.org/primeflex/ (accessed Sep. 03, 2023).
[34] "Python España." https://es.python.org/ (accessed Sep. 01, 2023).
[35] "FastAPI." https://fastapi.tiangolo.com/ (accessed Aug. 27, 2023).
[36] "Poetry - Python dependency management and packaging made easy." https://python-poetry.org/ (accessed Sep. 02, 2023).
[37] "Docker: Accelerated Container Application Development." https://www.docker.com/ (accessed Aug. 27, 2023).
[38] "Dockerfile reference | Docker Docs." https://docs.docker.com/engine/reference/builder/# (accessed Sep. 01, 2023).
[39] "Fly.io." https://fly.io/ (accessed Aug. 27, 2023).
[40] "Arquitecturas sin servidor." https://aws.amazon.com/es/lambda/serverless-architectures-learn-more/ (accessed Aug. 13, 2023).
[41] "Goodnotes | Notes Reimagined | Note-Taking App." https://www.goodnotes.com/ (accessed Sep. 03, 2023).

[42] "Brandmark Logo Maker - the most advanced AI logo design tool." https://brandmark.io/ (accessed May 25, 2023).
[43] "Generador de paletas de colores: extraer la paleta de colores de la imagen." https://products.aspose.app/html/es/color-palette-generator (accessed May 25, 2023).
[44] "Generador de imágenes de Microsoft Bing." https://www.bing.com/images/create?FORM=GENILP (accessed Sep. 03, 2023).
[45] "WAVE Evaluation Tool - Chrome Web Store." https://chrome.google.com/webstore/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh (accessed Sep. 06, 2023).
[46] "Software de administración de proyectos | Microsoft Project." https://www.microsoft.com/es-es/microsoft-365/project/project-management-software (accessed May 04, 2023).
[47] N. E. Fenton and J. Bieman, *Software metrics : a rigorous and practical approach*, 2nd ed. London: PWS Pub., 1997.
[48] "Ideal Modeling & Diagramming Tool for Agile Team Collaboration." https://www.visual-paradigm.com/ (accessed Aug. 07, 2023).
[49] "draw.io." https://app.diagrams.net/ (accessed Aug. 07, 2023).
[50] "PlantUML Web Server." http://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000 (accessed Sep. 03, 2023).
[51] I. Jacobson, *El Proceso Unificado de Desarrollo de Software (Spanish Edition)*. Addison Wesley Publishing Company, 2000.
[52] P. Y. Reyes-Delgado, M. Mora, H. A. Duran-Limon, L. C. Rodríguez-Martínez, R. V. O'Connor, and R. Mendoza-Gonzalez, "The strengths and weaknesses of software architecture design in the RUP, MSF, MBASE and RUP-SOA methodologies: A conceptual review," *Comput Stand Interfaces*, vol. 47, pp. 24–41, Aug. 2016, doi: 10.1016/j.csi.2016.02.005.
[53] A. Durán and T. Beatriz Bernárdez Jiménez, "Metodología para la Elicitación de Requisitos de Sistemas Software Versión 2.1".
[54] "What is Component Diagram?" https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/ (accessed Sep. 05, 2023).
[55] "Pinia | The intuitive store for Vue.js." https://pinia.vuejs.org/ (accessed May 25, 2023).
[56] "What is Vuex? | Vuex." https://vuex.vuejs.org/ (accessed May 25, 2023).
[57] "Getting Started - vue.js." https://012.vuejs.org/guide/ (accessed Aug. 30, 2023).
[58] "Design Patterns with UML: Facade Pattern." http://design-patterns-with-uml.blogspot.com/2013/02/facade-pattern.html (accessed Aug. 30, 2023).
[59] "Documentación de Google Maps Platform | Places API | Google for Developers." https://developers.google.com/maps/documentation/places/web-service?hl=es-419 (accessed May 25, 2023).
[60] "About | Cohere." https://cohere.com/about (accessed Aug. 27, 2023).
[61] "Web Standards | W3C." https://www.w3.org/standards/ (accessed Sep. 05, 2023).
[62] "Acorn - Firefox Design Systems." https://acorn.firefox.com/latest/acorn-aRSAh0Sp (accessed Sep. 05, 2023).
[63] "IBM Design Language." https://www.ibm.com/design/language/ (accessed Sep. 05, 2023).
[64] "About | Node.js." https://nodejs.org/en/about (accessed Jul. 04, 2023).

[65] "'PWA' | Can I use... Support tables for HTML5, CSS3, etc." https://caniuse.com/?search=PWA (accessed Sep. 05, 2023).
[66] "Prototipos orientados al mercado." https://tcue.usal.es/prototipos-orientados-al-mercado (accessed Sep. 06, 2023).