

Grado universitario en Ingeniería Informática
2020-2021

Trabajo Fin de Grado

“Hermes: Sistema recomendador y planificador turístico”

Álvaro Galisteo Álvarez

Tutor

Raquel Fuentetaja Pizán

Leganés, 2021



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

RESUMEN

En este proyecto se ha desarrollado un sistema recomendador y planificador turístico de código abierto denominado Hermes, que introduce cambios y mejoras respecto a proyectos existentes comerciales y/o académicos y que puede beneficiar al entorno social y económico actual. Este sistema se ha dividido en cuatro módulos: un sistema recomendador basado en una arquitectura híbrida, un sistema planificador que hace uso del lenguaje PDDL, una interfaz de usuario que hace uso de las librerías Bootstrap, Font Awesome y Leaflet y una API REST que permite al sistema interactuar con todos los módulos, gestionar la base de datos y enviar correos electrónicos. Este sistema está inicialmente disponible en español para la ciudad de Madrid.

Además del desarrollo, se han realizado pruebas con usuarios y validaciones que han mostrado que el sistema es efectivo y útil. Finalmente se destaca un importante objetivo por la seguridad, preservar la privacidad de los usuarios y mantener la ideología de código abierto y las licencias libres. La plataforma ha sido desplegada de forma gratuita en internet.

Palabras clave: Turismo, Sistema recomendador, Recomendación de puntos de interés, Sistema planificador, Planificación de itinerarios, Privacidad, Código Abierto, Node.js, Rust, WebApp, Docker, Despliegues

ABSTRACT

This project develops an open source tourism recommender and planner system called Hermes, which introduces changes and improvements with respect to existing commercial and/or academic projects and can benefit the current social and economic environment. This system has been divided into four modules: a recommender system based on a hybrid architecture, a planner system that makes use of the PDDL language, a user interface that makes use of the Bootstrap, Font Awesome and Leaflet libraries and a REST API that allows the system to interact with all the modules, manage the database and send emails. This system is initially available in Spanish for the city of Madrid.

In addition to the development, user experimentation and validation have shown that the system can be effective and useful. Finally, an important focus on security, preserving user privacy and maintaining the open source ideals and free licenses is highlighted. The platform has been deployed for free on the Internet.

Keywords: Tourism, Recommender system, Points of interest recommendation, Planning system, Itinerary planning, Privacy, Open Source, Node.js, Rust, WebApp, Docker, Deployments.

AGRADECIMIENTOS

Dedicado a mi abuelo, a quien le hubiese gustado ver todo lo que he llegado a poder hacer con la tecnología.

Deseo expresar mi agradecimiento a todos los que me han acompañado y ayudado a la elaboración de este Trabajado de Fin de Grado.

Me gustaría agradecer a mis padres, a mi hermana y a mi pareja por apoyarme durante todo este tiempo, por darme sugerencias, y por dejarme evaluar el proyecto con ellos. Además, gracias a mi hermana, esta plataforma ha podido tener un logotipo.

También me gustaría agradecer a Raquel Fuentetaja por ser mi tutora y haber apoyado en este proyecto.

Finalmente, quiero agradecer también a mis amigos por ayudarme a evaluar el proyecto y a ofrecerme sugerencias y ayuda durante el desarrollo de este, y a la comunidad Open Source con la que ha sido posible crear este trabajo.

ÍNDICE DE CONTENIDOS

Resumen.....	III
Abstract.....	IV
Agradecimientos.....	VI
Índice de contenidos.....	VII
Índice de figuras.....	XII
Índice de tablas.....	XIV
Lista de abreviaturas.....	XIX
1. Introducción.....	1
1.1. Motivación del Trabajo.....	1
1.2. Objetivos.....	2
1.3. Organización del documento.....	3
2. Estado del Arte.....	4
2.1. Descripción de un sistema recomendador.....	4
2.1.1. Metodologías de recomendación.....	4
2.1.2. Algoritmos de recomendación.....	7
2.1.3. Medidas de similitud.....	8
2.2. Descripción de un sistema planificador.....	14
2.2.1. Algoritmos de planificación.....	15
2.3. Marco actual.....	17
2.3.1. Sistemas existentes académicos.....	17
2.3.2. Sistemas existentes comerciales.....	19
2.4. Tecnología.....	21
2.4.1. Interfaces y frontend.....	21
2.4.2. Backend.....	22

2.4.3. Librerías y proyectos adicionales.....	24
2.4.4. Bases de datos.....	25
2.4.5. Seguridad.....	26
HTTPS.....	27
CORS, CSP y Cross Site Scripting.....	27
Protección de datos en reposo.....	28
2.4.6. Despliegues.....	29
3. Análisis del Sistema.....	31
3.1. Diseño.....	31
3.2. Usuarios.....	35
3.3. Requisitos.....	36
3.3.1. Requisitos de usuario.....	37
3.3.2. Requisitos de software.....	41
3.3.2.1. Requisitos funcionales.....	41
3.3.2.2. Requisitos no funcionales.....	49
3.3.3. Matriz de trazabilidad.....	52
3.4. Casos de uso.....	54
3.5. Prototipo.....	68
3.6. Clases.....	73
4. Implementación.....	86
4.1. Entorno operacional.....	86
4.2. Modelo Relacional.....	87
4.3. Obtención y procesado de datos.....	90
4.4. Interfaz de usuario.....	94
4.4.1. Mapas.....	95
4.4.2. Landing page.....	95

4.4.3. Registro e inicio de sesión.....	96
4.4.4. Recuperación de contraseña.....	98
4.4.5. Recomendaciones.....	99
4.4.6. Planes.....	101
4.4.7. Puntuaciones.....	104
4.4.8. Perfil.....	106
4.4.9. Ayuda.....	107
4.5. Sistema recomendador.....	108
4.5.1. Filtrado colaborativo basado en usuarios.....	108
4.5.2. Filtrado basado en contenido.....	110
4.5.3. Modelo híbrido.....	111
4.6. Sistema planificador.....	112
4.7. Backend.....	114
4.7.1. API REST.....	115
4.7.2. Logging.....	115
4.7.3. Seguridad.....	116
4.7.4. Controladores.....	118
4.7.5. Envío de correos.....	119
4.7.6. Cronjobs.....	119
4.8. Despliegue.....	119
4.9. Colaboración.....	121
5. Experimentación y Validación.....	123
5.1. Pruebas de interfaz.....	123
5.2. Pruebas unitarias API.....	127
5.3. Pruebas del recomendador.....	132
5.4. Pruebas del planificador.....	133

6. Marco Regulador.....	137
6.1. Privacidad.....	137
6.2. Ley de Propiedad Intelectual y licencias.....	138
7. Organización del Proyecto.....	141
7.1. Planificación.....	141
7.1.1. Metodología.....	144
7.2. Presupuesto.....	144
7.2.1. Costes de personal.....	144
7.2.2. Costes de materiales y licencias.....	145
7.2.3. Costes totales.....	146
8. Entorno Socio-económico.....	147
9. Conclusiones.....	149
9.1. Trabajos futuros.....	150
Bibliografía.....	152
Anexo.....	165
I. Evaluaciones de usuarios.....	165
II. Resultados pruebas del sistema.....	168
III. Inglés.....	170
1. Introduction.....	171
1.1. Motivation.....	171
1.2. Objectives.....	171
2. Current state of the art.....	172
2.1. Recommender systems.....	172
2.2. Planning systems.....	173
2.3. Academic and commercial projects.....	174
2.3.1. Academic projects.....	174

2.3.2. Commercial projects.....	176
2.4. Technologies.....	177
2.4.1. User Interfaces and frontend.....	177
2.4.2. Backend.....	178
2.4.3. Libraries and additional projects.....	179
2.4.4. Databases.....	180
2.4.5. Security.....	180
2.4.6. Deployments.....	181
3. System description.....	183
3.1. Operational environment.....	183
3.2. Data collection and processing.....	184
3.3. Modules.....	187
3.4. Database model.....	191
3.5. Deployment.....	194
3.6. Contributing.....	195
4. Experimentation and testing.....	196
4.1. User Interfaces testing.....	196
4.2. API unit testing.....	197
4.3. Recommender system testing.....	198
4.4. Planning system testing.....	198
5. Regulatory framework.....	200
5.1. Privacy.....	200
5.2. Intellectual Property Law and licenses.....	201
6. Socio-economical Framework.....	202
7. Conclusions.....	203
7.1. Future work.....	204

ÍNDICE DE FIGURAS

Figura 3.1: Arquitectura de la plataforma Hermes.....	33
Figura 3.2: Estructura de red distribuida de la plataforma Hermes.....	34
Figura 3.3: Inicio de sesión.....	69
Figura 3.4: Proceso de registro.....	69
Figura 3.5: Recomendaciones.....	70
Figura 3.6: Listado de planes.....	71
Figura 3.7: Detalle del plan.....	71
Figura 3.8: Listado de puntuaciones.....	72
Figura 3.9: Perfil de usuario.....	73
Figura 4.1: Esquema relacional de la base de datos.....	89
Figura 4.2: Identificación de zonas con Google My Maps.....	92
Figura 4.3: Ejemplo de cuadrícula con nivel de zoom 13.....	95
Figura 4.4: <i>Landing page</i>	96
Figura 4.5: Inicio de sesión.....	96
Figura 4.6: Registro.....	97
Figura 4.7: Recuperación de contraseña.....	99
Figura 4.8: Recomendaciones.....	99
Figura 4.9: Modal de planificación.....	100
Figura 4.10: Listado de planes.....	101
Figura 4.11: Información del estado de un plan.....	102
Figura 4.12: Detalle del plan (izquierda) y edición del plan (derecha).....	102
Figura 4.13: Vista móvil del plan.....	103
Figura 4.14: Puntuaciones del usuario.....	105
Figura 4.15: Crear puntuación.....	105

Figura 4.16: Perfil de usuario.....	106
Figura 4.17: Ayuda de Hermes.....	107
Figura 4.18: Algoritmo de cálculo de usuarios similares.....	109
Figura 4.19: Algoritmo de generación de recomendaciones mediante filtrado colaborativo.....	110
Figura 4.20: Algoritmo de generación de recomendaciones mediante filtrado basado en contenido.....	111
Figura 4.21: Ejemplos de correos electrónicos.....	119
Figura 4.22: Panel de control de Umami.....	120
Figura 4.23: Página de estado de Hermes.....	121
Figura 5.1: Implementación del tutorial.....	127
Figura 5.2: Iconografía basada en las categorías de un lugar.....	127
Figura 5.3: Buscador de punto de inicio.....	127
Figura 5.4: Mejora de las descripciones de algunos elementos.....	127
Figura 7.1: Diagrama de Gantt del proyecto.....	143
Figure 3.1: Hermes architecture.....	191
Figure 3.2: Hermes' database relational schema.....	193
Figure 3.3: Hermes' distributed architecture.....	195

ÍNDICE DE TABLAS

Tabla 2.1: Ejemplo de recomendación.....	5
Tabla 3.1: Plantilla de Requisitos.....	37
Tabla 3.2: Requisito RU-1.....	37
Tabla 3.3: Requisito RU-2.....	38
Tabla 3.4: Requisito RU-3.....	38
Tabla 3.5: Requisito RU-4.....	38
Tabla 3.6: Requisito RU-5.....	38
Tabla 3.7: Requisito RU-6.....	39
Tabla 3.8: Requisito RU-7.....	39
Tabla 3.9: Requisito RU-8.....	39
Tabla 3.10: Requisito RU-9.....	39
Tabla 3.11: Requisito RU-10.....	40
Tabla 3.12: Requisito RU-11.....	40
Tabla 3.13: Requisito RU-12.....	40
Tabla 3.14: Requisito RU-13.....	40
Tabla 3.15: Requisito RU-14.....	41
Tabla 3.16: Requisito RU-15.....	41
Tabla 3.17: Requisito RF-1.....	41
Tabla 3.18: Requisito RF-2.....	41
Tabla 3.19: Requisito RF-3.....	42
Tabla 3.20: Requisito RF-4.....	42
Tabla 3.21: Requisito RF-5.....	42
Tabla 3.22: Requisito RF-6.....	42
Tabla 3.23: Requisito RF-7.....	43

Tabla 3.24: Requisito RF-8.....	43
Tabla 3.25: Requisito RF-9.....	43
Tabla 3.26: Requisito RF-10.....	43
Tabla 3.27: Requisito RF-11.....	44
Tabla 3.28: Requisito RF-12.....	44
Tabla 3.29: Requisito RF-13.....	44
Tabla 3.30: Requisito RF-14.....	44
Tabla 3.31: Requisito RF-15.....	45
Tabla 3.32: Requisito RF-16.....	45
Tabla 3.33: Requisito RF-17.....	45
Tabla 3.34: Requisito RF-18.....	45
Tabla 3.35: Requisito RF-19.....	46
Tabla 3.36: Requisito RF-20.....	46
Tabla 3.37: Requisito RF-21.....	46
Tabla 3.38: Requisito RF-22.....	46
Tabla 3.39: Requisito RF-23.....	47
Tabla 3.40: Requisito RF-24.....	47
Tabla 3.41: Requisito RF-25.....	47
Tabla 3.42: Requisito RF-26.....	47
Tabla 3.43: Requisito RF-27.....	47
Tabla 3.44: Requisito RF-28.....	48
Tabla 3.45: Requisito RF-29.....	48
Tabla 3.46: Requisito RF-30.....	48
Tabla 3.47: Requisito RF-31.....	48
Tabla 3.48: Requisito RF-32.....	49
Tabla 3.49: Requisito RF-33.....	49

Tabla 3.50: Requisito RF-34.....	49
Tabla 3.51: Requisito RNF-1.....	49
Tabla 3.52: Requisito RNF-2.....	50
Tabla 3.53: Requisito RNF-2.....	50
Tabla 3.54: Requisito RNF-3.....	50
Tabla 3.55: Requisito RNF-4.....	50
Tabla 3.56: Requisito RNF-5.....	50
Tabla 3.57: Requisito RNF-6.....	51
Tabla 3.58: Requisito RNF-7.....	51
Tabla 3.59: Requisito RNF-8.....	51
Tabla 3.60: Requisito RNF-9.....	51
Tabla 3.61: Requisito RNF-10.....	52
Tabla 3.62: Requisito RNF-11.....	52
Tabla 3.63: Requisito RNF-12.....	52
Tabla 3.64: Matriz de trazabilidad, requisitos de usuario y requisitos de software.....	52
Tabla 3.65: Plantilla de Casos de Uso.....	54
Tabla 3.66: CU1: Registro.....	56
Tabla 3.67: CU2: Inicio de sesión.....	57
Tabla 3.68: CU3: Explorar recomendaciones.....	58
Tabla 3.69: CU4: Añadir lugar al plan.....	59
Tabla 3.70: CU5: Crear plan.....	60
Tabla 3.71: CU6: Editar plan.....	61
Tabla 3.72: CU7: Crear nueva puntuación de lugar.....	62
Tabla 3.73: CU8: Cambiar preferencias.....	63
Tabla 3.74: CU9: Cambiar datos personales.....	64
Tabla 3.75: CU10: Cerrar sesión.....	65

Tabla 3.76: CU11: Recuperar contraseña.....	66
Tabla 3.77: CU12: Enviar recordatorio de plan.....	67
Tabla 3.78: CU13: Enviar recordatorio de puntuación.....	68
Tabla 3.79: Plantilla de clases.....	73
Tabla 3.80: Clase AccessLog.....	74
Tabla 3.81: Clase ApplicationLog.....	75
Tabla 3.82: Clase Category.....	75
Tabla 3.83: Clase Distance.....	76
Tabla 3.84: Clase Hour.....	77
Tabla 3.85: Clase Login.....	77
Tabla 3.86: Clase Neighbor.....	78
Tabla 3.87: Clase PasswordRequest.....	78
Tabla 3.88: Clase PlaceCategory.....	79
Tabla 3.89: Clase Place.....	79
Tabla 3.90: Clase PlanItem.....	81
Tabla 3.91: Clase Plan.....	82
Tabla 3.92: Clase PopularTime.....	83
Tabla 3.93: Clase Rating.....	83
Tabla 3.94: Clase Recommendation.....	84
Tabla 3.95: Clase Session.....	84
Tabla 3.96: Clase UserCategory.....	84
Tabla 3.97: Clase UserView.....	85
Tabla 3.98: Clase User.....	85
Tabla 4.1: Categorías.....	90
Tabla 4.2: Zonas identificadas.....	92
Tabla 4.3: Simulación cálculo de vecinos.....	108

Tabla 4.4: Acciones definidas.....	112
Tabla 4.5: Predicados y funciones numéricas definidas.....	113
Tabla 5.1: Cambios extraídos de la evaluación con usuarios.....	126
Tabla 5.2: Pruebas unitarias de la API.....	128
Tabla 5.3: Mejores 5 resultados recomendador.....	133
Tabla 6.1: Software y material utilizado y sus licencias.....	138
Tabla 7.1: Costes de personal.....	145
Tabla 7.2: Coste de materiales y licencias.....	145
Tabla 7.3: Costes totales.....	146
Tabla 1: Evaluación del usuario 1.....	165
Tabla 2: Evaluación del usuario 2.....	165
Tabla 3: Evaluación del usuario 3.....	166
Tabla 4: Evaluación del usuario 4.....	166
Tabla 5: Evaluación del usuario 5.....	167
Tabla 6: Evaluación del usuario 6.....	167
Tabla 7: Resultados pruebas recomendador.....	168
Tabla 8: Resultados pruebas planificador.....	169
Table 3.1: Categories.....	184
Table 3.2: Zones.....	185

LISTA DE ABREVIATURAS

- **AMD:** Advanced Micro Devices
- **API:** Application Programming Interfaces
- **ARM:** Advanced RISC Machines
- **ASP.NET:** Active Server Pages .NET
- **AWS:** Amazon Web Services
- **BFS:** Breadth-First Search
- **CA:** Certificate Authority
- **CAPTCHA:** Completely Automated Public Turing test to tell Computers and Humans Apart
- **CBP:** Cost-Based Planner
- **CBR:** Case-Based Reasoning
- **CD:** Continuous Deployment
- **CDN:** Content Delivery Network
- **CI:** Continuous Intergration
- **CORS:** Cross-origin Resource Sharing
- **CSP:** Content Security Policy
- **CSS:** Cascading Style Sheets
- **DBSCAN:** Density-based spatial clustering of applications with noise
- **DFBB:** Depth-First Branch and Bound
- **DFS:** Depth-First Search
- **DOM:** Document Object Model
- **FD:** Fast Downward
- **FF:** Fast Forward
- **GBFS:** Greedy Best First Search
- **HiLo:** High Visual + Low Functionality
- **HSTS:** HTTP Strict Transport Security
- **HTML:** HyperText Markup Language
- **HTTP:** HyperText Transfer Protocol
- **HTTPS:** Hypertext Transfer Protocol Secure

- **ICAPS:** International Conference on Automated Planning and Scheduling
- **IP:** Internet Protocol
- **IPC:** International Planning Competition
- **IPv4:** Internet Protocol versión 4
- **IPv6:** Internet Protocol versión 6
- **IU:** Interfaz de Usuario
- **JMSD:** Jaccard Mean Squared Difference
- **JS:** JavaScript
- **JSON:** JavaScript Object Notation
- **JSP:** Java Server Pages
- **JSX:** JavaScript XML
- **LPG:** Local search for Planning Graphs
- **LXC:** Linux Containers
- **MPIP:** Modified Proximity, Impact, Popularity
- **MAE:** Mean Absolute Error
- **MSD:** Mean Squared Difference
- **MSE:** Mean Squared Error
- **MVC:** Model-view-controller
- **NHSM:** New Heuristic Similarity Model
- **NVMe:** Non-Volatile Memory Express
- **OPTICS:** Ordering points to identify the clustering structure
- **PDDL:** Planning Domain Definition Language
- **PDF:** Portable Document Format
- **PHP:** PHP: Hypertext Preprocessor
- **PIB:** Producto Interior Bruto
- **PIP:** Proximity, Impact, Popularity
- **RAM:** Random Access Memory
- **REST:** Representational State Transfer
- **RFC:** Request For Comments
- **SABRE:** Semi-Automatic Business Research Environment
- **SATA:** Serial AT Attachment

- **SOAP:** Simple Object Access Protocol
- **SQL:** Structured Query Language
- **SSD:** Solid-state Storage
- **SVD:** Singular Value Decomposition
- **TLS:** Transport Level Security
- **TSP:** Travelling Salesman Problem
- **URL:** Uniform Resource Locator
- **WASM:** WebAssembly

1. INTRODUCCIÓN

A continuación se exponen en este primer capítulo las motivaciones y objetivos que han marcado la realización de este trabajo de fin de grado desde el inicio hasta el final. Posteriormente, se presentará la organización que sigue esta memoria.

1.1. Motivación del Trabajo

El turismo es una industria que supuso, en 2019, el 10.3% PIB global donde 1 de cada 10 empleos está dedicado a esta industria [1]. Además, el turismo es una industria que se podría considerar como “paraguas”, ya que engloba y ayuda a muchos otros sectores como el comercio y la hostelería, generando un gran impacto económico, sociocultural y medioambiental [2], [3].

Por otra parte, el turismo está principalmente movido por la información [4] y por la confianza, donde millones de personas deciden sus visitas e itinerarios de viaje basados en recomendaciones de otras personas y comentarios en internet presentes en internet [5].

Es por ello, que la industria siempre ha estado abierta a nuevas tecnologías. Desde los sistemas de reserva de las compañías aéreas, como el Magnetronic Reservisor [6] o SABRE [7], introducidos en 1956 y en la década de 1960 respectivamente, hasta las plataformas de reservas de hoteles como Expedia y Booking.com o los buscadores de experiencias culturales como Civitatis, han permitido a los viajeros mejorar su experiencia a la hora de realizar turismo a lo largo de todo el planeta.

Sin embargo, hay un aspecto del proceso de viaje que todavía requiere de trabajo manual o semi-manual por parte de una persona que se corresponde con la búsqueda y planificación de sus visitas. Aunque existen servicios por parte de terceras personas que se encargan de realizar estas tareas [8], [9], no todo el mundo se encuentra cómodo utilizando estos servicios, ya sea por motivos económicos o incluso de privacidad.

Similarmente, aunque existen servicios de búsqueda de actividades y puntos de interés (como por ejemplo, Civitatis o Tripadvisor) y aplicaciones de planificación gratuitas (como por ejemplo, Tour o Roadtrippers) que facilitan esta tarea al usuario, estas, a menudo, no son intercompatibles entre sí, debido a que las aplicaciones no están disponibles en todas las plataformas móviles o no implementan funcionalidad para compartir o exportar datos. Además, las recomendaciones que realizan suelen estar basadas en promociones o según su popularidad en vez de ser individualizadas y personalizadas para cada usuario y pueden requerir de una búsqueda previa por parte del usuario.

1.2. Objetivos

En este trabajo se plantea crear una plataforma gratuita y de código abierto, con un sistema personalizado que permita a cada usuario planificar sus viajes, a través de una aplicación web, y que respete la privacidad de los usuarios.

Es por ello que el principal objetivo de este trabajo está dividido en dos. Por un lado, se presenta el desarrollo de un sistema de recomendación de puntos de interés que se base en información personal proporcionada por un usuario, así como de puntuaciones generadas por otros usuarios. Por otro lado, se presenta el desarrollo de un sistema de planificación, interconectado con el sistema de recomendación anteriormente mencionado, que permita al usuario planificar de forma automática su viaje en función de unos determinados parámetros.

No obstante, aunque la planificación se presenta como automática, el usuario podrá modificarla a su gusto en cualquier momento.

Dado que nos encontramos ante un objetivo muy ambicioso, en este trabajo se presenta un proyecto de plataforma focalizado en la ciudad de Madrid. Así, una vez realizado, el siguiente objetivo a proponer es la escalabilidad y adaptación de la plataforma a una mayor cantidad de ciudades.

También se plantean los siguientes subobjetivos:

1. Creación de un sistema de puntuaciones en el que cualquier usuario registrado pueda participar.
2. Creación de un sistema de recogida y tratamiento de datos de carácter personal.
3. Consiste en la creación de una base de datos con información de usuarios y puntos de interés que puedan ser usados por todos los componentes del sistema.
4. Creación y estudio de una interfaz que sea fácil de utilizar por cualquier usuario.

Con la implementación de este proyecto se desea adquirir conocimientos sobre algoritmos de recomendación y planificación automática, y toda su tecnología implícita, así como el tratamiento de forma segura de datos y el diseño de interfaces de usuario que sean usables y sencillas de usar.

Por último, para este proyecto se plantea el uso de tecnologías y herramientas de código abierto en su mayor medida posible.

1.3. Organización del documento

A continuación se resume la organización de esta memoria. El documento se compone de 9 capítulos con un anexo y una bibliografía:

1. **Introducción:** En este capítulo se recogen las motivaciones y objetivos que sigue este trabajo. Además, se incluye la organización de este documento.
2. **Estado del Arte:** En este capítulo se describen las principales ideas útiles para este trabajo y se realiza un análisis de los distintos tipos de metodologías y algoritmos para un sistema de recomendación y planificación. Además, se estudian posibles herramientas y plataformas existentes en la literatura y de forma comercial.
3. **Análisis del sistema:** En este capítulo se presentan el diseño y el análisis de una posible implementación. Así mismo, se describen los requisitos que debe seguir el sistema desarrollado y posibles casos de uso.
4. **Implementación:** En este capítulo se detalla el proceso que se ha seguido para la implementación de la plataforma, así como la descripción de las herramientas y servicios utilizados para su desarrollo y el proceso de obtención de datos.
5. **Experimentación y validación:** En este capítulo se presentan los experimentos y validaciones realizados para comprobar la utilidad de la plataforma, utilizando para ello datos de prueba y usuarios reales.
6. **Marco regulador:** En este capítulo se expondrán normativas, leyes y estándares que afectan a la realización de este proyecto.
7. **Organización del proyecto:** En este capítulo se describe la planificación, los presupuestos, el coste y el tiempo de desarrollo del proyecto.
8. **Entorno socio-económico:** En este capítulo se describe el posible impacto económico y social para la industria del turismo.
9. **Conclusiones:** Finalmente, en este capítulo se recogen ideas y comentarios sobre el proyecto. Además, se plantean posibles futuras mejoras.

2. ESTADO DEL ARTE

En este capítulo se presenta la tarea de investigación y estudio realizada sobre los sistemas recomendadores, sistemas planificadores, tecnologías y el marco actual que han permitido recopilar ideas e información para el desarrollo del proyecto.

2.1. Descripción de un sistema recomendador

Un sistema recomendador es un programa informático cuyo objetivo es recomendar elementos, como por ejemplo, puntos de interés, que sean de gran interés para un usuario.

2.1.1. Metodologías de recomendación

A la hora de realizar recomendaciones, nos encontramos distintas metodologías, Se describen a continuación, las más comunes.

- **Filtrado colaborativo:** Los sistemas de recomendación basados en filtrado colaborativo se fundamentan, principalmente, en las puntuaciones de distintos elementos generadas por múltiples usuarios. Como la base de datos de los elementos es muy grande, existirán muchos “huecos” de puntuaciones sin llenar. De esta manera el sistema busca llenar los “huecos”, en donde las predicciones para estos con mayor puntuación serán los recomendados al usuario [10].

Por ejemplo, se consideran tres usuarios, Alice, Bob y Carol. Alice y Bob tienen gustos muy similares, mientras que Alice y Carol no. Si las puntuaciones que han realizado Alice y Bob sobre sitios comunes son muy similares, un algoritmo será capaz de determinar un valor alto de similitud entre Alice y Bob, mientras que si las puntuaciones de Carol son muy distintas a las de Alice, el mismo algoritmo será capaz de determinar un valor muy bajo de similitud entre los dos.

Existen dos aproximaciones para construir sistemas con esta metodología de recomendación [10]:

- **Métodos basados en memoria:** Este método es el más fácil de implementar. Se basa en las puntuaciones realizadas por los usuarios. Se puede diferenciar en:
 - *Filtrado colaborativo basado en usuarios:* Este método se basa en determinar una puntuación faltante a partir otros usuarios, en donde usuarios con alta correlación tendrán mayor peso a la hora de determinar

la recomendación. Se podría considerar que este tipo de sistema recomendador considera que “usuarios similares tienen gustos similares”.

Siguiendo el ejemplo anterior, Alice quiere recibir una nueva recomendación de un lugar que no haya visitado.

TABLA 2.1: EJEMPLO DE RECOMENDACIÓN

Usuario	Lugar 1	Lugar 2	Lugar 3	Lugar 4	Lugar 5
Alice	4	2	5	-	-
Bob	4	1	4	1	5
Charlie	3	4	2	5	2

Teniendo en cuenta la Tabla 1, el sistema recomendará a Alice el *Lugar 5*, ya que, como se comentaba, Alice y Bob son muy similares y el *Lugar 5* tiene una mayor puntuación marcada por Bob que el *Lugar 4*.

- *Filtrado colaborativo basado en ítems:* Este método se basa en determinar una puntuación faltante a partir de las similitudes entre elementos. Se podría considerar que este tipo de sistema recomendador considera que “elementos similares tendrán puntuaciones similares”.

Por ejemplo y siguiendo la Tabla 1 del ejemplo anterior, Lugar 1, Lugar 3 y Lugar 5 son muy similares entre sí, mientras que Lugar 2 y Lugar 4 son similares entre ellos dos. Como Alice ha puntuado positivamente Lugar 1 y Lugar 3, es probable que Lugar 5 reciba también una puntuación positiva, mientras que Lugar 4 es posible que reciba una puntuación negativa, ya que Lugar 2 y Lugar 4 han recibido una negativa.

- **Métodos basados en modelos:** Este método es complejo de implementar. Se basa en la creación de modelos, utilizando herramientas de aprendizaje automático y minería de datos que extraen el contexto de un elemento.

En este tipo de aproximación, es común el uso de árboles de decisión o de una serie de reglas [11]. En el caso de que el sistema haga uso de reglas, estas se determinan antes de la implementación del sistema. Estas reglas son definidas por expertos que establecen que puntuación se debe asignar a los lugares del usuario con puntuación pendiente en base a unas condiciones.

Por otra parte, los árboles de decisión, son similares. El sistema genera periódicamente árboles de decisión a partir de los datos que se encuentran en la base de datos. Estos árboles, con contienen condiciones, son usados para determinar de nuevo la puntuación faltante de los lugares del usuario.

Sin embargo, con este tipo de metodología, existen distintos problemas a la hora de realizar las recomendaciones:

- **Dispersión o arranque en frío:** Este es uno de los principales problemas de los sistemas recomendadores. Inicialmente, un sistema recomendador tiene muy poca cantidad de puntuaciones con las que comenzar, por lo que es complicado buscar usuarios o elementos similares con los que poder realizar las recomendaciones [10], [12].

Una posible solución se encuentra en la obtención de preferencias mediante *Active learning*, que consiste en que un agente obtiene información adicional del mundo de forma explícita (por ejemplo, preguntando al usuario) o implícita (por ejemplo, grabando el tipo de elementos que el usuario visita), de manera que obtenga información útil que pueda usar para realizar recomendaciones [13], [14].

Otra posible solución aparece con los sistemas híbridos, que se describirán más adelante.

- **Escalabilidad:** Este problema surge con la capacidad limitada de computación y la gran cantidad de datos disponibles. Ya que este tipo de metodología no puede trabajar con usuarios de forma independiente, limitar el número de datos de un método puede hacer que sea incapaz de generar resultados óptimos [15].

Como posibles soluciones aparece el uso de técnicas de reducción de dimensionalidad, como el método de descomposición de valores singulares (SVD) [16].

- **Shilling Attacks:** Este tipo de problema ocurre cuando muchos usuarios puntúan un elemento de forma engañosa. Un ejemplo de ello ocurre cuando una empresa con un producto poco fiable paga para que múltiples usuarios realicen reseñas positivas, creando así reseñas engañosas y ocultando reseñas reales [17]. Este es un problema muy común en grandes plataformas, como Amazon [18]–[20], que puede provocar que un sistema recomendador no sea preciso y relevante para un usuario.

Una posible solución ante este problema podría ser el uso de métricas. A partir de estas, es posible extraer patrones que permiten detectar este tipo de ataques [21]. Un ejemplo de ello puede aparecer cuando para un determinado elemento, la media de puntuaciones crece más de lo habitual en un determinado período de tiempo. Otras posibles soluciones se basan en el análisis periódico de centroides, donde, si un determinado grupo de usuarios provocan cambios significativos en los centroides de los clústeres de la base de datos, estos son categorizados como maliciosos y aislados de los datos [17].

- **Filtrado basado en contenido:** Los sistemas que utilizan filtrado basado en contenido recomiendan en función de los modelos generados para cada usuario

que surgen de los intereses de este (por ejemplo, según sus compras y visitas) y de las características de los elementos [10].

Por ejemplo, si Alice ha visitado Lugar 1 y Lugar 3, es posible que quiera visitar Lugar 5, ya que los tres lugares tienen categorías y características en común.

Aunque esta metodología no hace uso de las puntuaciones y del conocimiento colaborativo de otros usuarios, evitando los problemas anteriores, sí que presenta otra serie de problemas a destacar:

- **Diversidad:** En muchos casos, esta metodología genera recomendaciones que pueden ser obvias [10]. Debido a que estos sistemas se basan en las características de los elementos, es muy difícil que un elemento con características distintas a las modeladas para el usuario sea presentado como recomendación, reduciendo la diversidad.
- **Arranque en frío:** Al igual que en el filtrado colaborativo, los recomendadores basados en contenido necesitan información del usuario, tal como sus visitas o compras [22],[12].
- **Sistemas híbridos:** Los sistemas híbridos pretenden explotar distintas fuentes de información para realizar sus recomendaciones, reduciendo así los problemas que se presentan para las distintas metodologías [23]. Un ejemplo muy común aparece con la unión de los sistemas de filtrado colaborativo y basado en contenido [24].

2.1.2. Algoritmos de recomendación

En la práctica, existen distintos algoritmos que pueden ser usados para realizar recomendaciones.

- **k-NN:** Es un algoritmo desarrollado en 1951 por Evelyn Fix y Joseph Hodges basado en vecindarios con k vecinos más cercanos, ya sea para clasificación o regresión [25]. En el caso de clasificación, dada una entrada, esta se clasifica en la clase a la que pertenezcan la mayoría de sus vecinos, mientras que en regresión, el valor de salida para la entrada se determina con la media de sus vecinos.

En el caso de los sistemas recomendadores, k-NN puede ser usado para predecir el valor de una puntuación faltante de un usuario, o para clasificar usuarios en distintos grupos o clústeres.

- **k-medias:** Es un algoritmo cuyo objetivo es dividir un conjunto de elementos en k clústeres distintos, mediante el cálculo de centroides que permiten dividir el espacio de elementos en k conjuntos [26], [27].

Al igual que con k-NN, k-medias es usado para clasificar usuarios en distintos grupos o clústeres en donde los usuarios que pertenecen a ellos son similares entre sí.

- **DBSCAN y OPTICS:** DBSCAN es un algoritmo de clusterización, similar a k-medias, que permite clasificar a los usuarios en distintos grupos o clústeres. A diferencia de k-medias, que requiere de los datos en bruto, DBSCAN hace uso de métricas entre elementos [28], por lo que se puede usar con medidas de similitud como las que se muestran en el Capítulo 2.1.3. OPTICS es un algoritmo basado en DBSCAN que aborda uno de sus problemas principales, la detección de clusters en datos con densidad variable [29].
- **Aprendizaje por refuerzo:** Es una técnica de aprendizaje automático que busca que un agente aprenda por sí solo a realizar una tarea [30]. Se hace uso de Procesos de Decisiones de Markov para modelar los posibles estados y acciones del agente y se recompensa o penaliza al agente en función del resultado obtenido por la acción. En el caso de los sistemas recomendadores, el aprendizaje por refuerzo es una aplicación novedosa y reciente.

2.1.3. Medidas de similitud

Existen distintas medidas para medir la similitud entre dos usuarios.

- **Distancia euclídea:** Es la medida más sencilla de implementar. En este método, se representa a los usuarios como coordenadas. Es recomendable cuando los datos no son escasos y la magnitud de los valores de las puntuaciones son significativas [31].

$$\text{similarity}(a, b)_{\text{Euclidean}} = \sqrt{\sum_{p \in I} (r_b^p - r_a^p)^2} \quad (1)$$

donde I es el conjunto de elementos comunes al usuario a y b y r_u^p es la puntuación del usuario u para el elemento p .

- **Coeficiente de correlación de Pearson:** Este coeficiente mide el grado de relación entre dos variables, donde el valor obtenido varía entre -1 y 1 [32]. Sin embargo, este coeficiente a veces presenta problemas debido a que no tiene en cuenta que, a menudo, las puntuaciones son valores absolutos entre rangos positivos. Es una de las medidas más utilizadas.

$$\text{similarity}(a, b)_{\text{PCC}} = \frac{\sum_{p \in I} (r_a^p - \bar{r}_a)(r_b^p - \bar{r}_b)}{\sqrt{\sum_{p \in I} (r_a^p - \bar{r}_a)^2} \cdot \sqrt{\sum_{p \in I} (r_b^p - \bar{r}_b)^2}} \quad (2) [33]$$

donde \bar{r}_u es la media de todas las puntuaciones del usuario u .

- **Similitud por coseno:** Esta medida representa a cada uno de las preferencias de un usuario como vectores, de manera que el ángulo entre ellos sea el que determine la similitud entre ambos vectores, donde valores cercanos a 0 indicarán que los usuarios son similares [34]. Junto al coeficiente de correlación de Pearson, es una de las más utilizadas.

$$\text{similarity}(a, b)_{\cos} = \frac{\vec{r}_a \cdot \vec{r}_b}{\|\vec{r}_a\| \cdot \|\vec{r}_b\|} \quad (3) [33]$$

donde \vec{r}_u es el vector de puntuaciones del usuario u y $\|\vec{r}_u\|$ es la magnitud del vector anterior.

- **Coeficiente de correlación de Pearson con restricciones:** Esta medida es una variante del Coeficiente de Correlación de Pearson, salvo qué este se define de manera que se tenga en cuenta que las puntuaciones son valores absolutos, solucionando el problema que se presenta en el coeficiente de correlación de Pearson [33].

$$\text{similarity}(a, b)_{CPCC} = \frac{\sum_{p \in I} (r_a^p - r_{med})(r_b^p - r_{med})}{\sqrt{\sum_{p \in I} (r_a^p - r_{med})^2} \cdot \sqrt{\sum_{p \in I} (r_b^p - r_{med})^2}} \quad (4) [28].$$

donde r_{med} es la mediana de la escala de puntuación. Por ejemplo, en una escala de 1 al 5, la mediana es 3.

- **Coeficiente de Jaccard:** Esta medida se define como el ratio entre la intersección y la unión de dos conjuntos. En ella, solamente se tiene en cuenta el tamaño del conjunto de los elementos comunes entre dos usuarios, ignorando las puntuaciones de cada uno [35].

$$\text{similarity}(a, b)_{Jaccard} = \frac{|I_a \cap I_b|}{|I_a \cup I_b|} \quad (5) [28]$$

donde I_u es el conjunto de todas las puntuaciones del usuario u

- **Mean Squared Difference (MSD):** Esta medida tiene en cuenta un conjunto de elementos, sin embargo, solamente tiene en cuenta los valores absolutos de las puntuaciones, pero no los elementos en común [33].

$$\text{similarity}(a, b)_{MSD} = 1 - \frac{\sum_{p \in I} (r_a^p - r_b^p)^2}{|I|} \quad (6) [28]$$

- **Jaccard Mean Squared Difference (JMSD):** El coeficiente de Jaccard y MSD se pueden combinar bajo una medida que intenta solucionar los problemas de las dos medidas anteriores [33].

$$\text{similarity}(a, b)_{JMSD} = \text{similarity}(a, b)_{Jaccard} \cdot \text{similarity}(a, b)_{MSD} \quad (7) [28]$$

Por otra parte, se han presentado recientemente varias medidas nuevas de similitud, de las cuales algunas se muestran a continuación:

- **Proximity, Impact, Popularity (PIP):** En 2008 se presenta la medida PIP [36]. En su desarrollo, los autores exponen 3 requisitos que se deberían tener en cuenta:
 1. La medida debería utilizar significativos de datos específicos del dominio.
 2. La medida debería permitir una fácil conexión con los sistemas de CF existentes, sustituyendo únicamente las medidas de similitud de los sistemas, sin requerir una gran re-implementación o una recopilación de datos adicional.
 3. La medida no solo debería mostrar mejores resultados en condiciones de arranque en frío, sino también resultados comparables a otras medidas populares en condiciones de no arranque en frío.

Para ello, se introducen 3 factores:

- **Proximity:** El factor de proximidad determina, no solo la diferencia entre las puntuaciones de dos usuarios, sino también como de acuerdo están estas dos puntuaciones. Por ejemplo, dados un usuario a y dos usuarios b y c con los que buscar similitud y una puntuación r , donde $r_a = 3$, $r_b = 5$ y $r_c = 2$. Aunque $|r_a - r_b| = 2 = |r_a - r_c|$, el usuario c es más próximo a a , debido a que las puntuaciones de a y c son neutra y positiva respectivamente frente a las puntuaciones de a y b , que son neutra y negativa respectivamente.

- $D(r_a^p, r_b^p) = |r_a^p - r_b^p|$ sí $Agreement(r_a^p, r_b^p)$ es **true**
 - $D(r_a^p, r_b^p) = 2 \cdot |r_a^p - r_b^p|$ sí $Agreement(r_a^p, r_b^p)$ es **false**
- $$Proximity(r_a^p, r_b^p) = ((2 \cdot (r_{max} - r_{min}) + 1) - D(r_a^p, r_b^p))^2 \quad (8)$$

donde r_{max} y r_{min} son los valores máximos y mínimos de la escala de puntuación y

$$\begin{aligned} Agreement(r_a^p, r_b^p) &= \text{false} \text{ si } r_a^p > r_{med} \text{ y } r_b^p > r_{med} \text{ o } r_a^p < r_{med} \text{ y } r_b^p < r_{med} \text{ y} \\ Agreement(r_a^p, r_b^p) &= \text{true} \text{ en cualquier otro caso.} \end{aligned} \quad (9)$$

- **Impact:** El factor de impacto determina el grado de preferencia de los usuarios para un elemento, priorizando cuando tienen puntuaciones positivas.
 - $Impact(r_a^p, r_b^p) = (|r_a^p - r_{med}| + 1)(|r_b^p - r_{med}| + 1)$ sí $Agreement(r_a^p, r_b^p)$ es **true**

- $Impact(r_a^p, r_b^p) = \frac{1}{(|r_a^p - r_{med}| + 1)(|r_b^p - r_{med}| + 1)}$ si $Agreement(r_a^p, r_b^p)$ es **false** (10)
- **Popularity:** El factor de popularidad determina la distancia entre los dos usuarios a comparar con la puntuación media de un elemento, ya que una diferencia alta con la media por parte de los dos usuarios muestra mayor relación entre ellos.
 - $Popularity(r_a^p, r_b^p) = 1 + (\frac{r_a^p + r_b^p}{2} + \mu_p)$ si $r_a^p > \mu_p$ y $r_b^p > \mu_p$ o $r_a^p < \mu_p$ y $r_b^p < \mu_p$
 - $Popularity(r_a^p, r_b^p) = 1$ en cualquier otro caso

Donde μ_p es la media global del elemento p . (11)

Por lo tanto, la medida se formaliza como:

$$similarity(a, b)_{PIP} = \sum_{p \in I} (Proximity(r_a^p, r_b^p) \cdot Impact(r_a^p, r_b^p) \cdot Popularity(r_a^p, r_b^p)) \quad (12)$$

- **TrustWalker Sigmoid Similarity Measure:** En 2009 se presenta TrustWalker [37], un sistema recomendador basado en filtrado colaborativo basado en ítems. En el, se define la siguiente medida de similaridad para dos ítems, en donde se usa la función sigmoide. Esta función sirve principalmente para evitar que se favorezca una lista $UC_{p,q}$ antes que otra en función de su tamaño.

$$similarity(p, q)_{Sigmoid} = \frac{1}{1 + \exp\left(\frac{-|UC_{p,q}|}{2}\right)} \cdot corr(p, q) \quad (13)$$

donde $corr(p, q)$ se define como:

$$corr(p, q) = \frac{\sum_{u \in UC_{p,q}} (r_u^p - \bar{r}_u)(r_u^q - \bar{r}_u)}{\sqrt{\sum_{p \in UC_{p,q}} (r_u^p - \bar{r}_u)^2} \cdot \sqrt{\sum_{p \in UC_{p,q}} (r_u^q - \bar{r}_u)^2}} \quad (14)$$

y donde $UC_{p,q}$ es el conjunto de usuarios que han puntuado el elemento p y q .

- **New Heuristic Similarity Model (NHSM):** La medida de similitud NHSM surge en 2014 [33] con el objetivo de solucionar los problemas de las medidas anteriores. En esta medida de similitud se tienen en cuenta 3 medidas adicionales:

- **Proximity, Significance, Singularity:** Esta medida tiene 3 factores:

$$similarity(a, b)_{PSS} = \sum_{p \in I} (Proximity(r_a^p, r_b^p) \cdot Significance(r_a^p, r_b^p) \cdot Singularity(r_a^p, r_b^p)) \quad (15)$$

- **Proximity:** El factor de proximidad determina, no solo la diferencia entre las puntuaciones de dos usuarios, sino también como de acuerdo están estas dos puntuaciones. Por ejemplo, dados un usuario a y dos usuarios b y c con los que buscar similitud y una puntuación r , donde $r_a = 3$, $r_b = 5$ y $r_c = 2$. Aunque $|r_a - r_b| = 2 = |r_a - r_c|$, el usuario c es más próximo a a , debido a que las puntuaciones de a y c son neutra y positiva respectivamente frente a las puntuaciones de a y b , que son neutra y negativa respectivamente.

$$Proximity(r_a^p, r_b^p) = 1 - \frac{1}{\exp(-|r_a^p - r_b^p|)} \quad (16)$$

- **Significance:** El factor de importancia determina como de distantes son las calificaciones con respecto a la mediana, priorizando las calificaciones que se alejen de ella.

$$Significance(r_a^p, r_b^p) = \frac{1}{1 + \exp(-|r_a^p - r_{med}| \cdot |r_b^p - r_{med}|)} \quad (17)$$

- **Singularity:** El factor de singularidad determina como de diferentes son las calificaciones con respecto al resto de calificaciones del mismo elemento.

$$Singularity(r_a^p, r_b^p) = 1 - \frac{1}{1 + \exp(-|\frac{r_a^p + r_b^p}{2} - \mu_p|)} \quad (18)$$

- **Coeficiente de Jaccard Modificado:** Esta medida se basa en el coeficiente de Jaccard original. A diferencia de este, esta medida consiste en el ratio entre la intersección y el producto de los dos conjuntos de puntuaciones.

$$similarity(a, b)_{Jaccard} = \frac{|I_a \cap I_b|}{|I_a| \times |I_b|} \quad (19)$$

- **User Rating Preference:** Esta medida sirve para tener en cuenta las preferencias de los usuarios a la hora de puntuar. Por ejemplo, hay muchos usuarios que tienden a puntuar a la alta, mientras que hay otros que tienden a puntuar a la baja. Con esta medida, se tienen en cuenta estos casos.

$$similarity(a, b)_{URP} = 1 - \frac{1}{1 + \exp(-|\mu_a - \mu_b| |\sigma_a - \sigma_b|)} \quad (20)$$

$$\sigma_u = \sqrt{\frac{\sum_{p \in I_u} (r_u^p - \bar{r}_u)^2}{|I_u|}} \quad (21)$$

donde μ_u es la media del usuario u en el conjunto I común entre ambos usuarios.

Uniendo las fórmulas (8), (12) y (13), la medida de NHSM se formaliza como:

$$similarity(a, b)_{NHSM} = similarity(a, b)_{PSS} \cdot similarity(a, b)_{Jaccard} \cdot similarity(a, b)_{URP} \quad (22)$$

- **Modified Proximity, Impact, Popularity (MPIP):** En agosto de 2020 se presenta la medida MPIP [38], como mejora de la medida PIP. Principalmente, propone cambios de manera que se reduzca el intervalo de salida a $(0, 1)$. Para ello, se modifican los 3 factores de la siguiente manera:

- **Proximity**

$$D(r_a^p, r_b^p) = |r_a^p - r_b^p|$$

- $Proximity_M(r_a^p, r_b^p) = \left(\frac{D(r_a^p, r_b^p) - \frac{r_{med}^p + r_{med}^n}{2}}{r_{max} - r_{min}} \right)^2$ si $Agreement(r_a^p, r_b^p)$ es **true**

- $Proximity_M(r_a^p, r_b^p) = \delta \left(\frac{D(r_a^p, r_b^p)}{r_{max} - r_{min}} \right)^2$ si $Agreement(r_a^p, r_b^p)$ es **false**

donde r_{med}^p y r_{med}^n es la mediana positiva y negativa respectivamente, es decir, r_{med}^p es la mediana entre r_{med} y r_{max} y r_{med}^n es la mediana entre r_{med} y r_{min} y

- $\delta = 0.75$ si $D(r_a^p, r_b^p) > r_{med}$
- $\delta = 0.5$ si $D(r_a^p, r_b^p) = r_{med}$
- $\delta = 0.25$ en caso contrario (24)

- **Impact**

- $Impact_M(r_a^p, r_b^p) = \exp\left(\frac{-1}{(|r_a^p - r_{med}|+1)(|r_b^p - r_{med}|+1)}\right)$ sí $Agreement(r_a^p, r_b^p)$ es **true**
- $Impact_M(r_a^p, r_b^p) = \frac{1}{(|r_a^p - r_{med}|+1)(|r_b^p - r_{med}|+1)}$ sí $Agreement(r_a^p, r_b^p)$ es **false** (25)
- **Popularity:**
 - $Popularity_M(r_a^p, r_b^p) = \log_{10}\left(2 + \left(\frac{r_a^p + r_b^p}{2} + \mu_p\right)^2\right)$ sí $r_a^p > \mu_p$ y $r_b^p > \mu_p$ o $r_a^p < \mu_p$ y $r_b^p < \mu_p$
 - $Popularity_M(r_a^p, r_b^p) = 0.3010$ en cualquier otro caso

Donde μ_p es la media global del elemento p . (26)

Por lo tanto, la medida se formaliza como:

$$similarity(a, b)_{PIP} = \sum_{p \in I} (Proximity(r_a^p, r_b^p) \cdot Impact(r_a^p, r_b^p) \cdot Popularity(r_a^p, r_b^p)) \quad (27)$$

2.2. Descripción de un sistema planificador

Un sistema planificador es un programa informático cuyo objetivo es determinar un plan que consiste en una serie de acciones para alcanzar un objetivo desde un estado inicial.

ICAPS (International Conference on Automated Planning and Scheduling) es un foro mundial académico cuyos objetivos principales consisten en promover el campo de la planificación y programación automática mediante reuniones, conferencias y competiciones [39]. ICAPS es además la encargada de organizar IPC (International Planning Competition), una competición en la que múltiples equipos presentan algoritmos y herramientas de planificación automática [40].

Esta competición está compuesta por una serie de dominios de planificación y su correspondiente conjunto de problemas. Con estos problemas, cada herramienta de cada equipo participante es evaluada en función del tiempo de planificación y de la calidad del plan generado. Además, estos problemas están definidos en el lenguaje PDDL (Planning Domain Definition Language), creado en 1998 para la competición IPC de 1999/2000 [41]. Este lenguaje es un intento de estandarización de todos los lenguajes de modelado de problemas de planificación existentes hasta la fecha.

PDDL está construido siguiendo una lógica de predicados [42]. Un dominio escrito en PDDL está compuesto por una serie de objetos y predicados, que permiten definir un

estado concreto del problema, así como de una serie de reglas a seguir que permiten moverse entre estados [42]. Por otra parte, un problema define una serie de instancias de objetos y predicados, es decir, un estado inicial, y unos objetivos que se deben cumplir y para considerar el problema como resuelto [42]. Adicionalmente se pueden incluir numerales [43], que permiten realizar cálculos matemáticos e incluso servir como métrica que el planificador debe minimizar para encontrar una solución óptima.

El uso de PDDL ha conseguido que la investigación en el ámbito de la planificación automática sea más sencillo, accesible y comparable [44].

2.2.1. Algoritmos de planificación

La idea de que los buenos algoritmos eran necesariamente específicos de un dominio quedó desmentida cuando las heurísticas h^{add} y h^{max} , independientes del dominio, obtuvieron excelentes resultados en IPC de 1998 [45]. Esto provocó una gran aumento en investigaciones sobre la planificación independiente del dominio. Además, esta investigación en planificación automática es la que sigue dominando, ya que supone una gran dificultad encontrar un planificador que funcione en todo tipo de casos [46].

Es habitual que este tipo de problemas se representen como sistemas de estado-transición. PDDL usa este sistema. Esta representación, también llamada dominio de planificación clásica [45], consiste de una tupla $\Sigma = (S, A, \gamma, \text{coste})$ donde

- S es un conjunto finito de estado del sistema.
- A es un conjunto finito de acciones que se pueden realizar.
- γ es una función que determina, desde un estado, qué acciones se pueden realizar.
- coste es una función que determina el coste, ya sea en tiempo o arbitrario, al realizar una acción desde un estado.

Es en este tipo de problemas en el que se centra la planificación clásica. La mayoría de planificadores clásicos actuales se basan en realizar búsquedas heurísticas hacia delante en el espacio de estados, utilizando heurísticas independientes de dominio [45].

A menudo, los planificadores clásicos hacen uso del algoritmo **GBFS** (Greedy Best First Search), debido a que, generalmente, las soluciones que no son óptimas se consideran aceptables [45], lo que permite generar soluciones de forma más rápida y con menor coste espacial. Este algoritmo tiene un funcionamiento similar a A*, donde la función de coste es igual a la función heurística.

Asimismo, este tipo de búsqueda también considera algoritmos como:

- **BFS** (Breadth-First Search) [45] es un algoritmo que recorre un grafo, explorando primero todos los nodos vecinos antes de pasar a subsecuentes niveles de profundidad, sin tener en cuenta el coste. En problemas de planificación, BFS siempre terminará y devolverá una solución si existe. Esta solución puede ser la mas corta, aunque no tiene porqué ser la más optima en coste.
- **DFS** (Depth-First Search) [45] es un algoritmo que, al contrario que BFS, recorre un grafo pasando primero a subsecuentes niveles de profundidad antes que a los nodos vecinos, sin tener en cuenta el coste. Al igual que BFS, en problemas de planificación, DFS siempre terminará y devolverá una solución si existe. Esta solución puede ser la mas corta, aunque no tiene porqué ser la más optima en coste.
- **A*** [45] es un algoritmo similar a BFS, sin embargo, a diferencia de BFS, tienen en cuenta el coste y hace uso de una función heurística para la búsqueda. Además, A* posee una serie de ventajas, como que es un algoritmo completo, es decir, que siempre devuelve solución si existe; y optimo, ya que si la heurística es admisible, la solución encontrada será optima. Sin embargo, presenta una necesidad de recursos en espacio bastante grande.
- **DFBB** (Depth-First Branch and Bound) [45] es un algoritmo basado en DFS. Su funcionamiento es similar, salvo que, a diferencia de DFS, a pesar de haber encontrado una solución, DFBB continua con otras ramas encontradas durante la búsqueda, en busca de una posible mejor solución.

Al contrario que la búsqueda hacia delante, también existe la búsqueda hacia atrás. Esta comienza por las metas y avanza, hacia atrás, buscando las acciones que se pueden aplicar para conseguir la metas deseada [45].

Una función heurística es una función que evalúa alternativas en los algoritmos de búsqueda en cada paso, basándose en la información disponible para decidir qué rama seguir. Un ejemplo de funciones, anteriormente mencionadas son h^{add} y h^{max} , funciones que aproximan el coste de alcanzar las metas de forma recursiva, relajando los borrados de las acciones [45].

Además de la planificación clásica, existen también el campo de investigación de planificación probabilística. Estos enfoques se basan principalmente en métodos de optimización con programación dinámica y en procesos de decisión de Markov, y son usados en problemas de planificación no deterministas [45].

También es posible el uso de algoritmos genéticos en la tarea de planificación. Los algoritmos genéticos son metaheurísticas inspiradas por los procesos de selección natural. Son comúnmente usados en problemas de optimización y búsqueda, como la planificación automática. Un ejemplo de aplicación es AgPlan [47], un planificador que

hace uso de un algoritmo genético y que requiere de problemas definidos mediante el lenguaje STRIPS, similar a PDDL.

2.3. Marco actual

En este apartado se presentan sistemas similares al trabajo planteado, que han sido desarrollados en el ámbito académico o que están implementados de forma comercial.

2.3.1. Sistemas existentes académicos

Se pueden encontrar varios sistemas de recomendación y/o planificación en la literatura académica.

SAMAP [48] es un sistema desarrollado por investigadores de la Universidad Carlos III de Madrid, en conjunto con la Universidad de Granada, la Universidad Autónoma de Barcelona, la Universidad Politécnica de Valencia y la UNED, publicado en 2008. Su objetivo es ayudar a las personas a visitar ciudades. Está compuesto por cuatro elementos, una ontología; un agente de usuario, es decir, una interfaz que permite al usuario interactuar con el sistema; un agente de recomendación, basado en Case-Based Reasoning (CBR) y un agente de planificación.

En la ontología, se recoge y se modela información sobre los lugares y el usuario. Esta incluye información sobre las ciudades, datos personales, lugares visitados y preferencias. Por otra parte, el método CBR hace uso del algoritmo k-NN. Este método, consiste en listar los k usuarios más similares al turista T , en función de los lugares visitados y las preferencias, que hayan visitado la ciudad que T quiere visitar. Posteriormente, se genera una lista de lugares posibles a visitar que al menos hayan sido visitados por uno de los usuarios k .

Una vez listados los posibles sitios, el agente de planificación usa éstos como entrada, además de las preferencias del usuario, los horarios de cada lugar y las ubicaciones. A partir de ello, el sistema planificador, crea un itinerario teniendo en cuenta las restricciones de tiempo de cada lugar, las distancias entre lugares. Además, como es posible que el usuario no pueda visitar todos los lugares de la lista generada por el agente recomendador, el agente planificador tiene en cuenta las preferencias del usuario, para mostrar los sitios más relevantes para él. El problema se traduce a formato PDDL que después puede ser resuelto por un planificador que admite este formato.

Una vez creado el itinerario, el agente de usuario genera un mapa con la ruta a seguir, en formato PDF.

PersonalTour [49] es un sistema recomendador de paquetes de viaje, desarrollado por investigadores de la Universidad Luterana de Brasil y la Universidad Católica de

Pelotas, en el año 2011. Su objetivo, a diferencia de SAMAP, es crear paquetes de viajes personalizados que puedan ser ofrecidos por agencias de viaje.

Este sistema está basado en agentes, que pueden interactuar entre sí. El objetivo de cada uno de los agentes es especializarse en recomendaciones de servicios específicos, como por ejemplo, vuelos, hoteles, transporte, etc... Para ello, cada agente tiene un nivel de confianza para cada servicio, y los agentes que mayor nivel de confidencia tienen para un servicio son los agentes que se encargarán de recomendar elementos para este servicio.

Estos niveles de confianza se calculan mediante las puntuaciones del usuario. Cada agente recomienda unos determinados ítems que tienen que ser puntuados por el usuario y en función de estas respuestas, los agentes aumentan o disminuyen su nivel de confianza para ese servicio. Además de las puntuaciones, se tiene en cuenta cuando se han realizado las puntuaciones, priorizando las más recientes.

Durante la evaluación de la propuesta, se extrajo conocimiento a partir de agencias de viajes, para crear la base de datos de cada agente. Adicionalmente, se realizaron una serie de pruebas para determinar el número ideal de agentes que debería tener el sistema. Posteriormente, se desplegó el sistema en una agencia de viajes durante un período de 3 meses y se recogieron datos de precisión de tres servicios: vuelos, hoteles y lugares a visitar. El 76,59% de paquetes de viajes generados por PersonalTour fueron preferidos por los clientes de la agencia de viaje, mientras que el 71,27% de paquetes de viaje generados por expertos humanos fueron preferidos por los clientes. Finalmente, se concluyó que las recomendaciones de PersonalTour era mejores que las de expertos humanos, aunque con muy poca diferencia.

GuideMe [12] es un sistema recomendador de puntos de interés desarrollado por investigadores del Instituto Superior de Ingeniería de Lisboa, en 2014. Está enfocado a la interacción social y a los dispositivos móviles. GuideMe es un servicio que ofrece al usuario guías de lugares a visitar y recomienda, en función de las visitas, más lugares que visitar. El sistema está compuesto por una aplicación móvil y una aplicación web, una API REST, un sistema recomendador y una base de datos. Adicionalmente, para la aplicación móvil, se dispone de un servicio que permite enviar notificaciones a los dispositivos móviles.

Por otra parte, el sistema recomendador se implementa mediante la librería Apache Mahout y se hace uso del algoritmo Slope One, un algoritmo de filtrado colaborativo que es ejecutado todos los días a las 3:00 AM. Este servicio, obtiene una lista de usuarios que son elegibles para mostrar recomendaciones y envía notificaciones en caso de que se encuentre en esta lista.

Finalmente, existe **PersTour** [50], desarrollado por investigadores de la Universidad de Melbourne en 2016. PersTour es un sistema generador de itinerarios cuyas recomendaciones y planificaciones están basadas en las preferencias del usuario.

El sistema está compuesto de tres componentes: un componente de recolección y análisis de datos, un componente de recomendación de itinerarios y una interfaz de usuario.

El componente de recolección de datos y análisis se encarga de obtener fotografías de la plataforma Flickr y procesar sus metadatos para inferir la popularidad del punto de interés y el tiempo medio de visita, además de extraer información de la Wikipedia para derivar las categorías correspondientes del punto de interés.

Por su parte, el componente de recomendación de itinerarios se encarga de crear itinerarios a partir de las preferencias del usuario y de restricciones de los puntos de interés como los horarios. Para generar los itinerarios hace uso de un algoritmo de optimización basado en colonias de hormigas, un tipo de algoritmo evolutivo. Este algoritmo comienza con una serie de agentes que van visitando todos los puntos de interés en función de las preferencias y las restricciones. Una vez recorridos todos los puntos hasta el final, el mejor camino escogido por los agentes se recuerda durante un tiempo limitado y se repite todo el proceso hasta optimizar el itinerario. Este algoritmo se basa en el comportamiento de las hormigas, donde estas van dejando rastros de feromonas por los caminos que recorren, lo que provoca que los mejores tramos sean los que mayor cantidad de feromonas acumulan.

Por último, la interfaz de usuario permite al usuario de la plataforma interactuar con el servicio. Esta interfaz se encarga de obtener datos del usuario, como los niveles de preferencia para cada categoría, horarios, duración del tour, etc...; interactuar con el resto de componentes y mostrar los resultados de las recomendaciones en forma de listado y de mapa.

2.3.2. Sistemas existentes comerciales

Con respecto a los sistemas existentes comerciales, se pueden clasificar en varias categorías:

1. **Manuales/Colaborativos:** Son itinerarios creados por personas, qué, o bien planifican visitas a sus clientes a partir de sus preferencias o bien publican rutas y recomendaciones de forma pública. Aunque la experiencia de las personas es muy importante, a menudo, estos servicios no son baratos, ya que requieren de la mano de obra de las personas que llevan a cabo estas ofertas. Esta categoría se puede dividir en dos subcategorías:

1. **Plataformas de viajes:** Son plataformas sociales en donde los viajeros pueden crear recomendaciones de lugares e incluso planes y compartirlas con otros usuarios. Un ejemplo de ellos aparece con *Triptipedia* (<https://www.triptipedia.com>), una enciclopedia colaborativa donde los usuarios pueden escribir recomendaciones en lugares de todo el mundo; *Tripspi* (<https://www.tripspi.com>), una plataforma que contiene itinerarios creados por usuarios de todo el mundo; y *Civitatis*, una plataforma que reúne itinerarios y excursiones populares (<https://www.civitatis.com/>).
2. **Servicios personalizados:** Son servicios llevado a cabo por particulares, en los que estas personas se encargan de planificar itinerarios individualizados a partir de una serie de datos ofrecidos por sus clientes. Un ejemplo de estos servicios se encuentra en *Cualquier destino* (<https://cualquierdestino.es/travel-planner/>) o *The Fearless Foreigner* (<https://www.thefearlessforeigner.com/personal-travel-planner/>).
2. **Por software:** Son sistemas o aplicaciones que permiten crear planes y/u ofrecen recomendaciones de sitios a visitar. A menudo, suelen utilizar servicios como Foursquare y Google Maps, de donde extraen los puntos de interés, las rutas y los mapas necesarios para llevar a cabo la tarea. Esta categoría se puede dividir en tres subcategorías:
 1. Los **planificadores manuales** son plataformas que permiten a un usuario crear un itinerario digital de forma manual. Estos planificadores no incluyen sistemas de recomendación, aunque a menudo suelen estar enfocados a la creación colaborativa. Un ejemplo de ello es *Pebblar* (<https://pebblar.com>) y *Marco* (<https://marco.app>), una página web y una aplicación respectivamente, que permiten crear itinerarios digitales personalizados y colaborativos.
 2. Los **planificadores manuales con recomendaciones** son similares a los planificadores manuales, con la diferencia de que estos ofrecen recomendaciones de posibles lugares a añadir en los itinerarios. Un ejemplo de ellos es *Tour* (<https://tourapp.co>), una aplicación móvil que permite crear de forma sencilla itinerarios y cuyas recomendaciones son extraídas y generadas en tiempo real por el servicio Foursquare.
 3. Los **planificadores automáticos** son sistemas que crean un itinerario de forma automática, a partir de una serie de preferencias que se establecen previas a la planificación. Un ejemplo de estas plataformas es *Triphobo* (<https://triphobo.com>) un planificador que genera itinerarios a partir de una serie de datos, como la edad de los viajeros, etc... Además de generar itinerarios con recomendaciones de lugares, también es capaz de buscar medios de transporte y alojamientos a partir de las preferencias establecidas. Estos itinerarios son modificables en todo momento. Además de Triphobo,

se encuentra *Roadtrippers* (<https://roadtrippers.com/>), una plataforma enfocada principalmente a turistas que disfrutan de la carretera. Esta permite generar un itinerario entre dos o más ciudades elegidas por el usuario, y muestra recomendaciones de posibles puntos de interés a visitar durante el trayecto, además de navegación paso a paso durante este.

2.4. Tecnología

En este apartado se describen las tecnologías existentes, como interfaces, seguridad y servidores, que permiten la implementación del proyecto.

2.4.1. Interfaces y frontend

La interfaz es una de las partes más importantes de toda plataforma, ya que permite que un usuario interactuar con el servicio, y suele ser el módulo que más tiempo de desarrollo suele conllevar en todo proyecto.

En el caso de las plataformas basadas en la web, son los lenguajes HTML, CSS y JavaScript las tecnologías interpretadas, aunque con ligeras diferencias, por todos los navegadores existentes, por ello, su uso es imprescindible para el desarrollo de cualquier aplicación cuya interacción se realice mediante la web. Aunque la web fue inicialmente planteada de forma estática, el gran crecimiento de internet ha provocado la necesidad de que esta se convierta en dinámica [51].

Por ello, surgen dos técnicas: *Server Side scripting* y *Client Side scripting*. En primer lugar *Server Side scripting* delega todas las tareas de maquetado y generación de HTML al servidor, de forma que todos los usuarios reciben una respuesta individual y dinámica [52]. Este tipo de páginas son creadas mediante los lenguajes PHP, ASP.NET y JSP [53], muy utilizados en entornos empresariales. Una ventaja de esta técnica, es que el usuario no tiene que esperar ante indicadores de carga y libera de carga de trabajo a los clientes, especialmente en dispositivos con baja capacidad de procesamiento, sin embargo, presenta varios problemas. Uno de ellos es la dependencia de la capacidad de cómputo de los servidores, ya que si esta no es suficiente, puede conllevar a tiempos de carga largos, deteriorando la experiencia del usuario [54]. A este, se unen la posible existencia de vulnerabilidades web shell, y el incremento de complejidad al hacer uso de la técnica de Caching, ya que cada página generada es individualizada.

En segundo lugar, *Client Side scripting*, delega todas las tareas al propio cliente, que se realizan mediante scripts escritos en el lenguaje JavaScript [52]. Esta técnica va unida al uso de Application Programming Interfaces (o APIs), para obtener información e interactuar con el servidor de forma dinámica. Aunque esta técnica facilita libera carga de trabajo en servidor y facilita el uso de Caching, no está exenta de posibles vulnerabilidades existentes en el servidor y puede provocar pésimas experiencias de

usuario en dispositivos con baja capacidad de cómputo o conexiones de red lentas o inestables [54].

Junto a estas técnicas, existen además distintas formas de implementación de interfaces web. La forma más básica es el denominado *Vanilla JS* o *Vanilla CSS*, que es simplemente una denominación al uso de estos lenguajes en “bruto”, es decir, sin hacer uso de librerías o *frameworks* [55], [56]. Aunque esta es la forma inicial en la que se desarrolló en la web, en 2006 aparecen jQuery y Sass, con el objetivo de facilitar el desarrollo web. jQuery es una librería de JavaScript que facilita muchas de las operaciones comunes realizadas sobre el DOM [57], mientras que Sass es un preprocesador de CSS, que extiende el lenguaje para facilitar la programación al desarrollador [58].

Más adelante, en 2011, surge desde Twitter, Bootstrap, un framework web implementado en HTML, Sass y JavaScript. Su objetivo es asegurar la consistencia visual entre páginas de la misma plataforma y estar enfocado en la experiencia móvil [59]. Aunque no es el único, es uno de los más usados [59].

A menudo, también aparece el MVC (Model-View-Vontroller) un patrón de arquitectura de software que separa la interfaz de usuario, de la interacción y del modelo de datos [60]. Esta arquitectura se presenta en la técnica de *Server Side scripting*, y se implementa en muchos frameworks full-stack, que se centran en el desarrollo de lógica de servidor y de interfaces en paralelo. Como ejemplos, se observan ASP.NET MVC, que hace uso del lenguaje .NET [61], Spring Framework, que hace uso de Java [62] y Symfony Framework, que hace uso del lenguaje PHP [63].

Aunque MVC normalmente ha sido usado en *Server Side scripting*, Google presenta en 2010 AngularJS [64], que hace uso de TypeScript, un lenguaje superconjunto de JavaScript, trayendo este patrón de arquitectura al cliente, aunque sigue requiriendo de interacción con APIs para poder realizar operaciones en servidor [65]. Este tipo de frameworks requieren de compilación previa a la publicación y despliegue de la plataforma.

Además de AngularJS, también existen otros frameworks, como React, desarrollado por Facebook, que hace uso de la extensión de JavaScript, JSX [66], o Vue.js, creado por Evan You e inspirado por AngularJS [67].

2.4.2. *Backend*

HTTP (Hypertext Transfer Protocol), es el protocolo que mueve internet. Desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, HTTP pasa a ser publicado en versión 1.1 en 1999 y pasa a definir la sintaxis y semántica que utilizan la

arquitectura web [68]. Por lo tanto, para el desarrollo de todo tipo de plataforma web, el uso de este protocolo es requerido.

Como se mencionaba anteriormente, a menudo es necesario la implementación de APIs para que un cliente pueda interactuar con un servidor, de manera que se almacenen y gestionen datos relacionados con un usuario de la plataforma.

Para el desarrollo de estas APIs, existen distintos estilos de arquitectura, como por ejemplo, SOAP, y REST:

- **SOAP** (Simple Object Access Protocol) es un paradigma con estado, orientado a los servicios que permiten operar con los datos, y que define la forma de enviar y recibir información entre un cliente y un servidor, cuya estructura está formada por un paquete, en cuyo interior contiene encabezados y un cuerpo [69]. Este paquete es, a menudo, transmitido mediante el protocolo HTTP, sin embargo, debido al uso de XML, es considerablemente más lento que otros paradigmas [70].
- **REST** (Representational State Transfer) es un paradigma sin estados, centrado en los datos, que define la forma de enviar información entre un cliente y un servidor. Se basa en la idea de un protocolo sin estado, en donde cada mensaje contiene la información necesaria para su comprensión, el uso de recursos direccionables mediante un identificador global y la definición de un conjunto de operaciones, como las operaciones GET, POST, PUT y DELETE definidas por HTTP [71]. Aunque es más joven que SOAP, es actualmente el paradigma más usado, gracias a su amplio soporte a una gran variedad de formato de datos, su mayor velocidad y a la posibilidad del uso de la técnica de Caching.

Sin embargo, es también necesario el uso de un lenguaje para implementar la lógica de servidor. Aunque el consejo general es escoger el lenguaje con el que los desarrolladores de la plataforma se sientan más cómodos, existen una serie de lenguajes que son los más utilizados para este tipo de tareas [72]:

- **PHP:** Un 80% de las páginas web usa este lenguaje, entre las que se incluyen Facebook y Yahoo. Está basado en la técnica de *Server Side scripting* y fue el primer lenguaje de programación diseñado especialmente para la web.
- **Java:** Es un lenguaje interpretado, desarrollado por Oracle. Es usado por muchas aplicaciones de grandes empresas, como las de la banca y los seguros, para comunicarse con otros sistemas.
- **Node.js:** Permite la ejecución de JavaScript en el lado del servidor y es muy adecuado para aplicaciones en tiempo real como chats y juegos. Además, una ventaja de usar JavaScript en el lado de servidor, es la posibilidad de reutilizar código para el *frontend* y el *backend*.

- **Python:** Aunque es un lenguaje comúnmente utilizado para la inteligencia artificial y el análisis estadístico, su sintaxis clara y fácil de uso, ha facilitado la adopción de este lenguaje en aplicaciones web, con usos notables por parte de Instagram y YouTube.

2.4.3. Librerías y proyectos adicionales

El uso de librerías en proyectos es cada vez más común, sobretodo cuando los proyectos son cada vez de mayor complejidad, por lo que no tiene sentido reescribir código que otros equipos han desarrollado con mayor eficiencia.

Con respecto a las recomendaciones, existen dos conocidas librerías de aprendizaje automático, Apache Mahout y Tensorflow. Apache Mahout, desarrollado en Java, es un framework para la construcción de algoritmos mediante álgebra lineal [73]. Esta librería ofrece funciones que se pueden usar para la creación de herramientas de aprendizaje automático ya sean de regresión, clustering o recomendaciones.

Por otra parte, Tensorflow es una librería de código abierto, desarrollada por Google e implementada en C++ con interfaces en Python, para la implementación de herramientas de aprendizaje automático [74]. Aunque la interfaz de Python es la más usada, también se ofrece una interfaz para JavaScript y Node.js [75]. Tensorflow está enfocado a la implementación de algoritmos de redes de neuronas, aunque también ofrece el algoritmo k-NN.

Con respecto a la planificación, existen diversos planificadores que utilizan el lenguaje PDDL:

- **FF** (Fast Forward) es un planificador heurístico de encadenamiento hacia delante [76]. El principio heurístico principal es obtener una estimación heurística, relajar la tarea P en cuestión en una tarea más simple P+. Su implementación, **Metric-FF**, es un sistema de planificación desarrollado por Joerg Hoffmann [77], diseñado como extensión del planificador FF, que añade el subconjunto de características definidas por PDDL 2.1.
- **FD** (Fast Downward) es un sistema de planificación basado en la búsqueda heurística [78]. Es un planificador de progresión, que busca el espacio de estados hacia adelante. A diferencia de otros sistemas de planificación, Fast Downward no utiliza directamente la representación PDDL de una tarea de planificación, si no que la entrada se traduce primero a una representación alternativa que hace explícitas muchas de las restricciones implícitas de una tarea de planificación. Aprovechando esta representación alternativa, Fast Downward utiliza descomposiciones jerárquicas de las tareas de planificación para calcular su función heurística.

- **CBP** (Cost-Based Planner) es un planificador PDDL que realiza una búsqueda heurística en el espacio de estados, utilizando varias heurísticas [79]. Por un lado, utiliza estados de anticipación basados en planes relajados, similar a Metric-FF, para acelerar la búsqueda. Por otro lado, la búsqueda también se guía utilizando una heurística numérica y una selección de acciones extraídas de un grafo de planificación relajado. El grafo de planificación relajado se construye teniendo en cuenta los costes de las acciones.
- **LPG** (Local search for Planning Graphs) es un planificador basado en búsqueda local y en grafos de planificación [80], [81]. El espacio de búsqueda de LPG consiste en "grafos de acción", subgrafos del grafo de planificación que representan planes parciales. Los pasos de búsqueda son ciertas modificaciones del grafo que transforman un grafo de acción en otro. LPG utiliza una representación compacta del grafo de planificación para definir vecinos en el espacio de búsqueda y evaluar sus elementos mediante una función heurística parametrizada.

Por último, una reciente herramienta es WASM o WebAssembly, un formato de instrucciones binarias que se ejecutan en una máquina virtual [82], normalmente integrada en navegadores. Al ser instrucciones binarias, estas se pueden ejecutar rápidamente y realizar cálculos con una velocidad similar a la de un lenguaje compilado [83] que sería muy costoso realizar en lenguajes interpretados como JavaScript.

Gracias a esta tecnología, se pueden desarrollar o *portar* librerías adicionales que no estén disponibles en otros módulos (como Tensorflow), para mejorar el rendimiento de algunos cálculos computacionales [84] (como por ejemplo, la función de similitud entre usuarios), que serían poco factibles en *Vanilla JS*.

2.4.4. Bases de datos

En el campo de las bases de datos, existen dos modelos importantes, las bases de datos relacionales y NoSQL.

Las bases de datos relacionales, son un tipo de base de datos que cumplen el modelo relacional, postulado por Edgar Frank Codd en 1970 [85]. En este modelo, una base de datos se compone por tablas con columnas y filas, en donde cada fila está identificada por una clave, y la existencia de claves ajenas permiten referencias filas en otras tablas.

Es el modelo más utilizado actualmente, lo que conlleva al desarrollo de Sistema de Gestión de Bases de Datos Relacionales. Los gestores actuales más populares son:

- **Microsoft SQL Server:** Es la implementación de un Sistema de Gestión de Bases de Datos Relacionales desarrollada por Microsoft [86].

- **Oracle SQL:** Es la implementación de un Sistema de Gestión de Bases de Datos Relacionales , desarrollada por Oracle [87]
- **PostgreSQL:** Es un Sistema de Gestión de Bases de Datos Relacionales gratuito y de código abierto. Destaca por su mayor rendimiento en cargas de trabajo [88].
- **MySQL/MariaDB:** MySQL es un Sistema de Gestión de Bases de Datos Relacionales gratuito y de código abierto, desarrollado por Oracle [89]. Por otro lado, MariaDB es un *fork*, de MySQL creado por los desarrolladores originales de MySQL debido a las incógnitas surgidas por la adquisición por parte de Oracle [90]. Está diseñado con un sustituto para MySQL, con las mismas características, aunque recientemente ha añadido más características diferentes de MySQL.

Por otra parte, las bases de datos NoSQL surgen de cara a mejorar las desventajas de las bases de datos relacionales, como la dificultad al manejar bloques de texto o las deficiencias al manejar datos multimedia o gran cantidad de datos [91].

Entre sus implementaciones, encontramos las siguientes:

- **MongoDB:** Es un sistema de base de datos orientado a documentos y de código abierto [92]. A diferencia de los sistemas relacionales, MongoDB guarda sus estructuras como documentos, que son archivos en formato JSON. Es adecuado para sistemas con gran cantidad de lecturas de datos y para manejar documentos, contenido y eventos.
- **Redis:** Es un sistema de base de datos de clave-valor, con almacenamiento en memoria y de código abierto [93]. A diferencia de los sistemas relacionales, Redis almacena los datos en tablas de *hash*, donde un clave corresponde a un único valor. Aunque es habitualmente utilizado en sistemas distribuidos para mantener cachés o datos compartidos entre instancias, puede ser también utilizado como una base de datos persistente [93].
- **Apache Cassandra:** Es una base de datos distribuida que es capaz de manejar gran capacidad de datos. A diferencia de otras bases de datos, Cassandra puede ser ejecutada en múltiples máquinas o nodos, aumentando la disponibilidad de la base de datos y mejorando el rendimiento [94]. A pesar de ser una base de datos NoSQL, su esquema de base de datos es muy parecido al de las bases de datos relacionales.

2.4.5. Seguridad

Los ataques en internet son amenazas muy constantes, por lo que otra parte vital de importancia en toda plataforma es proteger los datos de los usuarios, ya estén en transporte o en reposo.

HTTPS

En el caso de los datos en transporte, la seguridad viene provista por el protocolo HTTPS, una extensión del protocolo HTTP que implementa TLS, de manera que se cifren los datos que se transmiten, protegiendo la integridad y la privacidad de estos [95].

TLS hace uso de claves públicas y privadas [96], sin embargo, para garantizar la seguridad de las conexiones, no es recomendable el uso de claves autogeneradas. Para ello, aparecen las autoridades de certificación, o CAs. Estas CAs se encargan de generar certificados digitales, firmados por ellas, a sus clientes, que pueden usar para cifrar las comunicaciones entre los clientes y servidores [97]. Sin embargo, la expedición de estos certificados suele ser muy cara y tediosa [98]–[100].

Es por eso que en 2016, se pone en marcha Let's Encrypt, una autoridad de certificación que proporciona certificados gratuitos para el cifrado TLS a través de un proceso automatizado [101]. Así, pequeñas empresas y particulares pueden proteger sus páginas web sin costes, mejorando la seguridad de estas.

Adicionalmente, en el RFC 6797 se define HSTS (HTTP Strict Transport Security) [102]. HSTS es una política, definida por los servidores, que establece que todas las interacciones con el servidor se deben realizar mediante HTTPS y nunca mediante HTTP. De esta forma, se mitigan ataques de hombre en medio, cookie hijacking y downgrade attacks.

CORS, CSP y Cross Site Scripting

Además de HTTPS, muchas plataformas web implementan varios mecanismos para protegerlas de atacantes:

- **CORS** (Cross Origin Resource Sharing) [103] es un mecanismo implementado en los navegadores, que permite restringir el acceso a ciertos recursos por parte de una página web. Estas restricciones se implementan en los encabezados HTTP, enviados por los servidores web. Por ejemplo, una web maliciosa puede intentar acceder a los datos de un usuario de otra plataforma, mediante la ejecución de código en la consola del navegador o mediante páginas de Phising, sin embargo, como la otra plataforma ha establecido una restricción de acceso, las peticiones desde la web maliciosa nunca llegan a realizarse, ya que el navegador impide que se envíe esta a la otra plataforma.
- **CSP** (Content Security Policy) [104] es otro mecanismo, implementado en los navegadores, para limitar la ejecución de recursos en el navegador. Estas restricciones se implementan en los encabezados HTTP y establecen qué recursos se pueden ejecutar en la página. Por ejemplo, un servidor web puede

restringir la ejecución de código JavaScript que se encuentre en linea en elementos HTML. De esta manera, se puede reducir el impacto de vulnerabilidades como Cross-Site Scripting.

- **Cross-Site Scripting** es una vulnerabilidad que permite inyectar y ejecutar código JavaScript o CSS en una página que ven otros usuarios [105]. Este tipo de vulnerabilidad compone el 84% de todas las vulnerabilidades documentadas, según Symantec [106], y a menudo surge por la falta de validación de contenido generado por el usuario. Como soluciones a este tipo de vulnerabilidad, se presenta la validación, por parte del servidor, del contenido generado por el usuario, o la implementación de mecanismos como CSP, que impiden la ejecución de código inyectado en la página visitada.

Protección de datos en reposo

Cada vez más aparecen filtraciones de bases de datos de muchas plataformas en las que se publican datos de usuarios como correos electrónicos y contraseñas. Por ello, la protección de los datos en reposo es también muy importante.

En el ámbito de sistemas, esta tarea se suele llevar a cabo por administradores de sistemas, que se encargan de gestionar los sistemas que potencian las plataformas.

En el ámbito de las contraseñas, los usuarios, a menudo, usan las mismas claves para casi todos los servicios en los que están registrados [107], por lo tanto, una filtración en un servicio puede generar consecuencias en las cuentas de los mismos usuarios en otros servicios. Por ello, se establece que guardar contraseñas en texto plano es mala praxis. Para solucionarlo, se presentan varias técnicas para almacenar contraseñas de forma segura en base de datos:

- **Hashing** consiste en guardar el resultado de computar el hash de una contraseña en base de datos [108]. Aunque de esta forma, las contraseñas no son comprensibles, esta forma de almacenado puede ser atacada mediante tablas arco iris, que consisten en tablas con las contraseñas más comunes y sus correspondientes hash.
- **Hashing + salting** consiste en generar una cadena de caracteres de forma aleatoria, concatenarla con la contraseña, y guardar el resultado de computar el hash de esta concatenación junto a la cadena de caracteres generada de forma aleatoria [108]. De esta forma, se mitigan los ataques mediante tablas arco iris, ya que una contraseña conocida tendrá cada vez un resultado hash distinto.
- Otra forma de guardar contraseñas es mediante el uso del algoritmo **bcrypt**. Además de incorporar una *salting* para proteger de ataques de tabla arco iris, bcrypt es una función adaptativa, ya que con el tiempo, el recuento de

iteraciones puede aumentarse para hacerlo más lento, manteniendo su resistencia a ataques de fuerza bruta incluso con potencias de cálculo crecientes [109].

2.4.6. Despliegues

La escalabilidad es la capacidad de un sistema de adaptarse y reaccionar, cambiando sus recursos, de manera que no pierda calidad. Esta escalabilidad se puede dar de dos formas distintas, vertical y horizontal [110]. La escalabilidad vertical consiste en añadir más recursos a un sistema existente, bien sea aumentando la capacidad de cómputo o la capacidad de memoria, mientras que la escalabilidad horizontal consiste en añadir más sistemas, distribuyendo la carga de trabajo entre estos.

Esta capacidad de escalabilidad va unida al proceso de despliegue de toda plataforma, es decir, el proceso por el cual un sistema de software se hace disponible para su uso. Durante el proceso de despliegue, ha de tenerse en cuenta los sistemas en donde se va a ejecutar la plataforma. Se presentan dos variantes posibles, alojamiento en premisas y alojamiento en la nube [111].

En el alojamiento en premisas, todo el software se ejecutaba en las máquinas que la empresa tenía en sus propias oficinas. Sin embargo, este tipo de alojamiento a menudo presenta problemas de coste, ubicación, mantenimiento y seguridad [111]. Es por ello que actualmente, el software empieza a ejecutarse cada vez más en alojamiento en la nube, en plataformas como Amazon Web Services o Microsoft Azure, ya que estas ofrecen una mayor flexibilidad y mejoras en rendimiento y mantenimientos [112], [113].

Por otra parte, existen distintos lugares en los que ejecutar el software. El primero de ellos consiste en reservar servidores físicos en donde ejecutar programas directamente sobre ellos. De esta manera, si el software necesita interactuar directamente con el hardware, este puede hacerlo de forma sencilla [114].

Sin embargo, los servidores físicos presentan varias dificultades. Si un servicio necesita ampliarse, la puesta en marcha de nuevos servidores físicos puede tardar varios minutos en estar operativa. Además, los administradores de sistemas deben encargarse del mantenimiento, ya sea en el sistema operativo o en el hardware [111].

Las máquinas virtuales se presentan como una alternativa. Con las máquinas virtuales, como las que se encuentran en los servicios en la nube, los administradores de sistemas no tienen que preocuparse por mantenimiento del hardware o del sistema operativo, ya que éste lo realizan los propios servicios en la nube [115]. Otra ventaja es que se aprovechan mejor los recursos, ya que se pueden ejecutar múltiples máquinas virtuales en la misma máquina física [115].

Por último, los contenedores han surgido recientemente como una alternativa a la ejecución de software. Los contenedores vuelven a abstraer la ejecución del software, virtualizando y aislando el software entre los contenedores y el sistema anfitrión. Aunque presentan las mismas ligeras perdidas de rendimiento que las máquinas virtuales, los contenedores permiten ejecutar múltiples instancias de un programa de forma reproducible, eliminando la posibilidad de errores específicos de la máquina [116].

Entre los programas de gestión de contenedores más conocidos se encuentran LXC, lanzado en 2008 y que forma parte del núcleo de Linux [117], y Docker, lanzado en 2013 [118].

Otra característica de los contenedores es su gran escalabilidad horizontal. Es fácil arrancar múltiples instancias de un contenedor para manejar la demanda en el servicio [119], [120]. Ese es el objetivo de Kubernetes, un sistema de orquestación de contenedores de código abierto para automatizar el despliegue, el escalado y la gestión de contenedores [121], y de Docker Swarm, una utilidad para agrupar varios motores Docker que se ejecutan en diferentes máquinas bajo un solo motor Docker "virtual", lo que permite la automatización del despliegue y el escalado de contenedores [122].

3. ANÁLISIS DEL SISTEMA

En este capítulo se describe la solución planteada para la resolución del sistema. Asimismo, se incluyen los requisitos, casos de uso, prototipos y clases que la plataforma debe cumplir en todo momento.

3.1. Diseño

Como se comentaba anteriormente, el objetivo de este proyecto es desarrollar una plataforma con un sistema que recomiende y planifique itinerarios turísticos a sus usuarios. Se denominará Hermes, en honor al dios griego mensajero, de las fronteras y de los viajeros.

A diferencia de los proyectos presentados en el *Estado del Arte*, se desea dar más control a los usuarios de la plataforma a la hora de tomar decisiones, por lo tanto, se va a dividir en los siguientes módulos:

1. **Base de datos:** La base de datos contendrá toda la información necesaria para el funcionamiento de la plataforma.
2. **Sistema recomendador:** El sistema recomendador se encargará de generar recomendaciones de lugares o puntos de interés para los usuarios. Para ello, se seguirá un modelo de recomendación híbrido, en donde se utilizará filtrado colaborativo basado en usuarios (mediante las puntuaciones de los usuarios, usando la medida MPIP [38]) y filtrado basado en contenido (mediante las acciones observadas del usuario y sus preferencias). Este método híbrido permite solucionar el problema de *Shilling Attacks* y la baja diversidad.
3. **Sistema planificador:** El sistema planificador se encarga de crear itinerarios o planes. Este problema de planificador se puede considerar como un problema de TSP (Travelling Salesman Problem) con restricciones horarias. Para ello, hará uso del lenguaje PDDL y de un planificador clásico. Además, se deberá implementar un traductor de la representación interna de la plataforma al lenguaje de PDDL y otro traductor de la salida del planificador a la representación interna de los itinerarios.
4. **API/Backend:** El *backend*, es decir, la parte de servidor, se encarga de interactuar con la base de datos y los sistemas planificador y recomendador. Además, deberá implementar una API REST para interactuar con la interfaz de usuario y un servicio de correo electrónico para enviar notificaciones importantes a los usuarios y recordatorios para puntuar sitios visitados, de manera que se intente prevenir el problema de dispersión.

5. **Interfaz de usuario:** La interfaz de usuario es el medio por el que los usuarios interactuarán con la plataforma, encargándose de mostrar, solicitar y enviar información al *Backend*. La interfaz de usuario está dividida en los siguientes componentes:
 1. **Registro e inicio de sesión:** El componente de registro e inicio de sesión se encarga de crear nuevas cuentas e identificar al usuario en la plataforma. Deberá implementar medidas de seguridad para evitar el uso de herramientas automatizadas (o *bots*) y solicitar información al usuario durante el registro para sobrellevar el problema del arranque en frío.
 2. **Recuperación de contraseña:** Este componente se encarga de dar la posibilidad al usuario de recuperar su contraseña en caso de que la haya olvidado.
 3. **Recomendaciones:** Este componente deberá mostrar las recomendaciones al usuario de forma intuitiva. Para ello, usará un listado de lugares junto a un mapa interactivo. Además, permitirá al usuarios seleccionar lugares para añadir a un plan. A diferencia de las plataformas presentadas en el *Estado del Arte*, se ha decidido separar esta parte del planificador, ya que de esta manera se da más control al usuario y se le ofrece la posibilidad de descubrir los lugares relevantes para el de la ciudad a visitar, mejorando su experiencia de viaje y de usuario.
 4. **Planes:** Este componente mostrará todos los planes que el usuario ha creado, además de mostrar los itinerarios creados en detalle, permitiendo modificarlos y eliminarlos.
 5. **Puntuaciones:** Este componente mostrará al usuario las puntuaciones que ha creado, además de permitir al usuario crear nuevas puntuaciones y gestionar las existentes.
 6. **Perfil:** Este componente permitirá al usuario ver, modificar, descargar y eliminar todos sus datos personales, acorde a la política de privacidad creada; realizar gestiones de seguridad, como el cambio de contraseña o el cierre de sesiones; y modificar sus preferencias de notificaciones por correo.
 7. **Ayuda:** Este componente contendrá documentación de ayuda sobre la plataforma así como enlaces de contacto.

En la Figura 3.1 se muestra la arquitectura general de la plataforma Hermes.

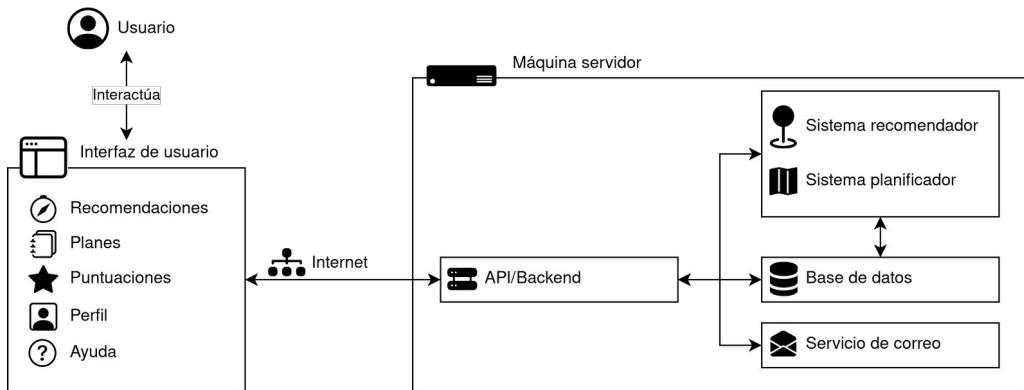


FIGURA 3.1: ARQUITECTURA DE LA PLATAFORMA HERMES

En el caso de la recolección de datos, solo se va a implementar el sistema para la ciudad de Madrid, aunque se deberá podrá ampliar fácilmente en el futuro. Se van a extraer datos de las plataformas Foursquare y OpenStreetMap. También se extraerán algunos datos faltantes de la plataforma Google Maps.

Se van a utilizar los lenguajes HTML, CSS y JavaScript junto a las librerías Bootstrap, Bootstrap Icons, Font Awesome y Leaflet para la interfaz de usuario, de manera que se pueda utilizar la técnica de *Caching*. Para el *backend* se utilizará Node.js con el lenguaje JavaScript, además del lenguaje Rust para crear librerías personalizadas para los casos en los que se requiera de cálculos intensivos. Además, se utilizará la librería *fastify* para la API REST, debido a su alto rendimiento [123], [124] y la librería *sequelize* para la interacción entre la base de datos y el backend.

Finalmente, el sistema se debe implementar aprovechando la escalabilidad horizontal. Para ello, se va a seguir la estructura de red que se muestra en la Figura 3.2.

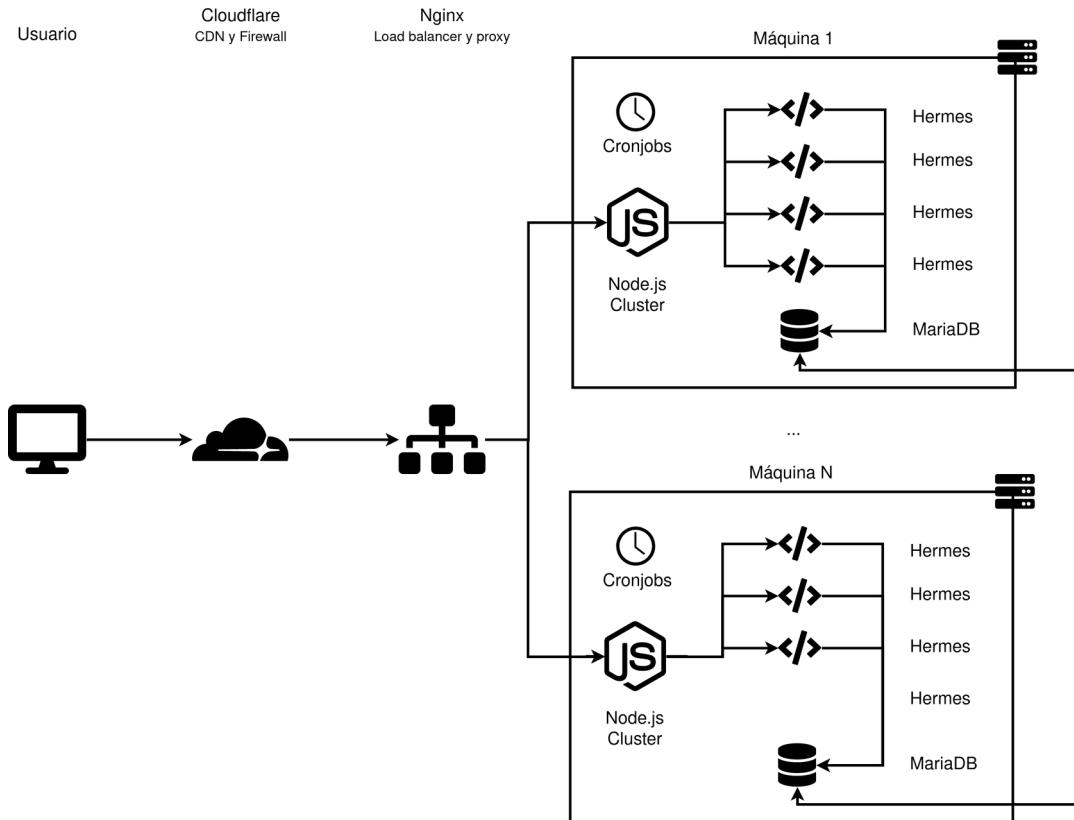


FIGURA 3.2: ESTRUCTURA DE RED DISTRIBUIDA DE LA PLATAFORMA HERMES

El objetivo de esta estructura es crear una plataforma que sea escalable horizontalmente y distribuida. Para ello, se va a utilizar la plataforma Cloudflare como CDN y Firewall, el software Nginx como Load Balancer y Proxy y contenedores de Docker para su fácil escalabilidad y distribución.

Así, el proceso de una petición (por ejemplo, cuando un usuario desea acceder a la plataforma) deberá ser el siguiente:

1. **Cloudflare:** La petición comienza entrando en la red de Cloudflare. Cloudflare, aunque es opcional, es utilizado como CDN y Firewall para prevenir ataques de denegación de servicio, para mejorar el rendimiento de la plataforma y para reducir el consumo de banda ancha de los servidores.
2. **Load balancing:** Tras pasar por Cloudflare, la petición se procesa en un servidor o servidores con Nginx. Este servidor es utilizado para distribuir la carga entre distintos servidores.
3. **Máquina:** Una vez Nginx ha elegido el servidor que procesará la petición, esta es enviada a una de las máquinas. Estas máquinas contienen tres contenedores:
 - **MariaDB:** Es la base de datos utilizada por el proyecto. Cuando hay más de una máquina en funcionamiento, la base de datos se distribuye entre todas ellas de manera que se reparta la carga de trabajo de la plataforma

- **Clúster Node.js:** Es un conjunto de varios procesos de Node.js. Estos procesos ejecutan el código implementado para la plataforma.
 - **Cronjobs:** Tareas que se ejecutan de forma periódica mediante el servicio Cron en el Linux.
4. **Respuesta:** La respuesta vuelve por el camino inverso de forma automática.

3.2. Usuarios

Para desarrollar el proyecto, es necesario establecer una serie de ejemplos de personas para conocer a los usuarios potenciales de la plataforma, y poder derivar requisitos, casos de uso y prototipos. En este apartado se definen tres ejemplos de usuarios y sus características principales.

Iván Durán

Personalidad y motivaciones

Iván Durán es un estudiante de ingeniería informática de 26 años. Es una persona alegre y extrovertida. Suele disfrutar pasando tiempo con sus amigos, especialmente en parques y zonas públicas y a menudo suele organizar escapadas con ellos. Está muy familiarizada con la tecnología y se preocupa por su privacidad. No le importa pagar por servicios que aseguran que sus datos personales son privados e intenta evitar aplicaciones con muchos servicios de analíticas.

Experiencia tecnológica

Iván está muy familiarizada con la tecnología y hace uso extensivo de medios digitales e internet. Tiene conocimientos avanzados sobre la tecnología y el software. Posee varios ordenadores en su domicilio, además de un teléfono móvil de última generación.

Objetivos

Para sus viajes, a Iván le encantaría encontrar una plataforma que le permitiese descubrir parques y lugares de recreo en los que pasar el tiempo. Esta plataforma debería ser sencilla de usar en dispositivos móviles además de respetar su privacidad y ser segura.

José Antonio Lorenzo

Personalidad y motivaciones

José Antonio Lorenzo tiene 32 años. Cuando su trabajo se lo permite, suele realizar viajes en la temporada de verano. Por lo general, no suele planificar mucho sus visitas,

pero si le gusta explorar plataformas de viaje para saber que lugares visitar sobre la marcha. No tiene muchas aficiones, aunque disfruta leyendo sobre ciencia, en especial física y química.

Experiencia tecnológica

José Antonio está familiarizado con la tecnología. Aunque dispone de un ordenador de sobremesa, suele utilizar mucho su teléfono móvil, principalmente cuando se encuentra fuera de casa. Aunque hace un uso extensivo de él, a menudo encuentra que muchas aplicaciones que utiliza son lentas o su experiencia de usuario es pésima.

Objetivos

Las plataformas que José Antonio suele utilizar no son muy personalizadas, donde sus contenidos son principalmente generados por usuarios. Por ello, le gustaría utilizar una plataforma más adaptada a sus preferencias que le permita descubrir que lugares puede visitar en uno de sus viajes.

Elena Martínez

Personalidad y motivaciones

Elena Martínez tiene 54 años y dos títulos, uno en Bellas Artes y otro en Historia del Arte. Es una persona un poco introvertida que pasa mucho tiempo rodeada de arte y literatura. A menudo suele visitar galerías de arte y museos y suele planificar al detalle sus visitas.

Experiencia tecnológica

Elena no es una persona muy experta en la tecnología, aunque se sabe manejar lo suficiente por internet para encontrar información sobre obras de arte y discutir sobre ellas en foros. Posee un portátil de gama media, con unos años de antigüedad.

Objetivos

A Elena le encantaría poder planificar de formas más sencillas sus visitas a museos y galerías de arte. Por ello, le gustaría descubrir una plataforma que le permitiese realizar esto, además de añadir notas y detalles sobre los lugares que planea visitar.

3.3. Requisitos

La plataforma debe cumplir con los siguientes requisitos que definen su comportamiento. Los tipos de requisitos pueden ser:

- **Usuario:** Son requisitos que describen las necesidades del usuario.
- Software: Son requisitos que describen las características y funcionalidades del software a implementar. Se clasifican en dos:
 - **Funcionales:** Son aquellos requisitos que definen las actividades o funciones que debe cumplir el proyecto.
 - **No funcionales:** Son aquellos requisitos que definen las restricciones o características que debe tener el proyecto.

Además, se seguirá la siguiente plantilla para su organización:

TABLA 3.1: PLANTILLA DE REQUISITOS

Identificador				
<i>Descripción</i>	<i>Necesidad</i>	<i>Prioridad</i>	<i>Estabilidad</i>	<i>Verificabilidad</i>
	(Baja-Media-Alta)			(Baja-Media-Alta)
	(Sí - No)			(Sí - No)

Los elementos que componen la plantilla son los siguientes:

- **Identificador:** Seguirán la estructura RX-Y, donde X es U para requisitos de usuario, F para requisitos funcionales y NF para requisitos no funcionales, e Y es un número entero.
- **Descripción:** Breve y conciso resumen del requisitos
- **Necesidad:** Necesidad de cumplimiento del requisito. Su valor puede ser Baja, Media o Alta
- **Prioridad:** Importancia del requisitos. Su valor puede ser Baja, Media o Alta.
- **Estabilidad:** Indica si el requisito puede sufrir modificaciones futuras o no. Su valor puede ser Sí o No.
- **Verificabilidad:** Indica si el requisito puede ser verificable mediante alguna prueba. Su valor puede ser Sí o No.

3.3.1. Requisitos de usuario

TABLA 3.2: REQUISITO RU-1

RU-1				
<i>Descripción</i>	<i>Necesidad</i>	<i>Prioridad</i>	<i>Estabilidad</i>	<i>Verificabilidad</i>
El usuario podrá registrarse en la plataforma.	Alta	Alta	Sí	Sí

TABLA 3.3: REQUISITO RU-2

RU-2			
<i>Descripción</i>	El usuario podrá iniciar sesión en la plataforma.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.4: REQUISITO RU-3

RU-3			
<i>Descripción</i>	El usuario podrá cerrar su sesión iniciada en la plataforma.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.5: REQUISITO RU-4

RU-4			
<i>Descripción</i>	El usuario podrá ver sus recomendaciones de lugares a visitar y ver información sobre ellos.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.6: REQUISITO RU-5

RU-5			
<i>Descripción</i>	El usuario podrá crear nuevos itinerarios o planes.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.7: REQUISITO RU-6

RU-6			
<i>Descripción</i>	El usuario podrá ver los itinerarios que tiene creados.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.8: REQUISITO RU-7

RU-7			
<i>Descripción</i>	El usuario podrá modificar los itinerarios que ha creado, añadiendo y eliminando lugares a visitar.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.9: REQUISITO RU-8

RU-8			
<i>Descripción</i>	El usuario podrá ver puntuaciones que ha creado.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.10: REQUISITO RU-9

RU-9			
<i>Descripción</i>	El usuario podrá crear puntuaciones.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.11: REQUISITO RU-10

RU-10			
<i>Descripción</i>	El usuario podrá eliminar las puntuaciones que ha creado.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.12: REQUISITO RU-11

RU-11			
<i>Descripción</i>	El usuario podrá ver y gestionar sus datos personales y preferencias.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.13: REQUISITO RU-12

RU-12			
<i>Descripción</i>	El usuario podrá modificar sus datos personales y preferencias.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.14: REQUISITO RU-13

RU-13			
<i>Descripción</i>	El usuario podrá acceder a la plataforma desde un navegador móvil o desde un navegador de escritorio.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.15: REQUISITO RU-14

RU-14			
<i>Descripción</i>	El usuario podrá consultar ayuda y solicitarla cuando lo necesite.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.16: REQUISITO RU-15

RU-15			
<i>Descripción</i>	El usuario podrá interactuar con la plataforma de forma segura y estará informado de ello.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

3.3.2. Requisitos de software

3.3.2.1. Requisitos funcionales

TABLA 3.17: REQUISITO RF-1

RF-1			
<i>Descripción</i>	La plataforma tendrá un módulo de base de datos.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.18: REQUISITO RF-2

RF-2			
<i>Descripción</i>	La plataforma almacenará los datos en base de datos de forma segura.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.19: REQUISITO RF-3

RF-3			
<i>Descripción</i>	La base de datos recogerá lugares, información sobre estos, datos de los usuarios y demás información necesaria.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.20: REQUISITO RF-4

RF-4			
<i>Descripción</i>	La plataforma tendrá un módulo recomendador de puntos de interés, con una estructura híbrida.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.21: REQUISITO RF-5

RF-5			
<i>Descripción</i>	La plataforma tendrá un módulo planificador.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.22: REQUISITO RF-6

RF-6			
<i>Descripción</i>	La plataforma permitirá el acceso a los usuarios de forma segura		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.23: REQUISITO RF-7

RF-7			
<i>Descripción</i>	La plataforma permitirá a los usuarios crear una nueva cuenta.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.24: REQUISITO RF-8

RF-8			
<i>Descripción</i>	La plataforma deberá recoger datos personales durante el registro.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.25: REQUISITO RF-9

RF-9			
<i>Descripción</i>	La plataforma deberá recoger preferencias del usuario durante el registro.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.26: REQUISITO RF-10

RF-10			
<i>Descripción</i>	La plataforma permitirá a los usuarios iniciar sesión.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.27: REQUISITO RF-11

RF-11			
<i>Descripción</i>	La plataforma rechazará intentos de inicio de sesión incorrectos.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.28: REQUISITO RF-12

RF-12			
<i>Descripción</i>	La plataforma permitirá recuperar contraseñas olvidadas.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Media
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.29: REQUISITO RF-13

RF-13			
<i>Descripción</i>	La plataforma permitirá cerrar sesiones iniciadas.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.30: REQUISITO RF-14

RF-14			
<i>Descripción</i>	La plataforma generará recomendaciones personalizadas para el usuario.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.31: REQUISITO RF-15

RF-15			
<i>Descripción</i>	La plataforma mostrará, en función del nivel de confianza, las recomendaciones al usuario.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.32: REQUISITO RF-16

RF-16			
<i>Descripción</i>	La plataforma permitirá al usuario añadir y eliminar lugares de un plan.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.33: REQUISITO RF-17

RF-17			
<i>Descripción</i>	La plataforma permitirá crear al usuario un nuevo plan.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.34: REQUISITO RF-18

RF-18			
<i>Descripción</i>	La plataforma deberá recoger datos de la visita, como las duraciones de la visita, el punto de inicio y fin de cada día y la duración del viaje, antes de crear un nuevo plan.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.35: REQUISITO RF-19

RF-19			
<i>Descripción</i>	La plataforma mostrará al usuario los planes que haya creado, ordenados por las secciones de actuales y pasados.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.36: REQUISITO RF-20

RF-20			
<i>Descripción</i>	La plataforma mostrará la información de un plan al usuario		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.37: REQUISITO RF-21

RF-21			
<i>Descripción</i>	La plataforma permitirá al usuario añadir y eliminar lugares de un plan activo.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.38: REQUISITO RF-22

RF-22			
<i>Descripción</i>	La plataforma permitirá cambiar el nombre y la descripción de un plan activo.		
<i>Necesidad</i>	Media	<i>Prioridad</i>	Baja
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.39: REQUISITO RF-23

RF-23			
<i>Descripción</i>	La aplicación permitirá eliminar planes.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.40: REQUISITO RF-24

RF-24			
<i>Descripción</i>	La plataforma permitirá ver las puntuaciones que ha creado un usuario.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.41: REQUISITO RF-25

RF-25			
<i>Descripción</i>	La plataforma permitirá crear nuevas puntuaciones.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.42: REQUISITO RF-26

RF-26			
<i>Descripción</i>	La plataforma permitirá eliminar puntuaciones existentes.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.43: REQUISITO RF-27

RF-27			
<i>Descripción</i>	La plataforma permitirá buscar lugares que estén en la base de datos.		
<i>Necesidad</i>	Baja	<i>Prioridad</i>	Baja
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.44: REQUISITO RF-28

RF-28			
<i>Descripción</i>	La plataforma permitirá al usuario ver su perfil.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.45: REQUISITO RF-29

RF-29			
<i>Descripción</i>	La plataforma permitirá al usuario modificar sus datos personales y preferencias.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.46: REQUISITO RF-30

RF-30			
<i>Descripción</i>	La plataforma permitirá al usuario cambiar su contraseña.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.47: REQUISITO RF-31

RF-31			
<i>Descripción</i>	La plataforma permitirá al usuario eliminar su cuenta.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.48: REQUISITO RF-32

RF-32			
<i>Descripción</i>	La plataforma dispondrá de artículos y textos de ayuda describiendo la plataforma.		
<i>Necesidad</i>	Media	<i>Prioridad</i>	Baja
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.49: REQUISITO RF-33

RF-33			
<i>Descripción</i>	La plataforma enviará correos electrónicos con información de seguridad.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.50: REQUISITO RF-34

RF-34			
<i>Descripción</i>	La plataforma enviará recordatorios de puntuaciones y planes.		
<i>Necesidad</i>	Media	<i>Prioridad</i>	Media
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

3.3.2.2. Requisitos no funcionales

TABLA 3.51: REQUISITO RNF-1

RNF-1			
<i>Descripción</i>	La plataforma se implementará como una aplicación web.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.52: REQUISITO RNF-2

RNF-2

<i>Descripción</i>	La plataforma usará el lenguaje JavaScript con Node.js y el lenguaje Rust.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.53: REQUISITO RNF-2

RNF-2

<i>Descripción</i>	La plataforma usará la base de datos MariaDB.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.54: REQUISITO RNF-3

RNF-3

<i>Descripción</i>	La plataforma usará las librerías <i>fastify</i> y <i>sequelize</i> .		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.55: REQUISITO RNF-4

RNF-4

<i>Descripción</i>	La plataforma usará la librería Bootstrap.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.56: REQUISITO RNF-5

RNF-5

<i>Descripción</i>	La plataforma implementará una API REST.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.57: REQUISITO RNF-6

RNF-6			
<i>Descripción</i>	La plataforma solo deberá aceptar peticiones HTTPS.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.58: REQUISITO RNF-7

RNF-7			
<i>Descripción</i>	La plataforma guardará las contraseñas en base de datos usando el algoritmo bcrypt		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.59: REQUISITO RNF-8

RNF-8			
<i>Descripción</i>	La plataforma hará uso de un sistema recomendador basado en filtrado colaborativo basado en usuarios y filtrado basado en contenido.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.60: REQUISITO RNF-9

RNF-9			
<i>Descripción</i>	La plataforma hará uso del lenguaje PDDL para el sistema planificador.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.61: REQUISITO RNF-10

RNF-10			
<i>Descripción</i>	La plataforma debe ser escalable y distribuida.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.62: REQUISITO RNF-11

RNF-11			
<i>Descripción</i>	La plataforma hará uso de contenedores Docker		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

TABLA 3.63: REQUISITO RNF-12

RNF-12			
<i>Descripción</i>	La plataforma hará uso de la medida de similitud MPIP para la implementación el sistema recomendador basado en filtrado colaborativo con usuarios.		
<i>Necesidad</i>	Alta	<i>Prioridad</i>	Alta
<i>Estabilidad</i>	Sí	<i>Verificabilidad</i>	Sí

3.3.3. Matriz de trazabilidad

Tras especificar todos los requisitos, es necesario comprobar que, para cada requisito de usuario existe al menos un requisito de software. Esto se confirma mediante la matriz de trazabilidad, en la Tabla 3.64.

TABLA 3.64: MATRIZ DE TRAZABILIDAD, REQUISITOS DE USUARIO Y REQUISITOS DE SOFTWARE

	RU-1	RU-2	RU-3	RU-4	RU-5	RU-6	RU-7	RU-8	RU-9	RU-10	RU-11	RU-12	RU-13	RU-14	RU-15
RF-1															

RF-2	X								X	X			
RF-3				X									
RF-4				X									
RF-5					X	X	X						
RF-6		X	X										
RF-7	X												
RF-8	X												
RF-9	X												
RF-10		X											
RF-11		X											
RF-12													
RF-13			X										
RF-14				X									
RF-15				X									
RF-16					X								
RF-17					X								
RF-18					X								
RF-19						X							
RF-20						X							
RF-21							X						
RF-22							X						
RF-23							X						
RF-24								X					
RF-25								X					
RF-26									X				
RF-27													
RF-28										X			
RF-29										X	X		
RF-30											X		
RF-31										X			

RF-32										X	
RF-33											
RF-34											
RNF-1										X	
RNF-2											
RNF-3											
RNF-4											
RNF-5											
RNF-6										X	
RNF-7										X	
RNF-8											
RNF-9											
RNF-10											
RNF-11											
RNF-12			X								

3.4. Casos de uso

En este apartado se definen los casos de uso del proyecto. Estos casos de uso definen ejemplos del comportamiento del sistema, así como el flujo que debe seguir para su funcionamiento en cada caso. Cada caso de uso se establece y se define con las siguiente plantilla:

TABLA 3.65: PLANTILLA DE CASOS DE USO

Identificador: Nombre del caso

Descripción

Actor

Flujo de actividad

Condiciones de entrada

Condiciones de salida

Diagrama de interacción

Imagen del diagrama de interacción

Los identificadores seguirán la estructura CUX, donde X es un número entero.

Los elementos que componen la plantilla son los siguientes:

- **Descripción:** Breve y conciso resumen de caso de uso.
- **Actor:** Elemento o elementos que componen el caso. Su valor puede ser Usuario y Sistema.
- **Flujo de actividad:** Consiste en el proceso que sigue el caso de uso para su realización.
- **Condiciones de entrada:** Establece que elementos se deben cumplir para que se pueda llevar a cabo el caso de uso.
- **Condiciones de salida:** Establece que elementos se deben cumplir para que se pueda llevar a cabo el caso de uso.
- **Diagrama de interacción:** Consiste en una ilustración que describe el caso de uso de forma visual.

Los casos de uso son los siguientes:

TABLA 3.66: CU1: REGISTRO

CU1: Registro

<i>Descripción</i>	Un nuevo Usuario quiere crear una nueva cuenta
<i>Actor</i>	Usuario y Sistema
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. Usuario introduce datos personales 2. Sistema verifica datos personales 3. Usuario introduce preferencias 4. Usuario introduce puntuaciones 5. Usuario finaliza la creación de cuenta
<i>Condiciones de entrada</i>	Datos personales válidos Preferencias del usuario Puntuaciones
<i>Condiciones de salida</i>	Se creará correctamente una nueva cuenta

Diagrama de interacción

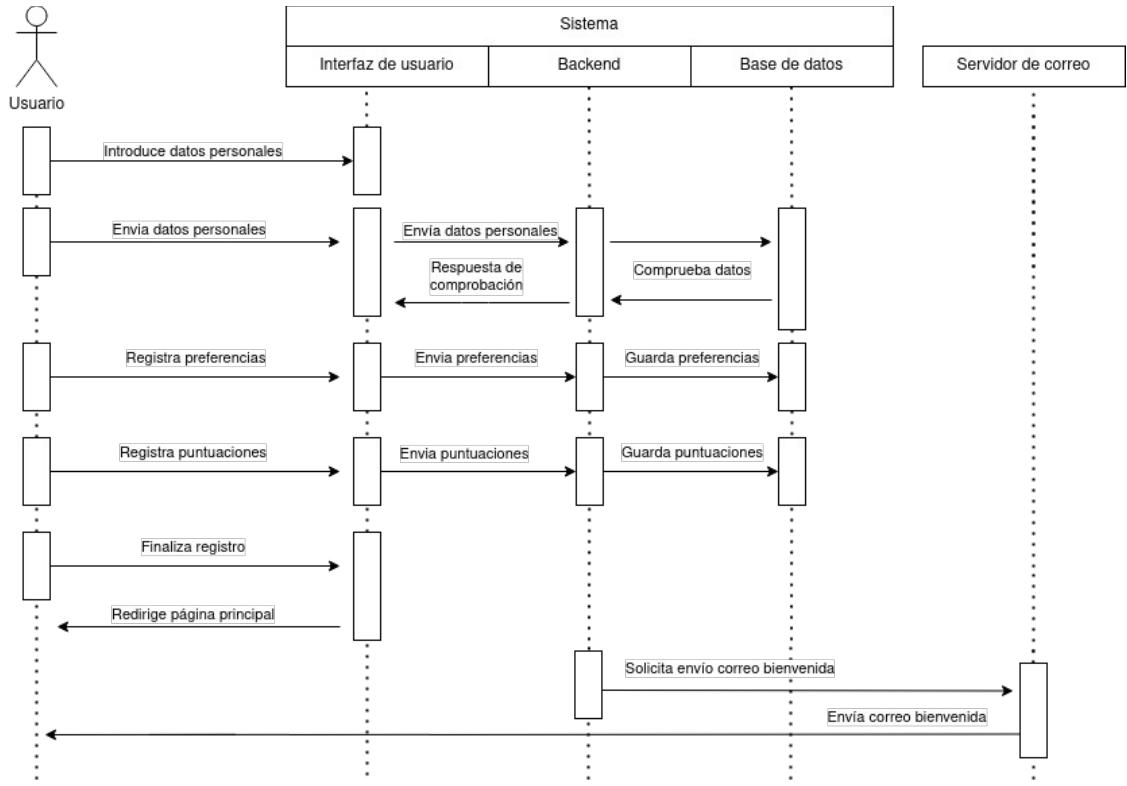


TABLA 3.67: CU2: INICIO DE SESIÓN

CU2: Inicio de sesión

<i>Descripción</i>	Un Usuario existente desea iniciar sesión
<i>Actor</i>	Usuario
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. Usuario introduce correo electrónico y contraseña 2. Usuario Inicia Sesión
<i>Condiciones de entrada</i>	Datos de inicio de sesión correctos
<i>Condiciones de salida</i>	Se creará un nuevo inicio de sesión

Diagrama de interacción

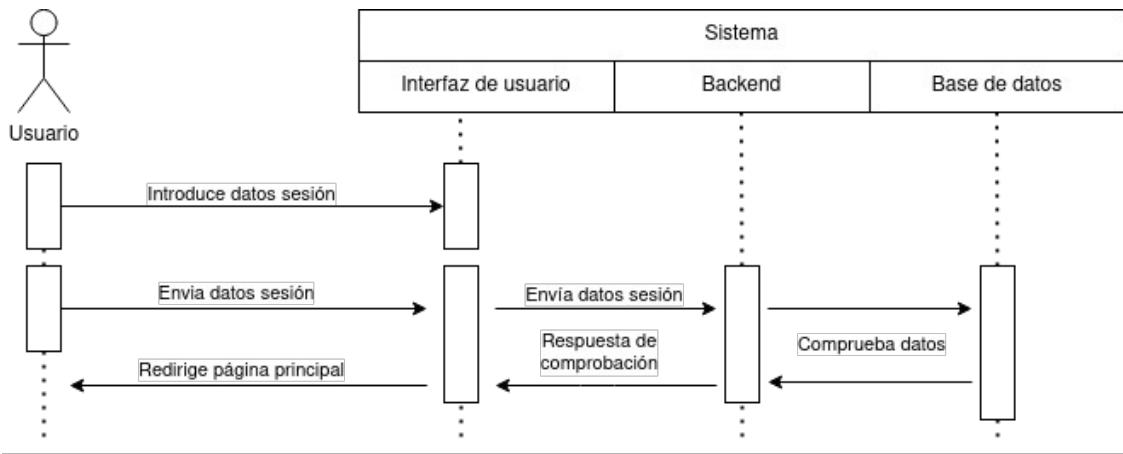


TABLA 3.68: CU3: EXPLORAR RECOMENDACIONES

CU3: Explorar recomendaciones

<i>Descripción</i>	El usuario quiere explorar las recomendaciones para el
<i>Actor</i>	Usuario
<i>Flujo de actividad</i>	1. Usuario explora recomendaciones <ul style="list-style-type: none"> • Mediante el mapa • Mediante el listado de lugares
<i>Condiciones de entrada</i>	Ninguna
<i>Condiciones de salida</i>	Se mostrarán las recomendaciones y se podrán interactuar con ellas

Diagrama de interacción

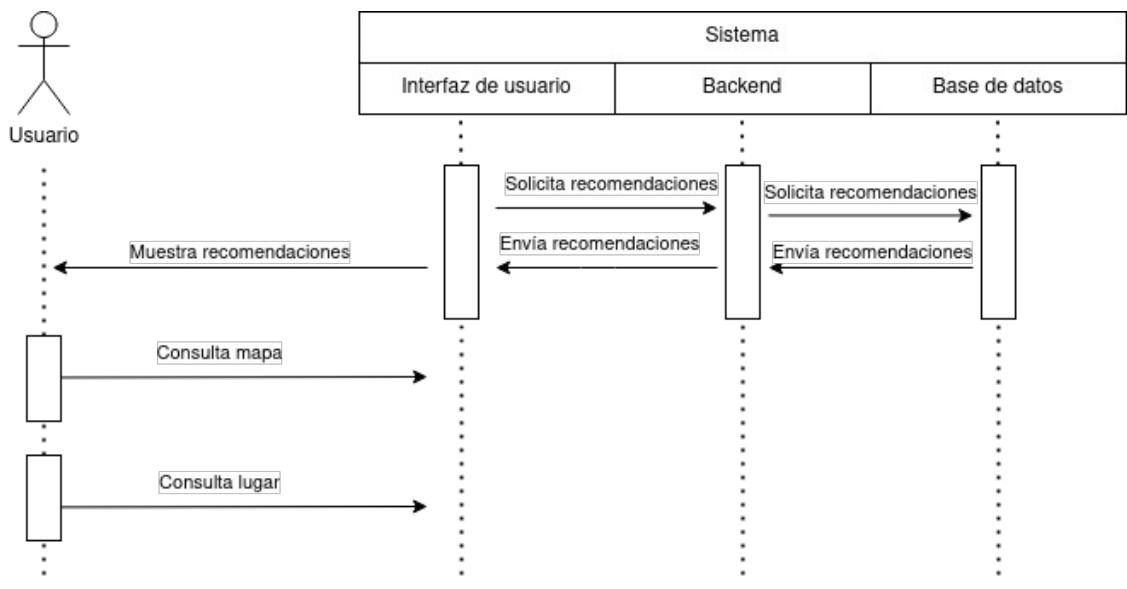


TABLA 3.69: CU4: AÑADIR LUGAR AL PLAN

CU4: Añadir lugar al plan

<i>Descripción</i>	El usuario quiere añadir un lugar al plan
<i>Actor</i>	Usuario
<i>Flujo de actividad</i>	1. Usuario añade lugar en recomendaciones al plan
<i>Condiciones de entrada</i>	Lugar a añadir de las recomendaciones
<i>Condiciones de salida</i>	Se añadirá un lugar al plan

Diagrama de interacción

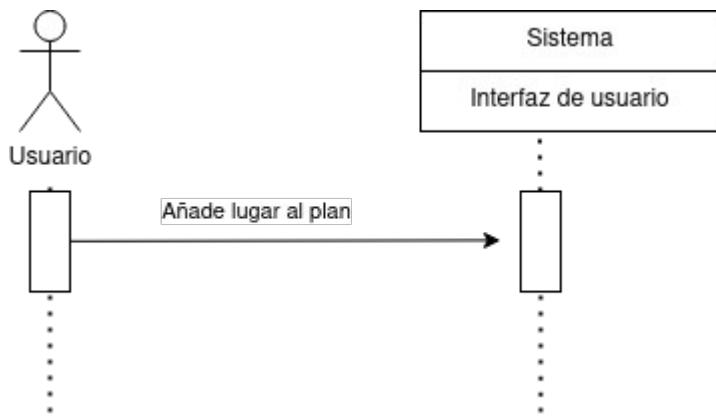


TABLA 3.70: CU5: CREAR PLAN

CU5: Crear plan

<i>Descripción</i>	El usuario desea crear un nuevo plan
<i>Actor</i>	Usuario y Sistema
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. Usuario pulsa en acción de planificar 2. Usuario introduce datos de la visita 3. Usuario pulsa botón crear 4. Sistema valida los datos de entrada 5. Sistema crea tarea de planificación
<i>Condiciones de entrada</i>	<p>Listado de lugares a visitar Información de la visita</p>
<i>Condiciones de salida</i>	<p>Se creará un nuevo plan Ocurrirá una redirección al nuevo plan</p>

Diagrama de interacción

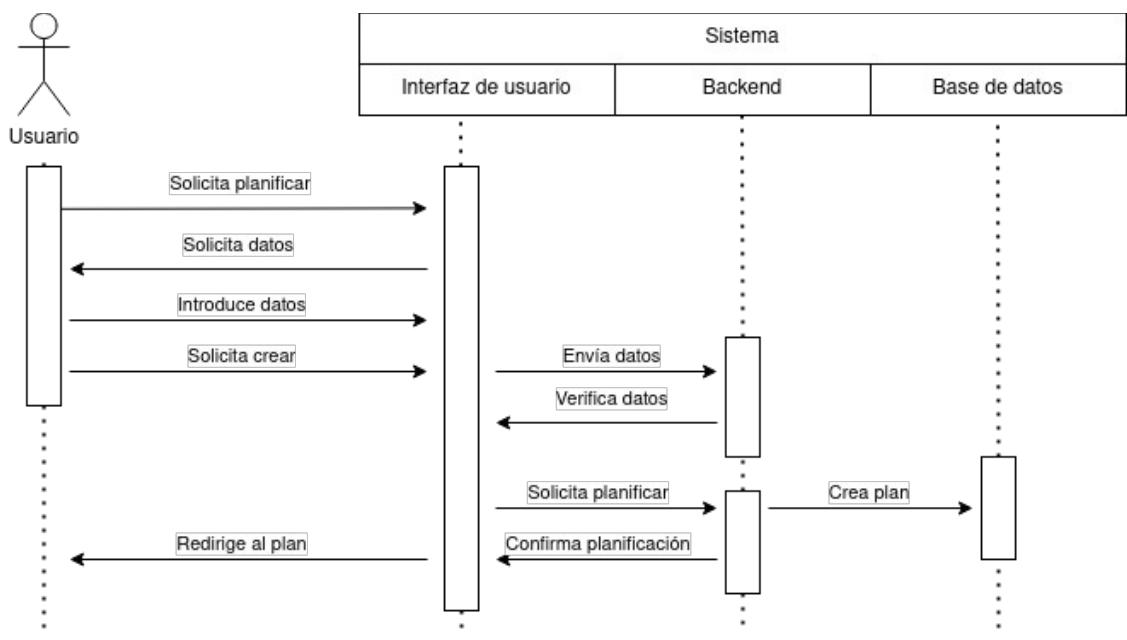


TABLA 3.71: CU6: EDITAR PLAN

CU6: Editar plan

Descripción El usuario desea editar un plan existente

Actor Usuario

Flujo de actividad

1. Usuario entra en modo edición
2. Usuario realiza cambios en el plan
3. Usuario guarda cambios del plan
4. Confirmación de cambios guardados

Condiciones de entrada Plan existente activo

Condiciones de salida Se realizará un cambio en el plan

Diagrama de interacción

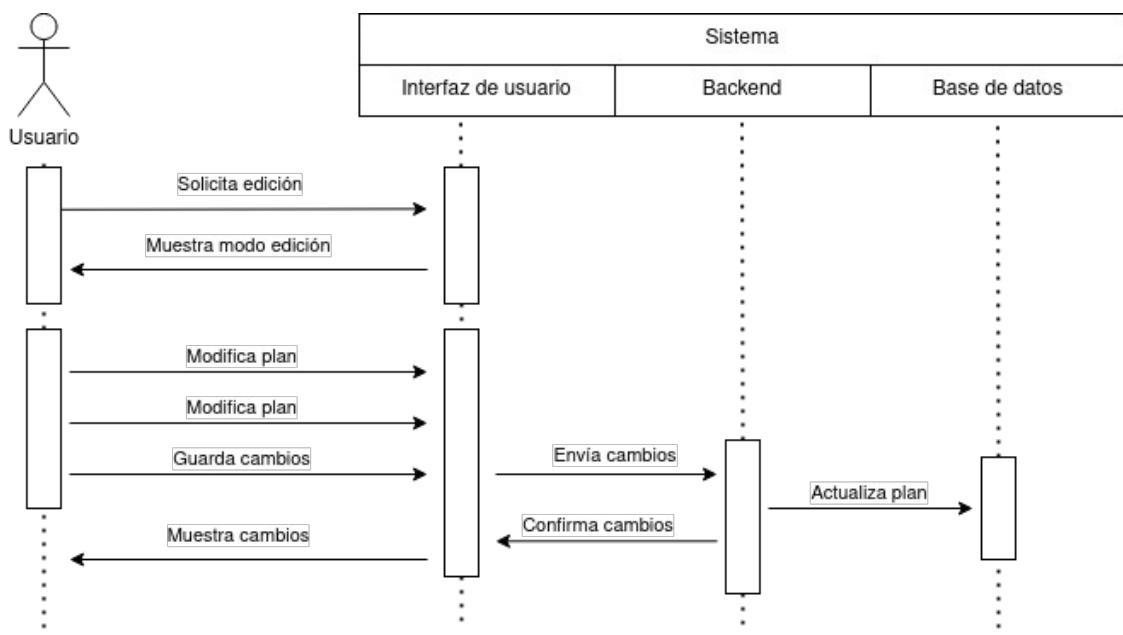


TABLA 3.72: CU7: CREAR NUEVA PUNTUACIÓN DE LUGAR

CU7: Crear nueva puntuación de lugar

Descripción El Usuario desea crear una nueva puntuación

Actor Usuario

Flujo de actividad

1. Usuario pulsa en crear nueva puntuación
2. Usuario busca lugar para puntuar
3. Usuario puntuá lugar

Condiciones de entrada Lugar a puntuar
Puntuación en una escala del 1 al 5

Condiciones de salida Se creará una nueva puntuación

Diagrama de interacción

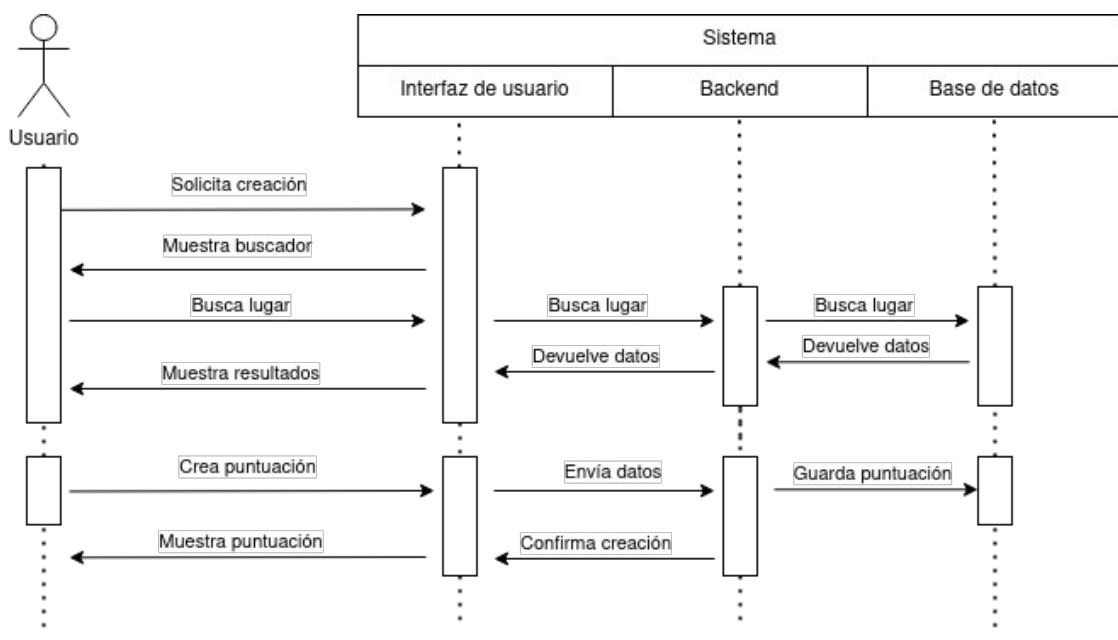


TABLA 3.73: CU8: CAMBIAR PREFERENCIAS

CU8: Cambiar preferencias

<i>Descripción</i>	El usuario desea cambiar sus preferencias
<i>Actor</i>	Usuario
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. El Usuario introduce nuevas preferencias 2. El Usuario guarda los cambios 3. Confirmación de guardado correcto
<i>Condiciones de entrada</i>	Preferencias nuevas válidas
<i>Condiciones de salida</i>	Se realizará un cambio de preferencias

Diagrama de interacción

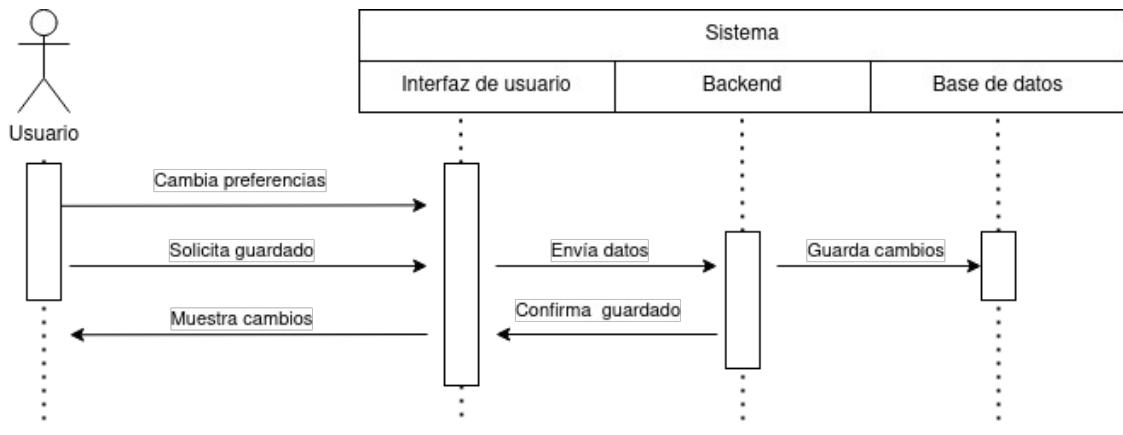


TABLA 3.74: CU9: CAMBIAR DATOS PERSONALES

CU9: Cambiar datos personales

<i>Descripción</i>	El usuario desea cambiar sus datos personales
<i>Actor</i>	Usuario
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. El Usuario introduce nuevos datos personales 2. El Usuario guarda los cambios 3. Confirmación de guardado correcto
<i>Condiciones de entrada</i>	Datos personales nuevos válidos
<i>Condiciones de salida</i>	Se realizará un cambio de datos personales

Diagrama de interacción

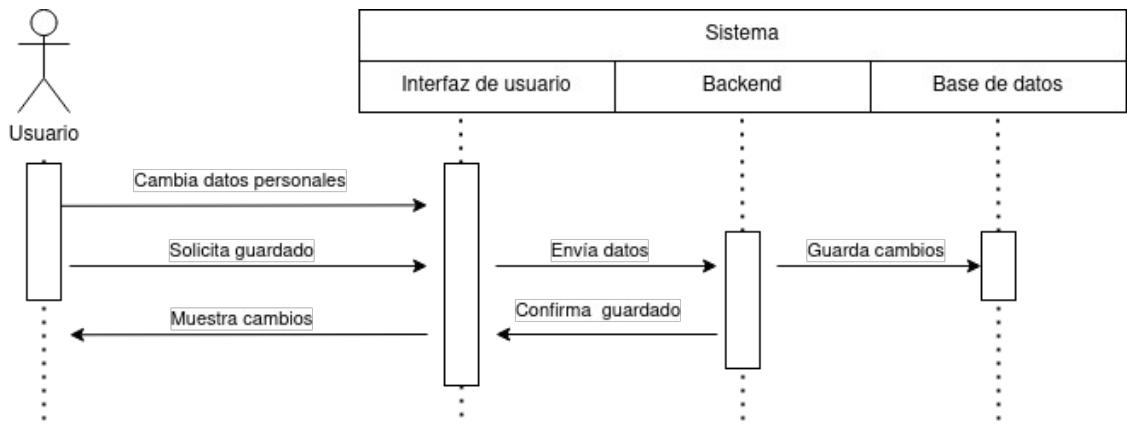


TABLA 3.75: CU10: CERRAR SESIÓN

CU10: Cerrar sesión

Descripción El usuario puede cerrar sesión en la plataforma

Actor Usuario

Flujo de actividad 1. Usuario pulsa Cerrar sesión

Condiciones de entrada Sesión iniciada

Condiciones de salida Se cerrará la sesión

Diagrama de interacción

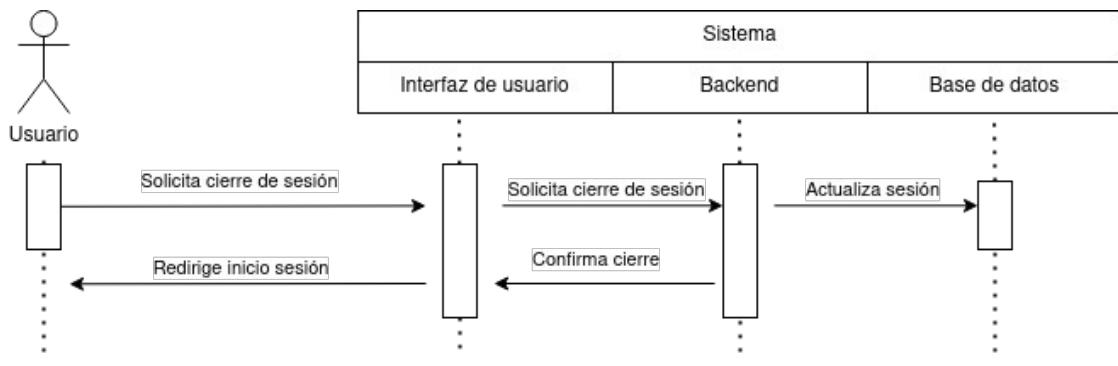


TABLA 3.76: CU11: RECUPERAR CONTRASEÑA

CU11: Recuperar contraseña

<i>Descripción</i>	El usuario desea recuperar la contraseña de su cuenta porque la ha perdido u olvidado
<i>Actor</i>	Usuario y Sistema
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. Usuario solicita recuperar su contraseña 2. Usuario introduce su correo electrónico 3. Sistema comprueba el correo electrónico <ul style="list-style-type: none"> • Si la cuenta existe, envía correo electrónico • Si la cuenta no existe, no envía correo y finaliza el flujo 4. Usuario recibe correo de recuperación 5. Usuario cambia la contraseña
<i>Condiciones de entrada</i>	<p>El usuario no recuerda la contraseña El correo electrónico del usuario es válido</p>
<i>Condiciones de salida</i>	Se producirá un cambio de contraseña

Diagrama de interacción

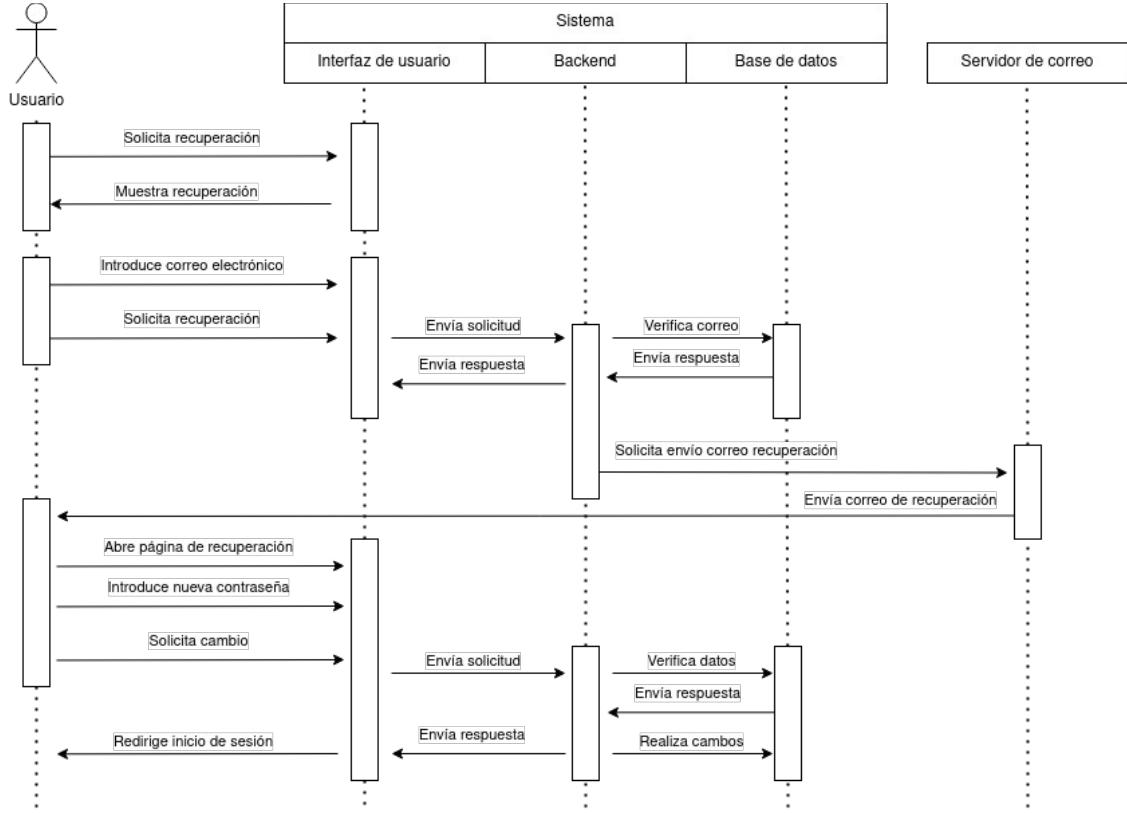


TABLA 3.77: CU12: ENVIAR RECORDATORIO DE PLAN

CU12: Enviar recordatorio de plan

<i>Descripción</i>	La plataforma envía correos electrónicos para recordar al usuario que uno de sus planes empieza al día siguiente de recibirlo
<i>Actor</i>	Sistema
<i>Flujo de actividad</i>	<ol style="list-style-type: none"> 1. Comprobación del día 2. Creación de correo electrónico 3. Envío de correo electrónico
<i>Condiciones de entrada</i>	Día previo al comienzo de un plan activo
<i>Condiciones de salida</i>	Se enviará un correo electrónico recordatorio

Diagrama de interacción

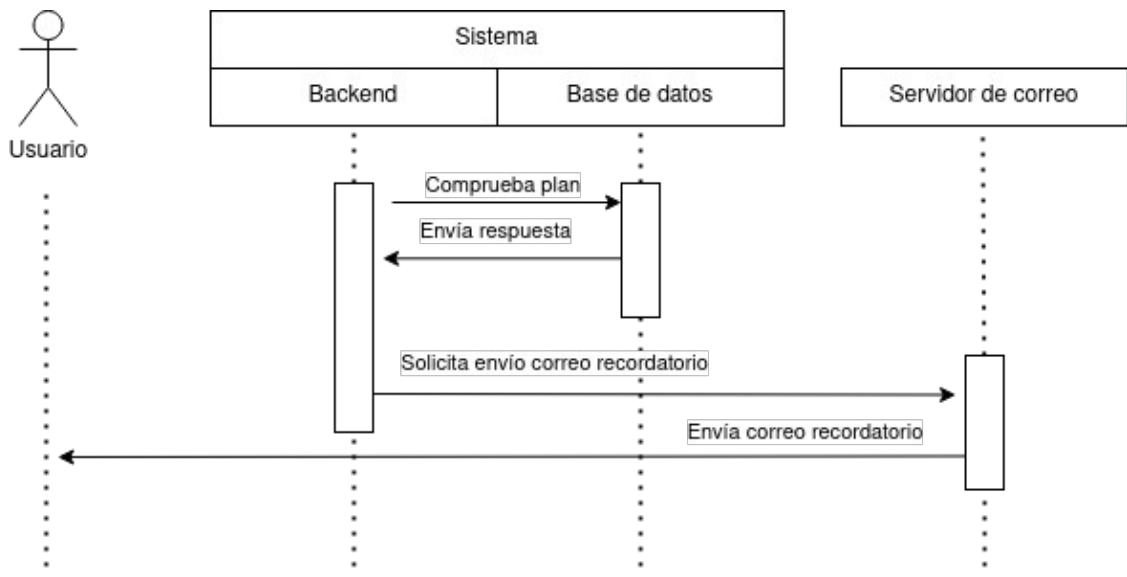


TABLA 3.78: CU13: ENVIAR RECORDATORIO DE PUNTUACIÓN

CU13: Enviar recordatorio de puntuación

Descripción La plataforma envía correos electrónicos para recordar al usuario que puntúe los lugares que ha visitado en el día anterior, para mejorar la precisión de la plataforma

Actor Sistema

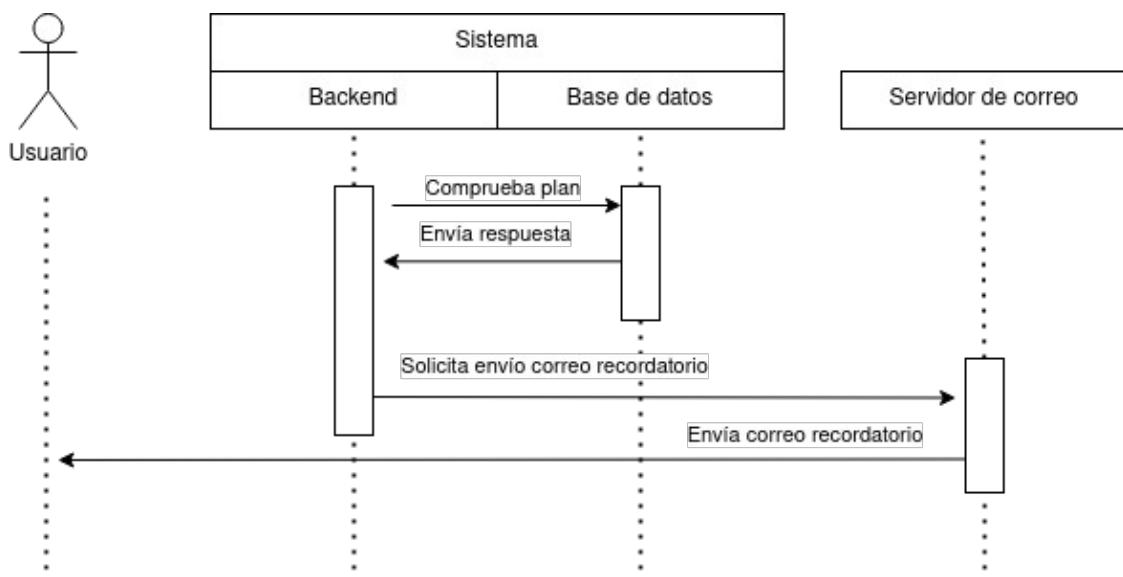
Flujo de actividad

1. Comprobación del día
2. Creación de correo electrónico
3. Envío de correo electrónico

Condiciones de entrada Fin de un día que se encuentra dentro de un plan

Condiciones de salida Se enviará un correo electrónico recordatorio

Diagrama de interacción



3.5. Prototipo

Tras analizar los casos de uso y los requisitos, y a partir de la información expuesta en el *Estado del Arte*, se ha creado el siguiente prototipo HiLo (High Visual + Low Functionality) de la interfaz de usuario.

HERMES

Inicia sesión para continuar

Correo electrónico
userio@example.com

Contraseña
.....

He olvidado mi contraseña

INICIAR SESIÓN

Recordarme

¿No tienes cuenta? [Únete](#)

FIGURA 3.3: INICIO DE SESIÓN

En la Figura 3.3, se muestra el diseño del proceso de inicio de sesión. Este es un proceso sencillo, donde el usuario tiene que introducir su correo electrónico y su contraseña. Adicionalmente se presentan enlaces para crear una cuenta si es necesario o para recuperar la contraseña de forma sencilla

HERMES

Crea una nueva cuenta
¿Tienes cuenta? [Iniciar sesión](#)

Nombre:

Añadir foto de perfil

Apellidos:

Correo electrónico:

Contraseña:

Acepto las [términos y condiciones](#) del servicio

CREAR CUENTA

¡COMENZAMOS!

¿De dónde eres?

¿Qué te gusta?

Selecione sus 3 preferencias
¿Has visitado alguno de estos sitios?

Calificación:

Cargar más sitios
Las recomendaciones se basan en tus preferencias y preferencias. Cuanto más visitas, más precisas serán las recomendaciones.

FINALIZAR

FIGURA 3.4: PROCESO DE REGISTRO

En la Figura 3.5 se muestra el diseño del proceso de registro en la plataforma. Este está dividido en dos partes. La primera parte, solicita al usuario datos personales, como el nombre completo, el correo electrónico y la contraseña. En el segundo paso, se le solicita datos de sus preferencias y puntuaciones de sitios que conozca. Esto se hace para sobrellevar el problema de arranque en frío anteriormente mencionado.

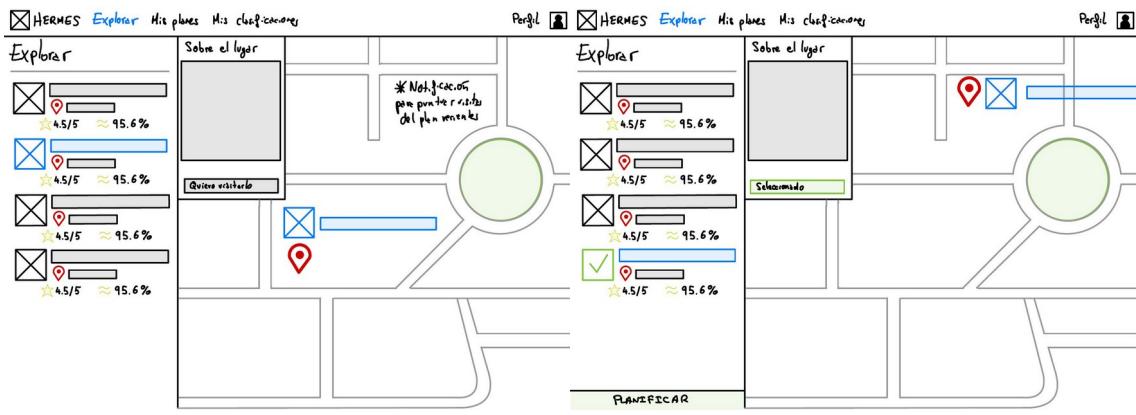
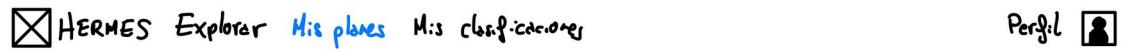


FIGURA 3.5: RECOMENDACIONES

En la Figura 3.5 se muestra el diseño del componente de recomendaciones. Este es considerado la página principal una vez el usuario haya iniciado sesión y es la interfaz en donde puede explorar la ciudad y los sitios que la plataforma le ha recomendado. Este componente se divide en tres secciones: una barra lateral, una ventana emergente de información y un mapa. En la barra lateral aparece un listado de sitios recomendados, junto a información adicional como la dirección, su puntuación media y el porcentaje de recomendación de la plataforma. En la ventana emergente de información, se muestra información sobre el lugar que se haya seleccionado, ya sea en el mapa o en la barra lateral. Además, se muestra un botón que permite añadir el lugar al plan que desee crear.

Una vez hayan añadido más de 3 lugares, un botón aparecerá en la barra lateral que le permitirá crear un nuevo plan o itinerario a partir de los lugares que ha seleccionado para visitar.



Plan actual * Correo electrónico al final del día para puntuar

NOMBRE	03/01/2020	5 días 01/01/2020 - 05/01/2020
Próxima visita		
	RUTA	SALTAR
		GUARDAR
		ELIMINAR

Planes pasados * Si: heg planes futuros, a parecerlo aquí

NOMBRE	5 días 01/01/2019 - 05/01/2019	
PUNTUAR	REPETIR	ELIMINAR

NOMBRE	5 días 01/01/2018 - 05/01/2018
REPETIR	ELIMINAR

FIGURA 3.6: LISTADO DE PLANES

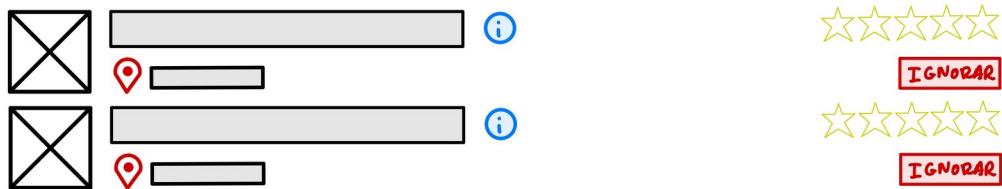
En la Figura 3.6 se muestra el diseño del listado de planes. En este componente, se mostrarán todos los planes, divididos en dos secciones. La sección de planes actuales muestra todos los planes que están activos en el momento o que están pendientes de realizar. En la sección de planes pasados, se muestran los planes que ya han sido completados.

NOMBRE		Duplicar	Editar	Eliminar
2 días 01/01/2020 - 02/01/2020				
01/01/2020				
10:00	12:45	15:30	18:15	21:00
Museo 1 MAPA	PoI 1 MAPA	Restaurante 1 MAPA	Museo 2 MAPA	PoI 2 Restaurante 2 MAPA
02/01/2020				
10:00	12:45	15:30	18:15	21:00
PoI 3 MAPA	PoI 4 MAPA	Restaurante 1 MAPA	Museo 5 MAPA	PoI 5 Restaurante 3 MAPA

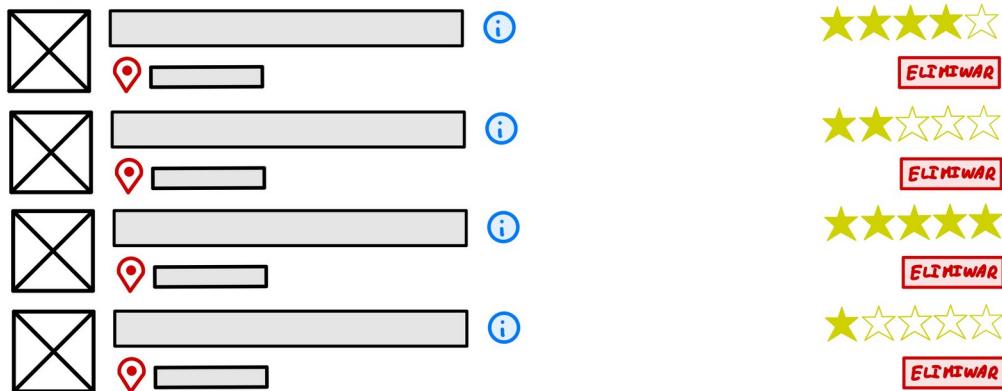
FIGURA 3.7: DETALLE DEL PLAN

Una vez el usuario cree un nuevo plan, se le redirigirá al componente de detalle del plan, cuyo diseño se muestra en la Figura 3.7. En esta vista, se muestra la información del plan, junto a los itinerarios a realizar cada día. Además, se le permite al usuario modificar el plan y la información de este, a su gusto.

Pendientes de calif.cas



Calificados



AÑADIR CALIFICACIÓN

FIGURA 3.8: LISTADO DE PUNTUACIONES

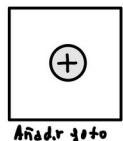
En la Figura 3.8 se muestra el diseño del componente de puntuaciones o calificaciones. De nuevo, este componente se separa en dos secciones. En la primera, se muestran los lugares pendientes de calificar, es decir, los lugares que han sido añadidos a un plan y que no han sido puntuados. En la segunda sección, se muestran todas las puntuaciones que ha realizado el usuario. Además, los usuarios podrán crear puntuaciones cuando deseen, mediante un botón.

Mi perfil

[Cerrar sesión](#)

Mis datos

Nombre _____
Pedro



Apellidos _____
García

Añadir foto
de perfil

Correo electrónico _____
usuario@ejemplo.com

[Cambiar contraseña](#)

[Guardar](#)

Mis preferencias

d'Qué te gusta? _____
 ▽

* Selecciona al menos 3 preferencias

[Guardar](#)

FIGURA 3.9: PERFIL DE USUARIO

Por último, en la Figura 3.9 se diseña la página del perfil de usuario. Desde aquí, el usuario podrá cerrar sesión, gestionar sus datos personales y preferencias y cambiar su contraseña.

3.6. Clases

En este apartado se muestran, de forma tabulada, las clases identificadas y sus atributos. Estas clases se definirán mediante la siguiente plantilla:

TABLA 3.79: PLANTILLA DE CLASES

Nombre		
Descripción		
Atributos		
Nombre	Tipo	Descripción
Atributo 1		

TABLA 3.80: CLASE ACCESSLOG

AccessLog		
<i>Descripción</i>	Almacena un registro por cada petición a la plataforma	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del registro.
level	Número	<p>Nivel del mensaje:</p> <ul style="list-style-type: none"> 0. Mensaje de depuración. 1. Mensaje de información. 2. Mensaje de alerta. 3. Mensaje de error. 4. Mensaje de error grave.
userId	Número	ID de usuario que realizó la petición. 0 si no está identificado.
ip	Texto	Dirección IP del solicitante.
location	Texto	URL que sirvió la petición.
status	Número	Estado HTTP devuelto por el servidor.
message	Texto	Texto adicional.
hostname	Texto	Nombre del servidor que procesó la petición.
worker	Número	Proceso del clúster que sirvió la petición. 0 si solo hay un único proceso.
time	Número	Tiempo en servir la petición, en milisegundos.

TABLA 3.81: CLASE APPLICATIONLOG

ApplicationLog		
<i>Descripción</i>	Almacena un registro de eventos de la plataforma.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del registro.
level	Número	<p>Nivel del mensaje:</p> <ul style="list-style-type: none"> 5. Mensaje de depuración. 6. Mensaje de información. 7. Mensaje de alerta. 8. Mensaje de error. 9. Mensaje de error grave.
userId	Número	ID de usuario que realizó la petición. 0 si no está identificado.
ip	Texto	Dirección IP del solicitante. Vacío si se realizó de forma interna.
location	Texto	URL o función que realizó el evento.
status	Número	Estado HTTP devuelto por el servidor.
message	Texto	Texto adicional.
hostname	Texto	Nombre del servidor que procesó la petición.
worker	Número	Proceso del clúster que sirvió la petición. 0 si solo hay un único proceso.

TABLA 3.82: CLASE CATEGORY

Category		
<i>Descripción</i>	Categoría que un lugar puede tener.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador de la categoría.
fsqId	Texto	Identificador de la categoría en Foursquare.
name	Texto	Nombre de la categoría.

TABLA 3.83: CLASE DISTANCE

Distance		
<i>Descripción</i>	Distancia entre dos lugares. Bidireccional.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
PlaceId1	Número	Identificador del primer lugar.
PlaceId2	Número	Identificador del segundo lugar.
meters	Número	Distancia entre los lugares, en metros.
time	Número	Tiempo entre los lugares, en minutos.
mode	Texto	Modo de viaje. Puede ser “walking” o “car”.

TABLA 3.84: CLASE HOUR

Hour		
<i>Descripción</i>		
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
PlaceId	Número	Identificador del lugar
day	Número	Día. Puede ser: 1. Lunes. 2. Martes. 3. Miércoles. 4. Jueves. 5. Viernes. 6. Sábado. 7. Domingo.
monthStart	Número	Mes de inicio en el que se cumple el horario, siendo 0 Enero y 11 Diciembre. -1 si no hay restricciones de meses.
monthEnd	Número	Mes final en el que termina de cumplirse el horario, siendo 0 Enero y 11 Diciembre. -1 si no hay restricciones de meses.
timeStart	Texto	Hora de inicio, en formato “HHMM”.
timeEnd	Texto	Hora de fin, en formato “HHMM”. Si el formato es “+HHMM”, indica que la hora de fin se encuentra en el día siguiente.

TABLA 3.85: CLASE LOGIN

Login		
<i>Descripción</i>		
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del evento.
UserId	Número	Identificador del usuario.
ip	Texto	Dirección IP desde donde se realizó el inicio de sesión.

TABLA 3.86: CLASE NEIGHBOR

Neighbor		
<i>Descripción</i>	Vecino de un usuario, usado en recomendaciones.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
UserId1	Número	Identificador del primer usuario.
UserId2	Número	Identificador del usuario. Es el vecino del primer usuario.
similarity	Número	Porcentaje de similitud entre usuarios.

TABLA 3.87: CLASE PASSWORDREQUEST

PasswordRequest		
<i>Descripción</i>	Describe una solicitud de recuperación de contraseña.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
key	Texto	Cadena de caracteres o <i>token</i> que identifica la solicitud.
UserId	Número	Identificador del usuario a recuperar la contraseña.

TABLA 3.88: CLASE PLACECATEGORY

PlaceCategory		
<i>Descripción</i>		Relaciona un lugar con una categoría.
Atributos		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
CategoryId	Número	Identificador de la categoría.
PlaceId	Número	Identificador del lugar.

TABLA 3.89: CLASE PLACE

Place		
<i>Descripción</i>		Describe un lugar.
Atributos		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del lugar.
fsqId	Texto	Identificador del lugar en Foursquare.
osmId	Texto	Identificador del lugar en OpenStreetMap.
gmapsUrl	Texto	URL del lugar en Google Maps.
tripadvisorUrl	Texto	URL del lugar en TripAdvisor. Vacío si no existe.
lat	Número	Latitud.
lon	Número	Longitud.
name	Texto	Nombre del lugar.
description	Texto	Descripción del lugar.
timeSpent	Número	Tiempo que un visitante está en el lugar, en minutos.
wikidata	Texto	Identificador del lugar en Wikidata. Vacío si no existe.
wikipedia	Texto	Identificador del lugar en Wikipedia. Vacío si no existe.
wheelchair	Booleano	Indica si el lugar tiene accesos para personas en silla de ruedas.
images	Número	Cantidad de imágenes que tiene el lugar en la

plataforma.

zone	Texto	Zona en la que se encuentra el lugar. Actualmente solo está disponible “Madrid”.
address	Texto	Calle y número.
postalCode	Texto	Código postal.
city	Texto	Ciudad.
state	Texto	Estado o comunidad autónoma.
country	Texto	País.
placeUrl	Texto	URL del lugar. Vacío si no existe.
phone	Texto	Número de teléfono del lugar. Vacío si no existe.
twitter	Texto	Usuario en Twitter del lugar. Vacío si no existe.
facebook	Texto	Usuario en Facebook del lugar. Vacío si no existe.
instagram	Texto	Usuario en Instagram del lugar. Vacío si no existe.
rating	Número	Puntuación media del lugar.
count	Número	Número de puntuaciones recibidas.

TABLA 3.90: CLASE PLANITEM

PlanItem		
<i>Descripción</i>	Ítem de un plan.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del ítem.
PlanId	Número	Identificador del plan.
order	Número	Orden del ítem en el plan.
day	Número	Día que se realiza el ítem en el plan.
startTime	Texto	Hora de inicio de la visita del lugar, en formato “HHMM”.
endTime	Texto	Hora de fin de la visita del lugar, en formato “HHMM”.
timeSpent	Número	Tiempo de visita, en minutos.
type	Número	Tipo de ítem. Puede ser: <ul style="list-style-type: none"> 0. Sin definir. 1. Lugar. 2. Punto de inicio. 3. Tiempo de espera. 4. Tiempo de descanso. 5. Visita personalizada.
travelNext	Número	Tiempo de traslado hasta el siguiente ítem. -1 si es el último ítem del día.
travelMode	Texto	Modo de viaje hasta el siguiente ítem. Puede ser “walking” o “car”.
description	Texto	Notas del usuario sobre el ítem.
PlaceId	Número	Identificador del lugar. Vacío si <i>type</i> es distinto a 1.

TABLA 3.91: CLASE PLAN

Plan		
<i>Descripción</i>	Describe un plan de un usuario.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del plan.
UserId	Número	Identificador del usuario.
name	Texto	Nombre del plan.
description	Texto	Descripción del plan.
startDate	Fecha	Fecha de inicio del plan.
endDate	Fecha	Fecha de fin del plan.
dayStart	Texto	Hora de inicio de cada día, en formato “HHMM”.
dayEnd	Texto	Hora de fin de cada día, en formato “HHMM”.
zone	Texto	Zona donde ocurre el plan. Actualmente solo está disponible “Madrid”.
status	Número	Estado del plan. Puede ser: -1. Error interno. 0. Activo. 1. Planificando. 2. Tiempo de planificación superado. 3. Ruta sin solución. 4. Completado.
startLat	Número	Latitud del punto de inicio.
startLon	Número	Longitud del punto de inicio.

TABLA 3.92: CLASE POPULARTIME

PopularTime		
<i>Descripción</i>	Horas populares del lugar.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
PlaceId	Número	Identificador del lugar.
day	Número	Día. Puede ser: 8. Lunes. 9. Martes. 10. Miércoles. 11. Jueves. 12. Viernes. 13. Sábado. 14. Domingo.
monthStart	Número	Mes de inicio en el que se cumple el horario, siendo 0 Enero y 11 Diciembre. -1 si no hay restricciones de meses.
monthEnd	Número	Mes final en el que termina de cumplirse el horario, siendo 0 Enero y 11 Diciembre. -1 si no hay restricciones de meses.
timeStart	Texto	Hora de inicio, en formato “HHMM”.
timeEnd	Texto	Hora de fin, en formato “HHMM”. Si el formato es “+HHMM”, indica que la hora de fin se encuentra en el día siguiente.

TABLA 3.93: CLASE RATING

Rating		
<i>Descripción</i>	Puntuación de un usuario para un lugar.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
UserId	Número	Identificador del usuario.
PlaceId	Número	Identificador del lugar.
rating	Número	Puntuación para el lugar, entre 1 y 5

TABLA 3.94: CLASE RECOMMENDATION

Recommendation		
<i>Descripción</i>	Recomendación de un lugar para un usuario.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
UserId	Número	Identificador del usuario
PlaceId	Número	Identificador del lugar
probability	Número	Nivel de confianza
from	Texto	Método por el que se generó la recomendación. “cf” para filtrado colaborativo y “cb” para filtrado basado en contenido.

TABLA 3.95: CLASE SESSION

Session		
<i>Descripción</i>	Sesión de un usuario en la plataforma.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
UserId	Número	Identificador del usuario.
key	Texto	Valor de la <i>cookie</i> de la sesión.
maxAge	Número	Tiempo de vida máximo de la sesión, en segundos.

TABLA 3.96: CLASE USERCATEGORY

UserCategory		
<i>Descripción</i>	Preferencia de un usuario.	
<i>Atributos</i>		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
UserId	Número	Identificador del usuario.
CategoryId	Número	Identificador de la categoría.

TABLA 3.97: CLASE USERVIEW

UserView		
<i>Descripción</i>	Número de consultas que un usuario ha realizado para una categoría.	
Atributos		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
UserId	Número	Identificador del usuario.
CategoryId	Número	Identificador de la categoría.
views	Número	Número de consultas para esa categoría.

TABLA 3.98: CLASE USER

User		
<i>Descripción</i>	Usuario de la plataforma.	
Atributos		
<i>Nombre</i>	<i>Tipo</i>	<i>Descripción</i>
id	Número	Identificador del usuario.
name	Texto	Nombre del usuario.
surname	Texto	Apellido(s) del usuario.
country	Texto	País del usuario, en formato ISO 639-2.
email	Texto	Correo electrónico del usuario.
password	Texto	Contraseña del usuario, en formato bcrypt.
attempts	Número	Número de intentos de inicio de sesión fallidos.
lastAttempt	Fecha y hora	Fecha y hora del último intento fallido de inicio de sesión.
lastNeighbor	Fecha y hora	Fecha y hora de la última generación de vecinos.
lastRecommended	Fecha y hora	Fecha y hora de la última generación de recomendaciones.
views	Número	Número de consultas totales.
notificationsPlans	Booleano	Permiso de notificación de recordatorio de planes.
notificationsRatings	Booleano	Permiso de notificación de recordatorio de puntuaciones.

4. IMPLEMENTACIÓN

En este capítulo se describe el proceso seguido para la implementación del proyecto, dividido en 9 secciones:

1. **Entorno operacional:** En este apartado se describe el entorno de desarrollo del software de la plataforma y los sistemas en los que se ha desplegado la plataforma.
2. **Modelo relacional:** En este apartado se describe el modelo de la base de datos.
3. **Obtención y procesado de datos:** En este apartado se describen las tareas realizadas para obtener y procesar los datos necesarios para el funcionamiento de la plataforma.
4. **Interfaz de usuario:** En este apartado se describe la interfaz de usuario implementada y sus características.
5. **Sistema recomendador:** En este apartado se describe el enfoque seguido para la implementación del sistema recomendador.
6. **Sistema planificador:** En este apartado se describe la implementación del sistema planificador.
7. **Backend:** En este apartado se describe la lógica de servidor y la seguridad implementada.
8. **Despliegue:** En este apartado se describe el proceso realizado para el despliegue de la plataforma.
9. **Colaboración:** En este apartado se introducen las normas y reglas creadas para la colaboración en el proyecto de código abierto.

4.1. Entorno operacional

El entorno operacional sobre el que se ejecuta el proyecto se describe en este apartado. En primer lugar, se ha implementado usando los lenguajes Node.js y Rust para la parte de servidor, seguido de los lenguajes HTML, CSS, JavaScript y Bootstrap para la parte de la interfaz de usuario.

Durante el desarrollo se ha llevado un control de versiones mediante la herramienta *git*. El repositorio donde se almacena el código del proyecto se encuentra en la plataforma GitHub de forma pública (<https://github.com/SrGMC/hermes>). Asimismo, el proyecto se organiza en los siguientes directorios:

- **data:** Este directorio contiene los scripts necesarios para cargar los datos en la base de datos. Además, contiene los distintos archivos de base de datos con los

lugares, separados por ciudades, y que facilitan la colaboración de cualquier persona.

- **docs:** Este directorio contiene la documentación del repositorio.
- **scrapers:** Este directorio contiene los scripts que se han creado para la extracción de datos, descrita en el apartado 4.3.
- **server:** Este directorio contiene el código de la implementación del *backend*
- **web:** Este directorio contiene la interfaz web de la plataforma.

La máquina en la que se ha realizado el desarrollo tiene las siguientes características y los siguientes programas:

- Lenovo Thinkpad con procesador AMD Ryzen 7, 64 bits, 4 núcleos y 8 hilos, 2.3 GHz
- 16GB de memoria RAM DDR4 2400MHz
- 512GB de disco M.2 NVMe
- Sistema operativo Fedora 34
- LibreOffice 7.2

Por otro lado, se van a utilizar dos máquinas para el despliegue. Este se realizará mediante contenedores y la herramienta Docker. Para las tareas de distribución se utiliza el software Nginx, que se encuentra posterior a Cloudflare, según la descripción de la Figura 3.2. Las características de ambas máquinas son:

- Odroid HC1 con procesador ARM v7, 32 bits, 8 núcleos, 2.1GHz
- 2GB de memoria RAM
- 128GB de disco SSD SATA 3
- Ethernet 10Mbps
- Sistema operativo Diet-Pi (basado en Debian 10)

4.2. Modelo Relacional

El esquema del modelo relacional de la base de datos se muestra en la Figura 4.1. Las tablas creadas se corresponden a las clases establecidas en el apartado 3.6, pluralizando el nombre de las clases para el nombre de las tablas. Adicionalmente, todas las tablas contienen, de forma interna, los siguientes atributos:

- `createdAt`: Indica la fecha de creación
- `updatedAt`: Indica la fecha de actualización

- deletedAt: Indica la fecha de eliminación. Su valor es NULL si no se ha eliminado. Esto permite mantener las tuplas que se hayan eliminado en base de datos en caso de que sea necesario recuperarlas (por ejemplo, si un usuario elimina su cuenta por accidente).

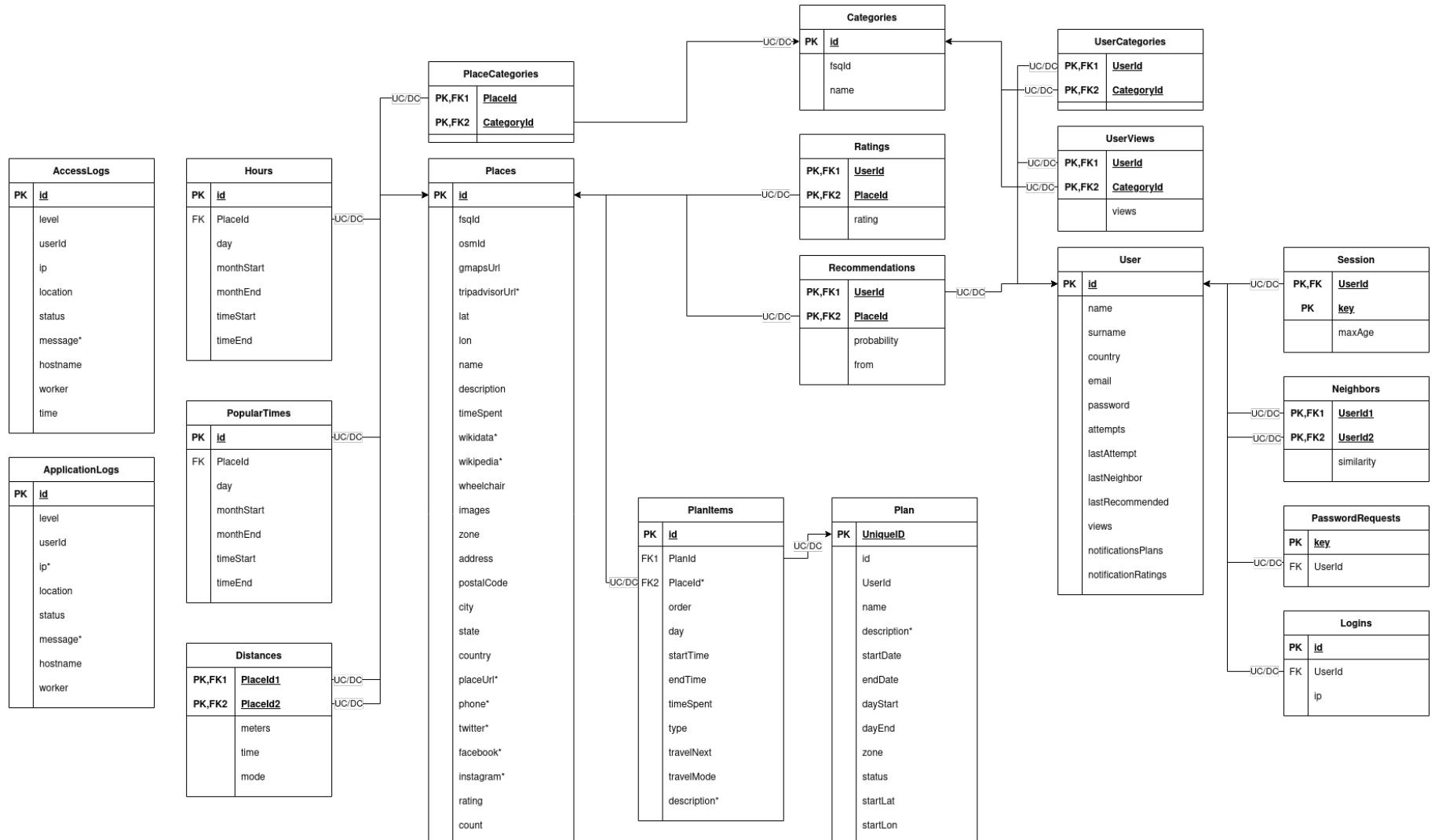


FIGURA 4.1: ESQUEMA RELACIONAL DE LA BASE DE DATOS

4.3. Obtención y procesado de datos

Para generar el contenido de la base de datos, se ha usado la API pública de Foursquare. En primer lugar, se ha explorado la API de Foursquare para conocer las rutas accesibles para obtener información. Cabe destacar que esta API solo permite un máximo de 99.500 peticiones y 500 peticiones premium [125]. Las rutas principales identificadas son las siguientes:

- **/venues/search:** Permite buscar lugares alrededor de unas coordenadas geográficas. Esta ruta tiene un radio limitado a 100km y a 50 resultados máximos. Adicionalmente, se puede ofrecer una lista de categorías para limitar los resultados a lugares que contengan estas.
- **/venues/VENUE_ID:** Devuelve información adicional de un lugar a partir de un identificador en formato hexadecimal.
- **/venues/VENUE_ID/hours:** Devuelve información sobre las horas de apertura de un lugar y sus horas más populares.

Una vez reconocidas las rutas de la API a usar, se ha creado una lista con las categorías de lugares más importantes:

TABLA 4.1: CATEGORÍAS

ID	Nombre	ID de Foursquare
1	Salas de conciertos	5032792091d4c4b30a586d5c
2	Mezquitas	4bf58dd8d48988d138941735
3	Jardines botánicos	52e81612bcfc57f1066b7a22
4	Esculturas al aire libre	52e81612bcfc57f1066b79ed
5	Plazas peatonales	52e81612bcfc57f1066b7a25
6	Salones de ópera	4bf58dd8d48988d136941735
7	Cuevas	56aa371be4b08b9a8d573511
8	Templos	4bf58dd8d48988d13a941735
9	Jardines	4bf58dd8d48988d15a941735
10	Zoológicos	4bf58dd8d48988d17b941735
11	Mercados	4bf58dd8d48988d1fa941735
12	Arte público	507c8c4091d498d9fc8c67a9
13	Lugares conmemorativos	5642206c498e4bfca532186c
14	Museos de arte	4bf58dd8d48988d18f941735
15	Miradores	4bf58dd8d48988d165941735
16	Lagos	4bf58dd8d48988d161941735
17	Estadios	4bf58dd8d48988d184941735
18	Arte callejero	52e81612bcfc57f1066b79ee
19	Miradores elevados	4bf58dd8d48988d133951735

ID Nombre	ID de Foursquare
20 Planetarios	4bf58dd8d48988d192941735
21 Cementerios	4bf58dd8d48988d15c941735
22 Ayuntamientos	4bf58dd8d48988d129941735
23 Galerías de arte	4bf58dd8d48988d1e2931735
24 Faros	4bf58dd8d48988d15d941735
25 Monumentos	4bf58dd8d48988d12d941735
26 Sinagogas	4bf58dd8d48988d139941735
27 Capitolios	4bf58dd8d48988d12a941735
28 Castillos	50aaa49e4b90af0d42d5de11
29 Auditorios	4bf58dd8d48988d173941735
30 Monasterios	52e81612bc57f1066b7a40
31 Jardines de esculturas	4bf58dd8d48988d166941735
32 Templos budistas	52e81612bc57f1066b7a3e
34 Santuarios	4eb1d80a4b900d56c88a45ff
35 Exibiciones zoologicas	58daa1558bbb0b01f18ec1fd
36 Museos de ciencia	4bf58dd8d48988d191941735
37 Plazas públicas	4bf58dd8d48988d164941735
38 Palacios	52e81612bc57f1066b7a14
39 Acuarios	4fceea171983d5d06c3e9823
40 Fuentes	56aa371be4b08b9a8d573547
41 Iglesias y catedrales	4bf58dd8d48988d132941735
42 Juntas municipales	52e81612bc57f1066b7a38
43 Museos de historia	4bf58dd8d48988d190941735
44 Lugares históricos	4deefb944765f83613cdba6e
45 Bahías	56aa371be4b08b9a8d573544
47 Templos Sikh	5bae9231bedf3950379f89c9
48 Puentes	4bf58dd8d48988d1df941735
49 Parques	4bf58dd8d48988d163941735
50 Anfiteatros	56aa371be4b08b9a8d5734db
51 Teatros	4bf58dd8d48988d137941735

Solamente se han escogido categorías que corresponden a puntos de interés de visita, como por ejemplo, Museos, Playas o Plazas, excluyéndose lugares comerciales como Centros Comerciales o Tiendas. A cada uno de las categorías se les ha asignado un identificador numérico, de acuerdo a la Tabla 3.82.

Posteriormente, se han utilizado páginas turísticas como Civitatis o Tripspi, para extraer zonas de interés de donde extraer lugares. Estas zonas se han establecido debido a las limitaciones de la ruta /venues/search de la API de Tripadvisor. Para extraer zonas de interés, se han colocado los lugares de interés más populares de estas plataformas en un mapa, como se muestra en la Figura 4.2, de manera que se puedan identificar clústeres o

zonas de puntos de interés, extrayéndose las zonas que se muestran en la tabla Tabla 4.2.

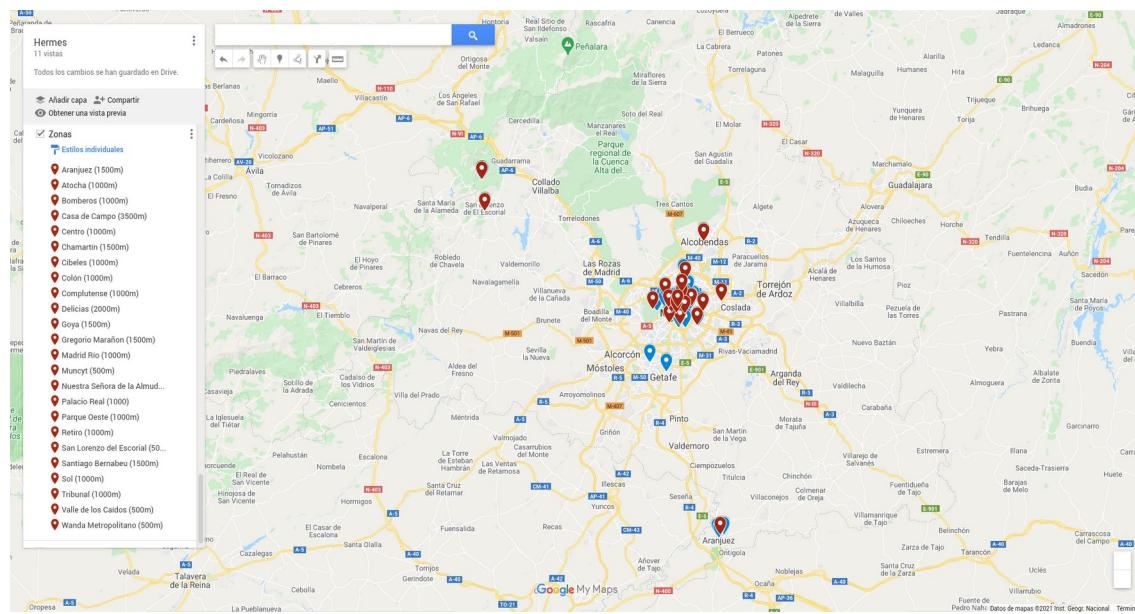


FIGURA 4.2: IDENTIFICACIÓN DE ZONAS CON GOOGLE MY MAPS

TABLA 4.2: ZONAS IDENTIFICADAS

Zona	Coordenadas	Radio de búsqueda
Aranjuez	40.03986, -3.60213	1500m
Atocha	40.40875, -3.69194	1000m
Bomberos	40.39619, -3.65511	1000m
Casa de Campo	40.42329, -3.75875	3500m
Centro	40.41149, -3.70757	1000m
Chamartín	40.47317, -3.68321	1500m
Cibeles	40.41933, -3.69308	1000m
Colón	40.42519, -3.6903	1000m
Complutense	40.44561, -3.72991	1000m
Delicias	40.39712, -3.6961	2000m
Goya	40.42807, -3.66893	1500m
Gregorio Marañón	40.43796, -3.69082	1500m
Madrid Río	40.40018, -3.72076	1000m
MUNCYT	40.53772, -3.64126	500m
Nuestra Señora de la Almudena	40.42019, -3.64231	1000m
Palacio Real	40.41755, -3.71438	1000m
Parque Oeste	40.42649, -3.72191	1000m
Retiro	40.41526, -3.68449	1000m
San Lorenzo del Escorial	40.58885, -4.14846	500m
Santiago Bernabéu	40.45224, -3.69073	1500m

Zona	Coordenadas	Radio de búsqueda
Sol	40.41641, -3.70379	1000m
Tribunal	40.42577, -3.70083	1000m
Wanda Metropolitano	40.43619, -3.59946	500m

Tras identificar las zonas, se ha creado un *script* en JavaScript que realiza peticiones de la ruta `/venues/search`. Estas peticiones se realizan por pares de zonas y categorías, realizando un total de 1050 peticiones. Estas peticiones resultaron en un total de 1223 puntos de interés, que posteriormente se redujeron en un total de 259 puntos de interés. Durante el proceso de filtrado, se eliminaron puntos de interés que no poseían dirección postal o cuyo nombre contuviese las subcadenas “polideportivo” o “campo de futbol”. Tras este pre-proceso de filtrado, se realizó un filtrado de forma semi-manual, automatizado parcialmente mediante el uso de la herramienta Puppeteer, de manera que de forma automática abriese un navegador web con la URL de Foursquare del lugar y la URL del lugar en Google Maps, pudiendo verificar de forma cruzada los datos de cada lugar.

A la hora de decidir si el lugar se debe añadir, se siguieron la siguiente serie de reglas, en donde si una de las respuestas a alguna de estas era negativa, se rechazaba la adición del lugar en la base de datos:

1. ¿El lugar tiene un nombre apropiado?
2. ¿El lugar de Foursquare coincide con el lugar en Google Maps?
3. ¿Es un lugar con interés turístico o histórico?

Por otro lado, si todas las respuestas eran positivas, se buscó el ID de OpenStreetMap para el lugar, la URL del lugar en Tripadvisor (si existiese), la página de Wikipedia del lugar (si existiese), se modificaron los datos incorrectos y se les asignó un ID numérico a cada lugar.

Una vez que se completó el pre-procesado de datos, se ejecutaron varios scripts de extracción de datos:

1. **Extracción de datos de OpenStreetMap:** Gracias a la herramienta Nominatim de OpenStreetMap, se trajeron los siguientes datos adicionales para cada lugar:
 - Descripciones de lugares
 - Enlaces a páginas de Wikipedia
 - IDs de Wikidata
 - Accesibilidad en silla de ruedas

y se corrigieron las coordenadas geográficas y las direcciones postales, ya que los resultados de OpenStreetMap han sido más precisos que los de Foursquare.

2. **Extracción de fotografías de Wikipedia:** Se utilizaron estas fotografías debido a su licencia Creative Commons. Todas las fotografías se convirtieron del formato PNG al formato JPG y cada una de ellas se organizó en directorios nombrados según el ID asignado a cada lugar.
3. **Obtención de datos adicionales de Foursquare:** A partir de cada ID de Foursquare de cada lugar, se extrajeron los datos de contacto del lugar (páginas web, usuarios en las redes sociales y números de teléfono) y sus correspondientes descripciones.
4. **Obtención de horarios de Foursquare:** Al igual que con el proceso anterior, se obtuvieron los horarios y los horarios populares desde Foursquare.

Una vez recogidos todos los datos, se consolidaron en un único archivo JSON, que contiene una lista de objetos donde cada objeto representa un lugar. Además, se corrigieron y añadieron descripciones faltantes de cada lugar.

Finalmente, se desarrollaron una serie de scripts para:

1. Migrar y/o inicializar la base de datos
2. Añadir las categorías a la base de datos
3. Añadir los lugares en la base de datos, excluyendo lugares ya añadidos que coincidan con nombre, latitud, longitud e ID de OpenStreetMap.
4. Añadir las distancias precalculadas entre lugares de la base de datos. Esto se realiza debido al alto coste del servicio de búsqueda de rutas de Google Maps una vez sobrepasados un límite de peticiones.

4.4. Interfaz de usuario

De acuerdo al Capítulo 3, la interfaz del usuario está dividida en varios componentes. Cada uno de estos componentes contiene su propia página en formato HTML, sus propios estilos en CSS y sus propios scripts en JavaScript. Además, se comparten unos estilos y scripts comunes, contenidos en los archivos *common.css* y *common.js*.

Por otra parte, para mantener una consistencia y evitar posibles problemas de compatibilidad, las librerías de Leaflet, Bootstrap, Bootstrap Icons y Font Awesome se alojan junto a los scripts y estilos de cada componente, evitando que estos recursos se soliciten a servidores externos. Asimismo, este alojamiento permite reducir el número de peticiones que se realizan a servidores externos, mejorando la privacidad de los usuarios.

4.4.1. Mapas

Los mapas en la web utilizan un sistema de cuadrículas o *tiles* y niveles de zoom. Cada una de estas cuadrículas tiene un tamaño estándar, de 256 píxeles por 256 píxeles y están asociadas a un nivel de zoom, que va desde 0 (vista de toda la Tierra) hasta 20 (vista a nivel de calle). Un ejemplo de cuadrícula se muestra en la Figura 4.3.



FIGURA 4.3: EJEMPLO DE CUADRÍCULA CON NIVEL DE ZOOM 13

Debido al alto coste de servicios como Google Maps, Mapbox o MapTiler, que ofrecen cuadrículas para mapas de todo el mundo, se va a alojar un servidor basado en OpenTileServer que ofrecerá cuadrículas de mapas para la plataforma.

La generación de estas cuadrículas se extrae de un archivo que contiene el mapa de la Comunidad de Madrid ofrecido por MapTiler.

4.4.2. *Landing page*

El objetivo de la *Landing page* es atraer al usuario para que utilice la plataforma.

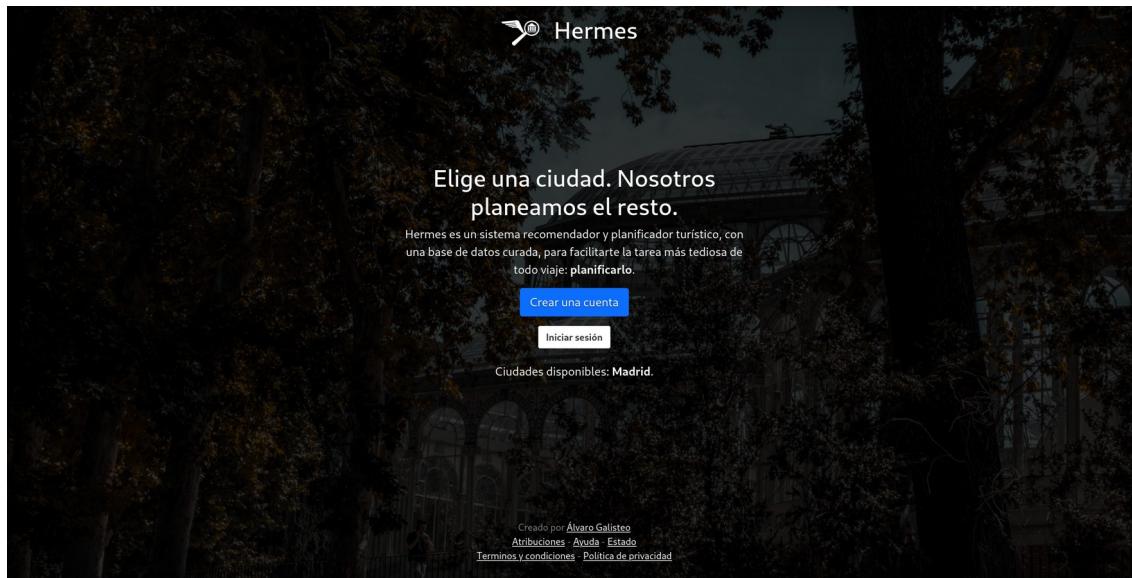


FIGURA 4.4: LANDING PAGE

La página, mostrada en la Figura 4.4, está compuesta por un encabezado y una descripción de la plataforma, junto al logotipo de esta. Dos botones se encuentran en el centro, invitando al usuario a crear una cuenta si no dispone de ella, o a iniciar sesión si dispone de ella. En el fondo, un *carousel* de imágenes va cambiando con una transición de fundido, destacando las ciudades disponibles en la plataforma.

4.4.3. Registro e inicio de sesión

El componente de registro e inicio de sesión se encarga de crear nuevas cuentas e identificar al usuario en la plataforma.

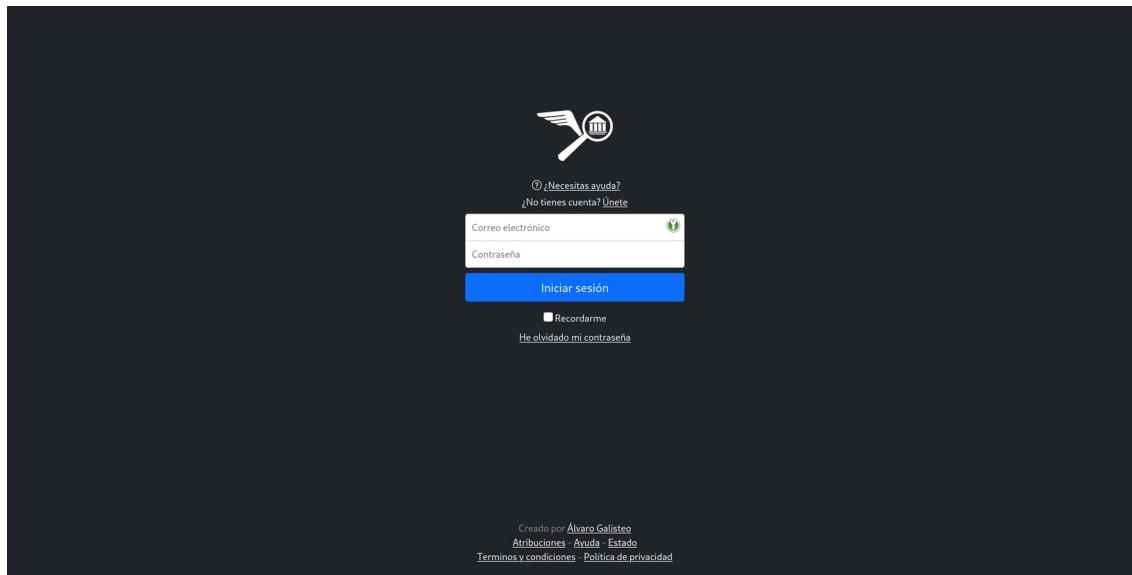


FIGURA 4.5: INICIO DE SESIÓN

Durante el inicio de sesión, mostrado en la Figura 4.5, se solicita al usuario el correo electrónico de su cuenta, su contraseña y si desea que la plataforma le recuerde o no. Si el usuario desea que se le recuerde, su sesión existirá durante 7 días, comparado con las 24h por defecto.

Por otra parte, a la hora de enviar la contraseña al servidor, esta se procesa primero por la función de SHA256. El hash generado por parte de esta función es el que se envía al servidor para su verificación. Este proceso es descrito en el apartado 4.7.3.

The figure consists of two screenshots of a web application. The top screenshot shows a login or registration page with a dark background. It features a logo at the top, followed by a '¿Necesitas ayuda?' link and a 'Ya tienes cuenta? Acceder' link. Below these are input fields for 'Nombre', 'Apellidos', 'Correo electrónico', and 'Contraseña'. A note below the email field says: 'Te recomendamos una contraseña de entre 8 y 16 caracteres y que sea distinta a la que usas en otros servicios.' There is a CAPTCHA section with 'I am human' and a reCAPTCHA logo. A large blue 'Siguiente' button is centered. Below it, a checkbox for accepting the 'Política de privacidad y los términos y condiciones' is shown. At the bottom, there is small text: 'Creado por Álvaro Galisteo', 'Atribuciones - Ayuda - Estado', and links for 'Terminos y condiciones' and 'Política de privacidad'. The bottom screenshot shows a '¿De dónde eres?' section with a dropdown menu for selecting a country. Below it is a grid of category buttons: 'Aqua y costas', 'Animales', 'Arte público', 'Artes escénicas', 'Deportes', 'Monumentos', 'Gobierno', 'Jardines', 'Lugares de culto', 'Zonas públicas', 'Miradores', and 'Museos'. A note below the grid says: 'Selecciona al menos 3 categorías. Las recomendaciones se basan en tus puntuaciones y preferencias.' A large blue 'Crear cuenta' button is at the bottom. To the right, there is a '¿Conoces algún sitio?' section with a note: 'Las recomendaciones se basan en tus puntuaciones y preferencias. Puntúa solo sitios que conoczas o hayas visitado.' It lists several recommended sites with star ratings: 'Viaducto de Segovia' (4 stars), 'Iglesia de San Miguel Arcángel de Chamartín' (4 stars), 'La Casa Encendida' (4 stars), 'Espacio Ronda' (4 stars), and 'Iglesia Catedral de las Fuerzas Armadas' (4 stars). Buttons for 'Buscar' and 'Cargar más sitios' are available, along with a large blue 'Finalizar' button.

FIGURA 4.6: REGISTRO

El registro, mostrado en la Figura 4.6, se encarga de crear una nueva cuenta para el usuario. Durante el proceso de registro, se solicita al usuario su nombre completo, su correo electrónico y una contraseña. Al igual que el inicio de sesión, la contraseña se procesa por la función SHA256 antes de enviarse al servidor. Además, para proteger a la plataforma ante *bots*, se requiere que el usuario cumpla un CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). Este CAPTCHA es creado por el servicio hCaptcha. Se ha elegido el servicio hCaptcha debido a su buena política de privacidad.

En el registro también se muestra el proceso implementado para recopilar la información necesaria, de forma que se evite el problema de arranque en frío y de dispersión. Antes de poder crear su cuenta, se le solicita al usuario que seleccione al menos tres categorías, organizadas por supercategorías. Estas categorías son las definidas en la Tabla 4.1. Asimismo, se le solicita al usuario que puntúe sitios que conozca y haya visitado. Ya que el proceso de puntuar a menudo puede ser tedioso, para reducir la fricción por parte del usuario se muestran 5 sitios de forma aleatoria, cuyo listado puede refrescarse generando más sitios aleatorios. También se ofrece la posibilidad al usuario de buscar lugares por su nombre. La puntuación de lugares no es requerida.

A diferencia del prototipo mostrado, el paso de selección de categorías y puntuaciones se ha dividido en dos pasos adicionales, facilitando el uso al usuario. Además, se ha simplificado y se ha hecho más visual la forma de añadir categorías, pasando de un campo de texto a un listado de ítems.

4.4.4. Recuperación de contraseña

En caso de que un usuario legítimo haya olvidado su contraseña, este puede recuperarla de forma sencilla, como se muestra en la Figura 4.7. Para comenzar, debe introducir su correo electrónico en el modal que se abre tras pulsar en *He olvidado mi contraseña*, en la pantalla de inicio de sesión.

Si el correo electrónico existe en la base de datos, se le enviará un correo con un enlace temporal para cambiar su contraseña. Si accede a este enlace, se le dirigirá a una página para recuperar su contraseña.

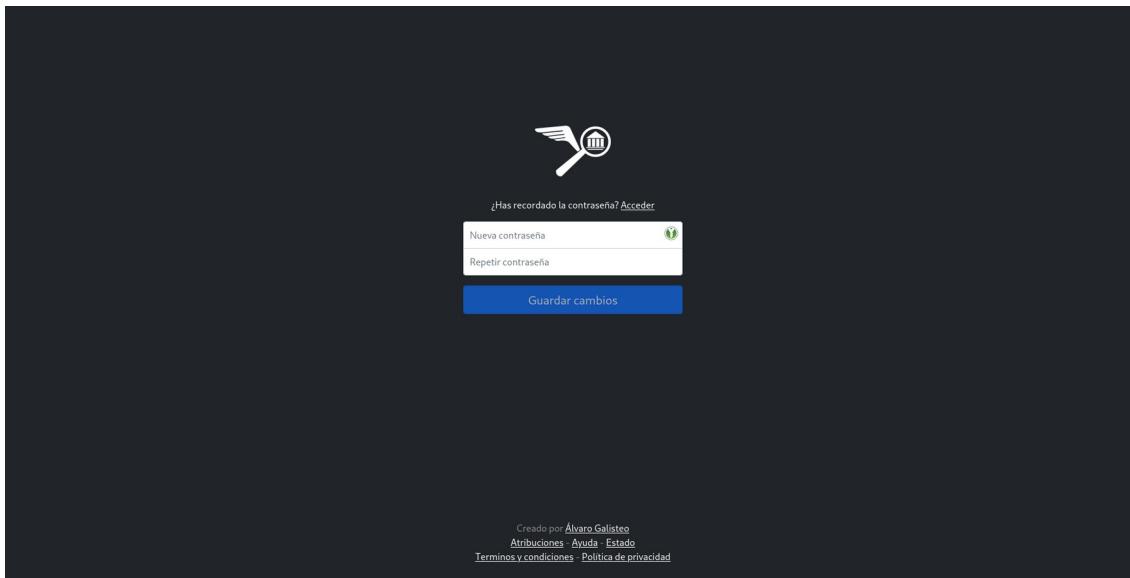


FIGURA 4.7: RECUPERACIÓN DE CONTRASEÑA

4.4.5. Recomendaciones

Este componente es la página principal de la plataforma tras iniciar sesión. Esta hace uso de un listado de lugares recomendados para el usuario junto a un mapa interactivo.

Left Screenshot (List View):

- Parque del Emir Mohamed I**: Recinto monumental de Madrid, situado en el barrio de Palacio del Retiro. Ofrece una gran variedad de espacios públicos con un sobre todo de estilo andaluz.
- Plaza de la Armería**: Famosa plaza que se encuentra entre la Catedral de la Almudena y la fachada sur del Palacio Real.
- Faro de Moncloa**: Torre de telecomunicaciones de 92 m construida en 1992, en cuya observación al alcance todos los gozos.
- Cerro Garabitas**: Cerro ubicado en el parque madrileño de la Casa de Campo, escenario de diversas batallas durante la Guerra Civil y la Batalla de Madrid.
- Palacio de La Moncloa**: Palacio ubicado dentro un complejo del gobierno, que sirve de Planificar

Right Screenshot (Detail View):

Lago de la Casa de Campo
Área recreativa suburbana con alquiler de kayaks y canoas, paseos junto al lago y restaurantes con parrilla.

• 3h 15min | ★ 5.0 | 96%

Real Jardín Botánico Alfonso XIII
Jardines botánicos, Parques
Situado en la Universidad Complutense de Madrid, fue construido como espacio central de investigación y divulgación botánica.

• 1h 45min | ★ 5.0 | 96%

FIGURA 4.8: RECOMENDACIONES

Similar al prototipo mostrado, la interfaz se divide en dos partes, según se muestra en la Figura 4.8. En la versión de escritorio, a la izquierda, se muestra un lista de lugares recomendados para el usuario, mientras que a la derecha se muestra un mapa interactivo con la información de cada lugar. Adicionalmente, se ha incluido la descripción del

lugar directamente en la barra lateral, sustituyéndola por la dirección que aparecía en el prototipo.

Además, esta interfaz se adapta fácilmente a la versión de móvil. Debido al tamaño reducido de pantalla, las dos partes de la interfaz de escritorio se separan de forma individual. Mientras que se muestra el mapa de forma permanente, el listado de lugares se puede mostrar u ocultar con el botón de la parte superior derecha.

Para facilitar la navegación en dispositivos móviles, el menú que se encontraba en la parte superior en la versión de escritorio se ha desplazado a la parte inferior en dispositivos móviles. Este cambio se propaga por el resto de componentes de la interfaz.

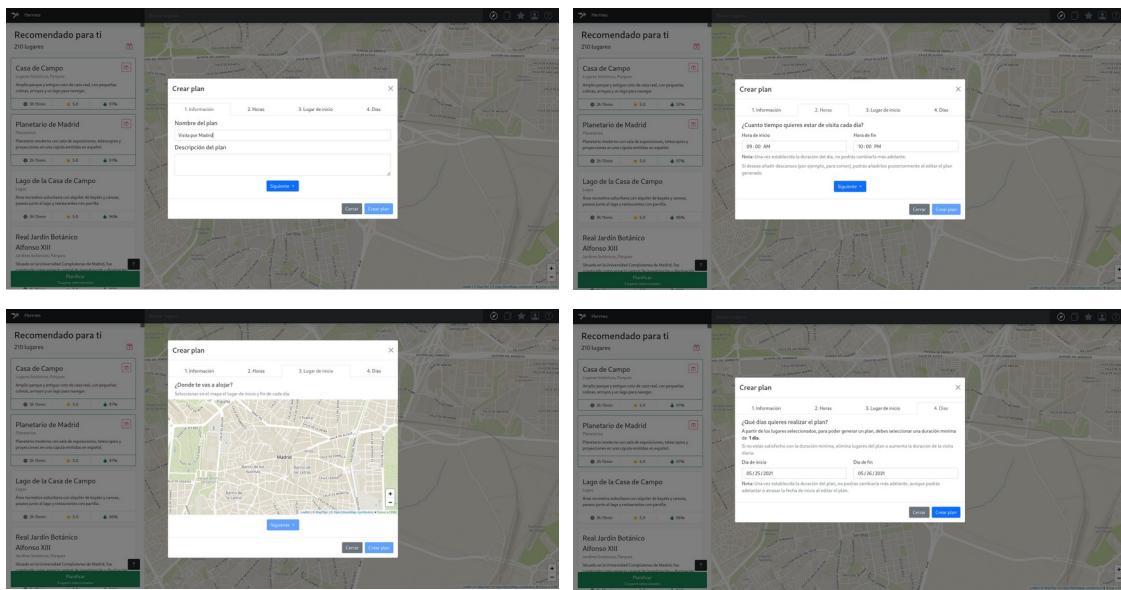


FIGURA 4.9: MODAL DE PLANIFICACIÓN

Por otra parte, como se muestra en la Figura 4.9, la creación de un nuevo plan está integrada en la interfaz de recomendaciones. Una vez el usuario haya añadido al plan un mínimo de 3 lugares (que aparecerán marcados en el mapa y en el listado de lugares), se activará el botón para planificar.

Una vez pulsado, se le guiará al usuario por el proceso de recogida de datos iniciales. En primer lugar, se solicita el nombre y una descripción para el plan, seguido del establecimiento de la duración de cada día de visita. Una vez establecida la duración, se solicitará al usuario que seleccione en un mapa el punto de inicio y fin de cada día.

Tras introducir todos los datos, estos se envían al servidor para calcular una duración mínima, de forma que el sistema planificador pueda encontrar una solución al problema de planificación. Este cálculo se detalla en el apartado 4.6. Finalmente, se solicita al usuario que establezca la duración en días del viaje o visita y se comienza a planificar el itinerario.

4.4.6. Planes

Este componente muestra todos los planes que el usuario ha creado, además de mostrar los itinerarios creados en detalle, permitiendo modificarlos y eliminarlos. Es por ello que está dividido en dos partes.

The screenshot shows the Hermes application interface. At the top, there is a dark header bar with the Hermes logo and several icons on the right. Below the header, the main content area is divided into two sections: 'Planes activos' (Active plans) and 'Planes antiguos' (Old plans).

The 'Planes activos' section has a light blue header bar with the text 'Sin planes activos' (No active plans).

The 'Planes antiguos' section has a white header bar with the text 'Museística Madrileña'. Below this, it says '3 días | 18/04/2021 → 20/04/2021'. It lists a single plan entry:
- Name: Museística Madrileña
- Duration: 3 days
- Dates: 18/04/2021 → 20/04/2021
- Status: Completado (Completed)
- Buttons: 'Ver plan' (View plan) and 'Eliminar' (Delete)

At the bottom of the page, there is a footer with links: 'Creado por Álvaro Galisteo', 'Atribuciones - Ayuda - Estado', 'Terminos y condiciones - Política de privacidad'.

FIGURA 4.10: LISTADO DE PLANES

La primera parte, mostrada en la Figura 4.10, se encarga de listar todos los planes del usuario, divididos, según lo establecido en apartado 3.1, en planes activos y planes antiguos. Estos planes pueden tener varios estados:

- **Activo**
- **En proceso de planificación**
- **Sin solución**
- **Completado**
- **Error interno**

El estado del plan se muestra, además de en el listado de planes, al consultar un plan en detalle, como se muestra en la Figura 4.11.

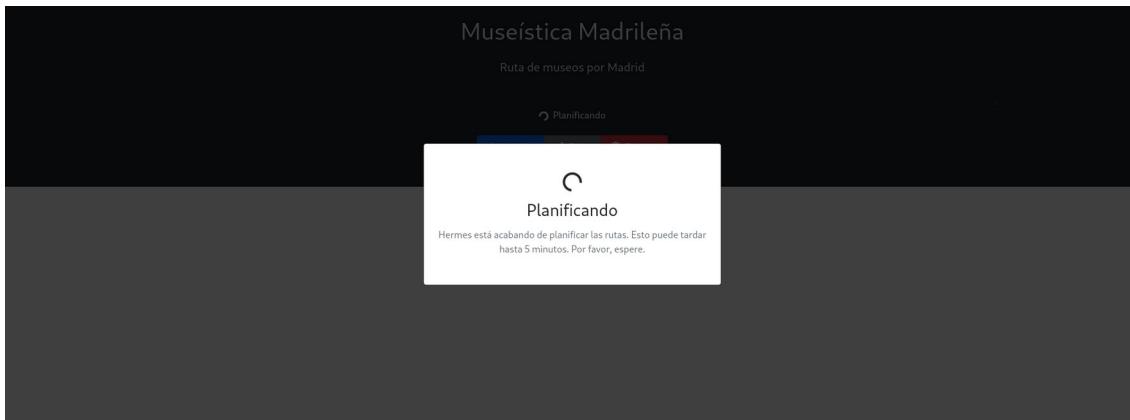


FIGURA 4.11: INFORMACIÓN DEL ESTADO DE UN PLAN

Además de poder ver los planes, a los usuarios se les ofrece la posibilidad de eliminarlos con previa confirmación.

The image displays two side-by-side screenshots of the Hermes app interface. The left screenshot shows the "Museística Madrileña" route plan for April 18, 2021, with a total duration of 3 days. It lists three stops: CaixaForum Madrid, Museo Nacional Centro de Arte Reina Sofía (MNCARS), and Museo Nacional de Ciencia y Tecnología. Each stop includes a thumbnail image, a summary, walking time, and social sharing buttons. The right screenshot shows the same route plan after it has been completed on April 20, 2021. The "Completed" status is indicated at the top, and the "Edit" and "Delete" buttons have been replaced by "Save" and "Cancel" buttons. The stop details remain the same, showing the same information as the left screenshot.

FIGURA 4.12: DETALLE DEL PLAN (IZQUIERDA) Y EDICIÓN DEL PLAN (DERECHA)

En la parte de detalle del plan, mostrada en la Figura 4.12, el usuario puede ver el plan generado por la plataforma. A diferencia del prototipo diseñado, el itinerario se muestra de forma vertical en vez de forma horizontal, de manera que sea más sencillo explorar el

plan desde la interfaz móvil, mostrada en la Figura 4.13. Adicionalmente, los días se muestran uno por uno, a diferencia de todos en la misma página como se diseño en el prototipo. De esta manera, se enfatiza cada día.

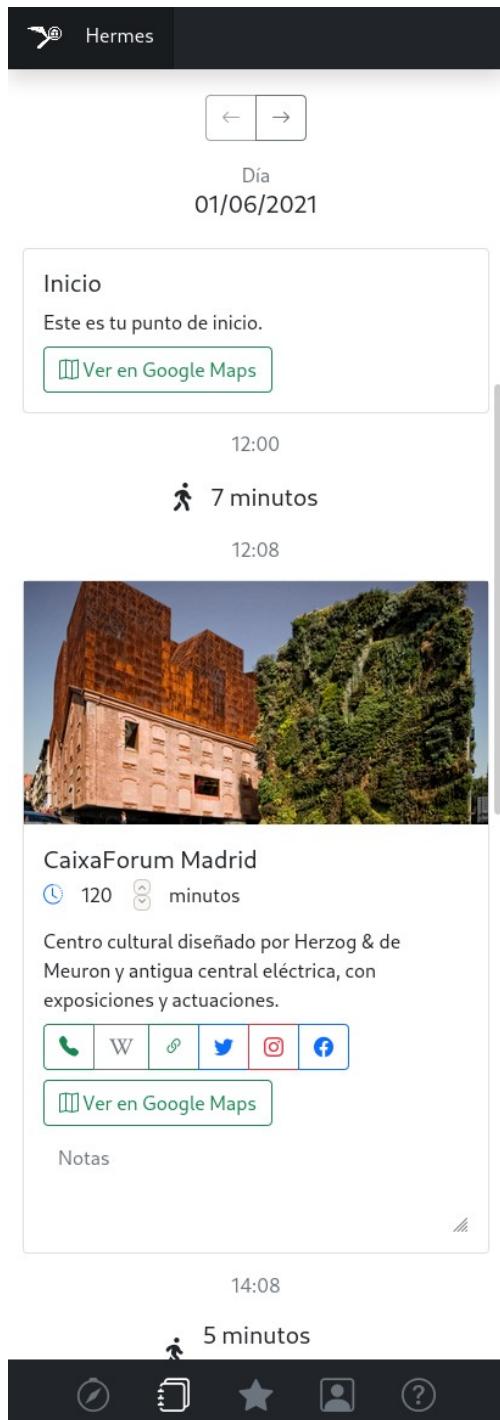


FIGURA 4.13: VISTA MÓVIL DEL PLAN

Una característica adicional que contiene la vista detallada se encuentra cuando el día actual se encuentra entre medias de los días de la visita de un plan. En este caso, se

cambia la ruta del día mostrada al día actual. Si el día actual se encuentra fuera de este período, se muestra por defecto la ruta del primer día.

Por otra parte, cuando el usuario entra en modo edición, la interfaz cambia, sustituyendo los tiempos de traslado por botones para insertar lugares entre dos lugares actuales. Los tipos de lugares disponibles son:

- **Inicio:** Es el punto de inicio y fin que se establecido al crear el plan.
- **Lugar:** Es un lugar que se encuentra en la base de datos y que ha sido añadido al plan antes de crearlo.
- **Descanso:** Es un período de descanso, como por ejemplo, la hora de la comida.
- **Espera:** Es un período de espera debido a que el siguiente lugar a visitar todavía no ha abierto.
- **Personalizado:** En un lugar personalizado que el usuario ha creado manualmente y que no se encuentra en la base de datos.

Además de poder insertar lugares, el usuario puede cambiar la duración de cada visita, añadir notas, eliminar lugares, modificar el título y la descripción y adelantar/trasesar la fecha de inicio (mientras la fecha sea posterior al día actual). Cuando el plan se guarde, se recalcularán los tiempos y las distancias en el servidor.

Finalmente, al igual que en el listado de planes, el usuario puede eliminar un lugar, con confirmación previa.

4.4.7. Puntuaciones

Este componente muestra al usuario las puntuaciones que ha creado, además de permitir al usuario crear nuevas puntuaciones y gestionar las existentes.

Las puntuaciones se organizan en dos secciones, como se muestra en la Figura 4.14: Pendientes y Realizadas. Las puntuaciones pendientes contienen lugares que el usuario aún no ha puntuado y que están presentes en planes que ha creado. Las puntuaciones realizadas son todas las puntuaciones que el usuario ya ha creado.

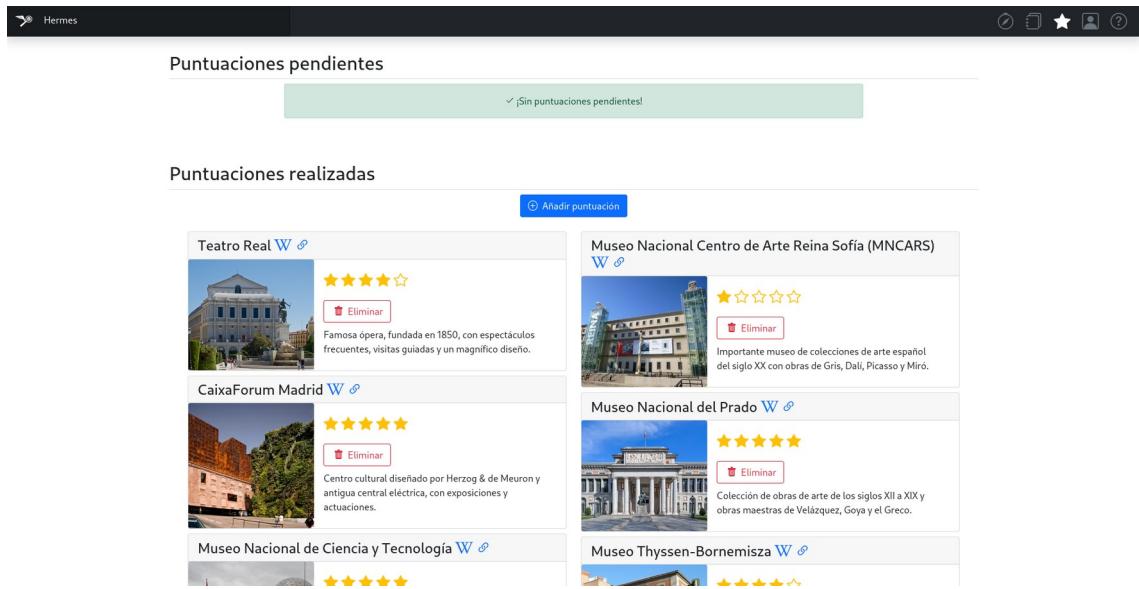


FIGURA 4.14: PUNTUACIONES DEL USUARIO

A diferencia de lo propuesto en el prototipo, las puntuaciones se ordenan en columnas, en vez de por filas. Además, las casillas de las puntuaciones muestran más información, como la imagen, la descripción, y enlaces a las páginas web y a la Wikipedia, a diferencia de la versión del prototipo cuya información quedaba relegada a un botón.

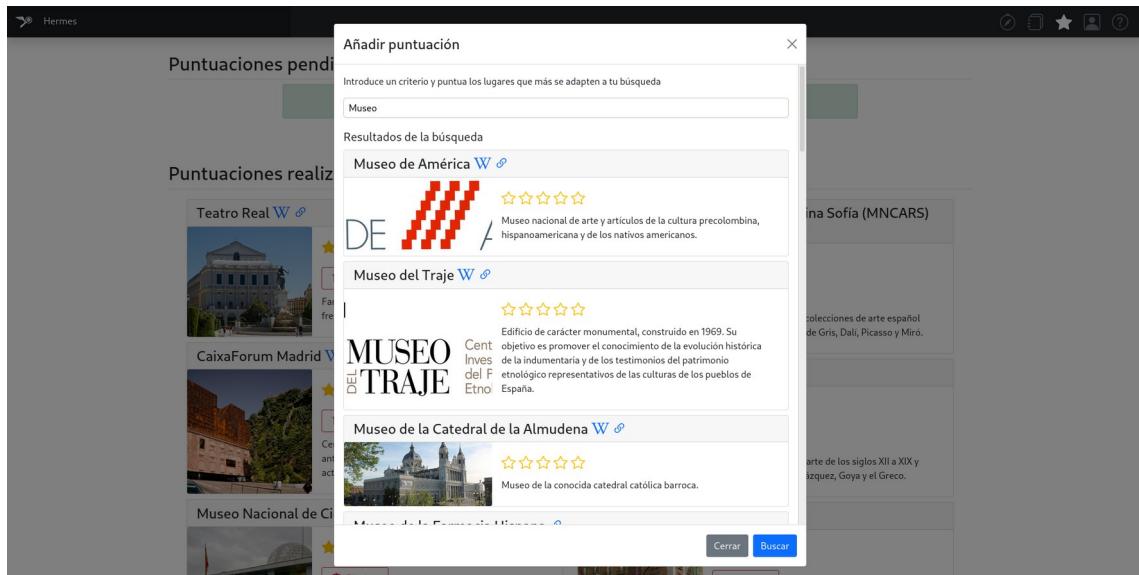


FIGURA 4.15: CREAR PUNTUACIÓN

En el caso de que el usuario desee crear una nueva puntuación, este deberá pulsar en el botón para añadir una nueva puntuación. Esto abrirá un modal, mostrado en la Figura 4.15, donde podrá buscar el nombre del lugar a puntuar y crear las puntuaciones de acuerdo a los resultados de la búsqueda. Las puntuaciones son guardadas de forma automática sin necesidad de confirmación.

4.4.8. Perfil

En el componente de perfil de usuario, el usuario podrá ver, modificar, descargar y eliminar todos sus datos personales, acorde a la política de privacidad creada; realizar gestiones de seguridad, como el cambio de contraseña o el cierre de sesiones; y modificar sus preferencias de notificaciones por correo.

Estas configuraciones están divididas por pestañas, mostradas en la Figura 4.16, donde cada una de ellas recoge un conjunto de configuraciones relacionadas. El cierre de sesión se encuentra dividido en dos partes. La primera se encuentra en la pestaña “Cuenta” y permite cerrar la sesión actual. La segunda se encuentra en la pestaña “Seguridad” y permite cerrar todas las sesiones iniciadas, excepto esta. En caso de que el usuario desee eliminar su cuenta, podrá hacerlo desde la pestaña “Avanzado”. Desde aquí podrá, además, descargar toda su información en formato JSON.

The figure consists of four screenshots of the Hermes application's profile interface, arranged in a 2x2 grid. Each screenshot shows a different tab selected:

- Top Left (Notificaciones Tab):** Shows settings for notifications, including 'Notificaciones de plan' (Plan notifications) and 'Notificaciones de puntuaciones' (Scoring notifications). It also displays the user's name ('Alvaro Galisteo') and links to 'Atribuciones', 'Ayuda', 'Estado', 'Terminos y condiciones', and 'Política de privacidad'.
- Top Right (Cuenta Tab):** Shows contact information fields for 'Nombre' (Name), 'Apellido' (Last Name), and 'Correo electrónico' (Email). It also displays the user's email ('alvaro@galisteo.me').
- Bottom Left (Seguridad Tab):** Shows a 'Cambiar contraseña' (Change password) form with fields for 'Nueva contraseña' (New password) and 'Repetir contraseña' (Repeat password). It also displays the user's name ('Alvaro Galisteo') and links to 'Atribuciones', 'Ayuda', 'Estado', 'Terminos y condiciones', and 'Política de privacidad'.
- Bottom Right (Avanzado Tab):** Shows 'Preferencias' (Preferences) for site categories. It lists various categories in a grid: Agua y costas, Animales, Arte público, Artes escénicas, Deportes, Monumentos, Gobierno, Jardines, Lugares de culto, Zonas públicas, Miradores, and Museos. A note says 'Selecciona al menos 3 categorías'. It also displays the user's name ('Alvaro Galisteo') and links to 'Atribuciones', 'Ayuda', 'Estado', 'Terminos y condiciones', and 'Política de privacidad'.

FIGURA 4.16: PERFIL DE USUARIO

4.4.9. Ayuda

Este componente contiene la documentación de ayuda sobre la plataforma. Como se observa en la Figura 4.17, la ayuda está dividida en varias partes. La primera destaca los artículos de ayuda junto a enlaces para reportar errores y sugerir nuevos lugares y ciudades.



FIGURA 4.17: AYUDA DE HERMES

En la parte inferior a esta se añaden enlaces para enviar mensajes por correo electrónico, para los casos en los que el usuario necesite más ayuda proporcionada por estas páginas. Además, se añaden enlaces a los términos y condiciones, a la política de privacidad y al repositorio público de GitHub.

4.5. Sistema recomendador

Según la especificación del Capítulo 3, el sistema recomendador tiene una estructura híbrida, compuesta por una enfoque de recomendación de filtrado colaborativo basado en usuarios y filtrado basado en contenido.

4.5.1. Filtrado colaborativo basado en usuarios

El enfoque de recomendación de filtrado colaborativo introduce el concepto de vecinos. Los vecinos de un usuario son otros usuarios cuya medida de similitud con el usuario objetivo es muy alta. La generación de estos vecinos se realiza calculando la similitud entre usuarios usando la métrica MPIP [38], descrita en el *Estado del Arte*. Para ello, se obtienen los N usuarios más similares al usuario objetivo.

TABLA 4.3: SIMULACIÓN CÁLCULO DE VECINOS

N	Usuarios	Tiempo (ms)	N	Usuarios	Tiempo (ms)
5	100	22	15	10000	39
10	100	21	20	10000	39
15	100	21	5	100000	205
20	100	21	10	100000	215
5	1000	46	15	100000	216
10	1000	23	20	100000	235
15	1000	22	5	1000000	2294
20	1000	22	10	1000000	2271
5	10000	37	15	1000000	2398
10	10000	29	20	1000000	2311

En la Tabla 4.3 se muestra el resultado de simular, usando Node.js, el cálculo de vecinos para un usuario en función de la cantidad de usuarios que hay en la base de datos. A partir de estos resultados, se observa que el tiempo de generación de vecinos es dependiente de la cantidad de usuarios en la base de datos. Similarmente, el cálculo de similitud es una tarea de cómputo intensiva.

Además, esta base de datos no es estable y va cambiando de forma constante, por lo que, por motivos de rendimiento, no es viable calcular la similitud de todos los usuarios con todos los usuarios, sobretodo cuando ocurran cambios en base de datos. Este problema es el denominado anteriormente como problema de escalabilidad.

Para solucionarlo, a la hora de generar vecinos, se seleccionan de forma aleatoria M (mayor que N) usuarios, que son usados para calcular la similitud con el usuario objetivo, independientemente de si el usuario ya es vecino. Una vez calculadas las similitudes, se seleccionan los N usuarios más similares. Posteriormente, se eliminan las asignaciones de vecinos del usuario objetivo ya existentes en la base de datos y se sustituyen por los nuevos vecinos computados. El proceso de generación de vecinos se muestra en la Figura 4.18.

Generación de vecinos

Entrada:

Usuario objetivo (U)

N mejores vecinos

M usuarios aleatorios

Salida:

Vecinos del usuario objetivo

P: Puntuaciones de U

V_m: M usuarios aleatorios

Para todo m en V_m:

S[m] = similitud_{MPIP}(m, U)

Ordenar S de mayor a menor

R = Coger N primeros usuarios de S

Devolver R

FIGURA 4.18: ALGORITMO DE CÁLCULO DE USUARIOS SIMILARES

Una vez encontrados los vecinos del usuario objetivo, se seleccionan todos los lugares que han sido puntuados por al menos uno de los vecinos, y para cada lugar se calcula un nivel de confianza, descartándose los lugares cuya confianza sea inferior al 60%. El algoritmo de generación de recomendaciones se describe en la Figura 4.19.

Generación recomendaciones (CF)

Entrada:

Usuario objetivo (U)

Vecinos de U (V_u)

Salida:

Lugares recomendados

L: Lugares visitados por usuarios de V_u

R: Recomendaciones = \emptyset

Para todo l en L:

$$p = \text{probabilidad}_{\text{CF}}(l, U)$$

Si $p > 0.6$:

$$R = R \cup \{l\}$$

Devolver R

FIGURA 4.19: ALGORITMO DE GENERACIÓN DE RECOMENDACIONES MEDIANTE FILTRADO COLABORATIVO

4.5.2. Filtrado basado en contenido

Por otra parte, en la generación de recomendaciones mediante filtrado basado en contenido, se tiene en cuenta las preferencias del usuario y las consultas del usuario para cada categoría. Cuando un usuario puntúa un lugar con unas determinadas categorías, se registra que el usuario ha realizado una consulta más para cada una de las categorías del lugar. Este valor de consultas también se incrementa considerando todos los lugares de un plan cuando este finaliza.

En primer lugar, para generar recomendaciones por filtrado basado en contenido, se calcula el porcentaje de consultas de un usuario para cada una de las categorías. Posteriormente se le asigna un valor del 100% a todas las categorías que coincidan con las preferencias del usuario, y se calcula el porcentaje de recomendación para todos los lugares en la base de datos. Este porcentaje de recomendación es la media de todos los porcentajes de consultas de las categorías que correspondan a cada lugar. Finalmente, se descartan los lugares cuyo porcentaje de probabilidad es inferior al 60%. El algoritmo de generación de recomendaciones se describe en la Figura 4.20.

Generación recomendaciones (CB)

Entrada:

Usuario objetivo (U)

Preferencias de U (P)

Consultas de U (C_o)

Salida:

Lugares recomendados

L: Todos los lugares en base de datos

C_a: Todas las categorías en base de datos

P_c: Porcentajes de consulta para cada categorías = Ø

R: Recomendaciones = Ø

Para todo c en C_a:

$$P_c[c] = C_o[c]$$

Si c en P:

$$P_c[c] = 1$$

Para todo l en L:

$$p = \text{probabilidad}_{CB}(l, U, P_c)$$

Si p > 0.6:

$$R = R \cup \{l\}$$

Devolver R

FIGURA 4.20: ALGORITMO DE GENERACIÓN DE RECOMENDACIONES MEDIANTE FILTRADO BASADO EN CONTENIDO

4.5.3. Modelo híbrido

Tras generar recomendaciones mediante los dos enfoque de recomendaciones, estos se combinan en una única lista de recomendaciones, que se almacena en base de datos durante un máximo de 24 horas, ordenada de mayor a menor confianza.

Adicionalmente, cuando se muestran las recomendaciones al usuario, se añaden cinco recomendaciones de lugares de forma aleatoria para aumentar la diversidad de las recomendaciones.

Para agilizar el proceso de cómputo de similitud y del nivel de confianza, estos cálculos se han implementado en una librería escrita en Rust y Neon Bindings. Ya que Rust es un lenguaje compilado, esto permite acelerar cómputos intensivos que serían más lentos si se calculasen con un lenguaje interpretado como Node.js.

Esta tarea de recomendación se ejecuta todos los días a las 02:00AM, durante las horas de baja carga de servidor.

Así, con el sistema recomendador implementado, se mitigan los problemas de arranque en frío y dispersión mediante la solicitud de preferencias y puntuaciones al usuario, se mitiga el problema de escalabilidad mediante la reducción de los usuarios vecinos utilizados para generar recomendaciones y se mitiga la falta de diversidad y *Shilling Attacks* mediante el diseño híbrido de este.

4.6. Sistema planificador

El sistema planificador implementado hace uso del lenguaje PDDL. Cada lugar se representa con un objeto *place*, identificado como “placeI”, donde I es el ID del lugar en la base de datos. Asimismo, los días son objetos de tipo *day*, identificados con el formato “mmmDDYYYY”, donde mmm es el mes de forma textual, DD es el día e YYYY es el año.

Cada uno de los problemas de planificación se identifican de la forma “planI”, donde I es el ID del plan en base de datos. Cada problema a resolver está compuesto por una serie de predicados y funciones matemáticas que conforman el estado inicial del plan, es decir, los lugares que quiere visitar el usuario, los horarios de apertura de cada lugar y las preferencias de horarios y fechas del usuario, además de las metas del problema. Las metas son las siguientes:

- Se debe estar en el lugar de inicio (representado por el objeto *end*)
- Se deben haber visitado todos los lugares
- Cada día debe tener al menos una visita

Además, se introduce el concepto de esperas. Este concepto aparece debido a que, durante la planificación, es posible que el siguiente lugar no esté abierto y el usuario deba esperar antes de la visita.

Por otra parte, el dominio contiene las acciones, predicados y funciones numéricas definidos en la Tabla 4.4 y en la Tabla 4.5.

TABLA 4.4: ACCIONES DEFINIDAS

Acción	Descripción
start-day ?place ?day	Comienza el día a planificar
end-day ?place	Finaliza el día
change-day ?day1 ?day2	Cambia de día
move ?place1 ?place2	Se traslada de un lugar a otro
visit ?place ?day	Se visita un lugar en un determinado día
start-wait ?place	Se comienza una espera
wait ?place	Se está esperando durante 15 minutos adicionales
end-wait ?place	Se finaliza la espera

TABLA 4.5: PREDICADOS Y FUNCIONES NUMÉRICAS DEFINIDAS

Ítem	Tipo	Descripción
visited ?place	Predicado	Indica que ?place ha sido visitado
unvisited ?place	Predicado	Indica que ?place no ha sido visitado
next-day ?day1 ?day2	Predicado	Indica que ?day2 es el día siguiente a ?day1
current-day ?day	Predicado	Indica que el día actual es ?day
current-place ?place	Predicado	Indica que se está visitando ?place actualmente
waiting ?place	Predicado	Indica que se está esperando en ?place
visit-duration ?place	F. Núm	Almacena la duración de la visita de ?place (en minutos)
opens ?place	F. Núm	Indica cuando abre ?place, en minutos desde las 00:00
closes ?place	F. Núm	Indica cuando cierra ?place, en minutos desde las 00:00
distance ?place1 ?place2	F. Núm	Indica la distancia entre ?place1 y ?place2, en metros
travel-time ?place1 ?place2	F. Núm	Indica el tiempo de traslado entre ?place1 y ?place2
visits ?day	F. Núm	Indica el numero de visitas realizas en ?day
day-start	F. Núm	Indica la hora de inicio de cada día de visita, en minutos desde las 00:00
day-end	F. Núm	Indica la hora de fin de cada día de visita, en minutos desde las 00:00
current-time	F. Núm	Indica el tiempo actual en la planificación, en minutos desde las 00:00
heuristic	F. Núm	Coste

A la hora de planificar, se establece una métrica que se debe minimizar. Este coste es la distancia recorrida total, en metros.

Además, se han implementado una serie de traductores que convierten de la representación interna de la plataforma a un problema de PDDL, y de la solución del planificador a la representación interna de la plataforma. Inicialmente se ha usado el planificador Metric-FF, aunque se estudiarán alternativas en el capítulo 5. Igualmente, se ha limitado la ejecución del planificador a 5 minutos, marcando el plan sin solución si la planificación supera este tiempo.

Por otra parte, se establecen varios tipos de estados en los que el plan puede estar y varios tipos de lugares que puede contener un plan. Estos son los mismos que los definidos en el apartado 4.4.6.

El proceso de planificación es el siguiente:

1. Se convierte de la representación interna de la plataforma al problema de PDDL.
2. Se escribe el problema PDDL en un fichero.
3. Se ejecuta el planificador.
 - o Si no ha terminado antes de 5 minutos, se marca el plan con el estado “Sin solución”.
4. Se convierte la solución, si existe, de la salida del planificador a la representación interna de la plataforma.
5. Se ordena la representación interna del plan, de manera que los sitios que se visitan estén almacenados de forma contigua en la base de datos, en función del orden de visita.
6. Se añade el plan a la base de datos y se marca como activo.

Cabe destacar que debido al coste, no es posible usar servicios de búsqueda de rutas como Google Maps Directions API. En sustitución, durante la traducción a los problemas y de las soluciones, se hace uso de las distancias que se encuentran precalculadas en la tabla *Distances* de la base de datos.

En el caso de que una distancia no se encuentre en esta tabla (como es el caso de una distancia entre el punto de inicio y de un lugar), se hace uso de una función que calcula la distancia en línea recta entre dos coordenadas geográficas que se multiplica por 1,2383493909051912, de forma que el resultado se acerque a la distancia que podría proveer un servicios de búsqueda de rutas. Este cálculo de la distancia se usa también para calcular el tiempo de traslado entre dos lugares, asumiendo una velocidad al andar de 1,4m/s.

El valor de 1,2383493909051912 se ha calculado mediante la media de proporciones entre las distancia de dos lugares que se encuentran en la base de datos y la distancia en línea recta entre sus coordenadas.

Por último, si el tiempo de traslado supera los 60 minutos, se hace uso del tiempo de traslado entre los dos lugares usando un vehículo en vez de el valor por defecto, a pie.

4.7. *Backend*

El *backend*, o servidor, es el módulo de la plataforma que se encargar de interactuar con la interfaz de usuario, con la base de datos y con el sistema recomendador. Su implementación se describe en esta sección.

4.7.1. API REST

La interfaz de usuario interactúa con el servidor mediante una API REST. Esta API se implementa mediante la librería *fastify*. A su vez, se hace uso de la librería *cluster*, una librería nativa de Node.js. Debido a que Node.js funciona con un único hilo [126], se pueden estar desperdiciando recursos del servidor. Mediante el uso de esta librería, se pueden crear varios hilos del mismo proceso, aprovechando mejor los recursos.

Asimismo, esta API ha sido definida [127] previamente a su implementación usando la especificación OpenAPI [128].

4.7.2. *Logging*

El registro de eventos es una característica importante, que permite recoger información sobre el sistema y la plataforma en tiempo real. En el proyecto, se implementan dos tipos de registros de eventos, el registro de eventos de aplicación y el registro de eventos de peticiones.

Mediante estos registros, se puede conocer en todo momento que fallos han ocurrido en la plataforma, qué eventos importantes han ocurrido e incluso realizar un análisis forense para casos en los que se haya comprometido la seguridad, ya sea de la plataforma o de un usuario.

El registro de peticiones está implementado mediante la clase de la Tabla 3.80. En cada petición que se realiza a la plataforma, ya sea a la API o a la interfaz web, se crea un registro que contiene:

- Nivel del registro.
- Servidor que procesó la petición.
- ID del hilo del cluster.
- ID de usuario que realizó la petición (si se realizó con sesión iniciada).
- Dirección IP (IPv4 o IPv6).
- URL que recibió la petición.
- Estado HTTP de la respuesta.
- Tiempo en responder (en milisegundos).
- Información adicional, como el agente de usuario.

Este registro de peticiones permite controlar la velocidad de respuesta del servidor, depurar posibles fallos que hayan podido ocurrir e informar de eventos de seguridad y de eventos importantes.

El registro de aplicación está implementado mediante la clase de la Tabla 3.81. En cada tarea que se realice dentro de la plataforma (por ejemplo, inicios de sesión fallidos, envíos de correos electrónicos, etc.) se crea un registro que contiene:

- Nivel del registro.
- Servidor que procesó la petición.
- ID del hilo del cluster.
- ID de usuario que realizó la petición (si se realizó con sesión iniciada).
- Dirección IP (IPv4 o IPv6) si la tarea se realizó en una API.
- URL o función que realizó la tarea.
- Información adicional.

Este registro de peticiones permite, depurar posibles fallos que hayan podido ocurrir e informar de eventos de seguridad y de resultados de tareas importantes.

4.7.3. Seguridad

Por lo general, todas las peticiones a la plataforma deben estar autenticadas mediante una cookie denominada *hermesSession*. Esta cookie contiene una clave de sesión de 255 caracteres, generada de forma aleatoria, que se almacena según lo definido en la clase de la Tabla 3.95. Al usar una clave de sesión, en vez del correo electrónico y la contraseña del usuario se puede prevenir el robo de credenciales mediante ataques de *Cross-Site Scripting* [105]. Estas cookies están gestionadas por la librería *fastify-cookie*, y son cifradas mediante una clave de 32 caracteres solo conocida por el servidor. Este cifrado y la generación aleatoria de estas claves permiten proteger ante ataques de Cookie Tampering [129] o Cookie Poisoning [130].

Para aumentar la seguridad de la plataforma, también se hace uso exclusivo HTTPS con un certificado generado por la Autoridad de Certificación Let's Encrypt. Todas las peticiones que se realizan a través de HTTP son rechazadas y solo se permiten peticiones con HTTPS. Esta regla es aplicada por Cloudflare y Nginx. Además, Nginx termina las conexiones TLS antes de que lleguen al servidor Node.js, convirtiéndolas en HTTP para que la biblioteca *fastify* las pueda manejar más fácilmente. Estas conexiones HTTP solo se utilizan de forma interna y nunca se abren a la Internet pública. Asimismo, Cloudflare y Nginx también aseguran la aplicación de HSTS y reglas de cortafuegos que previenen ante bots o herramientas automatizadas y de ataques (como los ataques de denegación de servicio distribuidos [131]).

Además de estas medidas, se hace uso de las librerías *fastify-helmet* y *fastify-cors*. *fastify-helmet* configura para cada petición una serie de cabeceras HTTP de seguridad, mientras que *fastify-cors* configura cabeceras para aplicar CORS, limitando que scripts,

estilos y conexiones son permitidas por el navegador, protegiendo al usuario ante ataques [103].

Otra medida de seguridad que se implementa es la transmisión de contraseñas mediante su transformación previa mediante la función SHA256. Aunque HTTPS cifra todas las comunicaciones, el envío del hash de la contraseña en vez de la contraseña en texto plano permite que, en caso de que ocurra un ataque de hombre en medio entre Cloudflare, Nginx y/o el servidor o se filtren los registros de peticiones, las credenciales reales de los usuarios siguen estando protegidas y no son visibles para los atacantes, mientras sean contraseñas seguras, debido a que el usuario puede haber utilizado las mismas contraseñas en otros servicios [107]. Además, el uso de SHA256 permite sobrellevar la limitación del tamaño máximo de entrada del algoritmo bcrypt. Aunque el máximo es de 72 bytes [132], el tamaño en bytes del hash de la contraseña (64 caracteres en formato UTF-8 que se convierten en 64 bytes) es siempre inferior al límite.

Por otra parte, todas las contraseñas son cifradas por el algoritmo *bcrypt*. Como la mayoría de usuarios suele usar la misma contraseña para muchos servicios [107], el uso de *bcrypt* asegura que las contraseñas no se almacenan en texto plano en la base de datos y evitan que se puedan usar contra otros servicios en caso de que ocurran filtraciones de la base de datos. Asimismo, el cálculo del número de ciclos de ejecución del algoritmo viene determinado por la fórmula $10 + (f_s(k) \bmod 7)$ (28), donde $f_s(k)$ es la suma de todos los números que se encuentran en la contraseña en formato hexadecimal k . Esta fórmula asegura que el número de ciclos se encuentra entre 10 y 16, de acuerdo al número de ciclos recomendados [133].

Otra medida de seguridad impuesta es el uso de *rate-limiting*. El uso de rate-limiting, implementada por la librería fastify-rate-limit, limita el número de peticiones consecutivas que se pueden realizar, protegiendo ante *bots* [134]. Además, durante el proceso de inicio de sesión, se limita el número de intentos fallidos a 5, añadiendo 1,5 segundos de ralentización por cada intento fallido. Una vez alcanzados 5 intentos fallidos, se bloquea el inicio de sesión al usuario durante 15 minutos, protegiendo al usuario de la plataforma de ataques de fuerza bruta [135]. Además, cada vez que se realiza un inicio de sesión, se registra la dirección IP desde donde se realizó, haciendo uso de la clase de la Tabla 3.85, y notificando al usuario si el inicio de sesión se produjo desde una dirección IP no vista antes.

Por último, cuando un usuario elimina su cuenta, esta se marca para eliminación y no se elimina de forma permanente hasta pasados 5 días. Esto permite que, en caso de que el usuario desee recuperarla, pueda contactar con la plataforma y solicitar la restauración de esta.

Todas estas medidas han sido implementadas con el objetivo establecido de crear una plataforma segura y que respeta la privacidad.

4.7.4. Controladores

La API está implementada mediante controladores. Estos controladores permiten, de forma modular, añadir nueva funcionalidad y rutas.

- **Account:** Este controlador se encarga de las tareas de gestión de cuentas. Implementa rutas que permiten obtener información del usuario, actualizar la información del usuario y sus preferencias, cambiar sus ajustes de notificaciones y descargar sus datos en la plataforma.
- **Auth:** Este controlador se encarga de las tareas de autenticación y gestiones de seguridad de las cuentas. En primer lugar, implementa una función que se ejecuta antes de procesar cada petición. Esta función se encarga de filtrar que peticiones son públicas y cuales privadas, redirigiendo a un inicio de sesión si las peticiones privadas no contienen la cookie de sesión o su sesión es inválida o ha expirado. Además, esta función se encarga de establecer el tiempo de vida en cache de navegador de los recursos, configurando un tiempo de vida de 0 segundos a peticiones de la API o de redirección.

En segundo lugar, se implementan funciones que permiten iniciar sesión, crear cuentas, cerrar sesiones, recuperar y cambiar contraseñas y eliminar cuentas. Estas funciones se implementan de acorde a la seguridad implementada en el apartado 4.7.3.

Además, durante el registro de cuentas, la clave generada al completar correctamente el CAPTCHA se comprueba con el servicio hCaptcha. Si la clave no es válida, se impide la creación de la nueva cuenta.

- **Places:** Este controlador se encarga de devolver lugares de forma aleatoria (para las puntuaciones iniciales durante el registro), de permitir la búsqueda de lugares y de obtener información adicional sobre un lugar específico.
- **Planner:** Este controlador permite crear nuevos planes, obtener el estado de los planes, generar duraciones mínimas, listar planes, actualizar planes y eliminar planes. Además, enlace con el sistema planificador durante la creación de planes.
- **Ratings:** Este controlador gestiona las puntuaciones del usuario, permitiendo la creación y eliminación de estas, además de listar puntuaciones del usuario y permitir la búsqueda de lugares para la creación de puntuaciones.

- **Recommender:** Este controlador se encarga de listar las recomendaciones del usuario, además de permitir la solicitud de generación de nuevas recomendaciones.
- **Session:** Este controlador no ofrece una ruta pública, por el contrario siendo usado por el resto de controladores. Su objetivo es crear, eliminar, comprobar y devolver información sobre las sesiones de usuario.

4.7.5. Envío de correos

Una de las funcionalidades de la plataforma es el envío de correos electrónicos. Estos se clasifican en correos electrónicos de cuenta y correos electrónicos de notificación. Los correos electrónicos de cuenta notifican de eventos de bienvenida, inicio de sesión desconocidos, cambios y recuperación de contraseña y eliminación de cuenta. Por otra parte, los correos electrónicos de notificación se encargan de notificar al usuario del inicio de un plan activo y de puntuar lugares visitado para mejorar la precisión de la plataforma.

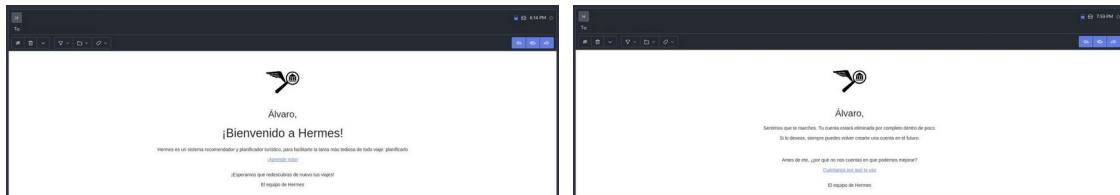


FIGURA 4.21: EJEMPLOS DE CORREOS ELECTRÓNICOS

Estos correos electrónicos se envían mediante la librería *nodemailer*, que enlaza con una cuenta de Gmail creada específicamente para la plataforma. Un ejemplo de este tipo de correos se muestra en la Figura 4.21.

4.7.6. Cronjobs

La plataforma requiere de una serie de tareas que se deben ejecutar periódicamente, como por ejemplo, la regeneración de recomendaciones o el envío de correos electrónicos recordatorios. Para ello, se hace uso del servicio Cron en Linux. Este servicio permite crear una tarea, que se ejecuta todos los días a las 02:00AM. Esta tarea se encarga de eliminar cuentas marcadas para eliminar desde hace más de 5 días, completar planes una vez finalicen, enviar correos electrónicos recordatorios y regenerar las recomendaciones de todos los usuarios.

4.8. Despliegue

En cada nueva versión de la plataforma, se realiza un despliegue al entorno de producción.

Durante el primer despliegue, se crea un archivo Dockerfile, que define como debe construirse la imagen del contenedor. Este archivo instala las librerías necesarias, compila el código Rust y copia el código de servidor y de la interfaz web. Asimismo, el uso de Docker permite escalar la plataforma fácilmente según las necesidades del servicio. Seguidamente, se crea un archivo de variables de entorno que definen:

- El dominio donde se encuentra la plataforma.
- El puerto HTTP del servidor.
- El nombre del servidor.
- La cadena de conexión a la base de datos.
- El secreto usado para cifrar las cookies de sesión.
- Las claves del servicio hCaptcha.
- Las claves de acceso al correo electrónico.

Una vez creada y configurada la imagen y sus variables, se procede al despliegue en el entorno de producción. En primer lugar se añaden scripts de analíticas a la interfaz web. Estas analíticas son provistas por Umami, como se muestra en la Figura 4.22, un software de analíticas web que no hace uso de cookies y que respeta la privacidad.

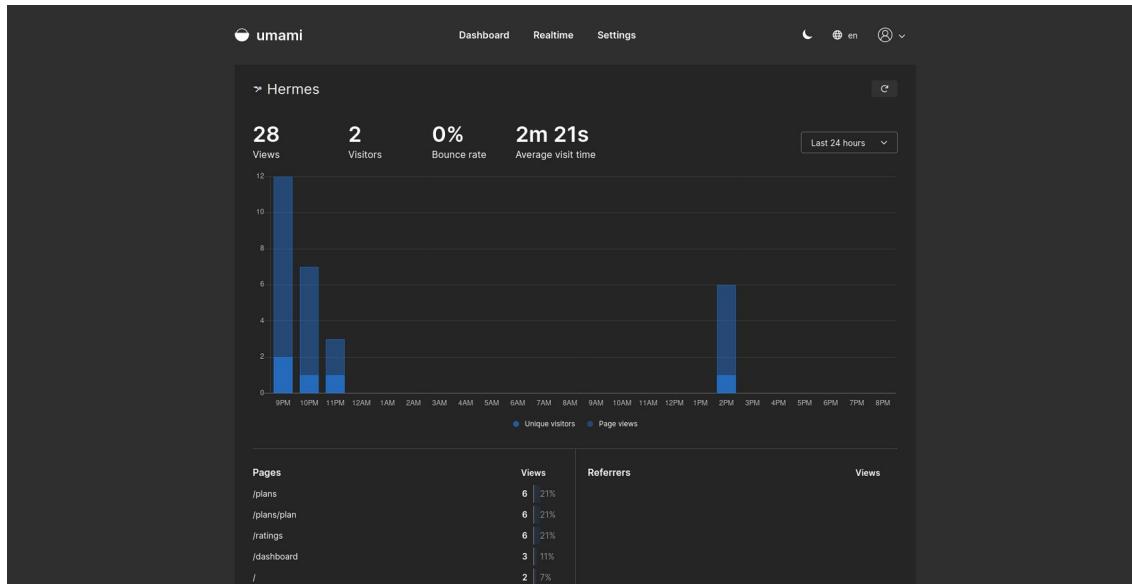


FIGURA 4.22: PANEL DE CONTROL DE UMAMI

Tras añadir las analíticas, se finaliza el despliegue iniciando los contenedores de la plataforma, creando o migrando la base de datos si es necesario y configurando Nginx para sus tareas de *Load balancing*. Además, se vacía la caché de Cloudflare de manera que se actualicen los archivos que se encuentran en la CDN. Esta serie de tareas y configuraciones permite que la plataforma se ajuste a la arquitectura diseñada en la Figura 3.2.

Por otra parte, para los despliegues y ser más transparentes con los usuarios, se ha creado una página de estado (<https://status.travelhermes.com>), que hace uso del proyecto *cState*, que permite conocer en todo momento el estado de la plataforma. Esta página de estado muestra mantenimientos programados e incidentes que estén ocurriendo y hayan ocurrido junto a información adicional. La página de estado se muestra en la Figura 4.23.

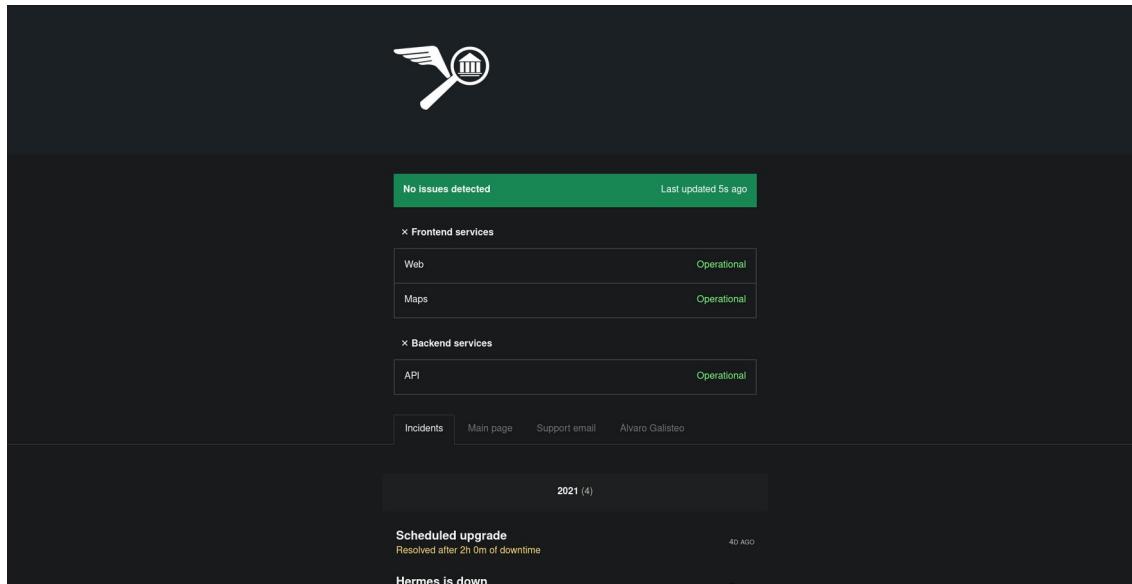


FIGURA 4.23: PÁGINA DE ESTADO DE HERMES

La plataforma Hermes está disponible de forma pública y para cualquier usuario bajo el dominio <https://travelhermes.com>.

4.9. Colaboración

El objetivo de esta plataforma es la creación de un proyecto Open Source (<https://github.com/SrGMC/hermes>), por lo que está abierta a mejoras, sugerencias y contribuciones por parte de otros usuarios. Para ello, se han establecido unas normas de colaboración para nuevos desarrolladores. Asimismo, el versionado se nombra siguiendo Semantic Versioning [136].

Se establecen unas plantillas que diferencian los tipos de sugerencias que los usuarios pueden tener. Todas las plantillas están en Español e Inglés.

1. Informe de errores: El objetivo de esta plantilla es que el usuario describa los errores que ha encontrado durante el uso de la plataforma, detallando donde ha ocurrido y cuales son los pasos para reproducir el error.
2. Solicitud de características: El objetivo de esta plantilla es que el usuario describa nuevas características que le gustaría ver o utilizar en la plataforma.

3. Solicitud de lugar/ciudades: El objetivo de estas plantillas es que el usuario pueda sugerir nuevos lugares o ciudades para la plataforma. Para ello, deberá detallar una serie de datos del lugar o ciudad.

Si cualquier persona desea contribuir con un nuevo lugar o corregir un lugar existente, se establecen una serie de pasos a realizar [137]:

1. El usuario debe llenar un objeto en formato JSON con los datos necesarios.
 - Las categorías disponibles aparecen en la Tabla 4.1.
 - La única zona disponible actualmente es “Madrid”.
2. El usuario debe añadir este objeto al archivo de base de datos correspondiente.
3. El usuario debe crear un *Pull request*, con los cambios realizados.
4. Los nuevos datos serán revisados antes de unirlos al repositorio principal y se añadirá a la base de datos en el caso de que se aprueben.

Si cualquier persona desea contribuir con nuevo código o mejoras, se establece un formato que se debe cumplir. Este formato se puede cumplir usando la herramienta Prettier con las configuraciones:

- Print width: 120.
- Tab width: 4.
- Single quotes: true.

Además, se establecen una serie de normas a cumplir [137]:

- Se debe escribir código que sea eficiente y rápido.
- Se debe escribir código que sea entendible y fácil de leer. Se debe evitar escribir código complejo.
- Se debe comentar todo el código que se añada, cambie o modifique y todo el código que sea relevante.
 - Las funciones y clases deben comentarse en formato JSDoc [138].

5. EXPERIMENTACIÓN Y VALIDACIÓN

En este capítulo se describen los procesos seguidos para la experimentación y las pruebas llevadas a cabo durante el desarrollo de este proyecto.

Las pruebas han sido divididas en cuatro partes:

1. **Pruebas de interfaz:** En estas pruebas, se realizan análisis y evaluaciones con usuarios para determinar la funcionalidad de la plataforma.
2. **Pruebas unitarias de la API:** Se han creado una serie de pruebas unitarias que permiten verificar la funcionalidad de la API.
3. **Pruebas de recomendador:** Se han realizado una serie de pruebas usando la base de datos 1m de MovieLens [139] para comprobar el rendimiento del recomendador.
4. **Pruebas de planificador:** Estas pruebas sirven para comprobar el rendimiento del proceso de planificación y para comparar distintos planificadores.

5.1. Pruebas de interfaz

En este apartado se describe el proceso de evaluación con usuarios realizado. Para ello, se ha establecido la siguiente metodología a seguir:

1. **Objetivos:** Se establecen una serie de objetivos que se desean obtener con la evaluación. Estos son:
 1. **Usabilidad y facilidad:** Se desea comprobar si el diseño visual es sencillo y ayuda al usuario a entender qué ocurre y cuándo y le invita a interactuar de forma sencilla con la interfaz.
 2. **Análisis de rendimiento:** Se desea comprobar cómo se comporta el sistema ante distintos usuarios y si este es eficiente y personalizado.
2. **¿Qué se desea conocer?:** A partir de los objetivos anteriores, se establecen una serie de parámetros que se quieren responder:
 1. **Chokepoints:** Se desea averiguar dónde se atasca el usuario y qué partes o elementos son más difíciles de utilizar o ambiguos.
 2. **Precisión de las recomendaciones:** Se desea averiguar si las recomendaciones del usuario son las que esperaba y si están adaptadas a su perfil.
 3. **Calidad de la planificación:** Se desea comprobar si los planes generados son los que esperados.

4. **Facilidad de uso:** Se desea conocer si la interfaz es fácil de usar, si ayuda al usuario a entender que ocurre y cuando y si le invita a interactuar de forma sencilla con ella.
3. **Asuntos prácticos:** Una vez establecidos los puntos anteriores, se procede a realizar la evaluación con el usuarios. En este apartado se realizan una serie de preguntas y se pone en contexto al usuario ante la tarea que debe realizar.

El contexto establecido es el siguiente: “Usted, el usuario, es un turista extranjero que ha decidido que desea visitar Madrid en su próximo viaje. Aunque no conoce nada sobre la ciudad, ha encontrado una plataforma que le permite crear planes personalizados con los sitios que visitar por esta ciudad en función de sus preferencias personales.” Por otra parte, se han realizado las siguientes preguntas a los usuarios. Cada pregunta está identificada con el formato PIX, donde X es un número entero.

1. **PI1:** ¿Cuál es su edad?
2. **PI2:** ¿Cuál es su género?
3. **PI3:** ¿Cuál es su experiencia con la tecnología?
4. **PI4:** ¿Suele viajar?
5. **PI5:** ¿Cuando viaja, suele planificarlo de antemano?
4. **Realización:** Durante este punto, se ha observado a cada usuario y se han realizado notas sobre la interacción y el comportamiento del usuario durante la evaluación.
5. **Preguntas finales:** Tras finalizar la interacción con el usuario, se han realizado las siguientes preguntas finales. Cada pregunta está identificada con el formato PFX, donde X es un número entero.
 1. **PF1:** ¿Cree que la interfaz es visualmente atractiva? ¿Es fácil de interactuar con ella?
 2. **PF2:** ¿Que elementos cree que son más complicados de entender?
 3. **PF3:** ¿Cree que las recomendaciones son apropiadas para usted?
 4. **PF4:** ¿Cree que las planificaciones son correctas y eficientes?
 5. **PF5:** ¿La respuesta de la plataforma es rápida?, es decir, ¿la espera durante las pantallas de carga han sido rápidas o lentas?

Una vez establecida la metodología, se ha evaluado la plataforma con 6 usuarios, de distintas edades y géneros. Las respuestas de cada usuario se encuentran en el Anexo I.

Tras realizar las pruebas, se han extraído las siguientes conclusiones:

1. **Chokepoints:** Se han encontrado varios puntos en los que usuarios se atascaban. Estos aparecen en la barra superior de navegación y en algunos casos en la interfaz de planificación, que se presenta en la interfaz de recomendaciones.
2. **Precisión de las recomendaciones:** Las recomendaciones han sido precisas, aunque para algunos usuarios han sido escasas. Esto se puede deber a la limitada respuesta de preferencias o a la poca cantidad de puntuaciones que han creado estos usuarios.
3. **Calidad de la planificación:** Los planes generados son, por lo general, ideales, aunque a menudo se presentaban casos de lugares muy cercanos entre sí, donde el planificador no ha planificado de forma que se visiten estos uno tras otros.
4. **Facilidad de uso:** Por lo general, a los usuarios les ha gustado la interfaz y les parece sencilla, aunque a menudo han encontrado elementos ambiguos o poco claros que les ha provocado la dificultad de interpretar las acciones de cada elemento.

Por lo tanto, a partir de estas conclusiones, se ha generado la Tabla 5.1, que contiene una lista de cambios y mejoras a realizar en la plataforma, algunos de ellos ya implementados.

TABLA 5.1: CAMBIOS EXTRAÍDOS DE LA EVALUACIÓN CON USUARIOS

Cambio	Implementado
Creación de tutoriales que se muestren la primera vez que se navegue por la plataforma	Sí
Renombrar algunos elementos (como por ejemplo, de Explorar a Recomendaciones para ti), de forma que sea mas clara la acción del elemento	Sí
Permitir buscar, no solo por nombre de lugar, si no también por categorías	Sí
Añadir iconos categorizados para facilitar la exploración	Sí
Situar de forma más clara, la acción de cerrar sesión	Sí
Especificar algunos elementos de forma que expliquen mejor su necesidad o acción	Sí
Ocultar elementos innecesarios, como la navegación entre fotografías cuando solo hay una	Sí
Añadir sugerencias cuando un usuario quiera añadir un lugar a un plan o crear una recomendación en vez de solo depender de la búsqueda	Sí
Limitar la longitud de algunos elementos que se guardan en base de datos	Sí
Añadir la distancia en kilómetros entre lugares en la vista de plan	Sí
Solucionar algunos fallos en donde se produce un <i>overflow</i> en el cálculo de horas con planes muy largos	Sí
Notificar en la vista de plan cuando algunos elementos añadidos manualmente pueden estar cerrados	No
Permitir buscar por dirección el punto de inicio de un plan, no solo pulsando en el mapa	Sí
Hacer más claros los botones para crear un plan o una puntuación	Sí
Reducir la dificultad del CAPTCHA durante el registro	No
Permitir actualizar puntuaciones	No
Repartir, durante la planificación, los lugares a visitar de forma más homogénea entre días, en vez de acumularlos los primeros días	No

En las figuras 5.1, 5.2, 5.3 y 5.4, se muestran algunos de los cambios implementados

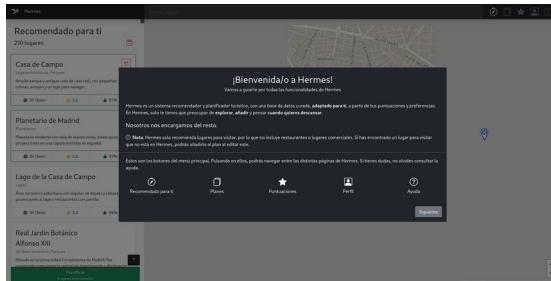


FIGURA 5.1: IMPLEMENTACIÓN DEL TUTORIAL

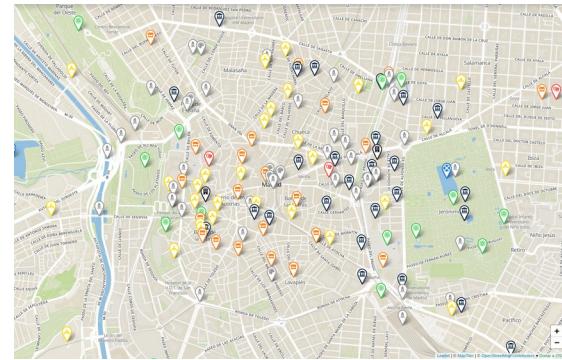


FIGURA 5.2: ICONOGRAFÍA BASADA EN LAS CATEGORÍAS DE UN LUGAR

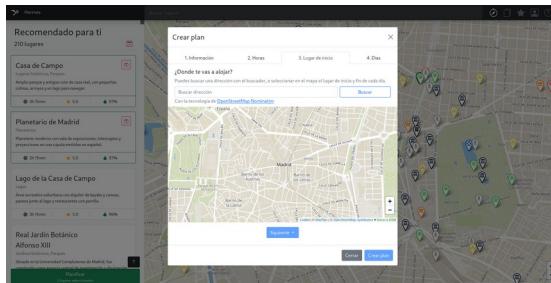


FIGURA 5.3: BUSCADOR DE PUNTO DE INICIO

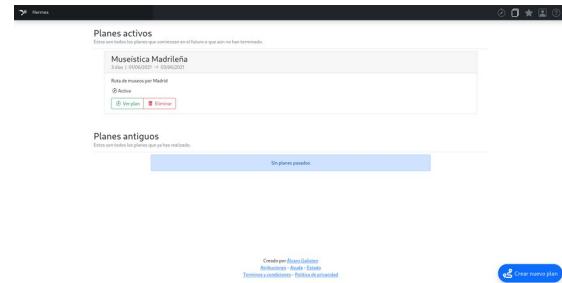


FIGURA 5.4: MEJORA DE LAS DESCRIPCIONES DE ALGUNOS ELEMENTOS

5.2. Pruebas unitarias API

En este apartado se describen las pruebas unitarias que se han realizado para la API REST implementada, de manera que permitan comprobar el funcionamiento de la API de acuerdo a las especificaciones establecidas.

Para ello, se va a hacer uso de la que contiene los siguientes campos:

- **Identificador:** Identifica la prueba. Este identificador tiene el formato PUX-Y, donde Y es un número entero y X puede ser:
 - S: Para el controlador de sesiones.
 - A: Para el controlador de cuentas.
 - L: Para el controlador de lugares.
 - P: Para el controlador del planificador.
 - R: Para el controlador de recomendaciones.
 - C: Para el controlador de puntuaciones.
 - Au: Para el controlador de autenticación.
- **Descripción:** Define la prueba que se va a realizar.

- **Completada:** Indica si la prueba ha sido completada satisfactoriamente o no.

Los resultados de las pruebas se muestran en la Tabla 5.2.

TABLA 5.2; PRUEBAS UNITARIAS DE LA API

Identificador	Descripción	Completada
Controlador de sesiones		
PUS-1	Comprobar que la ruta GET accountInfo no sea accesible sin sesión.	Sí
PUS-2	Comprobar que la ruta GET accountUpdate no sea accesible sin sesión.	Sí
PUS-3	Comprobar que la ruta PUT notificationsUpdate no sea accesible sin sesión.	Sí
PUS-4	Comprobar que la ruta GET accountDownload no sea accesible sin sesión.	Sí
PUS-5	Comprobar que la ruta GET logout no sea accesible sin sesión.	Sí
PUS-6	Comprobar que la ruta GET logoutAll no sea accesible sin sesión.	Sí
PUS-7	Comprobar que la ruta POST accountDelete no sea accesible sin sesión.	Sí
PUS-8	Comprobar que la ruta PUT passwordUpdate no sea accesible sin sesión.	Sí
PUS-9	Comprobar que la ruta GET placesSearch no sea accesible sin sesión.	Sí
PUS-10	Comprobar que la ruta POST placesSearch no sea accesible sin sesión.	Sí
PUS-11	Comprobar que la ruta GET placeInfo no sea accesible sin sesión.	Sí
PUS-12	Comprobar que la ruta GET ratingsGet no sea accesible sin sesión.	Sí
PUS-13	Comprobar que la ruta POST ratingCreate no sea accesible sin sesión.	Sí
PUS-14	Comprobar que la ruta POST ratingDelete no sea accesible sin sesión.	Sí

Identificador	Descripción	Completada
PUS-15	Comprobar que la ruta GET ratingsSearch no sea accesible sin sesión.	Sí
PUS-16	Comprobar que la ruta GET recommendationsGet no sea accesible sin sesión.	Sí
PUS-17	Comprobar que la ruta GET recommendationsRequest no sea accesible sin sesión.	Sí
PUS-18	Comprobar que la ruta POST plannerLength no sea accesible sin sesión.	Sí
PUS-19	Comprobar que la ruta POST plannerCreate no sea accesible sin sesión.	Sí
PUS-20	Comprobar que la ruta GET plannerGet no sea accesible sin sesión.	Sí
PUS-21	Comprobar que la ruta GET plannerList no sea accesible sin sesión.	Sí
PUS-22	Comprobar que la ruta GET plannerStatus no sea accesible sin sesión.	Sí
PUS-23	Comprobar que la ruta POST plannerDelete no sea accesible sin sesión.	Sí
PUS-24	Comprobar que la ruta PUT plannerUpdate no sea accesible sin sesión.	Sí
Controlador de cuentas		
PUA-1	Descargar datos de la cuenta	Sí
PUA-2	Obtener información de la cuenta	Sí
PUA-3	Actualizar la información de la cuenta con atributos válidos	Sí
PUA-4	Actualizar la información de la cuenta con menos de 3 preferencias	Sí
PUA-5	Actualizar la información de la cuenta con un correo electrónico no válido	Sí
PUA-6	Actualizar la información de la cuenta con atributos que faltan	Sí
PUA-7	Actualizar las notificaciones de la cuenta con atributos válidos	Sí
PUA-8	Actualizar las notificaciones de la cuenta con atributos	Sí

Identificador	Descripción	Completada
	que faltan	
Controlador de lugares		
PUL-1	Obtener información de lugares	Sí
PUL-2	Obtener lugares aleatorios sin ignorar	Sí
PUL-3	Obtener lugares aleatorios con ignorados	Sí
PUL-4	Buscar un lugar específico	Sí
PUL-5	Buscar varios lugares	Sí
PUL-6	Buscar lugares sin resultados	Sí
Controlador del planificador		
PUP-1	Obtener la longitud recomendada de un plan	Sí
PUP-2	Obtener la longitud recomendada de un plan con lugar desconocido	Sí
PUP-3	Obtener la duración recomendada de un plan con menos de 3 plazas	Sí
PUP-4	Obtener la longitud recomendada de un plan con horas no válidas	Sí
PUP-5	Obtener la longitud recomendada de un plan sin punto de partida	Sí
PUP-6	Crear un nuevo plan	Sí
PUP-7	Crear un nuevo plan con menos de 3 plazas	Sí
PUP-8	Crear un nuevo plan con una plaza desconocida	Sí
PUP-9	Crear un nuevo plan con atributos que faltan	Sí
PUP-10	Crear un nuevo plan con fechas no válidas	Sí
PUP-11	Crear un nuevo plan con horas no válidas	Sí
PUP-12	Crear un nuevo plan con un punto de partida no válido	Sí
PUP-13	Obtener todos los planes del usuario	Sí
PUP-14	Obtener el estado de un plan	Sí
PUP-15	Obtener el estado de un plan desconocido	Sí
PUP-16	Obtener información del plan	Sí
PUP-17	Obtener un plan desconocido	Sí
PUP-18	Actualizar un plan	Sí

Identificador	Descripción	Completada
PUP-19	Actualización de un plan cuyo punto de partida no es el primero	Sí
PUP-20	Actualización de un plan con lugar desconocido	Sí
PUP-21	Actualización de un plan en el que faltan atributos	Sí
PUP-22	Actualización de un plan desconocido	Sí
PUP-23	Actualización de un plan en el que la fecha de inicio sería anterior a la de hoy	Sí
PUP-24	Eliminación de un plan	Sí
PUP-25	Eliminación de un plan desconocido	Sí
Controlador de puntuaciones		
PUC-1	Crear puntuación	Sí
PUC-2	Crear una puntuación que ya existe	Sí
PUC-3	Crear puntuación fuera de los límites	Sí
PUC-4	Crear puntuación para un lugar que no existe	Sí
PUC-5	Crear una puntuación con atributos faltantes	Sí
PUC-6	Eliminar una puntuación	Sí
PUC-7	Eliminar una puntuación que no existe	Sí
PUC-8	Eliminar una puntuación de un lugar que no existe	Sí
PUC-9	Eliminar una puntuación con el cuerpo vacío	Sí
PUC-10	Obtener las puntuaciones de los usuarios	Sí
PUC-11	Buscar lugares no puntuados por el usuario	Sí
PUC-12	Obtener lugares aleatorios no puntuados por el usuario	Sí
Controlador de recomendaciones		
PUR-1	Solicitar nuevas recomendaciones	Sí
PUR-2	Obtener recomendaciones del usuario	Sí
PUR-3	Obtener recomendaciones del usuario de forma aleatoria	Sí
Controlador de autenticación		
PUAu-1	Iniciar sesión del usuario	Sí
PUAu-2	Cerrar la sesión del usuario	Sí
PUAu-3	Cerrar la sesión de todos los usuarios	Sí

Identificador	Descripción	Completada
PUAu-4	Cambiar la contraseña	Sí
PUAu-5	Cambio de contraseña con cuerpo vacío	Sí
	Cambiar la contraseña con una contraseña que no es un hash	
PUAu-6	Solicitud de restablecimiento de contraseña	Sí
	Solicitud de restablecimiento de contraseña sin proporcionar el correo electrónico	
PUAu-8	Cambiar la contraseña con una contraseña que no es hash	Sí
PUAu-9	Cambiar la contraseña con un atributo que falta	Sí
PUAu-10	Cambiar la contraseña con un token	Sí
PUAu-11	Cambiar la contraseña con un token no válido	Sí

5.3. Pruebas del recomendador

En este apartado se describen las pruebas realizadas para el recomendador. Estas pruebas tienen el objetivo de observar la precisión del sistema recomendador implementado. Para ello, se ha hecho uso de la base de datos de 1m de MovieLens [139], que contiene más de un millón de puntuaciones de películas, con 3900 películas y 6040 usuarios.

Esta base de datos permite comprobar las similitudes entre el nivel de confianza de la plataforma y las puntuaciones reales. Como las películas están categorizadas, similarmente a los puntos de interés, y las puntuaciones tienen un rango de 1 a 5, no ha sido necesario realizar cambios significativos en la base de datos.

Para estas pruebas se han establecido 3 parámetros, N, que es el número de vecinos que se generan; M, un número entero que multiplicado por N genera el número de usuarios aleatorios que se obtienen y D, que simula el número de días que ha estado funcionando el recomendador. Los valores posibles de estos parámetros son los siguientes:

- **N:** 5, 10 o 15 usuarios vecinos.
- **M:** 2, 4, 6, 8 o 10.
- **D:** 5, 10 o 20 días.

Debido al gran tiempo de cómputo que supone generar recomendaciones para 6040 usuarios, según se vio en la Tabla 4.3, estas pruebas se han realizado en una máquina

virtual ofrecida por el servicio Linode. Esta máquina virtual está compuesta por 2 núcleos y 8GB de memoria RAM. Las pruebas se ejecutaron durante 6 días.

Para el análisis, se ha utilizado el Error Cuadrático Medio (MSE) y el Error Absoluto Medio (MAE). Mediante el error cuadrático medio se puede estimar la diferencia entre los valores predichos y los observados en la misma escala que el valor observado. Mediante el Error Absoluto Medio se puede estimar los cuadrados de las desviaciones entre los valores predichos y los observados.

TABLA 5.3: MEJORES 5 RESULTADOS RECOMENDADOR

N	M	D	MSE	MAE
15	10	5	1,39	0,89
15	8	5	1,4	0,89
15	6	5	1,4	0,89
10	10	5	1,42	0,9
10	8	5	1,42	0,9

En el Anexo II se podrán observar todos los resultados obtenidos para esta prueba. En la Tabla 5.3 se muestran los 5 primeros valores más bajos.

En esta tabla se observa que la media de diferencias entre los valores predichos y los observados es de 1,39 puntos en la escala de 1 a 5, un 28% de diferencia.

Además, se observan varios eventos. En primer lugar, cuantos más vecinos se usen, más precisas son las recomendaciones. Por último, los valores altos de M, es decir, los usuarios aleatorios usados para buscar los vecinos, también generan recomendaciones precisas.

5.4. Pruebas del planificador

En este apartado se describen las pruebas realizadas para el planificador. Estas pruebas tienen el objetivo de comparar distintos planificadores con el dominio diseñado.

Para esta prueba se han utilizado los programas Metric-FF, CBP y LPG, introducidos en el capítulo 2.4.3. Además, se han planteado 4 casos de prueba:

1. **base_1_day.pddl**: Es un problema de planificación para generar un itinerario de un único día, visitando 3 lugares.
2. **base_m1_day.pddl**: Es un problema de planificación para generar un itinerario de más de un día, visitando 3 lugares.

3. **big.pddl**: Es un problema de planificación para generar un itinerario de una semana, visitando más de 50 lugares.
4. **restrictive.pddl**: Es un problema de planificación para generar un itinerario de tres días, visitando 4 lugares. Los días tienen un horario muy restrictivo, donde solo se dispone de 4 horas cada día para visitar lugares.

Por otra parte, además de establecerse un tiempo máximo de ejecución de 5 minutos, se establecido los siguientes parámetros para cada planificador:

- **Metric-FF**: Configuraciones de búsqueda Weighted A*, A*epsilon y EHC+H con A*epsilon.
- **CBP**:
 - **Heurísticas**:
 - 1. hlevel max propagation
 - 2. hlevel add propagation
 - 3. hmax
 - 4. hadd
 - 5. hadd with helpful actions
 - 6. hmax with helpful actions
 - 7. sum-hmax
 - 8. max-hadd
 - 9. sum-hmax with helpful actions
 - 10. max-hadd with helpful actions
 - **Algoritmos**:
 - 2. CEHC (EHC with and without costs)
 - 3. anytime BFS (bounded by g value and repeated states prune)
 - 4. anytime BFS + helpful and secondary lists
 - 6. anytime BFS + rescue nodes
 - 7. anytime BFS + lookahead + rescue nodes
 - 9. anytime BFS prune + lookahead + helpful and secondary lists
 - 10. depth first with chronological backtracking and Roller
 - 11. modified Hill-Climbing
 - 12. anytime BFS + Roller policy lookahead with horizon
 - 13. anytime BFS + Roller policy lookahead with horizon + rescue nodes

- 14. BFS+BnB (exhaustive) for learning
- 15. BFS+BnB (exhaustive) for learning using previous solution as upper bound
- 16. depth first with chronological backtracking sorting actions using ff heuristic (for Roller paper).
- 17. anytime BFS + FF heuristic policy lookahead with horizon (for Roller paper).
- 18. anytime BFS + FF heuristic policy lookahead with horizon + rescue nodes (for Roller paper).
- 19. CEHC ordering first level by $h(n)+\text{cost}(a)$
- 20. CEHC hdiff
- **LPG:** Generación de un máximo de 100 soluciones

En el Anexo II se encuentran los resultados de las pruebas realizadas, mientras que los resultados, divididos por problemas, son los siguientes:

1. **base_1_day.pddl:** El menor coste obtenido es de 4.096 m., con una longitud de 7 acciones. Estos han sido obtenidos por LPG, Metric-FF y CBP
2. **base_m1_day.pddl:** El menor coste obtenido es de 5.669 m., con una longitud de 12 acciones. Este ha sido obtenido por LPG y CBP.
3. **big.pddl:** El menor coste obtenido es de 263.062 m., con una longitud de 114 acciones. Este ha sido obtenido por Metric-FF con la configuración de búsqueda EHC+H con A*epsilon. Ningún otro planificador o configuración ha conseguido encontrar solución en el tiempo establecido.
4. **restrictive.pddl:** El menor coste obtenido es de 43.756 m., con una longitud de 13 acciones. Este ha sido obtenido por LPG y CBP.

A partir de los resultados obtenidos, se observa lo siguiente. En primer lugar, big.pddl es un problema muy grande, cuya solución es difícil de obtener, sobretodo con tiempo tan limitado. Es por ello que Metric-FF con su configuración por defecto ha sido la única ejecución que ha conseguido obtener solución, destacando que esta solución no es la óptima.

En segundo lugar, LPG ofrece muy buen rendimiento, encontrando la mejor solución en el tiempo establecido, en el resto de problemas.

Similar a LPG, CBP también ofrece un buen rendimiento, en especial los algoritmos:

- 2. CEHC (EHC with and without costs)
- 3. anytime BFS (bounded by g value and repeated states prune)
- 4. anytime BFS + helpful and secondary lists
- 9. anytime BFS prune + lookahead + helpful and secondary lists
- 11. modified Hill-Climbing
- 16. depth first with chronological backtracking sorting actions using ff heuristic (for Roller paper)
- 17. anytime BFS + FF heuristic policy lookahead with horizon (for Roller paper)
- 18. anytime BFS + FF heuristic policy lookahead with horizon + rescue nodes (for Roller paper).

Finalmente, se puede concluir que el uso de LPG o CBP puede ser beneficioso para el proyecto, debido a la calidad de sus planes frente a la implementación actual con Metric-FF. Sin embargo, deberá mantenerse el uso de Metric-FF con su configuración EHC+H con A*epsilon como alternativa para los casos en los que cualquiera de los dos planificadores no consiga encontrar una solución en el tiempo establecido.

6. MARCO REGULADOR

En este capítulo se introducen las normativas y leyes existentes en materia de privacidad digital y derechos de autor. Asimismo se introducen los elementos utilizados para cumplirlas.

6.1. Privacidad

Al igual que la seguridad, es muy importante preservar la privacidad de los usuarios. En este proyecto, la privacidad se ha tenido muy en cuenta a la hora de desarrollar el proyecto, prestando especial atención en respetarla en todo momento.

Actualmente, están en vigor dos leyes, la Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales y el Reglamento General de Protección de Datos Europeo.

La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [140], con entrada en vigor a fecha de 07 de diciembre de 2018, establece que todo ciudadano tiene derecho a el “acceso, rectificación, supresión, oposición, derecho a la limitación del tratamiento y derecho a la portabilidad” [140] de sus datos personales. Además, establece que debe existir un responsable y encargado de los datos y regula las infracciones por el incumplimiento de esta Ley.

Por otra parte, el Reglamento (EU) 2016/679 [141], conocido como Reglamento General de Protección de Datos Europeo, entrado en vigor el 24 de mayo de 2016, establece una serie de derechos digitales para todos los ciudadanos de la Unión Europea, similares a los de la Ley Orgánica 3/2018, de 5 de diciembre. Además, establece que todas las empresas extranjeras que poseen usuarios Europeos deben cumplir esta normativa, establece normativas que regulan los sucesos de como las violaciones y el filtrado de datos y establece nuevas sanciones por el incumplimiento de la Ley.

Como plataforma personalizada, los datos personales, las preferencias, las puntuaciones y las acciones de los usuarios dentro de estas pueden ser utilizados para mostrar anuncios, vender sus datos a terceras compañías y conocer mucho sobre una persona. Con el objetivo de ser transparentes, se ha creado una política de privacidad [142], disponible en la página del proyecto. Esta política de privacidad:

- Cumple las normativas de la Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales y el Reglamento General de Protección de Datos Europeo.

- Solo recogen los datos personales e identificativos estrictamente necesarios para el funcionamiento, desarrollo y mejora de la plataforma.
- Los datos de dispositivo son solamente utilizados con fines de análisis y de mejora de la plataforma y no se relacionan con ningún usuario.
- Ninguna información es compartida, enviada o vendida a terceros.
- No se comparte ninguna información con empresas de publicidad.
- No se extrae información para obtener tendencias personales y de comportamiento.
- No se monetiza ninguna información.
- Los datos del usuario son de su propiedad y puede eliminar, gestionar y descargar sus datos de forma libre y en cualquier momento.

6.2. Ley de Propiedad Intelectual y licencias

La propiedad intelectual [143] es una serie de derechos que se atribuyen a los autores y a sus titulares por sus obras. Estos derechos permiten a sus autores y sus titulares la explotación de sus obras, el reconocimiento de ellos por las mismas y la retribución económica por su uso por parte de personas o empresas ajenas. Asimismo, las leyes de propiedad intelectual protegen, de forma legal, la propiedad intelectual de las obras.

Por ello, en este trabajo se han respetado todos los derechos de autor de todos los materiales utilizados. Asimismo, uno de los objetivos planteados para este proyecto ha sido el uso primario de software y material Open Source o de licencia libre. Gracias a estas licencias, cualquiera puede modificar y utilizar el software o el material de forma libre, siempre y cuando se cumplan las restricciones que se impongan en estas [144].

A la hora de desarrollar este proyecto, se ha tenido especial cuidado en respetar las licencias y derechos de autor, además de utilizar todo el material y software libre posible. Todas las licencias utilizadas en el proyecto se muestran en la Tabla 6.1.

TABLA 6.1: SOFTWARE Y MATERIAL UTILIZADO Y SUS LICENCIAS

Nombre	Licencia	Autores
Software		
axios	MIT	Matt Zabriskie
bcrypt	MIT	Nicholas Campbell
Bootstrap	MIT	Twitter, Inc.
		Autores de Bootstrap
Bootstrap Icons	MIT	Twitter, Inc.
		Autores de Bootstrap
cState	MIT	Mantas Vilčinskas

Nombre	Licencia	Autores
csv-parser	MIT	Mathias Buus
fastify	MIT	Fastify Team
Font Awesome	CC BY 4.0	Font Awesome
	MIT	
geolib	MIT	Manuel Bieh
Leaflet	-	Vladimir Agafonkin
Leaflet.Curve	MIT	elfalem
MariaDB	GPL v2	MariaDB Corporation
mariadb	LGPL v2.1	MariaDB Corporation
Neon	Apache	Contribuyentes de Neon
nodemailer	-	Andris Reinman
OpenTileMaps	-	Klokan Technologies GmbH
ora	MIT	Sindre Sorhus
pg	MIT	Brian Carlson
pug	MIT	Contribuyentes de Pug
Puppeteer	Apache	Google Inc.
sequelize	MIT	Contribuyentes de Sequelize
sqlite3	BSD-3-Clause	MapBox
Umami	MIT	Mike Cao

Imágenes

Calle Gran Vía	Unsplash	Florian Wehde
Cartel luminoso Tío Pepe	Unsplash	Jhosef Anderson Cardich Palma
Jardín Botánico de Atocha	Unsplash	Mindaugas Petrusis
Lago de El Retiro	Unsplash	Javier Martinez
Palacio de Cristal de El Retiro	Unsplash	Javier Martinez
Puerta de Alcalá	Unsplash	Jean Estrella
Imágenes de lugares	CC0 CC-BY CC-BY-SA GPL/LGPL	Wikimedia Wikipedia

Mapas

OpenStreetMap	ODbL	OpenStreetMap
MapTiler Data	MapTiler data license	MapTiler

Datos

Datos de Foursquare	-	Foursquare Inc.
Datos de Google Maps	-	Google Inc.

Finalmente, cabe destacar que el proyecto, siguiendo el espíritu de la mayoría de licencias utilizadas, también viene acompañado de una licencia de software libre. En este caso, se ha optado por utilizar la licencia GPL v3.0. Esta licencia solamente restringe que se mencione la autoría correspondiente y que todos los cambios que se produzcan, ya sean contribuciones al proyecto o de forma comercial, se publiquen de forma libre. Asimismo, todas las contribuciones de terceras personas se les aplicará la misma licencia.

7. ORGANIZACIÓN DEL PROYECTO

En este capítulo se muestra la planificación, organización y costes durante el desarrollo del proyecto.

7.1. Planificación

Se han desarrollado las siguientes fases:

1. **Fase 1: Estado del arte** (15/12/2020 – 28/01/2021): A partir de la propuesta del proyecto, durante esta fase se ha investigado sobre posibles herramientas, algoritmos y tecnologías que permitiesen desarrollar el proyecto.
2. **Fase 2: Análisis del sistema** (29/01/2021 – 10/02/2021): Durante esta fase se ha analizado y establecido una posible solución para el sistema. En esta fase se ha establecido una descripción del proyecto; se han creado ejemplos de usuarios potenciales que han permitido establecer los requisitos y casos de uso; se ha creado un prototipo no funcional de la interfaz de usuario y se han establecido las clases de las que hará uso el sistema.
3. **Fase 3: Implementación** (11/02/2021 – 12/04/2021): Durante esta fase se ha desarrollado el sistema, a partir de lo obtenido en la fase anterior. Esta fase se divide en varias sub-fases:
 - **Obtención de datos** (11/02/2021 – 23/02/2021): Durante esta sub-fase se ha extraído y procesado el contenido que formará parte de la base de datos. Además, se ha diseñado el modelo relacional que deberá cumplir la base de datos.
 - **Interfaz de usuario** (24/02/2021 – 29/03/2021): Durante esta sub-fase se ha creado una interfaz de usuario y se le ha dotado de funcionalidad y comunicación con la API REST.
 - **Diseño de base de datos** (01/03/2021 – 02/03/2021): Durante esta sub-fase se ha creado la base de datos acorde al modelo relacional diseñado.
 - **API REST** (03/03/2021 – 10/03/2021): Durante esta sub-fase se ha implementado la API REST, junto a todos sus controladores de usuario, autenticación, recomendaciones, planificaciones y puntuaciones.
 - **Cronjobs** (04/03/2021 – 05/03/2021): Durante esta sub-fase se han diseñado y creado las tareas que se ejecutan de forma periódica.
 - **Sistema recomendador** (11/03/2021 – 23/03/2021): Durante esta sub-fase se ha implementado el sistema recomendador híbrido.

- **Sistema planificador** (24/03/2021 – 29/03/2021): Durante esta sub-fase se ha creado el sistema planificador que incluye traductores de lenguaje PDDL.
 - **Primer despliegue** (30/03/2021 – 30/03/2021): Durante esta sub-fase se ha desplegado la plataforma por primera vez, observando el rendimiento de esta.
 - **Mejoras y arreglos** (30/03/2021 – 12/04/2021): Durante esta sub-fase, y a partir del despliegue, se han implementado mejoras y arreglos encontrados.
4. **Fase 4: Experimentación y pruebas** (13/04/2021 – 27/04/2021): Durante esta fase se han diseñado y se han llevado a cabo la experimentación y pruebas de la plataforma para comprobar su rendimiento y funcionalidad. Esta fase se divide en varias sub-fases:
- **Pruebas unitarias** (14/04/2021 – 18/04/2021): Durante esta sub-fase se han creado y ejecutado pruebas unitarias para la API REST.
 - **Experimentación IU** (19/04/2021 – 25/04/2021): Durante esta sub-fase se han creado y llevado a cabo experimentos para la interfaz de usuario con usuarios.
 - **Pruebas recomendador** (21/04/2021 – 27/04/2021): Durante esta sub-fase se han creado y ejecutado pruebas para el sistema recomendador, usando la base de datos de MovieLens con 1 millón de puntuaciones.
 - **Pruebas planificador** (26/04/2021 – 27/04/2021): Durante esta sub-fase se han creado y ejecutado pruebas para el sistema recomendador con distintos planificadores.
5. **Fase 5: Memoria** (28/04/2021 – 10/05/2021): Durante esta fase final se ha desarrollado esta memoria por completo.

En la Figura 7.1 se muestra el diagrama del Gantt del proyecto dividido por semanas. En total se han dedicado 146 días a la realización de este proyecto.

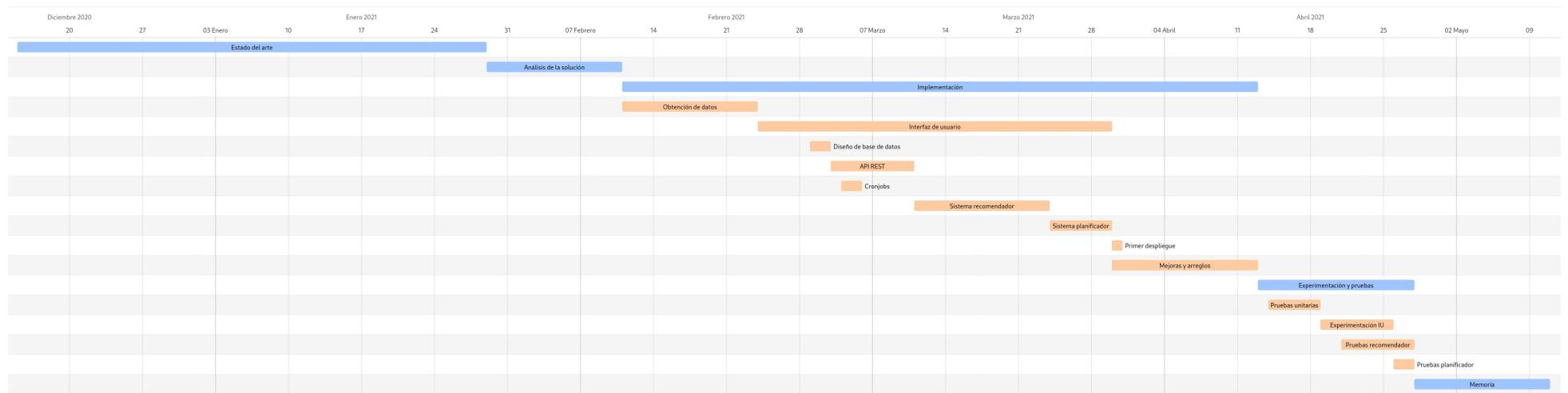


FIGURA 7.1: DIAGRAMA DE GANTT DEL PROYECTO

7.1.1. Metodología

Durante el desarrollo de este proyecto, se ha seguido el paradigma Agile, junto a la metodología DevOps. El paradigma Agile se basa en el proceso iterativo, en donde un proyecto se divide en pequeños segmentos que han de ser completados en un período de tiempo corto, llamado iteración [145]. El proceso de implementación se ha dividido en segmentos de una semana, y se han definido tareas que se deben cumplir durante esa semana. Una vez finalizadas las tareas, se ha comenzado el desarrollo del siguiente segmento. Este paradigma ha permitido que el desarrollo del proyecto sea flexible, rápido e iterativo. El control se ha realizado mediante el uso de tablas Kanban.

Un ejemplo de flexibilidad e iteración se muestra en la figura Figura 7.1, donde se observa que la implementación de la interfaz de usuario se ha extendido a lo largo de varias semanas, iterando continuamente sobre esta con mejoras.

Por otra parte, una vez realizado el primer despliegue, se ha seguido la metodología DevOps. Esta es una serie de tareas y buenas prácticas que permiten el despliegue rápido y continuo de nuevas mejoras y funcionalidades de una plataforma [146]. Tras el primer despliegue, se han realizado despliegues de nuevas versiones de desarrollo públicas cada 5 o 7 días, hasta la finalización del proyecto.

7.2. Presupuesto

Este trabajo es un proyecto de código abierto, por lo que el coste de este es nulo. Sin embargo, en este apartado se simulan y detallan los costes de materiales y licencias, de personal e indirectos.

7.2.1. Costes de personal

En la Tabla 7.1 se calculan los costes de personal que han intervenido en el proyecto. Se ha necesitado personal de jefa de proyecto (Raquel Fuentetaja, tutora), un desarrollador DevOps (Álvaro Galisteo, alumno), un diseñador de sistemas (Álvaro Galisteo, alumno) y un ingeniero de Quality Assurance (Álvaro Galisteo, alumno).

Los datos de salarios se han obtenido de la consultora HAYS [147] y de la encuesta a desarrolladores de StackOverflow de 2020 [148].

Las horas de trabajo se extraen del diagrama de Gantt de la Figura 7.1. Las horas de trabajo de la jefa de proyecto están extraídas de las reuniones realizadas con la tutora.

TABLA 7.1: COSTES DE PERSONAL

Personal	Coste/hora	Horas	Coste
Jefa de proyecto	21,40 €/hora	10 horas	214,04 €
Diseñador de sistemas	12,33 €/hora	80 horas	5.917,81 €
Desarrollador DevOps	13,70 €/hora	480 horas	6.575,34 €
Ingeniero Quality Assurance	12,33 €/hora	112 horas	1.380,82 €
		Total	14.088,01 €

El coste total de personal asciende a **catorce mil ochenta y ocho euros y un céntimo (14.088,01 €)**.

7.2.2. Costes de materiales y licencias

Los costes de materiales y licencias se detallan en la Tabla 7.2. En estos se incluyen las licencias de software (APIs, Base de datos y editores de código), los materiales de los entornos de desarrollo y de despliegue y los materiales del entorno de pruebas.

El cálculo para el proyecto se ha basado en estimar la vida útil de cada material en años o licencia para posteriormente aplicar la siguiente fórmula, teniendo en cuenta la duración del proyecto de 146 días.

$$\frac{\text{precio}}{\text{vida útil}} \cdot \frac{\text{duración proyecto}}{365} \quad (29)$$

TABLA 7.2: COSTE DE MATERIALES Y LICENCIAS

Material	Precio	Vida útil	Coste
Lenovo Thinkpad	912,38 €	5 años	72,99 €
Odroid HC1 x2	165,08 €	30 años	2,20 €
Kingston SSD 128GB x2	44 €	10 años	1,76 €
Sandisk microSD 32GB x2	21,98 €	5 años	1,76 €
Fuente de alimentación 5V/4A x2	24,08 €	20 años	0,48 €
Linode VPS 2 núcleos, 8GB RAM	6,88 €	0,50 años	5,50 €
Dominio .com	7,57 €	1 año	3,03 €
Licencia Sublime Text x2	132,78 €	-	132,78 €
API Foursquare (capa gratuita)	0 €	-	0 €
Licencia MariaDB	0 €	-	0 €
		Total	220,50 €

7.2.3. Costes totales

Finalmente, se incluyen los costes totales del proyecto. En se reúnen los costes de personal, de materiales y licencias, costes indirectos (electricidad y agua, 10% de los costes de materiales y licencias y personal), margen de riesgo (15% de los costes de materiales y licencias y personal), margen de beneficio (15% de los costes de materiales y licencias y personal) e IVA (21%). Estos se detallan en la Tabla 7.3.

TABLA 7.3: COSTES TOTALES

Descripción coste	Coste
Materiales y licencias	220,5 €
Personal	14.088,01 €
Indirectos (10%)	1.430,85 €
Margen de riesgo (15%)	2.146,27 €
Margen de beneficio (15%)	2.146,27 €
IVA (21%)	3.936,42 €
Total	22.681,32 €

El coste total final del proyecto es de **veintidós mil seiscientos ochenta y un euros con treinta y dos céntimos (22.681,32 €)**. Cabe de nuevo recordar que el coste de un proyecto de código abierto es cero (0€).

8. ENTORNO SOCIO-ECONÓMICO

En este apartado se muestra el entorno socio-económico de este proyecto, desarrollando el impacto social y económico que puede incurrir.

La inteligencia artificial se encuentra muy presente en nuestras vidas y tiene usos en gran cantidad de ámbitos, desde las recomendaciones de videos en YouTube, hasta la búsqueda de rutas con Google Maps, pasando por las *timelines* recomendadas de Twitter, Facebook e Instagram. Esta trae consigo avances muy beneficiosos para las personas y el ambiente. Sin embargo aunque estas plataformas se ofrecen de forma gratuita, su beneficio económico suele provenir, por lo general, mediante la recolección de gran cantidad de datos que además son necesarios para el uso de la plataforma. Esta recolección de datos también presenta sus problemas de privacidad y seguridad, como se ha podido demostrar con sucesos como el de Facebook y Cambridge Analytica, usado con el objetivo de influir en la opinión política [149], [150].

Por otra parte, debido a la situación de pandemia que se produjo durante el desarrollo de este proyecto, la industria del turismo ha sido uno de los sectores que más caídas ha sufrido debido al gran riesgo de salud que suponía el transporte y alojamiento de personas [151]. Sin embargo, tras un año del suceso y con la aparición de las vacunas, la industria turística se encuentra en un trayecto de crecimiento, especialmente con el turismo nacional [152], [153].

A menudo, muchos lugares, ciudades y países pasan desapercibidos, siendo eclipsados por otros más conocidos y populares. Aunque existen numerosas listas y publicaciones de “lugares poco conocidos recomendados para visitar” [154], [155], estos siguen siendo poco visitados.

Además, durante las experimentación y validación con usuarios, se ha descubierto que los usuarios pueden descubrir nuevos lugares para visitar que antes no conocían y que la plataforma ha encontrado que pueden ser interesantes para ellos.

En concreto, con este proyecto, se quiere ofrecer una plataforma que demuestre que es posible ofrecer contenido personalizado a los usuarios, respetando y protegiendo su privacidad. Además, se puede crear un impacto económico y social positivo, que puede ayudar a la recuperación económica de las zonas más afectadas por los sucesos de 2020 y 2021.

Finalmente, siempre que el contenido del proyecto se mantenga imparcial, puede ayudar a promocionar puntos de interés y ciudades previamente ocultos o poco visitados, animando a los usuarios a visitarlos de forma que pueda tener un efecto positivo en la

economía y sociedad local o nacional. Además, este efecto de promoción de lugares menos conocidos también puede contribuir a reducir el sobreturismo [156].

9. CONCLUSIONES

En este trabajo se ha desarrollado una plataforma, Hermes, de código abierto, que recomienda puntos de interés para visitar a un usuario, según sus preferencias y puntuaciones y permite planificar itinerarios a partir de estas recomendaciones, facilitando la organización de viajes y permitiendo descubrir de forma personalizada las ciudades. Además, uno de los objetivos planteados en esta plataforma ha sido el crear un servicio que respete la privacidad de los usuarios.

Para ello, se ha realizado un estudio de las posibles técnicas a utilizar para la implementación del sistema recomendador, el sistema planificador, la interfaz de usuario y el servidor. Se ha establecido el uso de un sistema híbrido con la métrica MPIP para el sistema recomendador; del lenguaje PDDL con un planificador clásico para el sistema planificador; del desarrollo de una aplicación web usando las librerías Bootstrap, Bootstrap Icons, Leaflet y Font Awesome; de una API REST usando los lenguajes JavaScript (en Node.js) y Rust, junto a las librerías *fastify*, *sequelize* y *Neon*; y de la base de datos MariaDB.

Durante el análisis se han presentado requisitos, casos de uso y prototipos, así como una descripción del la arquitectura distribuida fácilmente escalable. Posteriormente se ha descrito el proceso de obtención de datos, de diseño de la interfaz de usuario y de la implementación y seguridad de la plataforma, explicando en detalle los componentes y justificando las implementaciones y cambios. Además, se ha creado un proceso de despliegue que hace uso de contenedores con Docker, de servicios de analíticas privadas y de una página de estado de la plataforma. Junto a ello, se ha hecho uso de Cloudflare y Nginx para optimizar y balancear los recursos y el rendimiento de la plataforma.

Adicionalmente, se han establecido una serie de normas y procesos para que cualquier persona pueda contribuir con código o contenido a la plataforma, permitiendo a cualquiera mejorarla.

Posteriormente se han realizado una serie de experimentación y validación. En primer lugar, se han creado pruebas unitarias para la API, en las que todas se han completado satisfactoriamente. Se han establecido pruebas de interfaz mediante evaluación con usuarios que han ofrecido información de posibles cambios para la mejora de la experiencia de usuario, de los cuales algunos cambios ya han sido implementados.

Se ha descubierto que el recomendador tiene una diferencia de 1,39 puntos sobre la escala de 1 a 5, una precisión que se puede considerar aceptable, sobretodo a partir de los resultados de la evaluación con usuarios. Además cuantos más usuarios se usen para la generación de vecinos, más precisión se obtiene.

También, se han establecido pruebas con los planificadores Metric-FF, CBP y LPG. En estas pruebas se ha concluido que el uso de LPG y CBP puede ser beneficioso frente a la implementación actual con Metric-FF, aunque deberá usarse este último como alternativa para a casos complejos.

Por último, se ha analizado la legislación actual y se ha creado una política de privacidad transparente y respetuosa con el usuario.

9.1. Trabajos futuros

Durante el desarrollo de este trabajo se han realizado una serie de decisiones e implementaciones que, no obstante, pueden ser ampliadas o mejoradas en el futuro.

Además de los cambios no introducidos durante la experimentación y validación con usuarios, se propone la monetización de la plataforma, ya sea mediante anuncios, donaciones o suscripciones. De esta manera, se podría obtener un flujo de dinero que podría ser útil para mejorar y ampliar los servicios de la plataforma e incluso introducir nuevas ciudades y lugares para los usuarios.

Con este flujo financiero adicional, se propone el uso de servicios de mapas, como Google Maps, Mapbox o Maptiler, reduciendo la carga de servidor en el servicio de cuadrículas de mapas, además de el uso de planificadores de rutas como Google Maps Directions API o Here Maps API para mejorar los cálculos con distancias y tiempos de las rutas planificadas, ya que actualmente se hace uso de una tabla precalculada de distancias y tiempos.

Con respecto a las tareas que se ejecutan de forma periódica, se podría aumentar su frecuencia de ejecución, para reducir la carga de cómputo durante estos picos o incluso implementar sistemas que ejecuten estas tareas de forma dinámica en función de la carga real de la plataforma o de modelos de predicción de carga.

También sería conveniente hacer uso de la base de datos Redis, para optimizar las sesiones de los usuarios de forma que se eviten consultas innecesarias en la base de datos. Además, se podría realizar una tarea de *refactoring* para mejorar y optimizar el código del proyecto.

Durante el despliegue, se pueden implementar procesos de CI (Continuous Integration) y CD (Continuous Deployment) para optimizar los procesos de pruebas y de despliegues. Similarmente, se propone la migración a plataformas en la nube como AWS, Azure, fly.io, Linode, etc., con el objetivo de mejorar el rendimiento y eficiencia del servicio.

Otro posible campo a estudiar relacionado con el rendimiento y la eficiencia se encuentra en las bases de datos. El código de la plataforma es escalable horizontalmente y distribuido, sin embargo, la base de datos MariaDB solo es escalable verticalmente. Esto limita la base de datos a una sola máquina. Una alternativa que se presenta es Apache Cassandra, una base de datos NoSQL distribuida y altamente disponible que podría acelerar las consultas en base de datos.

También se propone investigar el uso de técnicas basadas en búsqueda local estocástica, como los algoritmos genéticos, para optimizar los planes generados. Además, se propone el desarrollo de otro enfoque de recomendación, donde se sustituye el cálculo de vecinos por una tarea de agrupación incremental mediante el uso de los algoritmos OPTICS o DBSCAN.

Por último, dado que actualmente el sistema solo está disponible para los usuarios de habla hispana y para la ciudad de Madrid, se propone la ampliación de las ciudades actualmente disponibles, empezando por los destinos europeos más populares, junto con un esfuerzo de internacionalización y traducción de la interfaz de usuario y los datos en la base de datos.

BIBLIOGRAFÍA

- [1] World Travel & Tourism Council, «Travel & Tourism Economic Impact», *World Travel & Tourism Council*, 2019. <https://wttc.org/Research/Economic-Impact> (accedido feb. 03, 2021).
- [2] E. Woo, M. Uysal, y M. J. Sirgy, «Tourism Impact and Stakeholders' Quality of Life», *J. Hosp. Tour. Res.*, vol. 42, n.º 2, pp. 260-286, feb. 2018, doi: 10.1177/1096348016654971.
- [3] R. Sharpley, *Tourism, tourists and society, fifth edition*. Taylor and Francis, 2018.
- [4] H. Bowcott, «Powered by data, driven by people: The travel sector's future», *McKinsey*, nov. 03, 2017. <https://www.mckinsey.com/industries/travel-logistics-and-transport-infrastructure/our-insights/powerd-by-data-driven-by-people-the-travel-sectors-future> (accedido feb. 03, 2021).
- [5] S. Staab y H. Werthner, «Intelligent Systems for Tourism», *IEEE Intell. Syst.*, vol. 17, n.º 6, pp. 53-66, nov. 2002, doi: 10.1109/MIS.2002.1134362.
- [6] The Teleregister Corporation, «Special Purpose Electronic Engineering... That Sets the Pace!», 1954. Accedido: feb. 03, 2021. [En línea]. Disponible en: <http://archive.computerhistory.org/resources/text/Teleregister/Teleregister.SpeciaIPurposeSystems.1956.102646324.pdf>.
- [7] Z. Wichter, «How One Computer System Tangled Up Several Airlines», *The New York Times*, abr. 29, 2019. <https://www.nytimes.com/2019/04/29/business/airlines-computer-glitch-sabre.html> (accedido feb. 03, 2021).
- [8] «Personal Travel Planner», *The Fearless Foreigner*. <https://www.thefearlessforeigner.com/personal-travel-planner/> (accedido feb. 03, 2021).
- [9] «TRAVEL PLANNER», *Cualquier Destino*. <https://cualquierdestino.es/travel-planner/> (accedido feb. 03, 2021).
- [10] C. C. Aggarwal, «An Introduction to Recommender Systems», en *Recommender Systems*, Springer International Publishing, 2016, pp. 1-28.
- [11] C. C. Aggarwal, «Model-Based Collaborative Filtering», en *Recommender Systems*, Springer International Publishing, 2016, pp. 71-138.

- [12] A. Umanets, A. Ferreira, y N. Leite, «GuideMe – A Tourist Guide with a Recommender System and Social Interaction», *Procedia Technol.*, vol. 17, pp. 407-414, 2014, doi: 10.1016/j.protcy.2014.10.248.
- [13] M. Elahi, F. Ricci, y N. Rubens, «A survey of active learning in collaborative filtering recommender systems», *Computer Science Review*, vol. 20. Elsevier Ireland Ltd, pp. 29-50, may 01, 2016, doi: 10.1016/j.cosrev.2016.05.002.
- [14] M. Elahi, F. Ricci, y N. Rubens, «Active learning in collaborative filtering recommender systems», en *Lecture Notes in Business Information Processing*, 2014, vol. 188, pp. 113-124, doi: 10.1007/978-3-319-10491-1_12.
- [15] Yu Xin, «Challenges in Recommender Systems : Scalability , Privacy , and Structured Recommendations», Massachusetts Institute of Technology, 2015.
- [16] F. O. Isinkaye, Y. O. Folajimi, y B. A. Ojokoh, «Recommendation systems: Principles, methods and evaluation», *Egyptian Informatics Journal*, vol. 16, n.º 3. Elsevier B.V., pp. 261-273, nov. 01, 2015, doi: 10.1016/j.eij.2015.06.005.
- [17] I. Gunes, C. Kaleli, A. Bilge, y H. Polat, «Shilling attacks against recommender systems: a comprehensive survey», *Artif. Intell. Rev.*, vol. 42, n.º 4, pp. 767-799, dic. 2014, doi: 10.1007/s10462-012-9364-9.
- [18] E. Trecul, «Amazon fake reviews: a problem and a 5-star industry - Red Points», *Red Points*. <https://www.redpoints.com/blog/amazon-fake-reviews/> (accedido feb. 08, 2021).
- [19] G. Magana, «Amazon has a fake product review problem - Business Insider - Business Insider», *Business Insider*, abr. 17, 2019. <https://www.businessinsider.com/amazon-fake-product-review-problem-2019-4> (accedido feb. 08, 2021).
- [20] C. Guthrie Weissman, «“We are getting our ass kicked”: Amazon sellers grapple with a fake review problem», *Modern Retail*, feb. 02, 2018. <https://www.modernretail.co/platforms/we-are-getting-our-ass-kicked-amazon-sellers-grapple-with-a-fake-review-problem/> (accedido feb. 08, 2021).
- [21] P. A. Chirita, W. Nejdl, y C. Zamfir, «Preventing shilling attacks in online recommender systems», en *Proceedings of the International Workshop on Web Information and Data Management WIDM*, 2005, pp. 67-74, doi: 10.1145/1097047.1097061.
- [22] P. Lops, M. de Gemmis, y G. Semeraro, «Content-based Recommender Systems: State of the Art and Trends», en *Recommender Systems Handbook*, Springer US, 2011, pp. 73-105.

- [23] C. C. Aggarwal, «Ensemble-Based and Hybrid Recommender Systems», en *Recommender Systems*, Springer International Publishing, 2016, pp. 199-224.
- [24] E. Çano y M. Morisio, «Hybrid recommender systems: A systematic literature review», *Intelligent Data Analysis*, vol. 21, n.º 6. pp. 1487-1524, 2017, doi: 10.3233/IDA-163209.
- [25] E. Fix y J. L. Hodges, «Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties», *Int. Stat. Rev. / Rev. Int. Stat.*, vol. 57, n.º 3, p. 238, dic. 1989, doi: 10.2307/1403797.
- [26] S. P. Lloyd, «Least Squares Quantization in PCM», *IEEE Trans. Inf. Theory*, vol. 28, n.º 2, pp. 129-137, 1982, doi: 10.1109/TIT.1982.1056489.
- [27] J. Macqueen, «Some methods for classification and analysis of multivariate observations», en *5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281-297.
- [28] M. Ester, M. Ester, H. Kriegel, J. Sander, y X. Xu, «A density-based algorithm for discovering clusters in large spatial databases with noise», pp. 226--231, 1996, Accedido: jun. 07, 2021. [En línea]. Disponible en: <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.1980>.
- [29] M. Ankerst, M. Ankerst, M. M. Breunig, H. Kriegel, y J. Sander, «OPTICS: Ordering Points To Identify the Clustering Structure», pp. 49--60, 1999, Accedido: jun. 07, 2021. [En línea]. Disponible en: <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.6542>.
- [30] M. Van Otterlo y M. Wiering, «Reinforcement learning and markov decision processes», en *Adaptation, Learning, and Optimization*, vol. 12, Springer Verlag, 2012, pp. 3-42.
- [31] C. Saluja, «Collaborative Filtering based Recommendation Systems exemplified», *Towards Data Science*. <https://towardsdatascience.com/collaborative-filtering-based-recommendation-systems-exemplified-ecbffe1c20b1> (accedido feb. 11, 2021).
- [32] Kent State University, «SPSS Tutorials: Pearson Correlation». <https://libguides.library.kent.edu/SPSS/PearsonCorr> (accedido feb. 11, 2021).
- [33] H. Liu, Z. Hu, A. Mian, H. Tian, y X. Zhu, «A new user similarity model to improve the accuracy of collaborative filtering», *Knowledge-Based Syst.*, vol. 56, pp. 156-166, ene. 2014, doi: 10.1016/j.knosys.2013.11.006.
- [34] Wikipedia, «Cosine similarity». https://en.wikipedia.org/wiki/Cosine_similarity (accedido feb. 11, 2021).

- [35] P. Jaccard, «THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.», *New Phytol.*, vol. 11, n.º 2, pp. 37-50, feb. 1912, doi: 10.1111/j.1469-8137.1912.tb05611.x.
- [36] H. J. Ahn, «A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem», *Inf. Sci. (Ny.)*, vol. 178, n.º 1, pp. 37-51, ene. 2008, doi: 10.1016/j.ins.2007.07.024.
- [37] M. Jamali y M. Ester, *TrustWalker: A Random Walk Model for Combining Trust-based and Item-based Recommendation*. 2009.
- [38] S. Manochandar y M. Punniyamoorthy, «A new user similarity measure in a new prediction model for collaborative filtering», *Appl. Intell.*, vol. 51, n.º 1, pp. 586-615, ago. 2020, doi: 10.1007/s10489-020-01811-3.
- [39] «ICAPS – International Conference on Automated Planning and Scheduling». <https://www.icaps-conference.org/> (accedido abr. 30, 2021).
- [40] International Conference on Automated Planning and Scheduling, «IPC 2018». <https://ipc2018.bitbucket.io/> (accedido may 05, 2021).
- [41] A. Barrett *et al.*, «The Planning Domain Definition Language». oct. 1998.
- [42] «What is PDDL? - Planning.wiki - The AI Planning & PDDL Wiki». <https://planning.wiki/guide/whatis/pddl> (accedido jun. 07, 2021).
- [43] «Numeric Fluents - PDDL 2.1 Requirements - Planning.wiki - The AI Planning & PDDL Wiki». <https://planning.wiki/ref/pddl21/req#numeric-fluents> (accedido jun. 07, 2021).
- [44] M. Fox y D. Long, «PDDL+: Modelling Continuous Time-dependent Effects», Durham. Accedido: abr. 30, 2021. [En línea]. Disponible en: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.5965>.
- [45] M. Ghallab, D. Nau, y P. Traverso, *Automated Planning and Acting*. Cambridge University Press, 2016.
- [46] D. Nau, «Current Trends in Automated Planning», *AI Magazine*, 2007.
- [47] M. Castilho, L. A. Kunzle, E. Lecheta, V. Palodeto, y F. Silva, «An Investigation on Genetic Algorithms for Generic STRIPS Planning». Accedido: jun. 07, 2021. [En línea]. Disponible en: <http://www.inf.ufpr.br/~marcoshttp://www.cpgei.cefetpr.br/~fabiano>.
- [48] L. Castillo *et al.*, «samap: An user-oriented adaptive system for planning tourist visits», *Expert Syst. Appl.*, vol. 34, n.º 2, pp. 1318-1332, feb. 2008, doi: 10.1016/j.eswa.2006.12.029.

- [49] F. Lorenzi, S. Loh, y M. Abel, «PersonalTour: A recommender system for travel packages», en *Proceedings - 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2011*, ago. 2011, vol. 2, pp. 333-336, doi: 10.1109/WI-IAT.2011.69.
- [50] K. H. Lim *et al.*, «PersTour: A personalized tour recommendation and planning system», en *CEUR Workshop Proceedings*, 2016, vol. 1628.
- [51] B. W. Benson, «JavaScript», *ACM SIGPLAN Not.*, vol. 34, n.º 4, pp. 25-27, abr. 1999, doi: 10.1145/312009.312023.
- [52] B. Doyle y C. Videira Lopes, «Survey of Technologies for Web Application Development», 2008.
- [53] «Server-side scripting», *Wikipedia*. https://en.wikipedia.org/wiki/Server-side_scripting#Languages (accedido abr. 30, 2021).
- [54] Shah Karan, «Client-side Vs. Server-side Rendering: What to choose when?», nov. 29, 2019. <https://www.solutelabs.com/blog/client-side-vs-server-side-rendering-what-to-choose-when> (accedido abr. 30, 2021).
- [55] «What is Vanilla CSS and why? - Stack Overflow», *Stack Overflow*, 2016. <https://stackoverflow.com/questions/40115768/what-is-vanilla-css-and-why> (accedido abr. 30, 2021).
- [56] «What is VanillaJS? - Stack Overflow», *Stack Overflow*, 2013. <https://stackoverflow.com/questions/20435653/what-is-vanillajs> (accedido abr. 30, 2021).
- [57] «jQuery». <https://jquery.com/> (accedido abr. 30, 2021).
- [58] «Sass: Syntactically Awesome Style Sheets». <https://sass-lang.com/> (accedido abr. 30, 2021).
- [59] «Bootstrap · The most popular HTML, CSS, and JS library in the world.» <https://getbootstrap.com/> (accedido abr. 30, 2021).
- [60] G. Krasner y S. Pope, «A cookbook for using the Model-View-Controller User Interface paradigm in Smalltalk-80», *J. Object-oriented Program. - JOOP*, vol. 1, 1998.
- [61] Microsoft, «ASP.NET MVC Pattern | .NET». <https://dotnet.microsoft.com/apps/aspnet/mvc> (accedido abr. 30, 2021).
- [62] Pivotal Software, «Spring Framework». <https://spring.io/projects/spring-framework> (accedido abr. 30, 2021).

- [63] Symfony community, «Symfony, High Performance PHP Framework for Web Development». <https://symfony.com/> (accedido abr. 30, 2021).
- [64] Google, «Angular». <https://angular.io/> (accedido abr. 30, 2021).
- [65] Google, «Angular - Get data from a server». <https://angular.io/tutorial/toh-pt6> (accedido abr. 30, 2021).
- [66] Facebook Open Source, «React – A JavaScript library for building user interfaces». <https://reactjs.org/> (accedido abr. 30, 2021).
- [67] E. You, «Vue.js». <https://vuejs.org/> (accedido abr. 30, 2021).
- [68] P. J. Leach, T. Berners-Lee, J. C. Mogul, L. Masinter, R. T. Fielding, y J. Gettys, «Hypertext Transfer Protocol -- HTTP/1.1». 1999, doi: 10.17487/RFC2616.
- [69] «SOAP Specifications», W3. <https://www.w3.org/TR/soap/> (accedido abr. 30, 2021).
- [70] A. Altvater, «SOAP vs. REST: The Differences and Benefits Between the Two Widely-Used Web Service Communication Protocols – Stackify», *Stackify*, 2017. <https://stackify.com/soap-vs-rest/> (accedido abr. 30, 2021).
- [71] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures», University of California, Irvine, 2000.
- [72] W.-M. Thor, «5 top programming languages to learn server-side web development», @twm, 2018. <https://twm.me/best-programming-languages-and-frameworks-for-server-side-web-development/> (accedido abr. 30, 2021).
- [73] The Apache Foundation, «Apache Mahout». <https://mahout.apache.org/> (accedido abr. 30, 2021).
- [74] «TensorFlow». <https://www.tensorflow.org/> (accedido feb. 09, 2021).
- [75] «TensorFlow.js | Machine Learning for Javascript Developers». <https://www.tensorflow.org/js> (accedido feb. 09, 2021).
- [76] J. Hoffmann, «FF: The Fast-Forward Planning System», 2001.
- [77] J. Hoffmann, «The Metric-FF Planning System: Translating “Ignoring Delete Lists” to Numeric State Variables», vol. 20, pp. 291-341, 2003.
- [78] M. Helmert, «The Fast Downward Planning System», *J. Artif. Intell. Res.*, vol. 26, pp. 191-246, 2011.
- [79] R. Fuentetaja, «The CBP planner», Universidad Carlos III de Madrid, 2011.

- [80] A. Gerevini, A. Saetti, y I. Serina, «An approach to efficient planning with numerical fluents and multi-criteria plan quality», *Artif. Intell.*, vol. 172, n.º 8-9, pp. 899-944, 2008, doi: 10.1016/j.artint.2008.01.002.
- [81] A. Gerevini, A. Saetti, y I. Serina, «An Empirical Analysis of Some Heuristic Features for Planning through Local Search and Action Graphs», *Fundam. Inform.*, vol. 107, n.º 2-3, pp. 167-197, 2011, doi: 10.3233/FI-2011-399.
- [82] «WebAssembly». <https://webassembly.org/> (accedido feb. 09, 2021).
- [83] T. J. Watson, «Comparing C WebAssembly and JavaScript performance». <https://tomjwatson.com/blog/c-wasm-vs-js/> (accedido feb. 09, 2021).
- [84] «Why using WebAssembly and Rust improves Node.js performance – IBM Developer». <https://developer.ibm.com/technologies/web-development/articles/why-webassembly-and-rust-together-improve-nodejs-performance/> (accedido feb. 09, 2021).
- [85] E. F. Codd, «A Relational Model of Data for Large Shared Data Banks», *Commun. ACM*, vol. 13, n.º 6, pp. 377-387, jun. 1970, doi: 10.1145/362384.362685.
- [86] Microsoft, «SQL Server 2019 | Microsoft». <https://www.microsoft.com/es-es/sql-server/sql-server-2019> (accedido abr. 30, 2021).
- [87] Oracle, «Database 19c and 21c | Oracle». <https://www.oracle.com/database/technologies/> (accedido abr. 30, 2021).
- [88] PostgreSQL, «PostgreSQL: The world's most advanced open source database». <https://www.postgresql.org/> (accedido abr. 30, 2021).
- [89] Oracle, «MySQL». <https://www.mysql.com/> (accedido abr. 30, 2021).
- [90] «Dead database walking: MySQL's creator on why the future belongs to MariaDB», *Computerworld*. https://www2.computerworld.com.au/article/457551/dead_database_walking_my_sql_creator_why_future_belongs_mariadb/ (accedido abr. 30, 2021).
- [91] N. Leavitt, «Will NoSQL Databases Live Up to Their Promise?», 2010. Accedido: abr. 30, 2021. [En línea]. Disponible en: <https://ieeexplore.ieee.org/document/5410700>.
- [92] MongoDB Inc., «The most popular database for modern apps | MongoDB». <https://www.mongodb.com/> (accedido abr. 30, 2021).
- [93] Redis Inc., «Redis». <https://redis.io/> (accedido abr. 30, 2021).

- [94] Apache Foundation, «Apache Cassandra». <https://cassandra.apache.org/> (accedido may 07, 2021).
- [95] E. Rescorla, «HTTP Over TLS». 2000, Accedido: may 01, 2021. [En línea]. Disponible en: <https://tools.ietf.org/html/rfc2818>.
- [96] Cloudflare, «How Does Public Key Encryption Work? | Public Key Cryptography and SSL | Cloudflare». <https://www.cloudflare.com/learning/ssl/how-does-public-key-encryption-work/> (accedido may 01, 2021).
- [97] «Certificate authority - Wikipedia», *Wikipedia*. https://en.wikipedia.org/wiki/Certificate_authority (accedido may 01, 2021).
- [98] GoDaddy, «Certificados SSL | Asegura tu web y protege tus datos - GoDaddy ES». <https://www.godaddy.com/es-es/seuridad-web/certificado-ssl> (accedido may 01, 2021).
- [99] DigiCert, «Compare TLS/SSL Certificates by Certificate Type | DigiCert.com». <https://www.digicert.com/tls-ssl/compare-certificates> (accedido may 01, 2021).
- [100] GlobalSign, «SSL / TLS CERTIFICATES». <https://shop.globalsign.com/en/ssl-tls-certificates> (accedido may 01, 2021).
- [101] Let's Encrypt, «About Let's Encrypt - Let's Encrypt». <https://letsencrypt.org/about/#> (accedido may 01, 2021).
- [102] J. Hodges, C. Jackson, y A. Barth, «HTTP Strict Transport Security (HSTS)». nov. 2012, Accedido: may 04, 2021. [En línea]. Disponible en: <https://tools.ietf.org/html/rfc6797>.
- [103] Mozilla, «Cross-Origin Resource Sharing (CORS) - HTTP | MDN». <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> (accedido may 01, 2021).
- [104] Mozilla, «Content Security Policy (CSP) - HTTP | MDN». <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP> (accedido may 01, 2021).
- [105] Google, «Cross-Site Scripting – Application Security – Google». <https://www.google.com/about/appsecurity/learning/xss/index.html> (accedido may 01, 2021).
- [106] Symantec, «Symantec Internet Security Threat Report», abr. 2008. Accedido: may 04, 2021. [En línea]. Disponible en: https://web.archive.org/web/20080625065121/http://eval.symantec.com/mktginfo/enterprise/white_papers/b-

whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf.

- [107] T. Hunt, «Troy Hunt: Password reuse, credential stuffing and another billion records in Have I been pwned», 2017. <https://www.troyhunt.com/password-reuse-credential-stuffing-and-another-1-billion-records-in-have-i-been-pwned/> (accedido may 04, 2021).
- [108] R. Dwivedi, «What is hashing and salting | Better Programming», 2020. <https://betterprogramming.pub/salting-and-hashing-explained-b76f5af83554> (accedido may 01, 2021).
- [109] N. Provos y D. Mazières, «A Future-Adaptable Password Scheme», en *Proceedings of 1999 USENIX Annual Technical Conference*, 1999, pp. 81-92, Accedido: may 01, 2021. [En línea]. Disponible en: http://www.usenix.org/events/usenix99/provos/provos_html/node1.html.
- [110] H. Fernandez, «Una visión diferente sobre escalabilidad y modelos de negocio», *EconomíaTIC*. <https://economiatic.com/concepto-escalabilidad/> (accedido may 01, 2021).
- [111] Atlassian, «Cloud vs. on-premise: which is better? | Atlassian». <https://www.atlassian.com/cloud/cloud-vs-on-premise> (accedido may 01, 2021).
- [112] Amazon Web Services, «Benefits». <https://aws.amazon.com/application-hosting/benefits/> (accedido may 01, 2021).
- [113] Microsoft, «What is Azure—Microsoft Cloud Services | Microsoft Azure». <https://azure.microsoft.com/en-us/overview/what-is-azure/> (accedido may 01, 2021).
- [114] Microsoft, «What is BareMetal Infrastructure on Azure? - Azure Baremetal Infrastructure | Microsoft Docs». <https://docs.microsoft.com/en-us/azure/baremetal-infrastructure/concepts-baremetal-infrastructure-overview> (accedido may 01, 2021).
- [115] Microsoft, «What Is a Virtual Machine and How Does It Work | Microsoft Azure». <https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/#what-used-for> (accedido may 01, 2021).
- [116] Docker Inc., «What is a Container? | App Containerization | Docker». <https://www.docker.com/resources/what-container> (accedido may 01, 2021).
- [117] Canonical Ltd., «Linux Containers». <https://linuxcontainers.org/> (accedido may 01, 2021).
- [118] Docker Inc., «Docker». <https://www.docker.com/> (accedido may 01, 2021).

- [119] Amazon Web Services, «AWS Auto Scaling».
<https://aws.amazon.com/autoscaling/> (accedido may 01, 2021).
- [120] Microsoft, «Use the cluster autoscaler in Azure Kubernetes Service (AKS) - Azure Kubernetes Service | Microsoft Docs».
<https://docs.microsoft.com/en-us/azure/aks/cluster-autoscaler> (accedido may 01, 2021).
- [121] Cloud Native Computing Foundation, «Kubernetes». <https://kubernetes.io/> (accedido may 01, 2021).
- [122] Docker Inc., «Swarm mode overview | Docker Documentation».
<https://docs.docker.com/engine/swarm/> (accedido may 01, 2021).
- [123] Fastify Team, «Benchmarks». <https://www.fastify.io/benchmarks/> (accedido may 03, 2021).
- [124] NearForm, «Reaching Ludicrous Speed with Fastify - NearForm», dic. 13, 2017.
<https://www.nearform.com/blog/reaching-ludicrous-speed-with-fastify/> (accedido may 03, 2021).
- [125] Foursquare, «Rate Limits | Places API».
<https://developer.foursquare.com/docs/places-api/rate-limits/> (accedido may 05, 2021).
- [126] «Why is Node.js single threaded? - Stack Overflow».
<https://stackoverflow.com/a/17959746> (accedido may 06, 2021).
- [127] Á. Galisteo Álvarez, «hermes_openapi.yaml», 2021.
https://github.com/SrGMC/hermes/blob/main/docs/api/hermes_openapi.yaml (accedido jun. 07, 2021).
- [128] OpenAPI Initiative, «OpenAPI Specification v3.1.0». 2021, Accedido: may 06, 2021. [En línea]. Disponible en: <https://spec.openapis.org/oas/v3.1.0>.
- [129] Imperva, «Cookie Tampering - Imperva Documentation Portal».
https://docs.imperva.com/bundle/on-premises-knowledgebase-reference-guide/page/cookie_tampering.htm (accedido may 06, 2021).
- [130] Imperva, «Cookie Poisoning - Imperva Documentation Portal».
https://docs.imperva.com/bundle/on-premises-knowledgebase-reference-guide/page/cookie_poisoning.htm (accedido may 06, 2021).
- [131] Cloudflare, «Cloudflare DDoS Protection | Intelligent DDoS Mitigation | Cloudflare». <https://www.cloudflare.com/ddos/> (accedido may 06, 2021).
- [132] N. Campbell, «bcrypt - npm». <https://www.npmjs.com/package/bcrypt> (accedido may 17, 2021).

- [133] «Recommended # of rounds for bcrypt - Information Security Stack Exchange». <https://security.stackexchange.com/questions/17207/recommended-of-rounds-for-bcrypt> (accedido may 06, 2021).
- [134] Fastify Team, «fastify/fastify-rate-limit: A low overhead rate limiter for your routes». <https://github.com/fastify/fastify-rate-limit> (accedido may 06, 2021).
- [135] A. Townsend, «Brute Force Attacks: Defend Your Systems | OneLogin», 2020. <https://www.onelogin.com/blog/brute-force-attacks> (accedido may 06, 2021).
- [136] «Semantic Versioning 2.0.0 | Semantic Versioning». <https://semver.org/> (accedido may 05, 2021).
- [137] Á. Galisteo Álvarez, «Contribuir a Hermes», 2021. <https://github.com/SrGMC/hermes/blob/main/docs/es/CONTRIBUTE.md#contribuir-con-lugares> (accedido jun. 07, 2021).
- [138] «Use JSDoc: Index». <https://jsdoc.app/> (accedido may 05, 2021).
- [139] F. Maxwell Harper y J. Konstan A., «The MovieLens Datasets: History and Context», *ACM Trans. Interact. Intell. Syst.*, vol. 5, n.º 4, p. 19, 2015, doi: 10.1145/2827872.
- [140] *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. Madrid, 2018.
- [141] *Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos*. Bruselas, Estrasburgo, 2016.
- [142] Á. Galisteo Álvarez, «Política de Privacidad - Hermes», 2021. <https://travelhermes.com/privacy/> (accedido jun. 08, 2021).
- [143] Ministerio de Cultura y Deporte, «La propiedad intelectual en general | Ministerio de Cultura y Deporte». <https://www.culturaydeporte.gob.es/cultura/propiedadintelectual/la-propiedad-intelectual/preguntas-mas-frecuentes/la-propiedad-intelectual.html> (accedido may 02, 2021).
- [144] GitHub Inc., «Choose an open source license | Choose a License». <https://choosealicense.com/> (accedido may 02, 2021).
- [145] Agile Alliance, «What is Agile Software Development?», *Agile Alliance*. <https://www.agilealliance.org/agile101/> (accedido jun. 07, 2021).

- [146] Microsoft, «What is DevOps? DevOps Explained | Microsoft Azure». <https://azure.microsoft.com/en-us/overview/what-is-devops/> (accedido jun. 07, 2021).
- [147] HAYS, «Guía del Mercado Laboral 2021 - Hays», 2021. <https://guiasalarial.hays.es/> (accedido may 09, 2021).
- [148] StackOverflow, «Stack Overflow Salary Report: Spain», 2020. Accedido: may 09, 2021. [En línea]. Disponible en: <https://info.stackflowsolutions.com/Talent-salary-report-spain.html>.
- [149] J. Pastor, «El escándalo de Cambridge Analytica resume todo lo que está terriblemente mal con Facebook», *Xataka*, 2018. <https://www.xataka.com/privacidad/el-escandalo-de-cambridge-analytica-resume-todo-lo-que-esta-terriblemente-mal-con-facebook> (accedido may 07, 2021).
- [150] M. González, «Qué ha pasado con Facebook: del caso Cambridge Analytica al resto de polémicas más recientes», *Xataka*, 2018. <https://www.xataka.com/legislacion-y-derechos/que-ha-pasado-con-facebook-del-caso-cambridge-analytica-al-resto-de-polemicas-mas-recientes> (accedido may 07, 2021).
- [151] Naciones Unidas, «Informe de políticas: La COVID-19 y la transformación del turismo», 2020. Accedido: may 07, 2021. [En línea]. Disponible en: https://www.un.org/sites/un2.un.org/files/policy_brief_covid-19_and_transforming_tourism_spanish.pdf.
- [152] L. Queijeiro y C. Gacía Baena, «Ahora sí se reactiva el turismo nacional: las reservas se disparan - NIUS», *NIUS*, 2021. https://www.niusdiario.es/economia/consumo/turismo-nacional-fin-estado-alarma-9-mayo-movilidad-disparan-reservas-hoteles-casas-rurales-vacaciones-verano-2021_18_3132945328.html (accedido may 07, 2021).
- [153] N. Mira y M. Lozano, «El fin del estado de alarma pronostica un boom del turismo nacional», *ABC*, 2021. https://www.abc.es/sociedad/abci-estado-alarma-pronostica-boom-turismo-nacional-202105052107_noticia.html (accedido may 07, 2021).
- [154] J. Rose Smith, «Why you should go to the world's least-visited countries», *CNN*, 2019. <https://edition.cnn.com/travel/article/least-visited-countries-travel/index.html>.
- [155] J. Pérez Pacheco, «26 Lugares secretos y poco conocidos que quizás no conocías en Madrid | Fotonazos - Viajes y fotografías», *Fotonazos*, 2018.

<https://www.fotonazos.es/2018/01/26-lugares-secretos-poco-conocidos-quizas-no-conocias-madrid/> (accedido may 07, 2021).

- [156] F. Street, «Overtourism: Can world be saved from too much travel? | CNN Travel», *CNN*, oct. 03, 2018. <https://edition.cnn.com/travel/article/overtourism-solutions/index.html> (accedido may 07, 2021).

ANEXO

I. Evaluaciones de usuarios

TABLA 1: EVALUACIÓN DEL USUARIO 1

Usuario	PI1	PI2	PI3	PI4	PI5
	23 años	Mujer	Media	Poco	Si
	PF1	Sí, aunque añadiría un tema claro.			
	PF2	Ninguno.			
	PF3	Si, son adecuadas.			
	PF4	Sí, aunque no rellena todos los días.			
1	PF5	Sí, ha sido rápida.			
Notas de evaluación					
Quitar botón de siguiente foto si solo hay una.					
El listado de búsqueda desaparece.					
Algunos títulos y elementos parecen poco descriptivos.					
Insertar lugares no queda claro.					
Editar un plan no indica elementos requeridos.					

TABLA 2: EVALUACIÓN DEL USUARIO 2

Usuario	PI1	PI2	PI3	PI4	PI5
	22 años	Hombre	Media	Nada	No
	PF1	Me parece poco intuitiva.			
	PF2	Los botones para planificar y los menús.			
	PF3	Si, son adecuadas.			
2	PF4	A medias. Me parecen poco entendibles.			
	PF5	Sí, es rápida.			
Notas de evaluación					
Volver a atrás en la ayuda no funciona.					
No le ha quedado claro que hacen algunos botones ni como empezar.					
CAPTCHA muy complicado.					

TABLA 3: EVALUACIÓN DEL USUARIO 3

Usuario	PI1	PI2	PI3	PI4	PI5
	22 años	Mujer	Media	Mucho	No
	PF1	Fácil, aunque cerrar sesión está muy oculto.			
	PF2	Ninguno.			
	PF3	Regular.			
	PF4	Si, están bien.			
3	PF5	Es rápida, aunque el mapa ha tardado un poco en cargar.			
Notas de evaluación					
Algunos títulos no son autoexplicativos.					
Faltan sugerencias al añadir lugares en un plan.					
Volver en la Ayuda no funciona.					
Cerrar sesión está muy oculto.					

TABLA 4: EVALUACIÓN DEL USUARIO 4

Usuario	PI1	PI2	PI3	PI4	PI5
	21 años	Hombre	Alta	A veces	A medias
	PF1	Si, es fácil de usar.			
	PF2	Nada, es sencilla.			
	PF3	Si, están bien.			
4	PF4	A medias. Lugares cercanos no salían seguidos.			
	PF5	Sí, aunque había fotos que no cargaban.			
Notas de evaluación					
El buscador en las recomendaciones necesita mayor contraste.					
Agrupar zonas cercanas en base de datos (p. ej. en Sol).					
Algunos lugares no tienen fotos.					

TABLA 5: EVALUACIÓN DEL USUARIO 5

Usuario	PI1	PI2	PI3	PI4	PI5
	50 años	Hombre	Alta	Mucho	Si
	PF1	A medias.			
	PF2	Muchos. No se que hace nada ni me dicen nada.			
	PF3	Si, son adecuadas.			
	PF4	Sí, salen bien.			
5	PF5	Si, es rápida.			
Notas de evaluación					
Hace falta un tutorial de inicio.					
Algunos títulos no son autoexplicativos.					
El buscador no busca por categorías, solo por lugares.					
No tiene claro que es la plataforma.					
Hacer más claros los botones de añadir puntuaciones.					

TABLA 6: EVALUACIÓN DEL USUARIO 6

Usuario	PI1	PI2	PI3	PI4	PI5
	50 años	Mujer	Media	Mucho	Si
	PF1	A medias.			
	PF2	Muchos. Por ejemplo, ¿porqué tengo que pulsar en el mapa para poner el punto de inicio?.			
	PF3	Si, son adecuadas.			
6	PF4	Sí, salen bien.			
	PF5	Rápida.			
Notas de evaluación					
Hace falta un tutorial de inicio.					
Algunos títulos no son autoexplicativos.					
Punto de inicio debería poder buscarse en vez de pulsar en el mapa.					
Hacer más fácil el proceso de crear un plan si no se está en el recomendador.					

II. Resultados pruebas del sistema

TABLA 7: RESULTADOS PRUEBAS RECOMENDADOR

N	M	D	MSE	MAE
15	10	5	1,39	0,89
15	8	5	1,4	0,89
15	6	5	1,4	0,89
10	10	5	1,42	0,9
10	8	5	1,42	0,9
15	4	5	1,42	0,9
10	6	5	1,43	0,9
10	4	5	1,45	0,91
15	2	5	1,47	0,92
5	10	5	1,48	0,92
5	8	5	1,48	0,92
5	6	5	1,51	0,93
10	2	5	1,51	0,93
15	10	10	1,52	0,93
15	10	20	1,52	0,93
15	8	10	1,53	0,93
15	8	20	1,53	0,93
15	6	20	1,53	0,93
15	6	10	1,53	0,93
5	4	5	1,54	0,94
10	10	10	1,54	0,93
10	10	20	1,54	0,93
10	8	20	1,54	0,94
10	8	10	1,54	0,94
15	4	10	1,54	0,94
15	4	20	1,55	0,94
10	6	20	1,55	0,94
10	6	10	1,56	0,94
10	4	10	1,57	0,95
10	4	20	1,57	0,95
15	2	10	1,57	0,95
15	2	20	1,58	0,95
5	10	20	1,58	0,95
5	10	10	1,58	0,95
5	8	10	1,59	0,95
5	8	20	1,59	0,95
5	6	10	1,6	0,96
5	6	20	1,6	0,96

N	M	D	MSE	MAE
10	2	10	1,6	0,96
5	2	5	1,61	0,96
10	2	20	1,61	0,96
5	4	10	1,62	0,97
5	4	20	1,63	0,97
5	2	10	1,67	0,98
5	2	20	1,68	0,98

TABLA 8: RESULTADOS PRUEBAS PLANIFICADOR

Debido al tamaño de la tabla, los resultados se encuentran en formato XSLX en el siguiente enlace:

<https://github.com/SrGMC/hermes/blob/main/server/test/planner/results.xlsx>

III. Inglés

1. INTRODUCTION

1.1. Motivation

Tourism is an industry that accounted, in 2019, for 10.3% of global GDP where 1 in 10 jobs is dedicated to this industry [1]. Moreover, tourism is an industry that could be considered as an "umbrella" industry, as it encompasses and helps many other sectors such as commerce and hospitality, generating a great economic, socio-cultural and environmental impact [2], [3]. Also, tourism is mainly driven by information [4] and trust, where millions of people decide their visits and travel itineraries based on recommendations from other people and internet reviews on the internet [5].

Therefore, the industry has always been open to new technologies, however, there is one aspect of the travel process that still requires manual or semi-manual work on the part of the traveler which is the researching and planning of visits. Although there are services provided by third parties that can perform these tasks not everyone is comfortable using these services, either for using these services, either for economic or even privacy reasons.

1.2. Objectives

In this project we propose the creation of a free and open source platform, with a personalised system that allows each user to plan their trips, through a web application, and that respects the privacy of the users, while using open source tools and technologies as much as possible

This main objective is divided two fold. On the one hand, we present the development of a points of interest recommendation system based on personal information provided by a user, as well as on ratings generated by other users. On the other hand, the development of a planning system is presented, interconnected with the aforementioned recommendation system, which allows the user to automatically plan his trip based on certain parameters. Although the planning is presented as automatic, the user will be able to modify it at any time.

Given that this is a very ambitious objective, this work presents a platform project focused, at this time, on the city of Madrid. Thus, once finished, the next objective to be proposed is the scalability and adaptation of the platform to a larger number of cities.

With the implementation of this project, the aim is to acquire knowledge of recommendation and automatic planning algorithms and all their implicit technology, as well as the secure processing of data and the design of user interfaces that are usable and easy to use.

2. CURRENT STATE OF THE ART

2.1. Recommender systems

A recommender system is software whose objective is to recommend items, such as points of interest, that are of interest to a user.

Several methodologies of recommendation are present, split in Collaborative filtering and Content-based filtering.

Recommender systems based on collaborative filtering are mainly based on the ratings of different items generated by multiple users. As the database of items is very large, there will be many unfilled "gaps" of ratings. In this way the system seeks to fill the "gaps", where the preconditions for these with the highest ratings will be those recommended to the user [10].

Content-based filtering systems are based on models generated for each user that arise from the user's interests (e.g. based on purchases and visits) and the characteristics of the items [10].

However, both methodologies have several problems, such as:

- **Cold-start:** This problem arises because, initially, a recommender system has very few ratings to start with, so it is difficult to search for users or similar items with which to make recommendations [10], [12].
- **Scalability:** This problem arises in collaborative filtering systems with the limited computing power and the large amount of data available. Since this type of methodology cannot work with users independently, limiting the amount of data in a method may make it unable to generate optimal results [15].
- **Diversity:** In many cases, content-based filtering can generate recommendations that may be obvious [10]. Because these systems are based on item characteristics, it is very difficult for an item with characteristics other than those modeled for the user to be presented as a recommendation, reducing diversity.
- **Shilling attacks:** This type of problem occurs when many users rating an item in a misleading way. An example of this occurs when a company with an unreliable product pays for multiple users to give positive reviews, creating misleading reviews and hiding real reviews [17].

A hybrid approach is usually used, mixing content-based and collaborative filtering to overcome these problems.

Along with this methodologies, user similarity functions are used to find how much different are users from each other. Common similarity function include Euclidean distance [31], Cosine similarity [34], Pearson coefficient correlation [33], Jaccard coefficient [35] and Mean Square Difference [33]. Along this functions, several new similarity measures were developed such as PIP [36], TrustWalker Sigmoid Similarity Measure [37], NHSM [33] and MPIP [38].

2.2. Planning systems

A planning system is software whose objective is to determine a plan consisting of a series of actions to achieve a goal from an initial state.

Every year, the International Planning Competition (IPC), organized by ICAPS, is held. IPC is a competition in which multiple teams present automatic planning algorithms and tools [40].

These problems are defined in the Planning Domain Definition Language (PDDL), created in 1998 for the 1999/2000 IPC competition [41]. This language is an attempt to standardize all existing planning problem modeling languages to date. PDDL is based on first order logic, containing objects and predicates that can be used to define states and actions that move between states [42]. The use of PDDL has made research in the field of automatic planning simpler, more accessible and comparable [44].

The idea that good algorithms were necessarily domain-specific was disproved in the 1998 IPC competition [45]. This led to a huge increase in research on domain-independent planning. Moreover, it is this research in automatic scheduling that continues to dominate, as it is very difficult to find a scheduler that works in all types of cases [46].

It is on such problems that classical planning focuses. Most of classical planners are focused on forward chaining heuristic search, using domain-independent heuristics [45]. Classical planners often use the GBFS (Greedy Best First Search), similar to A*, but can also use BFS (Breadth-First Search), DFS (Depth-First Search) A* and DFBB (Depth-First Branch and Bound) [45].

In contrast, the second approach, backward chaining, starts from the final goals and works backwards, looking for actions that can be applied to achieve the desired goal [45].

Additionally, heuristic functions are often part of the implementations of the previous branches to improve the search for results [45].

Besides classical planning, there is also the research field of probabilistic planning. These approaches are mainly based on optimization methods with dynamic programming and Markov decision processes, and are used in non-deterministic planning problems [45].

Recently, the possibility of using genetic algorithms in the planning task, metaheuristics inspired by natural selection processes, has emerged. They are commonly used in optimization and search problems, such as automatic planning. One example is AgPlan [47], a genetic algorithm-based domain-independent planner that uses the STRIPS language, a language similar to PDDL.

2.3. Academic and commercial projects

2.3.1. Academic projects

SAMAP [48] is a system with the objective of helping people visit cities. It is composed of four elements, an ontology; a user agent, i.e., an interface that allows the user to interact with the system; a recommendation agent, based on Case-Based Reasoning (CBR) and a planning agent.

In the ontology, information about the places and the user is collected and modeled. This includes information about cities, personal data, places visited and preferences. On the other hand, the CBR method makes use of the k-NN algorithm. This method consists of listing the k most similar users to the tourist T, based on the places visited and preferences, who have visited the city that T wants to visit. Subsequently, a list of possible places to visit that have been visited by at least one of the k users is generated.

Once the possible sites are listed, the planning agent uses these as input, in addition to the user's preferences, the schedules of each place and the locations. From this, the planning system creates an itinerary taking into account the time constraints of each location and the distances between locations. In addition, as the user may not be able to visit all the places in the list generated by the recommending agent, the planning agent takes into account the user's preferences to show the most relevant places for him. This planning is translated into a PDDL format problem that can then be solved by a planner that supports this format.

PersonalTour [49] is a travel package recommender system with the objective of creating customized travel packages that can be offered by travel agencies.

This system is based on agents, who can interact with each other. The goal of each agent is specializing in specific service recommendations, such as flights, hotels, transportation, etc... To do this, each agent has a confidence level for each service, and the agents that have the highest confidence level for a service are the agents that will be in charge of recommending items for this service.

These confidence levels are calculated using user scores. Each agent recommends certain items that have to be scored by the user and depending on these responses, the agents increase or decrease their confidence level for that service. In addition to the scores, it is taken into account when the scores have been made, prioritizing the most recent ones.

A series of tests were conducted to determine the ideal number of agents the system should have. Subsequently, the system was deployed in a travel agency for a period of 3 months and accuracy data was collected for three services: flights, hotels and places to visit. It was concluded that PersonalTour's recommendations were better than those of human experts, although with very little difference.

GuideMe [12] is a point-of-interest recommender system, focused on social interaction and mobile devices. GuideMe is a service that provides the user with guides of places to visit and recommends, based on the visits, more places to visit. The system is composed of a mobile application and a web application, a REST API, a recommender system and a database. Additionally, for the mobile application, there is a service that allows sending notifications to mobile devices.

The recommender system is implemented using the Apache Mahout library and makes use of the Slope One algorithm, a collaborative filtering algorithm that is executed every day at 3:00 AM. This service obtains a list of users who are eligible to show recommendations and sends notifications in case they are on this list.

PersTour [50], is an itinerary-generating system whose recommendations and planning are based on user preferences.

The system is composed of three components: a data collection and analysis component, an itinerary recommendation component, and a user interface.

The data collection and analysis component is responsible for obtaining photographs and metadata from several platforms, and processing this data to extract information, such as popularity or visit duration.

The itinerary recommendation component is responsible for creating itineraries based on user preferences and POI constraints such as timetables. To generate the itineraries, it makes use of the ant colony-based optimization algorithm, a type of evolutionary

algorithm. This interface is responsible for obtaining user data, such as preference levels for each category, schedules, tour duration, etc.; interacting with the rest of the components and displaying the results of the recommendations in the form of a list and a map.

2.3.2. Commercial projects

Existing commercial systems can be classified into several categories:

1. **Manual/Collaborative:** These are itineraries created by individuals, who either plan visits to their clients based on their preferences or publish routes and recommendations publicly. Although the expertise of individuals is very important, these services are often not cheap, as they require the labor of the people who carry out these offers. This category can be divided into two subcategories:
 1. **Travel platforms:** these are social platforms where travelers can create recommendations of places and even plans and share them with other users. An example of these is *Triptipedia* (<https://www.triptipedia.com>), *Tripspi* (<https://www.tripspi.com>), and *Civitatis* (<https://www.civitatis.com/>).
 2. **Personalized services:** These are services carried out by individuals, in which they are responsible for planning individualized itineraries based on a series of data provided by their clients. An example of these services can be found in *Cualquier Destino* (<https://cualquierdestino.es/travel-planner/>) or *The Fearless Foreigner* (<https://www.thefearlessforeigner.com/personal-travel-planner/>).
2. **By software:** These are systems or applications that allow the creation of automated plans and/or offer recommendations of places to visit. They often use services such as Foursquare and Google Maps, from which they extract the points of interest, routes and maps necessary to carry out the task.

This category can be divided into three subcategories:

1. **Manual planners** are platforms that allow a user to create a digital itinerary manually. These planners do not include recommendation systems, although they are often focused on collaborative creation. An example is *Pebblar* (<https://pebblar.com>) and *Marco* (<https://marco.app>), a website and an application, respectively, that allow the creation of personalized and collaborative digital itineraries.
2. **Manual planners with recommendations** are similar to manual planners, with the difference that they offer recommendations of possible places to add to itineraries. An example of this is *Tour* (<https://tourapp.co>), a mobile application that allows the simple creation of itineraries and whose

recommendations are extracted and generated in real time by the Foursquare service.

3. **Automatic planners** are systems that create an itinerary automatically, based on a series of preferences established prior to planning. An example of these platforms is *Triphobo* (<https://tripphobo.com>), a planner that generates itineraries based on a series of data, such as the age of the travelers. In addition to *Roadtrippers* (<https://roadtrippers.com/>) is a platform focused mainly on tourists who enjoy the road. It allows to generate an itinerary between two or more cities chosen by the user, and shows recommendations of possible points of interest to visit during the trip, as well as step-by-step navigation during the trip.

2.4. Technologies

2.4.1. User Interfaces and frontend

User interfaces one of the most important parts of any platform, since it allows a user to interact with the service, and it is usually the module that takes the longest development time in any project.

In the case of web-based platforms, HTML, CSS and JavaScript languages are the technologies interpreted, although with slight differences, by all existing browsers. Although the web was initially designed in a static way, the great growth of the Internet has led to the need for it to become dynamic [51].

Therefore, two techniques have emerged: Server Side scripting, which delegates all layout and HTML generation tasks to the server, so that all users receive an individual and dynamic response [52]; and Client Side scripting, which delegates all tasks to the client itself, which are performed by scripts written in the JavaScript language [52]. The latter technique is coupled with the use of Application Programming Interfaces (or APIs), to obtain information and interact with the server dynamically.

Along with these techniques, there are also different ways of implementing web interfaces. The most basic form is the so-called Vanilla JS or Vanilla CSS, which is simply a name for the use of these languages in "raw" form, i.e., without making use of libraries or frameworks [55], [56]. Along this, several frameworks, such as Bootstrap, exist with the goal of ensuring visual consistency between pages of the same platform and to be focused on the mobile experience [59].

Often, it also appears the MVC (Model-View-Controller) a software architecture pattern that separates the user interface, interaction and data model [60]. This architecture is presented in the Server Side scripting technique, and is implemented in many full-stack frameworks, which focus on the development of server logic and interfaces in parallel.

Although MVC has usually been used in server-side scripting, in 2010 Google introduced AngularJS [64], which makes use of TypeScript, a superset language of JavaScript, bringing this architecture pattern to the client.

In addition to AngularJS, there are also other frameworks, such as React from Facebook [66] and Vue.js [67].

2.4.2. Backend

HTTP (Hypertext Transfer Protocol) is the protocol that drives the Internet. Developed by the World Wide Web Consortium and the Internet Engineering Task Force, HTTP is published in version 1.1 in 1999 and goes on to define the syntax and semantics used by the web architecture [68]. Therefore, for the development of all types of web platforms, the use of this protocol is required.

As mentioned above, it is often necessary to implement APIs so that a client can interact with a server to store and manage data related to a user of the platform.

For the development of these APIs, there are different architecture styles, such as SOAP and REST:

- **SOAP (Simple Object Access Protocol)** is a stateful paradigm, oriented to services that allow to operate with data, and that defines the way to send and receive information between a client and a server, whose structure is formed by a packet, inside which contains headers and a body [69].
- **REST (Representational State Transfer)** is a stateless, data-centric paradigm that defines how information is sent between a client and a server. It is based on the idea of a stateless protocol, where each message contains the information necessary for its understanding, the use of addressable resources through a global identifier and the definition of a set of operations, such as GET, POST, PUT and DELETE operations defined by HTTP [71]. It is currently the most widely used paradigm, thanks to its broad support for a wide variety of data formats, its greater speed and the possibility of using the caching technique [70].

However, it is also necessary to use a language to implement the server logic. Although the general advice is to choose the language that the platform developers are most comfortable with, there are a number of languages that are most commonly used for this type of task, such as PHP, Java, Node.js or Python [72].

2.4.3. Libraries and additional projects

The use of libraries in projects is becoming more and more common, especially when projects are increasingly complex, so it does not make sense to rewrite code that other teams have developed more efficiently.

Regarding recommendations, there are two well-known machine learning libraries, Apache Mahout and Tensorflow. Apache Mahout, developed in Java, is a framework for building algorithms using linear algebra [73]. This library offers functions that can be used for the creation of machine learning tools for regression, clustering or recommendations.

On the other hand, Tensorflow is an open source library, developed by Google and implemented in C++ with Python interfaces, for the implementation of machine learning tools [74]. Although the Python interface is the most widely used, an interface for JavaScript and Node.js is also provided [75].

With respect to planning, there are several programs that use the PDDL language:

- **FF (Fast Forward)** is a forward chaining heuristic scheduler [76], whose principle is based on obtaining a heuristic estimate and relaxing the task P in question into a simpler task P+. Its implementation, Metric-FF, is a scheduling system developed by Joerg Hoffmann [77].
- **FD (Fast Downward)** is a planning system based on heuristic search [78]. It is a progression planner, which searches the state space of the world of a planning task in a forward direction. Unlike other planning systems, Fast Downward does not directly use the PDDL representation of a planning task, but the input is first translated into an alternative representation.
- **CBP (Cost-Based Planner)** is a PDDL planner that performs a heuristic search in the state space, using several heuristics [79]. On the one hand, it uses relaxed plan-based anticipation states, similar to Metric-FF, to speed up the search. On the other hand, the search is also guided using a numerical heuristic and a selection of actions extracted from a relaxed planning graph. The relaxed planning graph is constructed taking into account the costs of the actions.
- **LPG (Local search for Planning Graphs)** is a planner based on local search and planning graphs [80], [81]. The search space of LPG consists of "action graphs", subgraphs of the planning graph representing partial plans. Search steps are certain modifications of the network that transform one action network into another.

Finally, a recent tool is WASM or WebAssembly, a format of binary instructions that are executed in a virtual machine [82], usually embedded in browsers. Being binary

instructions, these can be executed quickly and perform computations with a speed similar to that of a compiled language [83] that would be very expensive to perform in interpreted languages such as JavaScript.

2.4.4. Databases

In the field of databases, there are two important models, relational and NoSQL databases.

Relational databases are currently the most widely used model, which leads to the development of Relational Database Management Systems. The most popular current managers are: Microsoft SQL Server [86], Oracle SQL [87], PostgreSQL [88], MySQL [89] and MariaDB , a fork, of MySQL created by the original MySQL developers due to the unknowns arising from the acquisition by Oracle [90].

On the other hand, NoSQL databases were created to improve the disadvantages of relational databases, such as the difficulty in handling blocks of text or the deficiencies in handling multimedia data or large amounts of data [91]. Among their implementations are MongoDB, an open source, document-oriented database system [92]; Redis, an open source, in-memory, key-value database system [93], often used as a distributed cache; and Apache Cassandra, a distributed database that is capable of handling large data capacity [94].

2.4.5. Security

In the case of transport data, security is provided by the HTTPS protocol, an extension of the HTTP protocol that implements TLS, so that the data being transmitted is encrypted, protecting its integrity and privacy [95].

TLS makes use of public and private keys [96], however, to ensure the security of connections, the use of self-generated keys is not recommended. This is why certification authorities, or CAs, are used. CAs are responsible for generating digital certificates, signed by them, to their clients, which they can use to encrypt communications between clients and servers [97]. However, as issuing these certificates is often very expensive and tedious [98]–[100] in 2016, Let's Encrypt, a certificate authority that provides free certificates for TLS encryption through an automated process, is launched [101].

Additionally, RFC 6797 defines HSTS (HTTP Strict Transport Security) [102], a policy, defined by servers, that states that all interactions with the server must be done via HTTPS and never via HTTP. This prevents man-in-the-middle attacks, cookie hijacking and downgrade attacks.

Along transport level security, browsers implement several techniques to improve security, such as CORS, a mechanism to restrict access to certain resources by a web page [103], via HTTP headers; and CSP, a mechanism to limit the execution of resources in the browser [104], defined also in HTTP headers.

At the server level, it is important to take into account the security of data at rest, so that these databases cannot be leaked or their leaks are limited in scope. This task is usually carried out by system administrators, who are responsible for managing the systems that power the platforms.

In the area of passwords, users often use the same passwords for almost all the services to which they are registered [107], so a leak in one service can have consequences for the accounts of the same users in other services. Therefore, it is established that storing passwords in plain text is malpractice. To solve this, several techniques are presented to store passwords securely in a database:

- **Hashing** consists of storing the result of computing the hash of a password in database [108]. Although in this way, passwords are not understandable, this form of storage can be attacked by means of rainbow tables, which consist of tables with the most common passwords and their corresponding hashes.
- **Hashing + salting** consists of generating a random string of characters, concatenating it with the password, and storing the result of computing the hash of this concatenation together with the randomly generated string of characters [108]. In this way, rainbow table attacks are prevented, since a known password will have a different hash result each time.
- Another way to store passwords is by using the **bcrypt** algorithm. In addition to incorporating salting to protect against rainbow table attacks, bcrypt is an adaptive function, as over time, the iteration count can be increased to make it slower, maintaining its resistance to brute force attacks even with increasing computational powers [109].

2.4.6. Deployments

Scalability is the ability of a system to adapt and react, changing its resources, so that it does not lose or degrade in quality. Scalability can take two different forms, vertical and horizontal [110]. Vertical scalability consists of adding more resources to an existing system, either by increasing computational capacity or memory capacity, while horizontal scalability consists of adding more systems, distributing the workload among them.

During the deployment process, the systems on which the platform will run must be taken into account. There are two possible variants, on-premise hosting [111], where all

the software runs on machines that the company has in its own offices; and cloud hosting [111], where all the software runs on servers in data centers, which provides greater flexibility and scalability.

On the other hand, there are different places to run the software. The first is bare metal, which consists of running programs without software abstractions. In this way, if the software needs to interact directly with the hardware, it can do so easily [114].

Virtual machines are presented as an alternative [115]. With virtual machines, such as those found in cloud services, system administrators do not have to worry about hardware or operating system maintenance, as this is done by the cloud services themselves [112], [113]. Another advantage is that resources are better utilized, as multiple virtual machines can be run on the same physical machine [115].

Finally, containers have recently emerged as an alternative to running software. Containers re-abstract software execution, virtualizing and isolating the software between the containers and the host system. Although they present the same slight performance losses as virtual machines, containers allow multiple instances of a program to run in a reproducible manner, eliminating the possibility of machine-specific errors [116].

Among the best-known container management programs are LXC, released in 2008 and part of the Linux kernel [117], and Docker, released in 2013 [118]. Another feature of containers is their high horizontal scalability [119], [120]. That is the goal of Kubernetes, an open source container orchestration system for automating container deployment, scaling and management [121], and Docker Swarm, a utility for grouping multiple Docker engines running on different machines under a single "virtual" Docker engine, enabling automation of container deployment and scaling [122].

3. SYSTEM DESCRIPTION

As mentioned above, the objective of this project is to develop a platform with a system that platform with a system that recommends and plans tourist itineraries to its users. It will be called Hermes, in honor of the Greek god messenger, of borders and travelers.

3.1. Operational environment

First, it has been implemented using Node.js and Rust for the server side, followed by HTML, CSS and JavaScript along with the Bootstrap framework for the user interface module.

During the development a version control has been carried out using the *git* tool. The repository where the project code is stored is located on the GitHub platform in public form (<https://github.com/SrGMC/hermes>). Likewise, the project is organized in the following directories:

- **data:** This directory contains the scripts needed to load the data into the database. In addition, it contains the different database files with the locations, separated by cities, and which facilitate the collaboration of any person.
- **docs:** This directory contains the repository documentation.
- **scrapers:** This directory contains the scripts that have been created for data extraction, described in section 4.3.
- **server:** This directory contains the code of the backend implementation - web: This directory contains the web interface of the platform.

The machine on which the development has been performed has the following characteristics and the following programs:

- Lenovo Thinkpad with AMD Ryzen 7 processor, 64 bits, 4 cores and 8 threads, 2.3 GHz.
- 16GB of RAM DDR4 2400MHz.
- 512GB M.2 NVMe disk.
- Fedora 34 operating system.
- LibreOffice 7.2.

On the other hand, two machines will be used for the deployment. This will be done using containers and the Docker tool. For the distribution tasks, the Nginx software is used, which is downstream of Cloudflare. The characteristics of both machines are:

- Odroid HC1 with ARM v7 processor, 32 bits, 8 cores, 2.1GHz.
- 2GB RAM.
- 128GB SATA 3 SSD disk.
- 10Mbps Ethernet.
- Diet-Pi operating system (based on Debian 10).

3.2. Data collection and processing

To generate the database content, the Foursquare public API has been used. First, the API has been explored to identify accessible routes. It should be noted that this API only allows a maximum of 99,500 requests and 500 premium requests [125].

Once the API routes to be used were recognized, a list of the most important place categories was created, where only categories corresponding to points of interest to visit were chosen, such as Museums, Beaches or Squares, excluding commercial places such as Shopping Malls or Stores. Each of the categories has been assigned a numerical identifier, as seen in the Table 3.1:

TABLE 3.1: CATEGORIES

ID	Name	Foursquare ID
1	Concert halls	5032792091d4c4b30a586d5c
2	Mosques	4bf58dd8d48988d138941735
3	Botanic gardens	52e81612bcfc57f1066b7a22
4	Outdoor sculptures	52e81612bcfc57f1066b79ed
5	Pedestrian squares	52e81612bcfc57f1066b7a25
6	Opera houses	4bf58dd8d48988d136941735
7	Caves	56aa371be4b08b9a8d573511
8	Temples	4bf58dd8d48988d13a941735
9	Gardens	4bf58dd8d48988d15a941735
10	Zoos	4bf58dd8d48988d17b941735
11	Markets	4bf58dd8d48988d1fa941735
12	Public art	507c8c4091d498d9fc8c67a9
13	Commemorative places	5642206c498e4bfca532186c
14	Art museums	4bf58dd8d48988d18f941735
15	Viewpoints	4bf58dd8d48988d165941735
16	Lakes	4bf58dd8d48988d161941735
17	Stadiums	4bf58dd8d48988d184941735
18	Street art	52e81612bcfc57f1066b79ee
19	Elevated viewpoints	4bf58dd8d48988d133951735
20	Planetariums	4bf58dd8d48988d192941735
21	Cemeteries	4bf58dd8d48988d15c941735

ID	Name	Foursquare ID
22	Town halls	4bf58dd8d48988d129941735
23	Art galleries	4bf58dd8d48988d1e2931735
24	Lighthouses	4bf58dd8d48988d15d941735
25	Monuments	4bf58dd8d48988d12d941735
26	Synagogues	4bf58dd8d48988d139941735
27	Capitols	4bf58dd8d48988d12a941735
28	Castles	50aaa49e4b90af0d42d5de11
29	Auditoriums	4bf58dd8d48988d173941735
30	Monasteries	52e81612bc57f1066b7a40
31	Sculpture gardens	4bf58dd8d48988d166941735
32	Budhist temples	52e81612bc57f1066b7a3e
34	Shrines	4eb1d80a4b900d56c88a45ff
35	Zoo exhibitions	58daa1558bbb0b01f18ec1fd
36	Science museums	4bf58dd8d48988d191941735
37	Public squares	4bf58dd8d48988d164941735
38	Palaces	52e81612bc57f1066b7a14
39	Aquariums	4fceea171983d5d06c3e9823
40	Fountains	56aa371be4b08b9a8d573547
41	Churches and cathedrals	4bf58dd8d48988d132941735
42	Town councils	52e81612bc57f1066b7a38
43	History museums	4bf58dd8d48988d190941735
44	Historic places	4defb944765f83613cdba6e
45	Bays	56aa371be4b08b9a8d573544
47	Sikh temples	5bae9231bedf3950379f89c9
48	Bridges	4bf58dd8d48988d1df941735
49	Parks	4bf58dd8d48988d163941735
50	Amphitheaters	56aa371be4b08b9a8d5734db
51	Theaters	4bf58dd8d48988d137941735

Subsequently, tourist sites such as Civitatis or Tripspi have been visited to extract areas of interest from which to extract places. These zones have been established due to the limitations of the Tripadvisor API. To extract zones of interest, the most popular places of interest of these platforms have been placed on a map, so that clusters or zones of points of interest can be identified, extracting the zones defined in Table 3.2.

TABLE 3.2: ZONES

Zone	Coordinates	Search radius
Aranjuez	40.03986, -3.60213	1500m
Atocha	40.40875, -3.69194	1000m
Bomberos	40.39619, -3.65511	1000m
Casa de Campo	40.42329, -3.75875	3500m

Zone	Coordinates	Search radius
Centro	40.41149, -3.70757	1000m
Chamartín	40.47317, -3.68321	1500m
Cibeles	40.41933, -3.69308	1000m
Colón	40.42519, -3.6903	1000m
Complutense	40.44561, -3.72991	1000m
Delicias	40.39712, -3.6961	2000m
Goya	40.42807, -3.66893	1500m
Gregorio Marañon	40.43796, -3.69082	1500m
Madrid Río	40.40018, -3.72076	1000m
MUNCYT	40.53772, -3.64126	500m
Nuestra Señora de la Almudena	40.42019, -3.64231	1000m
Palacio Real	40.41755, -3.71438	1000m
Parque Oeste	40.42649, -3.72191	1000m
Retiro	40.41526, -3.68449	1000m
San Lorenzo del Escorial	40.58885, -4.14846	500m
Santiago Bernabeu	40.45224, -3.69073	1500m
Sol	40.41641, -3.70379	1000m
Tribunal	40.42577, -3.70083	1000m
Wanda Metropolitano	40.43619, -3.59946	500m

After identifying the zones, a JavaScript script was created to make requests to the Foursquare API to extract data and interactively process the data obtained. In total, 1050 requests were made. These requests resulted in a total of 1223 points of interest, which were subsequently narrowed down to a total of 259 points of interest.

During the filtering process, points of interest that did not have a postal address or whose name contained the substrings " sports center" or "soccer field" were eliminated.

After this pre-filtering process, a semi-manual filtering was performed, partially automated through the use of Puppeteer, a library that automatically opened a web browser with the Foursquare URL of the place and the URL of the place in Google Maps, allowing the data of each place to be cross-checked manually.

When deciding whether the place should be added, the following set of rules were followed, where if one of the answers to any of these was negative, the place was rejected from being added to the database:

1. Does the place have an appropriate name?
2. Does the Foursquare place match the place on Google Maps?
3. Is it a place of tourist or historical interest?

If all answers were positive, the OpenStreetMap ID for the place, the URL of the place on Tripadvisor (if it existed), the Wikipedia page for the place (if it existed), the incorrect data were modified and a numerical ID was assigned to each place.

Once all data were completed, several data extraction scripts were run:

1. **Extraction of additional data from OpenStreetMap.**
2. **Extraction of photographs from Wikipedia:** these photographs were used due to their Creative Commons license. All photographs were converted to JPG format and each photograph was organized into directories named according to the ID assigned to each location.
3. **Obtaining additional data from Foursquare.**
4. **Obtaining timetables from Foursquare.**

Once all the data was collected, it was consolidated into a single JSON file, which contains a list of objects where each object represents a place. In addition, missing descriptions of each place were corrected and added.

Finally, a series of scripts were developed to:

1. Migrate and/or initialize the database
2. Add the categories to the database
3. Add the places into the database, excluding already added places matching name, latitude, longitude and OpenStreetMap ID.
4. Add the pre-calculated distances between places in the database. This is done due to the high cost of the Google Maps route search service once a limit of requests is exceeded.

3.3. Modules

In contrast to the projects presented in the *Current State of the Art*, the aim is to give more control to the users of the platform when it comes to making decisions, therefore, the system is divided into several modules.

The HTML, CSS and JavaScript languages have been used together with Bootstrap, Bootstrap Icons, Font Awesome and Leaflet libraries for the user interface, so that the Caching technique can be used. For the backend, Node.js will be used with the JavaScript language, in addition to the Rust language to create custom libraries for cases where intensive calculations are required. In addition, the *fastify* library will be used for the REST API, due to its high performance [123], [124] and the *sequelize* library for the interaction between the database and the backend.

- Database:** The database contains all the information necessary for the operation of the platform. MariaDB is being used as the database software.
- Recommender system:** The recommender system is in charge of generating recommendations of places or points of interest for users. For this purpose, it uses a hybrid recommendation approach, where user-based collaborative filtering and content-based filtering is used.

User-based collaborative filtering is implemented using neighbors. When a recommendation needs to be generated, M users are selected randomly and its user similarity measure (in this case, MPIP [38]) is computed with the objective user. Then, the N most similar users are selected. Once the neighbors have been selected, places that have been visited by at least one of the user's neighbors and whose confidence level is higher than a threshold are recommended. This approach is used to overcome scalability problems.

Content-based filtering is implemented through the use of user preferences and user queries for each category. These queries are stored in the database each time a user creates a rating or one of the visited plans is finished.

The use of this hybrid approach allows the system to also overcome Shilling Attacks and diversity problems. The recommendation task is executed every day at 02:00 AM.

- Planner system:** The planner system is in charge of creating itineraries or plans. For this purpose, it makes use of the PDDL language and a classical planner. In addition, several translators are implemented to convert between the internal representation and planner inputs/outputs. The planning task is limited to 5 minutes.

Before planning a new itinerary, the user is asked to input information about it's trip, such as visit length, starting place, etc.... Finally, before creating the request to plan, a minimum day length is computed so that the planning system can find a route.

In the PDDL domain designed, a problem is solved when the solution meets the following goals:

- User is back at the starting point
- All places have been visited
- There is at least one visit each day

It should be noted that due to the cost, it is not possible to use route finding services such as Google Maps Directions API. Instead, during the translation of problems and solutions, use is made of the distances that are pre-calculated in the Distances table of the database. In case a distance is not found in this table (such as a distance between the starting point and a place), a function that

calculates the straight line distance between two geographical coordinates is used.

Additionally, if the travel time is longer than 60 min, the journey is changed from walking to driving.

4. **API/Backend:** The backend, is in charge of interacting with the database and the scheduler and recommender systems. In addition, it implements a REST API to interact with the user interface and an email service to send important notifications and reminders. An API definition [127] has been made following the OpenAPI specification [128].

This API is implemented using controllers. Along this, several logging and security systems are in place. Logging systems store application and request events. This allows us to audit the system to find and diagnose application bugs and errors and also make security forensic analysis.

On the other hand, the platform implements the following security procedures:

- Passwords are sent to the server hashed with SHA256. This prevents a MITM attacker to know the real password, that might also be shared with other services.
 - Passwords are stored by making use of the bcrypt algorithm. The number of rounds is determined by the sum of the numbers in the SHA256 hash, with a minimum of 10 rounds [133].
 - Authentication is done via a signed cookie named *hermesSession*.
 - Only HTTPS requests are allowed, rejecting HTTP requests. HTTPS certificates are generated by Let's encrypt.
 - Rate limiting is implemented in order to deter bots and protect user accounts from brute force login attempts.
 - Logins are saved along with IP addresses to let the user know when an unknown login has been made.
 - CORS and CSP is configured to prevent cross-site scripting and cross-browser attacks.
5. **User interface:** The user interface is the means by which users will interact with the platform. To maintain consistency and compatibility, the Leaflet, Bootstrap and Font Awesome libraries are self-hosted. The user interface is divided into the following components:
 1. **Landing page:** The landing page attracts users to the platform. It is the first page that is displayed to the user when he/she accesses the page. It features a description of the platform, alongside the logo, links of interest, and sign in and sign up buttons.

2. **Registration and sign in:** The registration and sign in component is responsible for creating new accounts and identifying the user on the platform. It implements security measures, using the hCaptcha service, to prevent the use of automated tools (or bots) and requests information from the user during registration to overcome the cold start problem.

Before being able to create an account, the user is asked to select at least three categories, organized by supercategories. These categories are those defined in the Table 3.1. The user is also asked to rate sites that he/she knows and has visited. Rating sites is not required.

3. **Password recovery:** This component is in charge of giving the user the possibility to recover his password in case he/she has forgotten it.
4. **Recommendations:** This component shows the recommendations to the user in an intuitive way and allows to search places. To do so, it uses a list of places, together with an interactive map. It also allow the user to select places to add to a plan. Unlike the platforms presented in the *Current State of the Art*, it has been decided to separate this part from the planner, as this gives more control to the user and offers him/her the possibility to discover the relevant places for him/her in the city to visit, improving his/her travel and user experience.

As Google Maps, Mapbox or MapTiler services are costly, it was decided to host an OpenTileServer instance to serve map tiles to users without the need for a third party service.

5. **Plans:** This component shows all the plans that the user has created, as well as showing the itineraries created in detail, allowing them to be modified and deleted.

Plans can have 5 different states:

- **Active**
- **Planning**
- **No solution found**
- **Completed**
- **Internal error**

In addition, when the user edits a plan, he/she can modify the name, description, start date (as long as the start date is after the current day), the duration of the items in the plan, add and delete items, and add notes to each item. Items can have the following types:

1. **Starting place:** It is the starting and ending place of each day.
2. **Place:** It is a place that exists in the database.

3. **Rest:** It is a period of rest, such as lunchtime.
4. **Wait:** It is a waiting period because the next place to visit has not yet opened.
5. **Custom place:** In a custom place that the user has created manually and that is not in the database.
6. **Ratings:** This component shows the user the ratings he/she has created, as well as allowing the user to create new ratings and manage existing ones.
Ratings are split in two sections: pending ratings and ratings already made. Pending ratings are places that appear in plans that are not yet rated by the user, while Ratings already made are ratings created by the user.
When the user wants to create a new rating, a modal popup appears, allowing the user to search places and rate the ones that better match the search criteria.
7. **Profile:** This component allows the user to view, modify, download and delete all their personal data, according to the created privacy policy; perform security management, such as changing passwords or logging out; and modify their email notification preferences.
8. **Help:** This component contains help documentation about the platform as well as contact links.

The implemented architecture is shown in Figure 3.1.

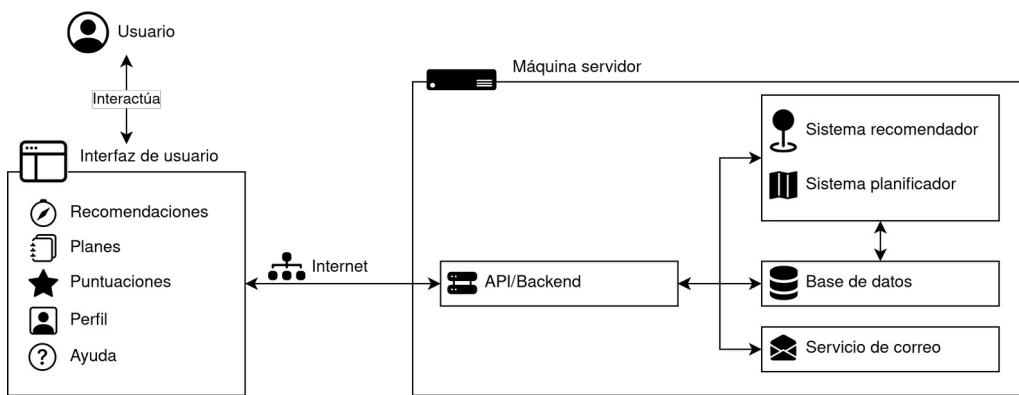


FIGURE 3.1: HERMES ARCHITECTURE

3.4. Database model

The schema of the relational model of the database is shown in Figure 3.2. In addition, all tables contain, internally, the following attributes:

- `createdAt`: Indicates the creation date.
- `updatedAt`: Indicates the date of update.

- deletedAt: Indicates the deletion date. Its value is NULL if it has not been deleted. This allows keeping the deleted tuples in the database in case they need to be recovered (for example, if a user accidentally deletes his account).

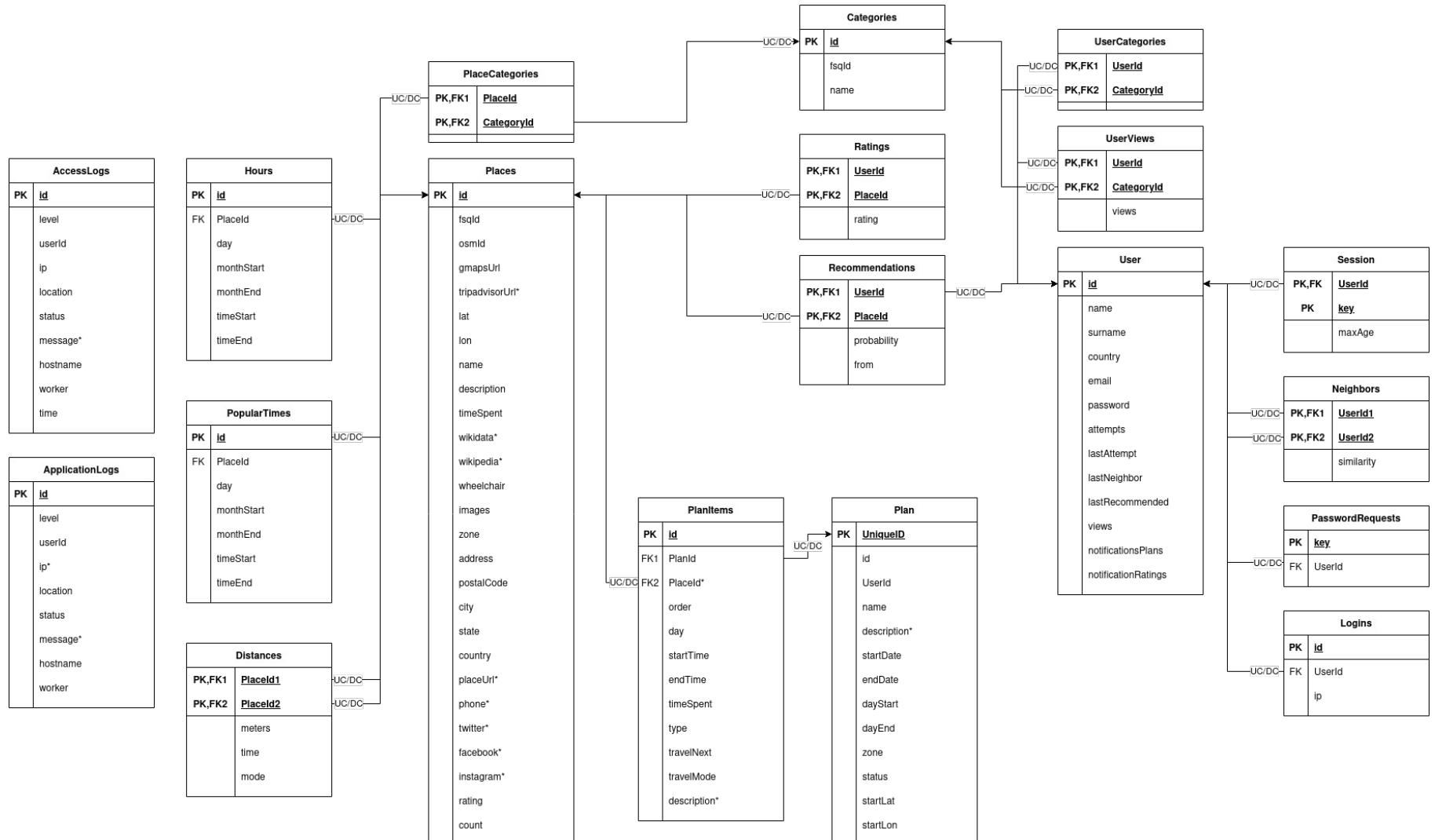


FIGURE 3.2: HERMES' DATABASE RELATIONAL SCHEMA

3.5. Deployment

Hermes is publicly available to any user under the domain <https://travelhermes.com>.

For each new version of the platform, a deployment to the production environment is performed. During the first deployment, a *Dockerfile* was created, which defines how the container image should be built. This file installs the necessary libraries, compiles the Rust code and copies the server and web interface code into the container image. Also, using Docker allows the platform to scale easily according to the needs of the service. Next, a file of environment variables is created that define:

- The domain where the platform is located.
- HTTP port of the server.
- Server name.
- Database connection string.
- The secret used to encrypt the session cookies.
- hCaptcha service keys.
- Email account keys.

Once the image and its variables have been created and configured, we proceed to the deployment in the production environment. First, analytic scripts are added to the web interface provided by Umami, an analytic service that does not use cookies and respects privacy. The deployment is completed by starting the platform containers, creating or migrating the database if necessary and configuring Nginx for its load balancing tasks. In addition, the Cloudflare cache is flushed so that the files in the CDN are updated.

On the other hand, for deployments and to be more transparent with users, a status page (<https://status.travelhermes.com>) has been created, which makes use of the cState project, allowing to know at all times the status of the platform. This status page shows scheduled maintenance and incidents that are occurring and have occurred along with additional information.

The system has been implemented in a way that takes advantage of horizontal scalability and is distributed, following the network structure of the Figure 3.3. Use has been made of the CloudFlare platform as CDN and Firewall [131], Nginx software as Load Balancer and Proxy and Docker containers for easy scalability and distribution.

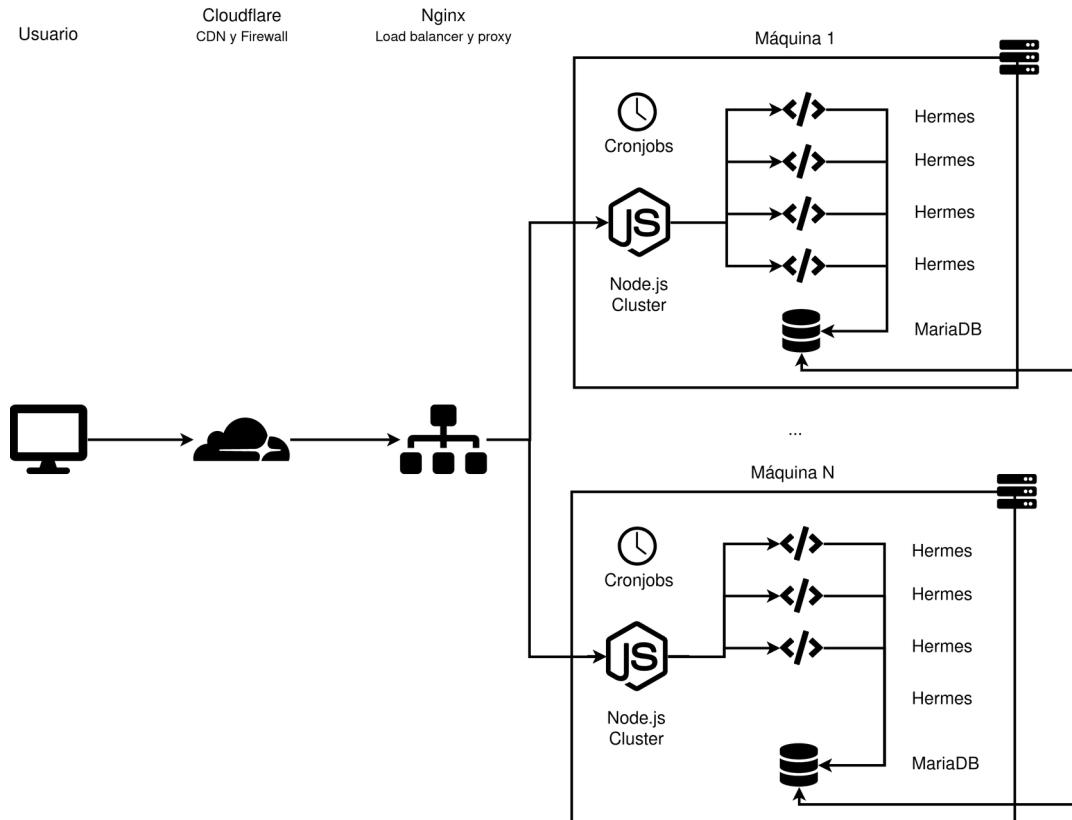


FIGURE 3.3: HERMES' DISTRIBUTED ARCHITECTURE

3.6. Contributing

As the objective of this platform is the creation of an Open Source project (<https://github.com/SrGMC/hermes>), it is open to improvements, suggestions and contributions from other users. For this purpose, contributing rules [137] have been established for new developers. Also, versioning is done following Semantic Versioning [136].

Templates have been established to differentiate the types of suggestions that users may have. All templates are in English and Spanish. These include bug report templates, feature requests and place/city requests.

If anyone wishes to contribute a new place or fix an existing place, there are a series of steps to perform, including creating Pull Requests and adding JSON objects in the required files.

Finally, if anyone wants to contribute new code or enhancements, a format is established that must be adhered to [137]. This format can be met using the Prettier tool with a few setting changes and all the code must be commented and the JSDoc format [138] must be used.

4. EXPERIMENTATION AND TESTING

Testing has been divided in four parts:

1. **User Interface testing:** To test the user interface, evaluations with users have been performed.
2. **API unit testing:** Several unit test have been made to test every API endpoint and controller.
3. **Recommender system testing:** Recommender system testing has been performed using the 1m database fro MovieLens [139].
4. **Planning system testing:** Several tests with different PDDL problem files have been made to test the efficiency of three different classical planners.

4.1. User Interfaces testing

This section describes the evaluation process carried out with users. For this purpose, the following methodology has been established:

1. **Objectives:** A series of objectives to be obtained with the evaluation are established. These are:
 1. **Usability and ease:** It is desired to check if the visual design is simple and helps the user to understand what happens and when and invites him to interact in a simple way with the interface.
 2. **Performance analysis:** We want to check how the system behaves with different users and if it is efficient and personalized.
2. What do we want to know: Based on the previous objectives, we establish a series of parameters that we want to answer:
 1. **Chokepoints:** We want to find out where the user gets stuck and which parts or elements are more difficult to use or ambiguous.
 2. **Accuracy of recommendations:** You want to find out if the user's recommendations are what he/she expected and if they are adapted to his/her profile.
 3. **Quality of the planning:** We want to check if the generated plans are as expected and if they are efficient.
 4. **Ease of use:** We want to know if the interface is easy to use, if it helps the user to understand what happens and when, and if it invites the user to interact with it in a simple way.

3. **Practical issues:** Once the previous points have been established, we proceed to the evaluation with the users. In this section, a series of questions are asked and the user is put in the context of the task to be performed. These questions range from age and gender to experience with technology and how much they travel.

The context is as follows: "You, the user, are a foreign tourist who has decided that you want to visit Madrid on your next trip. Although you know nothing about the city, you have found a platform that allows you to create personalized plans with the places to visit around this city based on your personal preferences."

4. **Final questions:** After finishing the interaction with the user, some final questions were asked, which aim to get insights about the UX, and efficiency of the different platform modules.

After performing the tests, the following conclusions were drawn:

1. **Chokepoints:** Several points were found where users got stuck. These appear in the top navigation bar and in some cases in the planning interface, present in the recommendations interface.
2. **Accuracy of recommendations:** Recommendations have been accurate, although for some users they have been low. This may be due to the limited response of preferences or the small number of scores created by these users.
3. **Quality of planning:** The plans generated are generally efficient, although there were often cases of locations in close proximity to each other, where the planner has not planned in such a way as to visit these one after the other.
4. **Ease of use:** In general, users liked the interface and found it simple, although they often found ambiguous or unclear elements that made it difficult to interpret the actions of each element.

Therefore, based on these conclusions, a list of changes was generated, some of them already implemented.

4.2. API unit testing

Unit tests have been performed for the implemented REST API, in order to verify that the API works according to the established specifications.

This unit tests are made for each API controller, and tests whether the API responds accordingly to valid and invalid requests. 198 different tests were made, where every test passed successfully.

4.3. Recommender system testing

This section describes the tests performed for the recommender system. These tests are aimed at observing the accuracy of the implemented recommender system. For this purpose, use has been made of the 1m MovieLens database [139], which contains more than one million movie ratings, with 3900 movies and 6040 users. This database allows to check the similarities between the platform's recommendation confidence level and the actual ratings. As the movies are categorized, similar to the POIs, and the scores have a range from 1 to 5, no significant changes to the database have been necessary.

For these tests, 3 parameters have been set, N, which is the number of neighbors generated; M, an integer that multiplied by N generates the number of random users obtained; and D, which simulates the number of days the recommender has been running. Due to the large computational time involved in generating recommendations for 6040 users, as seen in Table 4.3, these tests have been performed on a virtual machine offered by Linode.

For the analysis, the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) have been used. By means of the Mean Squared Error, the difference between the predicted and observed values can be estimated on the same scale as the observed value. By means of the Mean Absolute Error, the squares of the deviations between the predicted and observed values can be estimated.

After testing, it has been observed that the mean difference between the predicted and observed values is 1.39 points on the scale of 1 to 5, a 28% difference.

In addition, several events are observed. First, the more neighbors used, the more accurate the recommendations are. Finally, high values of M, i.e. random users used to search for neighbors, also generate accurate recommendations.

4.4. Planning system testing

This section describes the tests performed for the planning system. The purpose of these tests is to compare different planning programs, Metric-FF, CBP and LPG, with the designed domain. Testing problems are the following

1. **base_1_day.pddl:** Planning problem to generate a single day itinerary, visiting 3 places.
2. **base_m1_day.pddl:** Planning problem to generate an itinerary of more than one day, visiting 3 places.
3. **big.pddl:** Planning problem to generate an itinerary of one week, visiting more than 50 places.

4. **restrictive.pddl**: Planning problem to generate an itinerary of three days, visiting 4 places. The days have a very restrictive schedule, where only 4 hours are available each day to visit places.

Moreover, in addition to setting a maximum execution time of 5 minutes, different execution parameters have been set for each planner.

From the results obtained, the following can be observed. First of all, big.pddl is a very large problem, whose solution is difficult to obtain, especially with such limited time. That is why Metric-FF with its default configuration has been the only run that has managed to obtain a solution, emphasizing that this solution is not the optimal one. Secondly, LPG offers very good performance, finding the best solution in the established time, in the rest of the problems.

Finally, it can be concluded that the use of LPG or CBP can be beneficial for the project, due to its efficiencies compared to the current implementation with Metric-FF. However, the use of Metric-FF with its EHC+H configuration with A*epsilon should be maintained as an alternative for cases where either of the two planners fails to find a solution in the given time.

5. REGULATORY FRAMEWORK

5.1. Privacy

Like with security, it is very important to preserve the privacy of users. In this project, privacy has been taken into account when developing the project, paying special attention to respect it at all times.

Currently, two laws are in force, the *Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales* and the European General Data Protection Regulation.

The *Ley Orgánica 3/2018, del 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales* [140], with entry into effect as of December 07, 2018, establishes that every citizen has the right to "access, rectification, erasure, opposition, right to limitation of processing and right to portability" [140] of his or her personal data. In addition, it establishes that there must be a data controller and data processor and regulates infringements for non-compliance with this Law.

On the other hand, the Directive (EU) 2016/679 [141], known as the European General Data Protection Regulation, entered into force on May 24, 2016, establishes a series of digital rights for all European Union citizens, similar to those of *Ley Orgánica 3/2018*. In addition, it establishes that all foreign companies that own European users must comply with this regulation, establishes regulations governing the occurrences of such as data breaches and data filtering, and establishes new penalties for non-compliance with the Law.

As a personalized platform, users' personal data, preferences, ratings and actions within it can be used to display advertisements, sell their data to third party companies and learn a lot about a person.

In order to be transparent, a privacy policy [142] has been created, available on the project page. This privacy policy:

- Complies with the regulations of the *Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales* and the European General Data Protection Regulation.
- Only personal and identifying data strictly necessary for the operation, development and improvement of the platform are collected.
- Device data is only used for analysis and improvement of the platform and is not related to any user.

- No information is shared, sent or sold to third parties.
- No information is shared with advertising companies.
- No information is mined for personal and behavioral trends.
- No information is monetized.
- User data is the users' property and he/she can freely delete, manage and download its data at any time.

5.2. Intellectual Property Law and licenses

Intellectual property [143] is a set of rights attributed to authors and right holders for their works. These rights allow their authors and owners to exploit their works, to be recognized for them, and to receive financial compensation for their use by other people or companies. Also, intellectual property laws protect, in a legal way, the intellectual property of the works.

Therefore, in this work all copyrights of all materials used have been respected. Also, one of the objectives set for this project has been the primary use of Open Source or free license software and material. Thanks to these licenses, anyone can modify and use the software or material freely, as long as they comply with the restrictions imposed in these licenses [144].

In developing this project, special care has been taken to respect licenses and copyrights, and to use as much free software and material as possible.

Finally, it should be noted that the project, following the spirit of most of the licenses used, also comes with a free software license. In this case, the GPLv3.0 license was chosen. This license only restricts that the corresponding authorship is mentioned and that all changes, whether contributions to the project or commercially, are published freely. Likewise, all contributions from third parties will be licensed under the same license.

6. SOCIO-ECONOMICAL FRAMEWORK

Artificial intelligence is present in our lives and has uses in many areas, from video recommendations on YouTube, to route directions with Google Maps, to personalized timelines on Twitter, Facebook and Instagram.

However, although these platforms bring advances highly beneficial to people and the environment, and usually are offered free of charge, their economic benefit usually comes from the collection of large amounts of data, which are also necessary for the use of the platform. This data collection also presents its privacy and security problems, as seen in events such as Facebook and Cambridge Analytica, used to influence political opinion [149], [150].

On the other hand, due to the pandemic situation that occurred during the development of this project, the tourism industry has been one of the sectors that has suffered the most downfalls due to the great health risk involved in transportation and accommodation of people [151]. However, after a year of the event and with the emergence of vaccines, the tourism industry is on a growth trajectory, especially with domestic tourism [152], [153].

Often, many places, cities and countries go unnoticed, being overshadowed by others that are more well-known and popular. Although there are numerous lists and publications of "recommended little known places to visit" [154], [155] these usually remain under-visited.

In addition, during experimentation and validation with users, it has been found that users can discover new places to visit that they did not know before and that the platform has found to be of interest to them.

Specifically, with this project, we want to offer a platform that demonstrates that it is possible to offer personalized content to users, respecting and protecting their privacy. In addition, a positive economic and social impact can be created, which can help the economic recovery of the areas most affected by the events of 2020 and 2021.

Finally, as long as the content of the project remains unbiased, it can help promote previously hidden or little visited points of interest and cities, encouraging users to visit them in a way that can have a positive effect on the local or national economy and society. In addition, this effect of promoting lesser-known places can also help to reduce overtourism [156].

7. CONCLUSIONS

In this work an open source platform, Hermes, was developed, which recommends points of interest to visit to a user, according to their preferences and scores, and allows planning itineraries based on these recommendations, facilitating the organization of trips and allowing personalized discovery of cities. In addition, one of the objectives of this platform has been to create a service that respects the privacy of users.

To this end, a study of the possible techniques to be used for the implementation of the recommender system, the planner system, the user interface and the server has been carried out. It has been established the use of a hybrid system with the MPIP metric for the recommender system; the use of the PDDL language with a classic scheduler for the scheduler system; the development of a web application using the Bootstrap, Bootstrap Icons, Leaflet and Font Awesome libraries; the development a REST API using the JavaScript (in Node.js) and Rust languages, together with the fastify, sequelize and Neon libraries; and the use of MariaDB as a database.

During the analysis, requirements, use cases and prototypes were presented, as well as a description of the easily scalable distributed architecture. Subsequently, the process of data collection, user interface design and platform implementation and security has been described, explaining in detail the components and justifying the implementations and changes. In addition, a deployment process has been created that makes use of containers with Docker, private analytics services and a platform status page. In addition, Cloudflare and Nginx have been used to optimize and balance the resources and performance of the platform. Additionally, a series of rules and processes have been established so that anyone can contribute code or content to the platform, allowing anyone to improve it.

Subsequently, a series of experimentation and validation has been carried out. First, unit tests have been created for the API, all of which have been successfully completed. Interface tests have been set up through evaluation with users who have provided feedback on possible changes for the improvement of the user experience, of which some changes have already been implemented.

The recommender was found to have a difference of 1.39 points on a scale of 1 to 5, an accuracy that can be considered acceptable, especially based on the results of the user evaluation. In addition, the more users used for neighbor generation, the higher the accuracy.

Tests have also been carried out with Metric-FF, CBP and LPG planners. In these tests it has been concluded that the use of LPG and CBP can be beneficial compared to the

current implementation with Metric-FF, although the latter should be used as an alternative for complex cases.

Finally, the current legislation has been analyzed and a transparent and user-friendly privacy policy has been created.

7.1. Future work

During the development of this work, a series of decisions and implementations have been made, which, however, can be expanded and/or improved in the future.

In addition to the changes not implemented during experimentation and validation with users, monetization of the platform is proposed, either through advertisements, donations or subscriptions. In this way, a cash flow could be obtained that could be useful to improve and expand the platform's services and even introduce new cities and locations for users. With this additional financial flow, the use of map services such as Google Maps, Mapbox or Maptiler is proposed, reducing the server load on the map grid service, as well as the use of route planners such as Google Maps Directions API or Here Maps API to improve the calculations with distances and times of the planned routes, since currently a pre-calculated table of distances and times is used.

Regarding the tasks that are executed periodically, their execution frequency could be increased to reduce the computational load during these peaks or even implement systems that execute these tasks dynamically depending on the actual load of the platform or load prediction models.

It would also be convenient to make use of the Redis database, to optimize user sessions in order to avoid unnecessary database queries. In addition, a refactoring task could be performed to improve and optimize the project code.

During deployment, CI (Continuous Integration) and CD (Continuous Deployment) processes can be implemented to optimize testing and deployment processes. Similarly, migration to cloud platforms such as AWS, Azure, fly.io, Linode, etc., is proposed with the aim of improving service performance and efficiency.

Another possible area to study related to performance and efficiency is in databases. The platform code is horizontally scalable and distributed, however, the MariaDB database is only vertically scalable.

This limits the database to a single machine. An alternative presented is Apache Cassandra, a distributed and highly available NoSQL database that could speed up database queries.

It is also proposed the use of stochastic local search techniques, such as genetic algorithms, to optimize generated plans, along with the use of incremental clustering with OPTICS or DBSCAN for the recommendation task.

Finally, given that the system is currently only available to Spanish-speaking users and the city of Madrid, it is proposed to expand the cities currently available, starting with the most popular European destinations, along with an effort to internationalize and translate the user interface and the data in the database.