



UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Aplicación web para organizar viajes en grupo

Autor

Hugo Vázquez Docampo
Rodrigo de la Calle Alonso

Tutor F. Javier Gil Rubio

julio de 2024



AGRADECIMIENTOS

Nos gustaría comenzar agradeciendo a nuestras familias y sobre todo, nuestros padres, porque de no ser por ellos, no habríamos podido venir a estudiar este máster. Continuando por Madrid, esta inmensa y caótica ciudad que más allá de horrorizarnos nos ha enamorado y nos ha abierto las puertas a un futuro apasionante.

También agradecer a la Universidad Politécnica de Madrid y todos los profesores del Máster en Ingeniería Web por la formación recibida durante el curso. Por supuesto nuestros compañeros de clase que, aunque no ha sido mucho el tiempo que hemos pasado juntos, se ha sentido como una comunidad.

Por último, no podemos olvidar a todas esas cafeterías donde ya nos hemos convertido en habituales por pasar las tardes analizando, debatiendo, programando y documentando este proyecto.



RESUMEN

Este proyecto de fin de máster se basa en la creación de TripPlanner, una aplicación web diseñada para la organización de viajes en grupo, que facilita desde la elección de destinos y fechas hasta la administración completa de los planes de viaje. Desarrollada con Angular [6] y Spring Boot [7], la aplicación es completamente *responsive*, asegurando una experiencia de usuario fluida tanto en dispositivos móviles como en pantallas de mayores dimensiones. Mediante la integración de un sistema de reparto de gastos, gestión de pagos, tickets, mapas y un calendario de eventos, proporciona una herramienta completa para manejar todos los aspectos logísticos y financieros de los viajes grupales. El desarrollo se ha apoyado en las metodologías, tecnologías avanzadas y buenas prácticas de programación adquiridas a lo largo del máster, complementadas con una sólida batería de pruebas que respaldan la robustez del software.

Además, el proyecto ha integrado herramientas profesionales como Liquibase [5] para la gestión de bases de datos, SonarCloud [10] y GitHub Actions [9] para la integración y despliegue continuos (CI/CD), y Docker [8] para la contenerización del *backend* [4], lo que agiliza significativamente los despliegues. Asimismo, se han utilizado Swagger-UI [11] para la prueba de interfaces de *backend* [4] y Orval [1] para la generación automática de modelos en Angular [6]. El uso de estas utilidades ha Enriquecido el desarrollo y la gestión del proyecto, optimizando el proceso de desarrollo.

La experiencia laboral previa y los conocimientos adquiridos durante el máster han sido fundamentales para superar diversos desafíos técnicos, como la compatibilidad entre versiones de librerías, la configuración de arquitecturas de software y la gestión de herramientas de desarrollo. La implementación desde cero de GitHub Actions [9] y GitHub Projects ha permitido automatizar y optimizar los procesos de desarrollo continuo.

Este proyecto no solo refleja las competencias técnicas desarrolladas, sino que también destaca la capacidad de aplicar conocimientos prácticos en un contexto real y complejo, demostrando la preparación para enfrentar y resolver problemas técnicos avanzados en el ámbito de la ingeniería web.

Por último, es importante mencionar que realizar este trabajo en parejas nos permitió repartir las diferentes tareas que conforman el proyecto, aprovechando las fortalezas de cada uno para potenciar el proceso de desarrollo. Esta colaboración maximizó la calidad del software producido, al combinar habilidades complementarias y fomentar un entorno de constante aprendizaje y adaptación.

PALABRAS CLAVE

Spring Boot, Angular, PostgreSQL, RUP, Viajes, Planes, Pagos, Gestión, Responsive, UI/UX.



ABSTRACT

This master's thesis project is based on the creation of TripPlanner, a web application designed for organizing group travel, facilitating everything from choosing destinations and dates to complete management of travel plans. Developed with Angular [6] and Spring Boot [7], the application is fully *responsive*, ensuring a smooth user experience on both mobile devices and larger screens. By integrating a cost-sharing system, payment management, tickets, maps, and an event calendar, it provides a complete tool to handle all logistical and financial aspects of group travel.

The development has been supported by the methodologies, advanced technologies, and best programming practices acquired throughout the master's program, complemented by a solid set of tests that underpin the robustness of the software.

Additionally, the project has integrated professional tools such as Liquibase [5] for database management, SonarCloud [10] and GitHub Actions [9] for continuous integration and deployment (CI/CD), and Docker [8] for *backend* [4] containerization, significantly speeding up deployments. Swagger-UI [11] has also been used for *backend* [4] interface testing and Orval [1] for the automatic generation of models in Angular [6]. The use of these utilities has enriched the development and management of the project, optimizing the development process.

Previous work experience and knowledge gained during the master's program have been crucial in overcoming various technical challenges, such as compatibility between library versions, configuration of software architectures, and management of development tools. Implementing GitHub Actions [9] and GitHub Projects from scratch has allowed for the automation and optimization of continuous development processes.

This project not only reflects the technical skills developed but also highlights the ability to apply practical knowledge in a real and complex context, demonstrating the preparation to face and solve advanced technical problems in the field of web engineering.

Lastly, it is important to mention that doing this work in pairs allowed us to distribute the different tasks that make up the project, leveraging each other's strengths to enhance the development process. This collaboration maximized the quality of the software produced by combining complementary skills and fostering an environment of constant learning and adaptation.

KEYWORDS

Spring Boot, Angular, PostgreSQL, RUP, Trip, Plans, Bills, Management, Responsive, UI/UX.



TABLA DE CONTENIDOS

Contenido

Agradecimientos	3
Resumen.....	5
Palabras clave	5
Abstract.....	7
Keywords	7
Tabla de Contenidos	9
Objetivos	1
Capítulo 1: Introducción	3
1.1. Metodología.....	3
1.1. Modelo del dominio	5
Capítulo 2: Modelo de requisitos.....	7
2.1. Objetivos del sistema	7
2.2. Actores	9
2.3. Requisitos de información.....	10
2.4. Requisitos No Funcionales.....	11
2.5. Requisitos Funcionales.....	14
2.6. Matriz de Rastreabilidad	41
Capítulo 3: Diseño del Sistema.....	43
3.1. Diagrama de Arquitectura	43
3.2. Diagrama de Entidad Relación.....	43
3.3. Diagrama de Secuencia	44
3.4. Prototipo de Interfaz de Usuario	45
3.5. Diseño UI/UX	47
Capítulo 4: Implementación.....	49
4.1. Entorno de desarrollo	49
4.2. Semilla de la aplicación	50
4.4. Diseño MVC	52
4.5. Calidad de software.....	53
4.6. Pantalla <i>responsive</i>	54
Capítulo 5: Pruebas	58
5.1. Pruebas de DTOs.....	58
5.2. Pruebas de integración con la base de datos	58
5.3. Cobertura de pruebas y análisis de la calidad de código.....	59
Capítulo 6: Gestión del Proyecto	62
6.1. Cronograma.....	62

6.2. EDT (Estructura de Desglose de Trabajo)	63
Conclusiones y Posibles Ampliaciones	64
Bibliografía	66



Ilustraciones

Ilustración 1: Github Project con las pilas de trabajo.....	3
Ilustración 2: Rastreo de tareas mediante Github.....	4
Ilustración 3: Diagrama de clases.....	5
Ilustración 4: Objetivo Gestión de Usuarios	7
Ilustración 5: Objetivo Gestión de Viajes	7
Ilustración 6: Objetivo Gestión de Propuestas	8
Ilustración 7: Objetivo Gestión de Planes	8
Ilustración 8: Objetivo Gestión de Tickets.....	8
Ilustración 9: Objetivo Gestión de Pagos	9
Ilustración 10: Actor Usuario Anónimo	9
Ilustración 11: Actor Usuario	9
Ilustración 12: Requisito Información de Usuarios.....	10
Ilustración 13: Requisito Información de Viajes.....	10
Ilustración 14: Requisito Información de Planes.....	11
Ilustración 15: Requisito Usabilidad	11
Ilustración 16: Requisito Compatibilidad.....	12
Ilustración 17: Requisito Confidencialidad.....	12
Ilustración 18: Requisito Disponiblidad	13
Ilustración 19: Requisito Seguridad	13
Ilustración 20: Requisito Rendimiento	13
Ilustración 21: Grupo Funcional Gestión de Usuarios	14
Ilustración 22: Grupo Funcional Gestión de Viajes	15
Ilustración 23: Grupo Funcional Gestión de Planes	15
Ilustración 24: Grupo Funcional Gestión de Ubicación	16
Ilustración 25: Grupo Funcional Gestión de Horario	16
Ilustración 26: Grupo Funcional Gestión de Tickets.....	16
Ilustración 27: Grupo Funcional Gestión de Pagos	17
Ilustración 28: Grupo Funcional Gestión de Propuestas	17
Ilustración 29: Caso de Uso Iniciar Sesión.....	18
Ilustración 30: Diagrama de Secuencia Iniciar Sesión	18
Ilustración 31: Caso de Uso Registrarse.....	19
Ilustración 32: Diagrama de Secuencia Registrarse	19
Ilustración 33: Caso de uso Recuperar Contraseña	20
Ilustración 34: Diagrama de Secuencia Recuperar Contraseña.....	20
Ilustración 35: Caso de Uso Cerrar Sesión.....	21
Ilustración 36: Diagrama de Secuencia Cerrar Sesión	21
Ilustración 37: Caso de Uso Editar Perfil	22
Ilustración 38: Diagrama de Secuencia Editar Perfil	22
Ilustración 39: Caso de Uso Eliminar Cuenta	23
Ilustración 40: Diagrama de Secuencia Eliminar Cuenta	23
Ilustración 41: Caso de Uso CRUD Viaje	24
Ilustración 42: Diagrama de Secuencia CRUD Viaje	25
Ilustración 43: Caso de Uso Añadir Participante al Viaje	26
Ilustración 44: Diagrama de Secuencia Añadir Participante al Viaje	26
Ilustración 45: Caso de Uso Eliminar Participante del Viaje	27
Ilustración 46: Diagrama de Secuencia Eliminar Participante del Viaje	27
Ilustración 47: Caso de Uso CRUD Plan	28
Ilustración 48: Diagrama de Secuencia CRUD Plan	29

Ilustración 49: Caso de Uso Valorar Plan	29
Ilustración 50: Diagrama de Secuencia Valorar Plan.....	30
Ilustración 51: Caso de Uso CRUD Ubicación	30
Ilustración 52: Diagrama de Secuencia CRUD Ubicación.....	31
Ilustración 53: Caso de Uso CRUD Horario	32
Ilustración 54: Diagrama de Secuencia CRUD Horario.....	33
Ilustración 55: Caso de Uso CRUD Ticket	34
Ilustración 56: Diagrama de Secuencia CRUD Tickets	34
Ilustración 57: Caso de Uso CRUD de Pagos	35
Ilustración 58: Diagrama de Secuencia CRUD Pagos	36
Ilustración 59: Caso de Uso CRUD Propuesta.....	37
Ilustración 60: Diagrama de Secuencia CRUD Propuesta	38
Ilustración 61: Caso de Uso Valorar Propuesta.....	39
Ilustración 62: Diagrama de Secuencia Valorar Propuesta	39
Ilustración 63: Caso de Uso Marcar Como Finalista.....	40
Ilustración 64: Diagrama de Secuencia Marcar como Finalista	40
Ilustración 65: TRM-01	41
Ilustración 66: TRM-02	41
Ilustración 67: Diagrama de Arquitectura del Sistema.....	43
Ilustración 68 Diagrama de entidad relación	43
Ilustración 69: Diagrama de secuencia Registrarse	44
Ilustración 70: Diagrama de Secuencia Añadir participante al viaje	44
Ilustración 71: Diagrama de Secuencia CRUD Plan	45
Ilustración 72 Diagrama de Secuencia Marcar Como Finalista	45
Ilustración 73: Prototipo de Interfaz de Usuario	46
Ilustración 74: Selección de colores para la aplicación.....	47
Ilustración 75: Ejemplo de acción de GitHub	49
Ilustración 76: estructura scripts Liquibase	50
Ilustración 77: WorkFlows de GitHub	51
Ilustración 78: Ejemplo de consulta con myBatis	51
Ilustración 79: Ejemplo de uso Lombok	51
Ilustración 80 Modelo Vista Controlador [13]	52
Ilustración 81: Pantalla de tickets (ordenador)	54
Ilustración 82: Pantalla de tickets (móvil).....	54
Ilustración 83: fragmento de código responsive.....	55
Ilustración 84: Fragmento de estilos responsive	55
Ilustración 85: Pantalla responsive	56
Ilustración 86: Componente responsivo	56
Ilustración 87: Ejemplo de test de DTO	58
Ilustración 88: Ejemplo TestIT	59
Ilustración 89: insert Liquibase viajes	59
Ilustración 90: Análisis de Calidad - SonarCloud	60
Ilustración 91: Cronograma del Proyecto.....	62
Ilustración 92: EDT del proyecto	63





OBJETIVOS

El objetivo primordial de este proyecto es demostrar la aplicabilidad de los conocimientos adquiridos, a través de la implementación de un proyecto que simule las condiciones y desafíos de un entorno laboral real. Se busca, no solo poner a prueba la capacidad técnica en el uso de tecnologías punteras en el mercado (Angular [6], Spring [7]...), sino también evaluar la eficacia con la que podemos manejar un proyecto complejo en un contexto que simula la dinámica de trabajo en equipo en el mundo de la Ingeniería de Software.

Además, al tratarse de un desarrollo por parejas, tratamos de simular un entorno colaborativo real, donde la comunicación efectiva y la coordinación son esenciales para el éxito del producto final.

En términos prácticos, nuestro enfoque se centra en aplicar los conocimientos adquiridos en las diferentes asignaturas del máster. Esto incluye, desde la programación avanzada hasta la gestión de proyectos y el análisis de sistemas.

Este proyecto también se propone como una plataforma para integrar y aplicar algunas prácticas de desarrollo de software, incluyendo la programación ágil, la integración continua (CI), y la entrega continua (CD), utilizando herramientas como SonarCloud [10] y Docker [8], que facilitan la revisión de código y la gestión de despliegues de manera automatizada. A través de esta integración de herramientas y metodologías, buscamos crear un producto que no solo sea técnica y funcionalmente robusto, sino que también refleje las competencias interdisciplinarias fomentadas durante el máster.



CAPÍTULO 1: INTRODUCCIÓN

La aplicación web TripPlanner busca ser una solución integral y competitiva que centralice la gestión de todos los elementos implicados en la organización de viajes en grupo. Para materializar esta idea, se ha llevado a cabo un análisis de mercado, evaluando las fortalezas y debilidades de las opciones existentes.

Este estudio ha permitido desarrollar una propuesta que no solo busca ofrecer una experiencia de usuario óptima, sino también cubrir una necesidad que surge debido a la globalización y la popularización de los viajes como forma de ocio. TripPlanner busca captar la atención de los usuarios mediante una interfaz amigable y funcionalidades avanzadas, estableciéndose como la opción preferente en la categoría de planificación de viajes grupales.

1.1. Metodología

En el desarrollo de este proyecto, se ha optado por hacer uso de metodologías RUP (Rational Unified Process [12]) para gestionar y planificar el trabajo. Este enfoque se adaptada muy bien a los requisitos del proyecto.

La fase de implementación se ha organizado en torno a una serie de pilas de tareas, con el objetivo de mantener una mayor trazabilidad del proceso de desarrollo. Se muestran en la siguiente figura:

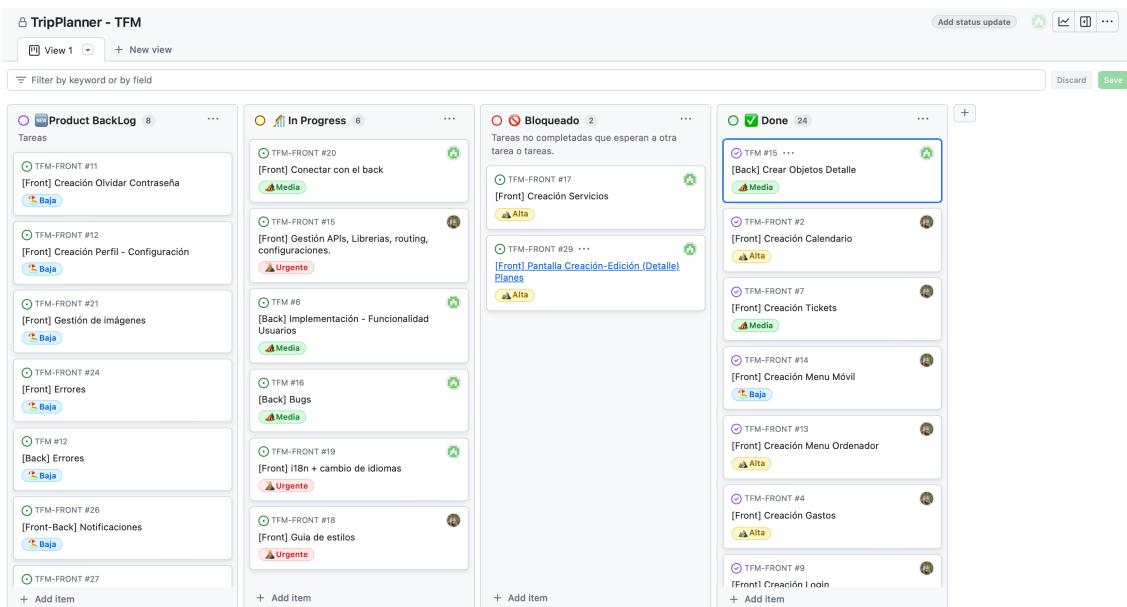


Ilustración 1: Github Project con las pilas de trabajo

- La pila del “**Producto**” contiene las diferentes tareas definidas a la hora de identificar los requisitos del proyecto.
- Por otra parte, la pila “**En progreso**”, muestra las tareas que se están desarrollando.

- La pila de “**Bloqueados**”, recoge aquellas tareas que no pueden continuar su desarrollo porque dependen de otras que se encuentran, o bien en desarrollo, o bien pendientes de implementar.
- Por último, la pila de “**Done**”, muestra aquellas tareas desarrolladas.

La pizarra de trabajo empleada ha sido personalizada para poder añadir atributos como el nivel de complejidad estimado para cada una de las tareas, así como el repositorio del que beben dichas tareas para que, al realizar los diferentes *commits* se guarde un registro del progreso de las tareas:

The screenshot shows a GitHub commit history for a repository named 'TripPlanner - TFM'. It highlights several commits related to task tracking:

- A commit from 'hugovdocampo' moved an issue from 'Product BackLog' to 'In Progress' on April 24.
- On April 24, 'hugovdocampo' added 7 commits referencing this issue:
 - inclusión de liquibase + base de datos en la nube + tabla core #1 (commit 9baff08)
 - intento de swagger parte 1 #1 (commit c690e1a)
 - intento de swagger parte 1 #1 (commit 81066d2)
 - sonar y slack #1 (commit bdfaef11)
 - relanzar job #1 (commit 58dd605)
 - reestructuración de ficheros #1 (commit 420e28d)
 - relanzar job #1 (commit a48ba69)
- On April 28, 'hugovdocampo' added 2 more commits referencing this issue:
 - sonar final #1 (commit 2cdcf9)
 - conflicts #1 (commit 7d1b748)

Ilustración 2: Rastreo de tareas mediante Github

Pese a considerar una metodología RUP, algunas de las herramientas de las metodologías ágiles como el uso de pilas de trabajo se han incluido en el proyecto, dejando en la pila del producto aquellas tareas consideradas secundarias para futuras mejoras de la aplicación (fuera del ámbito de este proyecto de fin de máster).

La priorización de las historias en la pila del producto se realiza en base a la importancia y las necesidades que se deben abordar. Se da prioridad a aquellas historias que son fundamentales para cumplir con los objetivos del proyecto y brindar un valor significativo a los usuarios.

El trabajo en el proyecto se planifica para el periodo que comprende entre el 1 de abril y el 25 de mayo, incluyendo el tiempo dedicado a la documentación.



1.1. Modelo del dominio

Tras presentar la aplicación a tratar en este proyecto y la metodología escogida para su desarrollo, se procede a presentar el diagrama de clases, tratando de recoger los aspectos principales del sistema a desarrollar. Esta es una parte fundamental del modelado de objetos y es esencial para el desarrollo basado en la programación orientada a objetos.

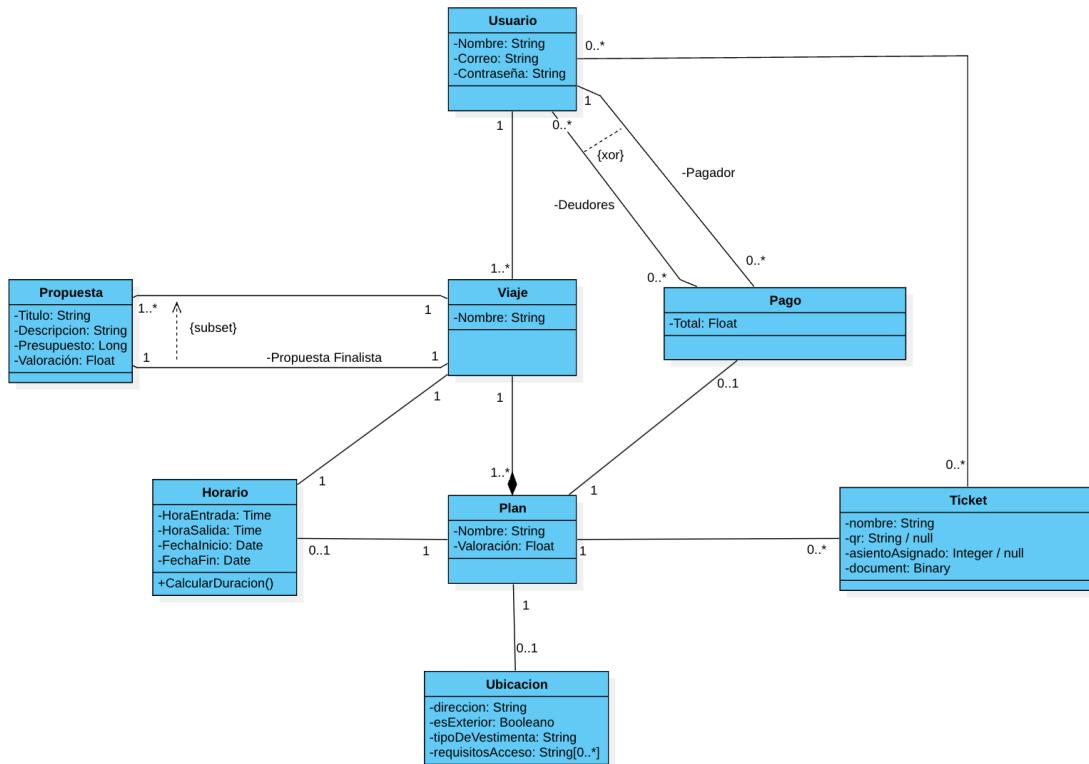


Ilustración 3: Diagrama de clases



CAPÍTULO 2: MODELO DE REQUISITOS

En esta sección se detallan los distintos requisitos del sistema, con el propósito de establecer un marco claro que delimita el alcance de la aplicación. Este enfoque permitirá definir las bases fundamentales necesarias para realizar una estimación precisa y válida del producto a desarrollar:

2.1. Objetivos del sistema

OBJ-001	Gestión de Usuarios
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Subobjetivos	No
Importancia	Alta
Urgencia	Máxima
Estabilidad	Alta
Descripción	Se debe de poder trabajar con la creación, edición y eliminación de usuarios además de identificarlos y almacenar toda la información necesaria de estos.

Ilustración 4: Objetivo Gestión de Usuarios

OBJ-002	Gestión de Viajes
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Subobjetivos	[OBJ-003] Gestión de Propuestas: se han de poder presentar y votar diferentes propuestas de viaje entre los participantes.
Importancia	Media
Urgencia	Media
Estabilidad	Alta
Descripción	Se debe de poder trabajar con la creación, edición y eliminación de Viajes presentes, pasados y futuros.

Ilustración 5: Objetivo Gestión de Viajes

OBJ-003	Gestión de Propuestas
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Subobjetivos	No
Importancia	Alta
Urgencia	Alta
Estabilidad	Alta
Descripción	Se debe de poder trabajar con la creación, edición y eliminación de Propuestas para un viaje. El objetivo es poder votar diferentes opciones para poder llegar a un acuerdo (propuesta finalista).

Ilustración 6: Objetivo Gestión de Propuestas

OBJ-004	Gestión de Planes
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Subobjetivos	[OBJ-005] Gestión de Tickets: se deben poder almacenar los tickets de un plan. [OBJ-006] Gestión de Pagos: se deben poder almacenar los pagos de un plan.
Importancia	Alta
Urgencia	Media
Estabilidad	Alta
Descripción	Se debe de poder trabajar con la creación, edición y eliminación de los planes que componen un viaje.

Ilustración 7: Objetivo Gestión de Planes

OBJ-005	Gestión de Tickets
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Subobjetivos	No
Importancia	Alta
Urgencia	Media
Estabilidad	Alta
Descripción	Se debe de poder trabajar con la creación, edición y eliminación de los tickets (entradas o billetes) asociados a un plan.

Ilustración 8: Objetivo Gestión de Tickets



OBJ-006	Gestión de Pagos
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Subobjetivos	No
Importancia	Alta
Urgencia	Media
Estabilidad	Alta
Descripción	Se debe de poder trabajar con la creación, edición y eliminación de los pagos asociados a un plan y varios usuarios para posteriormente repartir y equilibrar los gastos.

Ilustración 9: Objetivo Gestión de Pagos

2.2. Actores

ACT-001	Usuario Anónimo
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Descripción	Se trata de aquel actor que no ha iniciado sesión o no se ha registrado todavía.

Ilustración 10: Actor Usuario Anónimo

ACT-002	Usuario
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Descripción	Es aquel actor que está registrado en la plataforma.

Ilustración 11: Actor Usuario

2.3. Requisitos de información

En este apartado se definen los requisitos relativos al modelo de dominio del proyecto.

IRQ-001	Información de Usuarios	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	[OBJ-001]: Gestión de Usuarios	
Datos Específicos	Nombre, Apellidos, Email.	
Tiempo de Vida	Medio	Máximo
	4 años	10 años
Ocurrencias simultáneas	Medio	Máximo
	20	200
Importancia	Muy Alta	
Urgencia	Alta	
Estabilidad	Alta	

Ilustración 12: Requisito Información de Usuarios

IRQ-002	Información de Viajes	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	[OBJ-002]: Gestión de Viajes [OBJ-003]: Gestión de Propuestas	
Datos Específicos	Nombre, Fechas, Propuestas	
Tiempo de Vida	Medio	Máximo
	2 años	8 años
Ocurrencias simultáneas	Medio	Máximo
	10	100
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Alta	

Ilustración 13: Requisito Información de Viajes



IRQ-003	Información de Planes	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	[OBJ-004]: Gestión de Planes [OBJ-005]: Gestión de Tickets [OBJ-006]: Gestión de Pagos	
Datos Específicos	Título, Valoración, Horario, Ubicación, Tickets, Pagos	
Tiempo de Vida	Medio	Máximo
	2 años	8 años
Ocurrencias simultáneas	Medio	Máximo
	20	200
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Alta	

Ilustración 14: Requisito Información de Planes

2.4. Requisitos No Funcionales

En este apartado se definen los requisitos que forman parte del proyecto y que no se refieren directamente a las funciones específicas que provee.

NFR-001	Usabilidad
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Dependencias	No
Descripción	Deberá ser un software intuitivo que no requiera de formación o un proceso de aprendizaje concreto. Tiene que poder ser usado por personas con habilidades básicas de informática y en dispositivos móviles.
Importancia	Alta
Urgencia	Media
Estabilidad	Muy Alta

Ilustración 15: Requisito Usabilidad

NFR-002		Compatibilidad
Versión		1.0 (02/04/2024)
Autores		Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Dependencias		No
Descripción		El sistema deberá poder ser utilizado en todos los sistemas operativos compatibles con navegadores web actuales como Google Chrome.
Importancia		Alta
Urgencia		Media
Estabilidad		Muy Alta

Ilustración 16: Requisito Compatibilidad

NFR-003		Confidencialidad
Versión		1.0 (02/04/2024)
Autores		Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Dependencias		No
Descripción		El sistema deberá restringir los permisos de acceso a cada usuario del sistema. Se deben respetar los permisos marcados para cada tipo de Actor.
Importancia		Muy Alta
Urgencia		Alta
Estabilidad		Muy Alta

Ilustración 17: Requisito Confidencialidad



NFR-004	Disponibilidad
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Dependencias	No
Descripción	El sistema tiene que estar disponible durante todo el día (24h) salvo excepciones puntuales para labores de mantenimiento.
Importancia	Alta
Urgencia	Media
Estabilidad	Media

Ilustración 18: Requisito Disponibilidad

NFR-005	Seguridad
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Dependencias	No
Descripción	El sistema debe de mantener la seguridad de las comunicaciones e integridad de los datos de los usuarios.
Importancia	Muy Alta
Urgencia	Muy Alta
Estabilidad	Muy Alta

Ilustración 19: Requisito Seguridad

NFR-006	Rendimiento
Versión	1.0 (02/04/2024)
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso
Dependencias	No
Descripción	El sistema tiene que ser ligero para poder funcionar de forma fluida en dispositivos con capacidades computacionales básicas.
Importancia	Alta
Urgencia	Media
Estabilidad	Alta

Ilustración 20: Requisito Rendimiento

2.5. Requisitos Funcionales

Se definen los servicios del sistema.

2.5.1. Diagramas de Casos de Uso

En este apartado se muestra la comunicación y comportamiento del sistema mediante su interacción entre sus componentes.

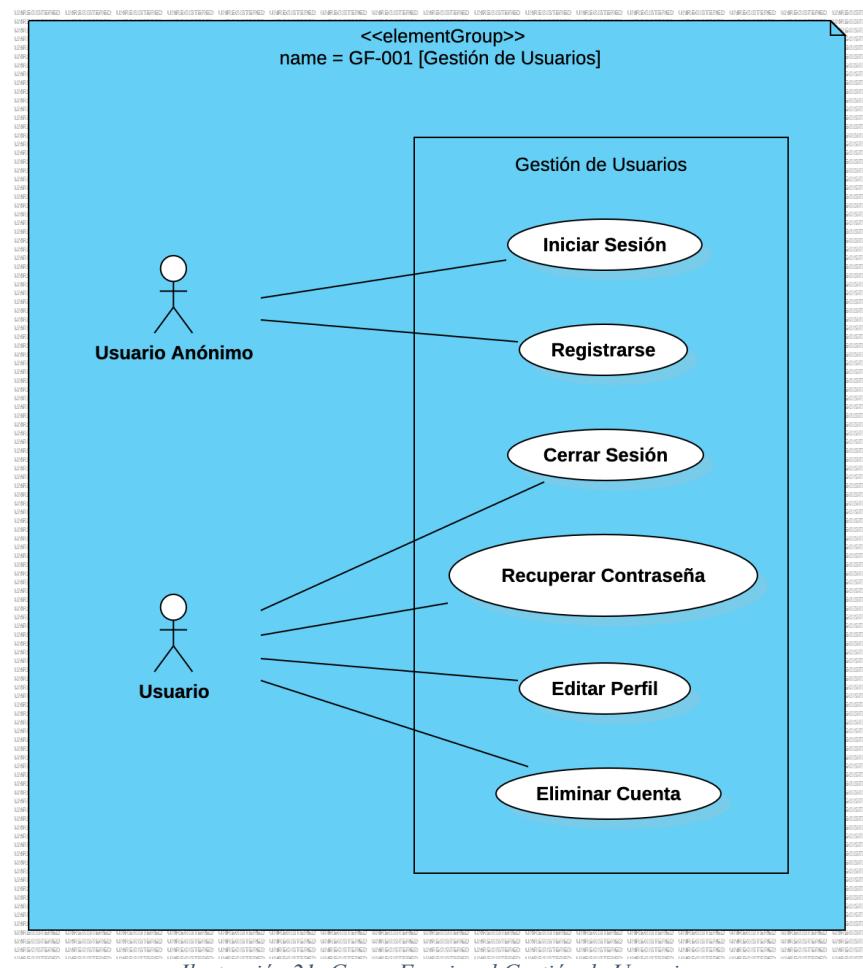


Ilustración 21: Grupo Funcional Gestión de Usuarios

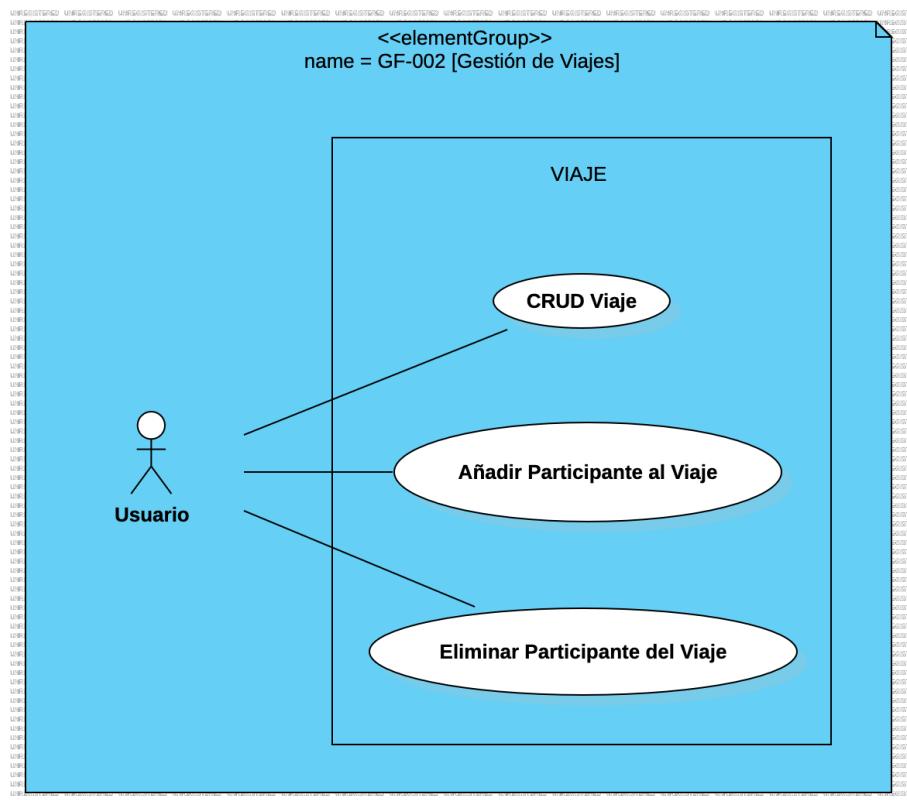


Ilustración 22: Grupo Funcional Gestión de Viajes

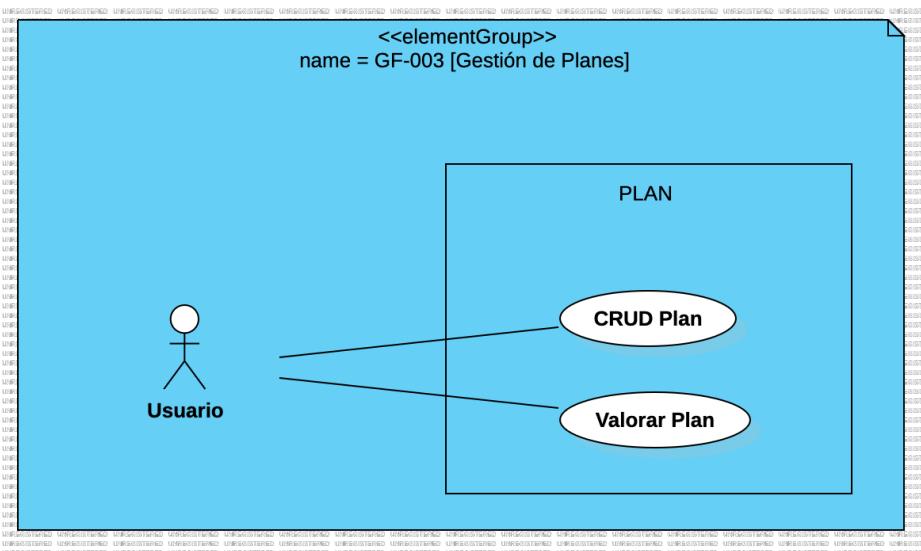


Ilustración 23: Grupo Funcional Gestión de Planes

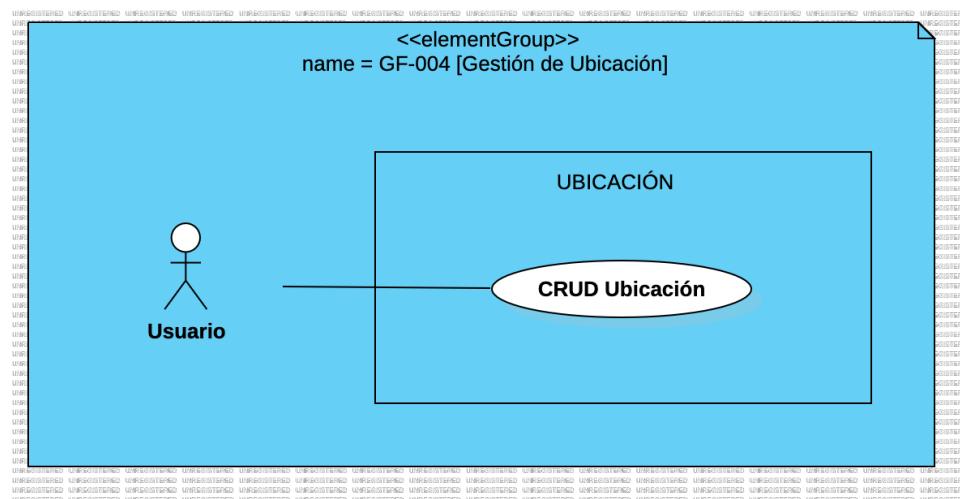


Ilustración 24: Grupo Funcional Gestión de Ubicación

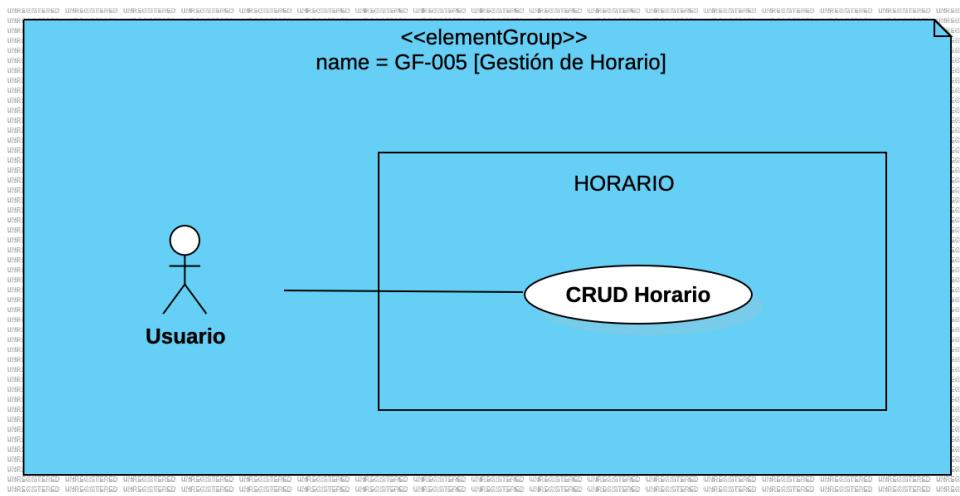


Ilustración 25: Grupo Funcional Gestión de Horario

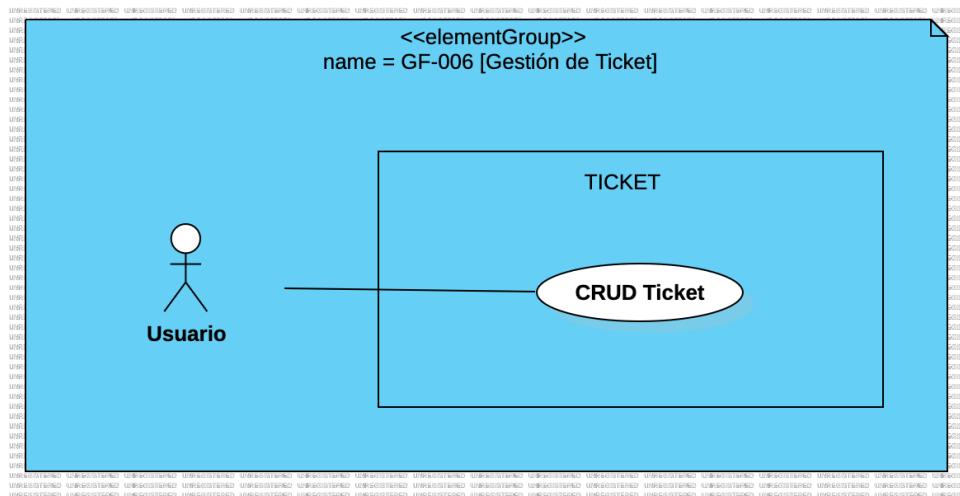


Ilustración 26: Grupo Funcional Gestión de Tickets

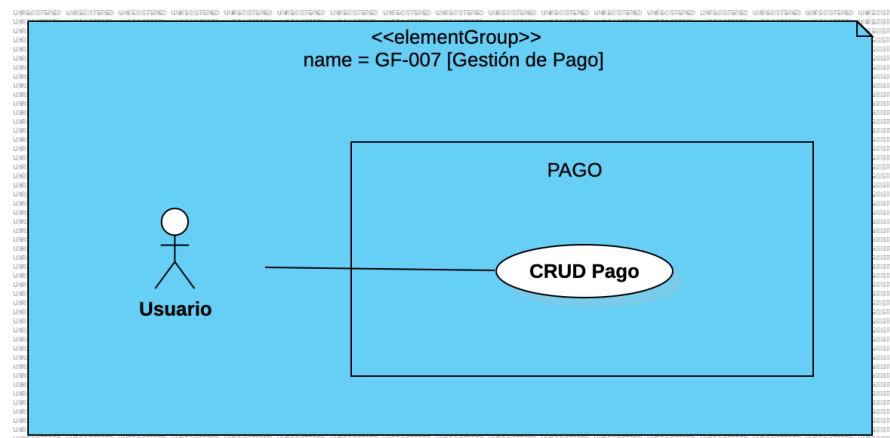


Ilustración 27: Grupo Funcional Gestión de Pagos

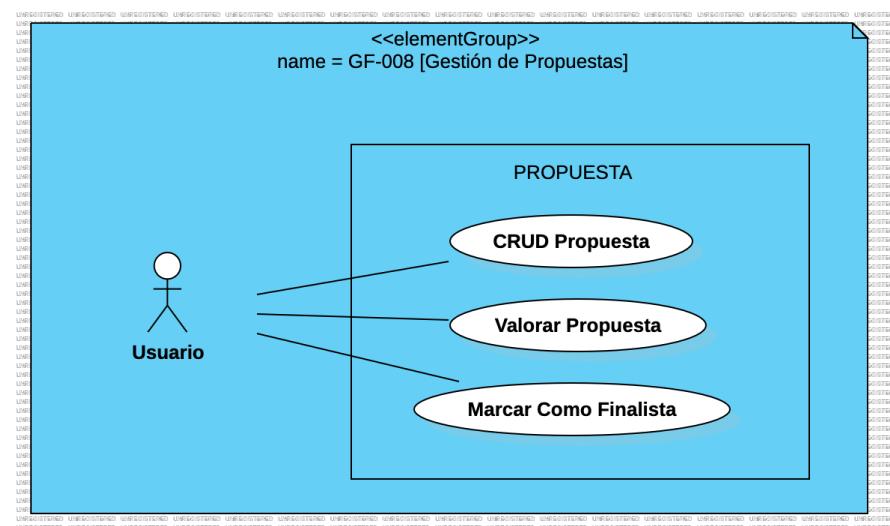


Ilustración 28: Grupo Funcional Gestión de Propuestas

2.5.2. Tablas de Casos de Uso

UC-001	Iniciar Sesión	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-001] Gestión de Usuarios [IRQ-001] Información de los Usuarios 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario entre en el sistema o después de cerrar sesión.	
Precondición	No	
Secuencia Normal	Paso	Acción
	1	El actor usuario anónimo [ACT-001] escribe el email y la contraseña.
	2	El actor usuario anónimo [ACT-001] pulsa la opción iniciar sesión.
	3	El actor usuario anónimo [ACT-001] inicia la sesión como un actor usuario [ACT-002].
Postcondición	No	
Excepciones	Paso	Acción
	3	Las credenciales no son correctas y se muestra un error.
Importancia	Muy Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 29: Caso de Uso Iniciar Sesión

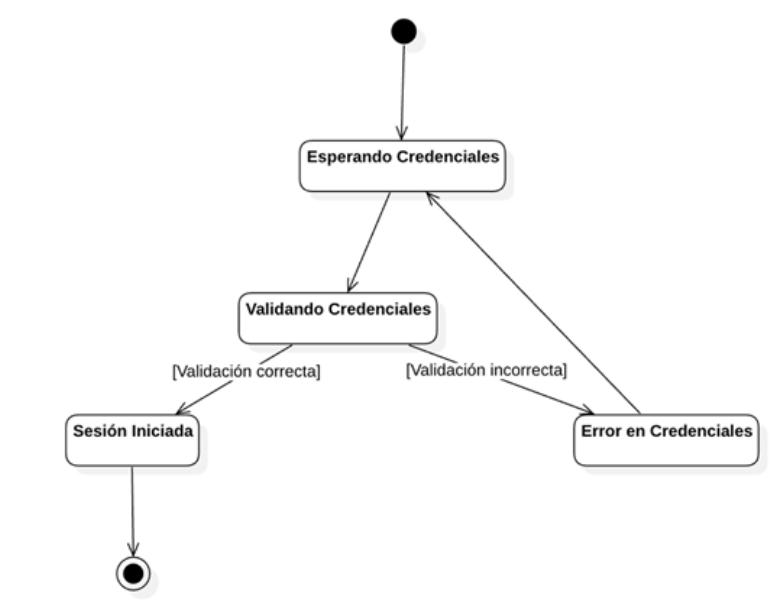


Ilustración 30: Diagrama de Secuencia Iniciar Sesión



UC-002	Registrarse	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-001] Gestión de Usuarios[IRQ-001] Información de los Usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera Registrarse.	
Precondición	No puede existir una cuenta en el sistema con el mismo correo electrónico.	
Secuencia Normal	Paso	Acción
	1	El actor usuario anónimo [ACT-001] completa los datos correspondientes.
	2	El actor usuario anónimo [ACT-001] selecciona la opción para registrarse.
	3	El actor usuario queda registrado en la plataforma.
Postcondición	No	
Excepciones	Paso	Acción
	3	Si se encuentra un error en los datos del usuario, devuelve un error.
Importancia	Muy Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 31: Caso de Uso Registrarse



Ilustración 32: Diagrama de Secuencia Registrarse

UC-003	Recuperar Contraseña	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-001] Gestión de Usuarios [IRQ-001] Información de los Usuarios 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera recuperar la contraseña olvidada.	
Precondición	Debe de haber un usuario registrado en el sistema con ese correo previamente.	
Secuencia Normal	Paso	Acción
	1	El actor usuario anónimo [ACT-001] escribe el email de su cuenta para recibir el correo de recuperar contraseña.
	2	El actor usuario anónimo [ACT-001] selecciona la opción para recuperar contraseña con su correo electrónico.
	3	Se envía el correo de recuperación de la contraseña.
Postcondición	No	
Excepciones	Paso	Acción
	3	Si se encuentra un error en los datos del usuario devuelve un error.
Importancia	Media	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 33: Caso de uso Recuperar Contraseña

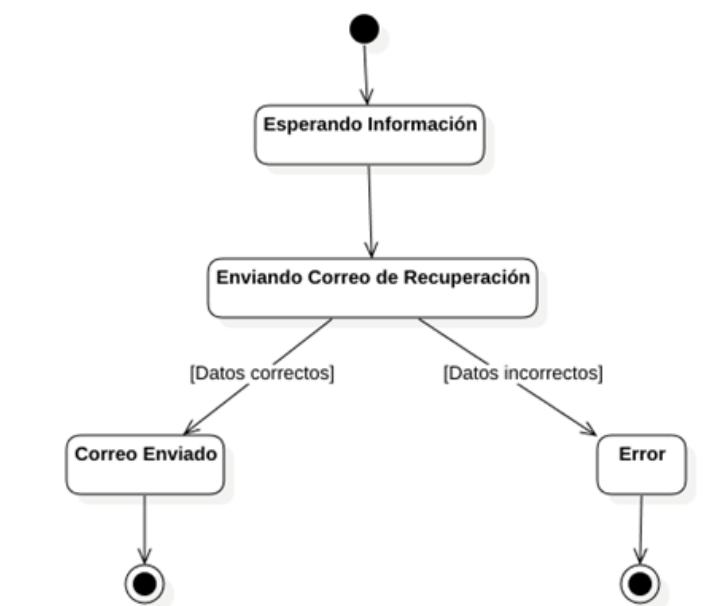


Ilustración 34: Diagrama de Secuencia Recuperar Contraseña



UC-004	Cerrar Sesión	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-001] Gestión de Usuarios[IRQ-001] Información de los Usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera cerrar la sesión iniciada.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] seleccionan la opción de cerrar sesión.
	2	La sesión queda cerrada.
	No	
Excepciones	No	
Importancia	Media	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 35: Caso de Uso Cerrar Sesión

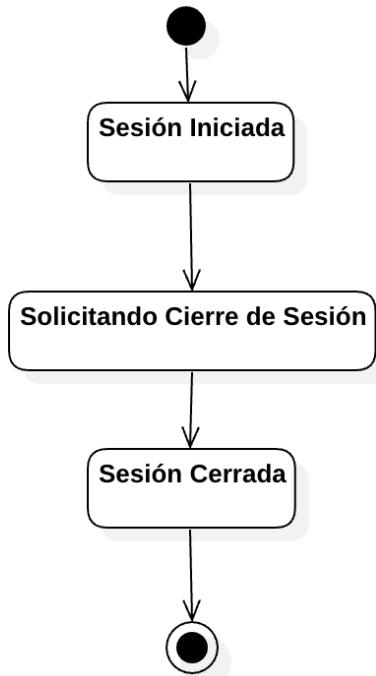


Ilustración 36: Diagrama de Secuencia Cerrar Sesión

UC-005	Editar Perfil	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-001] Gestión de Usuarios [IRQ-001] Información de los Usuarios 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera editar sus datos del perfil.	
Precondición	El actor usuario [ACT-002] tiene la sesión iniciada [UC-001].	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] modifica sus datos del perfil.
	2	El actor usuario [ACT-002] selecciona la opción para guardar los datos.
	3	Los datos son guardados en el sistema.
Postcondición	No	
Excepciones	Paso	Acción
	3	Si se encuentra un error en los datos del usuario, devuelve un error.
Importancia	Alta	
Urgencia	Media	
Estabilidad	Muy Alta	

Ilustración 37: Caso de Uso Editar Perfil

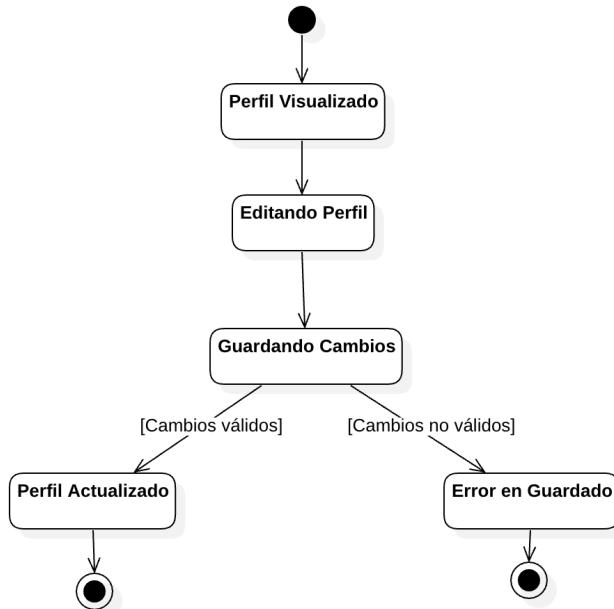


Ilustración 38: Diagrama de Secuencia Editar Perfil



UC-006	Eliminar Cuenta	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-001] Gestión de Usuarios[IRQ-001] Información de los Usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera cerrar la sesión iniciada.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la opción de eliminar la cuenta.
	2	El actor usuario [ACT-002] confirma que quiere verificar la cuenta.
	3	Se elimina la cuenta y todos los datos dependientes de ella
Postcondición	No	
Excepciones	Paso	Acción
	3	Si se encuentra un error en la eliminación de la cuenta, devuelve un error.
Importancia	Media	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 39: Caso de Uso Eliminar Cuenta

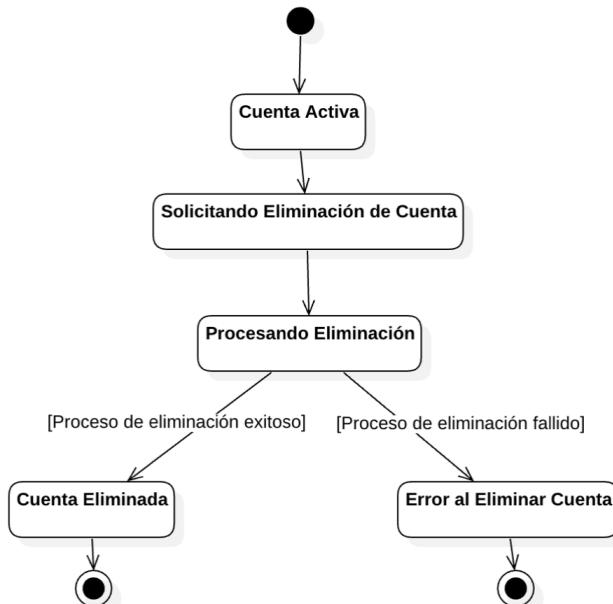


Ilustración 40: Diagrama de Secuencia Eliminar Cuenta

UC-007	CRUD Viaje	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-002] Gestión de Viajes [IRQ-002] Información de los Viajes 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar un viaje.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la vista de gestión de viajes.
	2	El actor usuario [ACT-002] visualiza los viajes.
	3.A	El actor usuario [ACT-002] selecciona crear un viaje nuevo.
	3.B	El actor usuario [ACT-002] selecciona modificar un viaje existente.
	3.C	El actor usuario [ACT-002] selecciona eliminar un viaje existente.
	4	El actor usuario [ACT-002] completa los datos.
Postcondición	5 El actor usuario [ACT-002] selecciona la opción confirmar los cambios.	
	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
Excepciones	Paso	Acción
	5	Si se encuentra un error en los datos del viaje, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 41: Caso de Uso CRUD Viaje

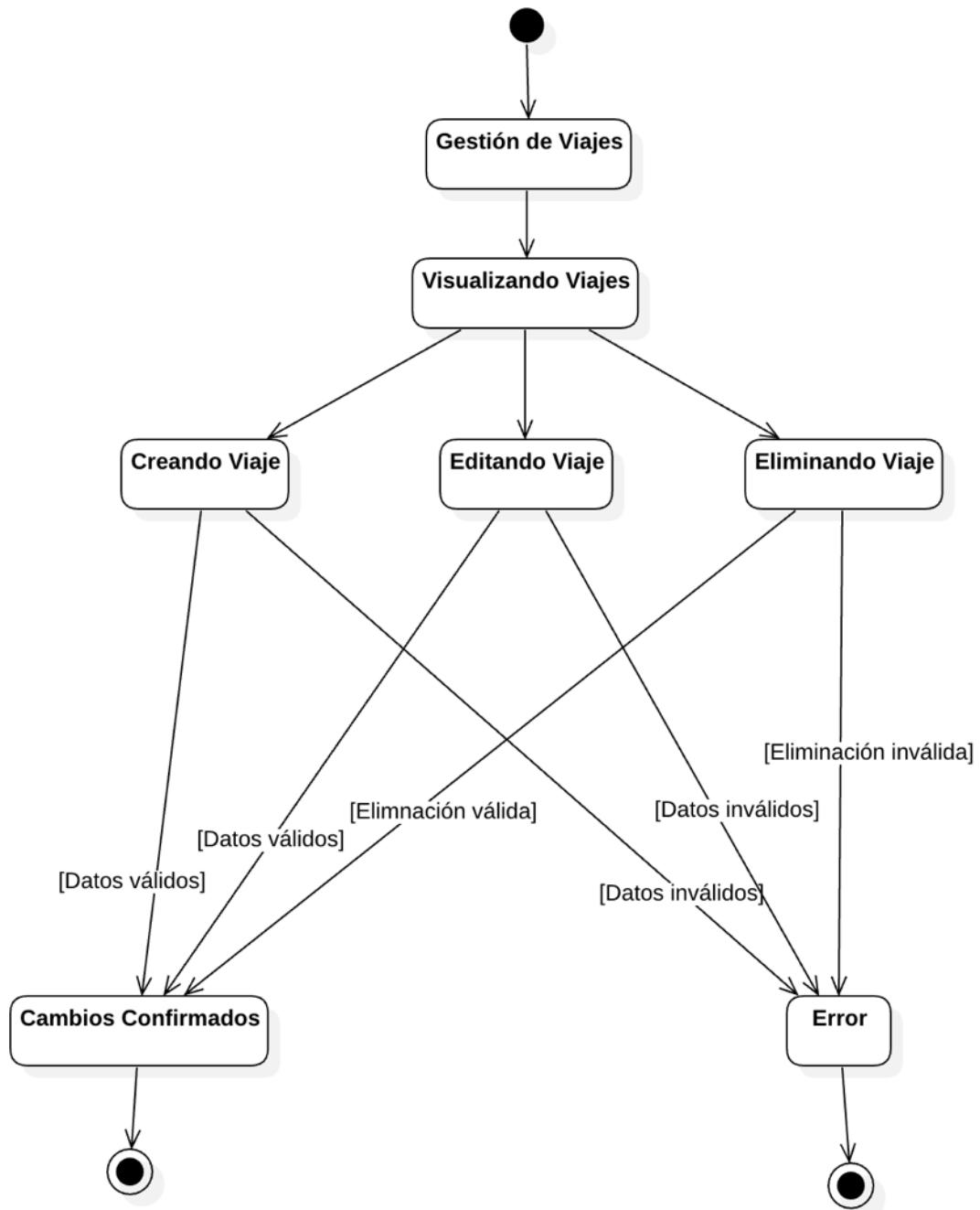


Ilustración 42: Diagrama de Secuencia CRUD Viaje

UC-008	Añadir Participante al Viaje	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-001] Gestión de Usuarios [OBJ-002] Gestión de Viajes [IRQ-001] Gestión de Usuarios [IRQ-002] Información de los Viajes 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera añadir un participante a un viaje.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El participante al que se quiere añadir está registrado en el sistema. Debe de haber un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la opción de participantes.
	2	El actor usuario [ACT-002] selecciona la opción de añadir un participante.
	3	El actor usuario [ACT-002] añade a un usuario por su correo electrónico.
Postcondición	Los cambios quedan almacenados en la base de datos.	
Excepciones	Paso	Acción
	3	Si el usuario participante al que se quiere añadir no está registrado en el sistema, devuelve un error.
Importancia	Media	
Urgencia	Media	
Estabilidad	Muy Alta	

Ilustración 43: Caso de Uso Añadir Participante al Viaje

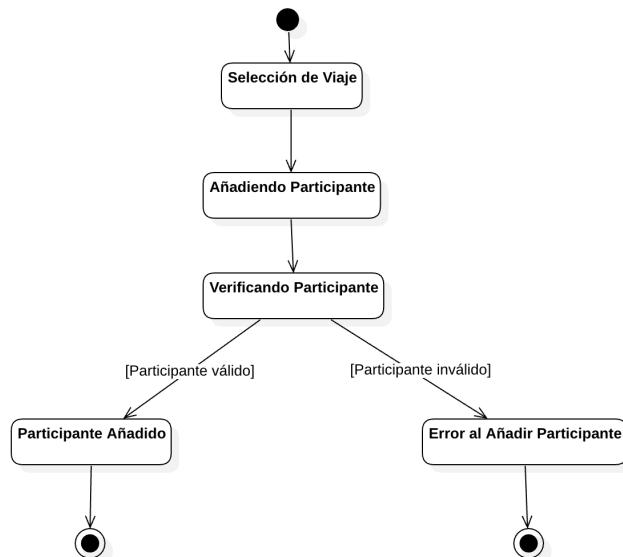


Ilustración 44: Diagrama de Secuencia Añadir Participante al Viaje



UC-009	Eliminar Participante del Viaje	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-001] Gestión de Usuarios[OBJ-002] Gestión de Viajes[IRQ-001] Gestión de Usuarios[IRQ-002] Información de los Viajes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera eliminar un participante de un viaje.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El participante al que se quiere añadir está registrado en el sistema. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la opción de participantes.
	2	El actor usuario [ACT-002] selecciona la opción de eliminar un participante.
	3	El actor usuario [ACT-002] confirma que quiere desvincular a ese usuario del viaje.
Postcondición	Los cambios quedan almacenados en la base de datos.	
Excepciones	No	
Importancia	Media	
Urgencia	Media	
Estabilidad	Muy Alta	

Ilustración 45: Caso de Uso Eliminar Participante del Viaje

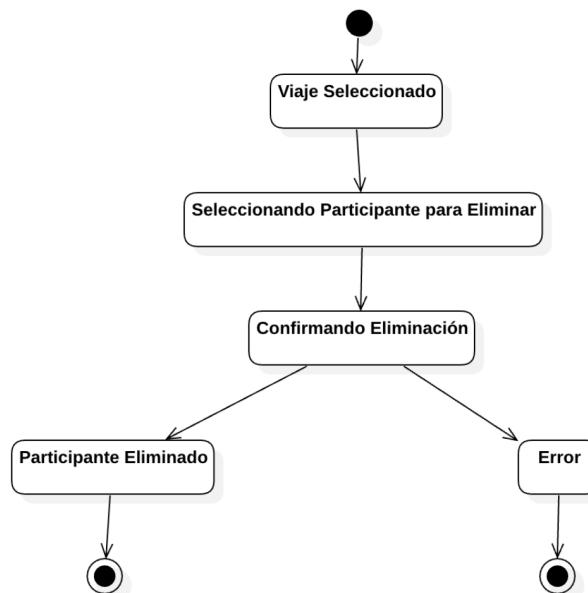


Ilustración 46: Diagrama de Secuencia Eliminar Participante del Viaje

UC-010	CRUD Plan	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-002] Gestión de Viajes [OBJ-004] Gestión de Planes [IRQ-002] Información de los Viajes [IRQ-003] Información de los Planes 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar un plan.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona un tipo de visualización de planes.
	2	El actor usuario [ACT-002] visualiza los planes.
	3.A	El actor usuario [ACT-002] selecciona crear un plan nuevo.
	3.B	El actor usuario [ACT-002] selecciona modificar un plan existente.
	3.C	El actor usuario [ACT-002] selecciona eliminar un plan existente.
	4	El actor usuario [ACT-002] completa los datos.
Postcondición	5	El actor usuario [ACT-002] selecciona la opción confirmar los cambios.
	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
Excepciones	Paso	Acción
	5	Si se encuentra un error en los datos del plan, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 47: Caso de Uso CRUD Plan

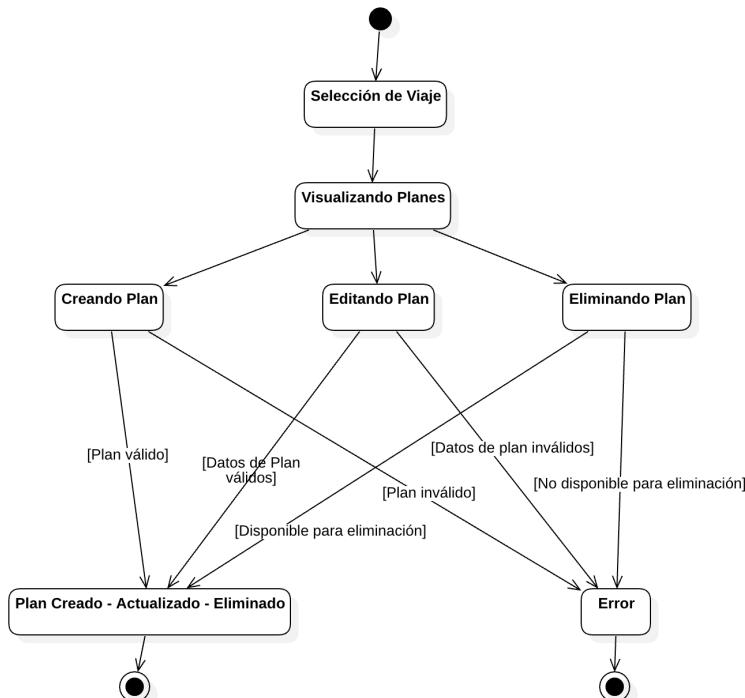


Ilustración 48: Diagrama de Secuencia CRUD Plan

UC-011	Valorar Plan	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-004] Gestión de Planes[IRQ-003] Información de los Planes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera cerrar la sesión iniciada.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona un tipo de visualización de planes.
	2	El actor usuario [ACT-002] visualiza los planes.
	3	El actor usuario [ACT-002] selecciona un plan.
	4	El actor usuario [ACT-002] valora el plan con una puntuación.
Postcondición	No	
Excepciones	No	
Importancia	Media	
Urgencia	Media	
Estabilidad	Muy Alta	

Ilustración 49: Caso de Uso Valorar Plan



Ilustración 50: Diagrama de Secuencia Valorar Plan

UC-012	CRUD Ubicación	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-004] Gestión de Planes [IRQ-003] Información de los Planes 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar una ubicación asociada a un plan.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona un tipo de visualización de planes.
	2	El actor usuario [ACT-002] visualiza los planes.
	3	El actor usuario [ACT-002] selecciona un plan.
	4.A	El actor usuario [ACT-002] selecciona añadir ubicación al plan.
	4.B	El actor usuario [ACT-002] selecciona modificar ubicación del plan.
	4.C	El actor usuario [ACT-002] selecciona eliminar ubicación del plan.
	5	El actor usuario [ACT-002] completa los datos.
Postcondición	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
	Paso	Acción
Excepciones	5	Si se encuentra un error en los datos de la ubicación, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 51: Caso de Uso CRUD Ubicación

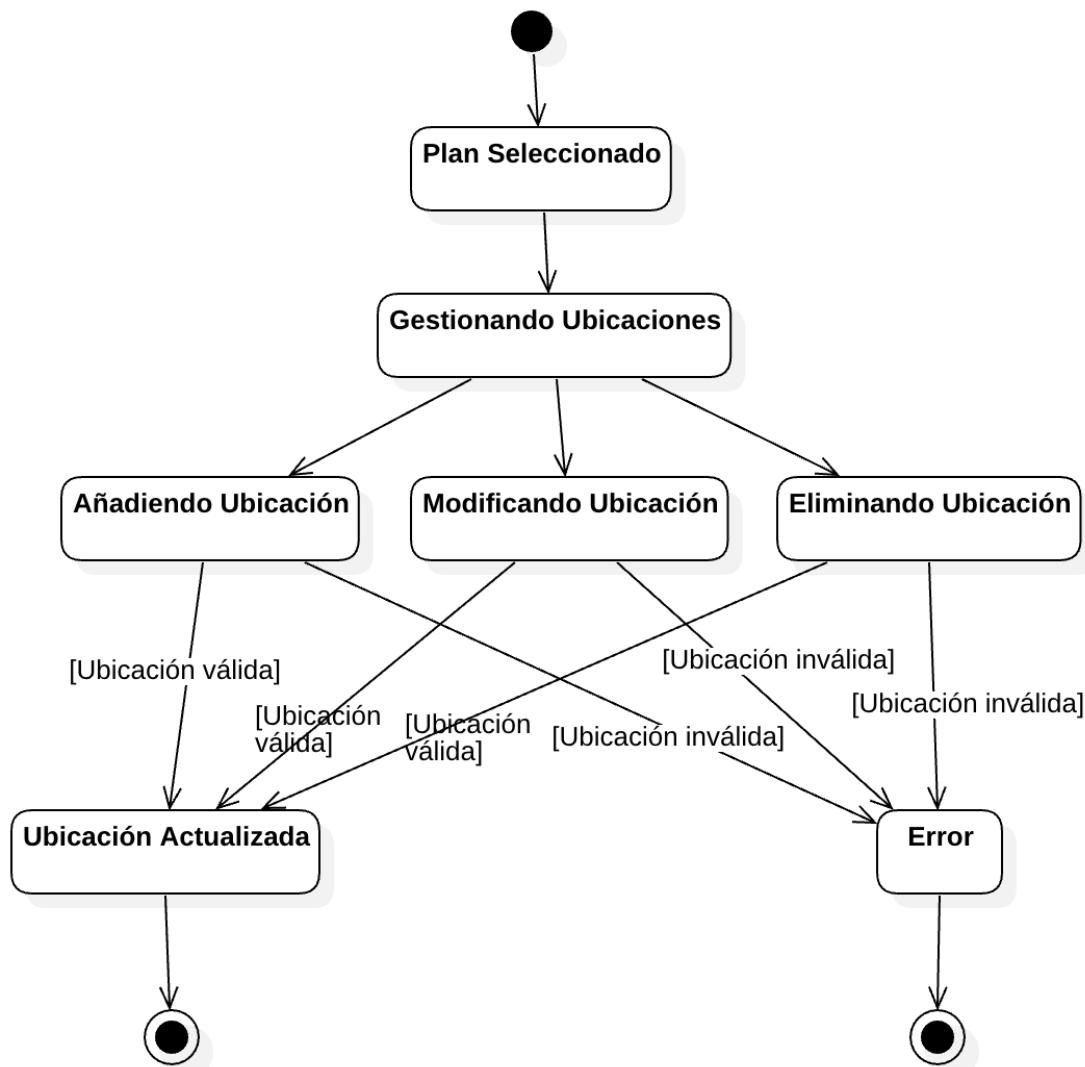


Ilustración 52: Diagrama de Secuencia CRUD Ubicación

UC-013	CRUD Horario	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> • [OBJ-004] Gestión de Planes • [IRQ-003] Información de los Planes 	
Descripción	<p>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar un horario asociado a un plan.</p>	
Precondición	<p>El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.</p>	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona un tipo de visualización de planes.
	2	El actor usuario [ACT-002] visualiza los planes.
	3	El actor usuario [ACT-002] selecciona un plan.
	4.A	El actor usuario [ACT-002] selecciona añadir horario al plan.
	4.B	El actor usuario [ACT-002] selecciona modificar horario del plan.
	4.C	El actor usuario [ACT-002] selecciona eliminar horario del plan.
	5	El actor usuario [ACT-002] completa los datos.
Postcondición	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
	Paso	Acción
Excepciones	5	Si se encuentra un error en los datos del horario, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 53: Caso de Uso CRUD Horario

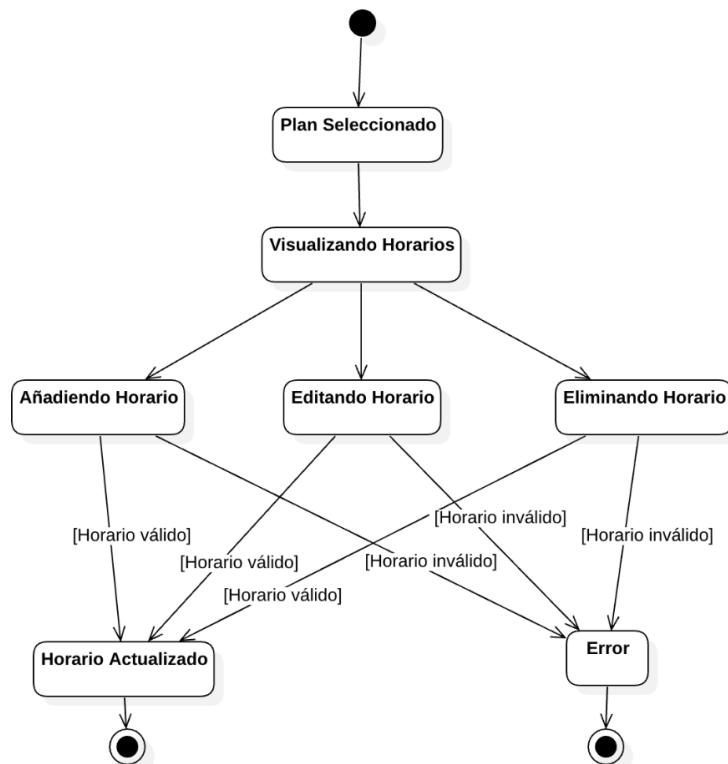


Ilustración 54: Diagrama de Secuencia CRUD Horario

UC-014	CRUD Ticket	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-004] Gestión de Planes[OBJ-005] Gestión de Tickets[IRQ-003] Información de los Planes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar tickets asociado a un plan.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona un tipo de visualización de planes.
	2	El actor usuario [ACT-002] visualiza los planes.
	3	El actor usuario [ACT-002] selecciona un plan.
	4.A	El actor usuario [ACT-002] selecciona añadir ticket al plan.
	4.B	El actor usuario [ACT-002] selecciona modificar ticket del plan.
	4.C	El actor usuario [ACT-002] selecciona eliminar ticket del plan.
	5	El actor usuario [ACT-002] completa los datos.
6		El actor usuario [ACT-002] selecciona la opción confirmar los cambios.

Postcondición	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
Excepciones	Paso	Acción
	5	Si se encuentra un error en los datos del ticket, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 55: Caso de Uso CRUD Ticket

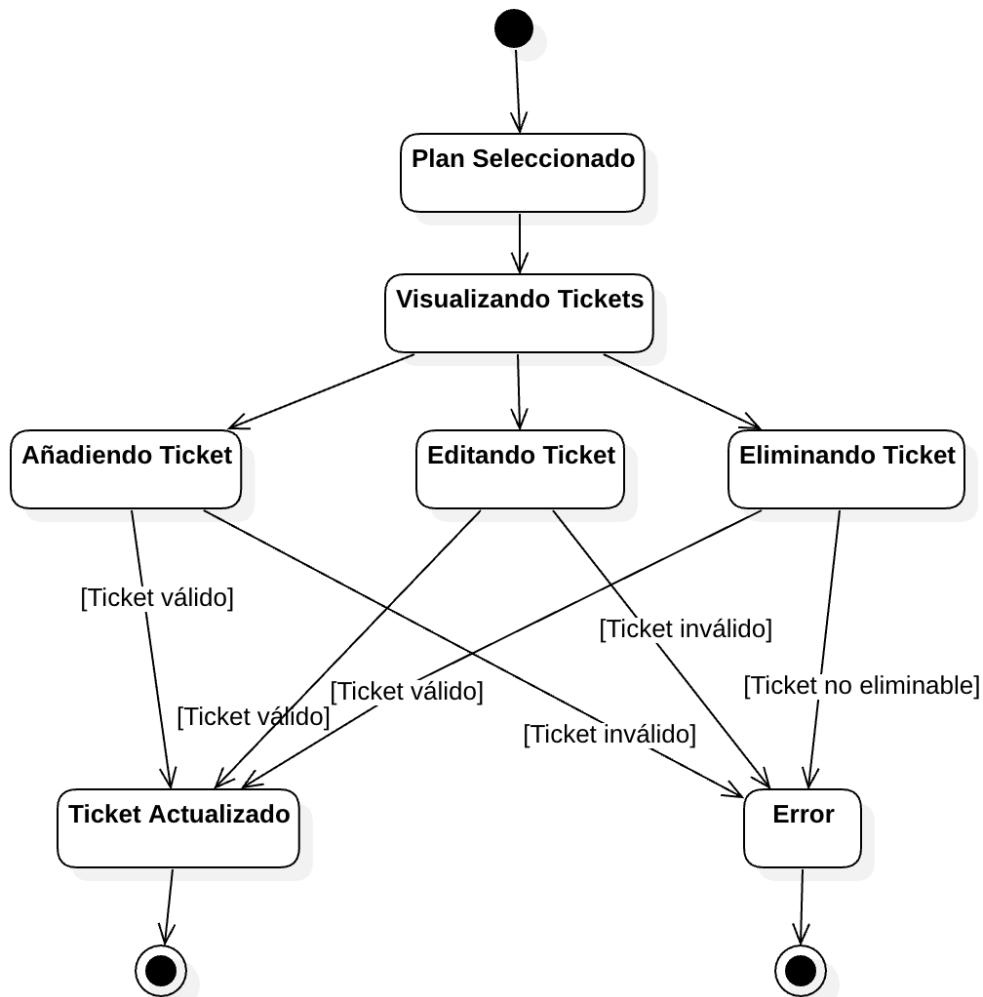


Ilustración 56: Diagrama de Secuencia CRUD Tickets



UC-015	CRUD Pago	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-004] Gestión de Planes[OBJ-006] Gestión de Pagos[IRQ-003] Información de los Planes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar un pago.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona un tipo de visualización de planes.
	2	El actor usuario [ACT-002] visualiza los planes.
	3	El actor usuario [ACT-002] selecciona un plan.
	4.A	El actor usuario [ACT-002] selecciona añadir pago al plan.
	4.B	El actor usuario [ACT-002] selecciona modificar pago del plan.
	4.C	El actor usuario [ACT-002] selecciona eliminar pago del plan.
	5	El actor usuario [ACT-002] completa los datos.
Postcondición	6 El actor usuario [ACT-002] selecciona la opción confirmar los cambios.	
	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
Excepciones	Paso	Acción
	5	Si se encuentra un error en los datos del pago, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 57: Caso de Uso CRUD de Pagos

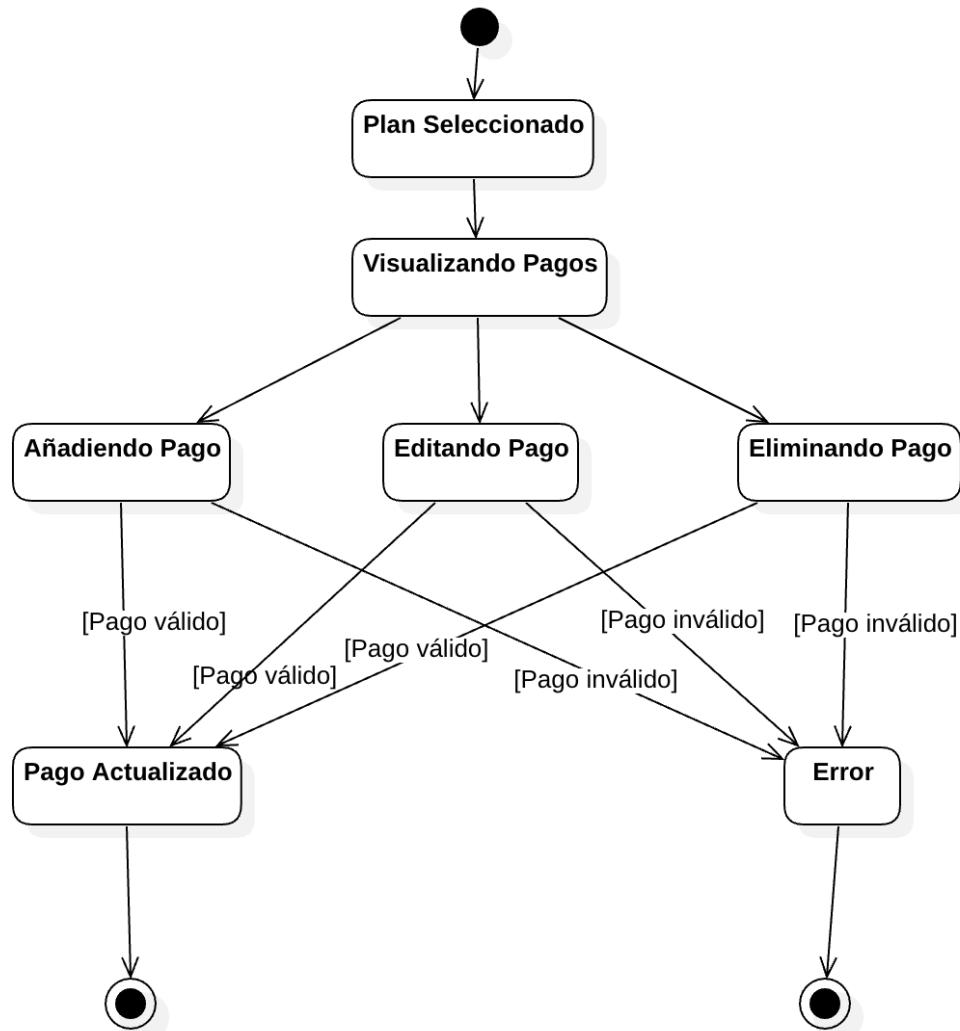


Ilustración 58: Diagrama de Secuencia CRUD Pagos



UC-016	CRUD Propuesta	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-002] Gestión de Propuestas[OBJ-003] Gestión de Viajes[IRQ-002] Información de los Viajes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera crear, visualizar, modificar o eliminar una propuesta.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada. El actor usuario [ACT-002] debe de tener un viaje seleccionado.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la vista de gestión de propuestas.
	2	El actor usuario [ACT-002] visualiza las propuestas.
	3.A	El actor usuario [ACT-002] selecciona crear una propuesta nueva.
	3.B	El actor usuario [ACT-002] selecciona modificar una propuesta existente.
	3.C	El actor usuario [ACT-002] selecciona eliminar una propuesta existente.
	4	El actor usuario [ACT-002] completa los datos.
Postcondición	5 Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
	Paso	Acción
Excepciones	5	Si se encuentra un error en los datos de la propuesta, devuelve un error.
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 59: Caso de Uso CRUD Propuesta

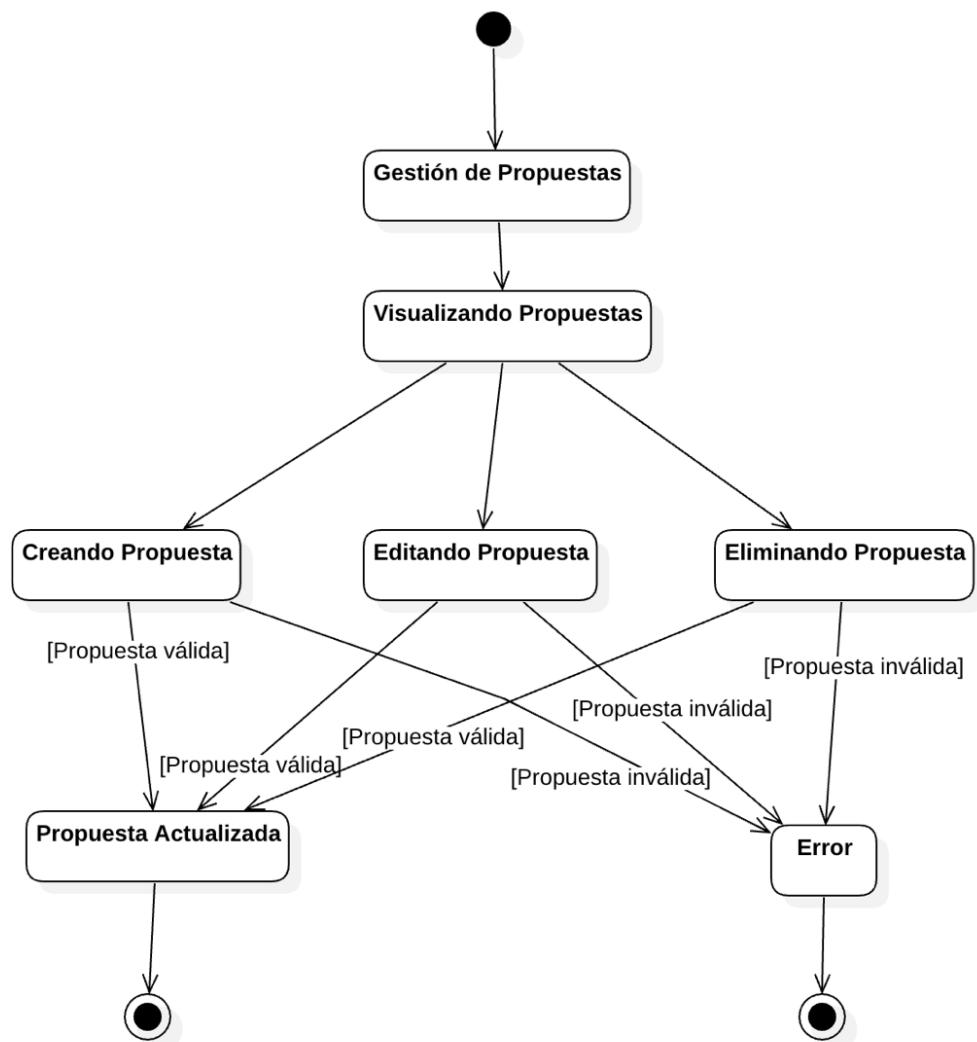


Ilustración 60: Diagrama de Secuencia CRUD Propuesta



UC-017	Valorar Propuesta	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none">[OBJ-002] Gestión de Propuestas[OBJ-003] Gestión de Viajes[IRQ-002] Información de los Viajes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera valorar una propuesta de viaje.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la vista de gestión de propuestas.
	2	El actor usuario [ACT-002] visualiza las propuestas.
	3	El actor usuario [ACT-002] selecciona una propuesta
	4	El actor usuario [ACT-002] valora la propuesta con una puntuación.
Postcondición	Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
Excepciones	No	
Importancia	Alta	
Urgencia	Alta	
Estabilidad	Muy Alta	

Ilustración 61: Caso de Uso Valorar Propuesta

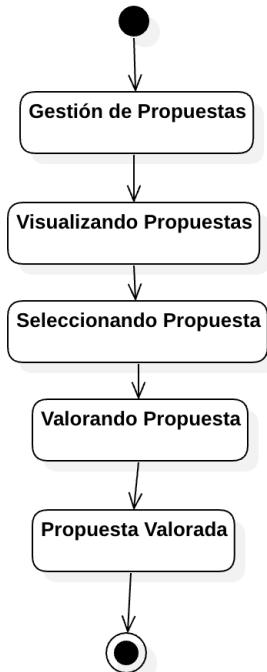


Ilustración 62: Diagrama de Secuencia Valorar Propuesta

UC-018	Marcar Como Finalista	
Versión	1.0 (02/04/2024)	
Autores	Hugo Vázquez Docampo Rodrigo de la Calle Alonso	
Dependencias	<ul style="list-style-type: none"> [OBJ-002] Gestión de Propuestas [OBJ-003] Gestión de Viajes [IRQ-002] Información de los Viajes 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor usuario [ACT-002] quiera marcar una propuesta como finalista del viaje.	
Precondición	El actor usuario [ACT-002] tiene una sesión iniciada.	
Secuencia Normal	Paso	Acción
	1	El actor usuario [ACT-002] selecciona la vista de gestión de propuestas.
	2	El actor usuario [ACT-002] selecciona elegir finalista.
	3	El actor usuario [ACT-002] selecciona una propuesta.
Postcondición	La propuesta queda asociada como ganadora para ese viaje. Los cambios, en caso de que se realicen, quedan almacenados en la base de datos.	
Excepciones	No	
Importancia	Media	
Urgencia	Media	
Estabilidad	Muy Alta	

Ilustración 63: Caso de Uso Marcar Como Finalista



Ilustración 64: Diagrama de Secuencia Marcar como Finalista



2.6. Matriz de Rastreabilidad

Mediante estas matrices se representan las relaciones entre los objetivos y requisitos de información, y entre los objetivos y los casos de uso.

TRM-001	OBJ-001	OBJ-002	OBJ-003	OBJ-004	OBJ-005	OBJ-006
IRQ-001	X	-	-	-	-	-
IRQ-002	-	X	X	-	-	-
IRQ-003	-	-	-	X	Xs	Xs

Ilustración 65: TRM-01

*Xs: subobjetivo

TRM-002	OBJ-001	OBJ-002	OBJ-003	OBJ-004	OBJ-005	OBJ-006
UC-001	X	-	-	-	-	-
UC-002	X	-	-	-	-	-
UC-003	X	-	-	-	-	-
UC-004	X	-	-	-	-	-
UC-005	X	-	-	-	-	-
UC-006	X	-	-	-	-	-
UC-007	-	X	-	-	-	-
UC-008	X	X	-	-	-	-
UC-009	X	X	-	-	-	-
UC-010	-	X	-	X	-	-
UC-011	-	-	-	X	-	-
UC-012	-	-	-	X	-	-
UC-013	-	-	-	X	-	-
UC-014	-	-	-	X	X	-
UC-015	-	-	-	X	-	X
UC-016	-	-	X	X	-	-
UC-017	-	-	X	X	-	-
UC-018	-	-	X	X	-	-

Ilustración 66: TRM-02



CAPÍTULO 3: DISEÑO DEL SISTEMA

En esta sección se expone el enfoque del diseño técnico del sistema.

3.1. Diagrama de Arquitectura

El diagrama de arquitectura proporciona una visión de alto nivel de la estructura global del sistema. Define la disposición y la interacción de los componentes de software, las capas de la aplicación y la infraestructura tecnológica. A través del siguiente diagrama se busca comprender la organización lógica del sistema:

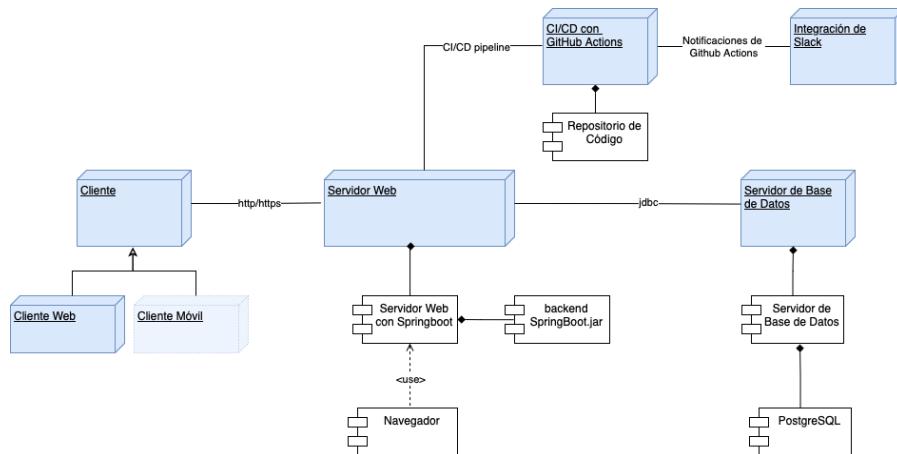


Ilustración 67: Diagrama de Arquitectura del Sistema

3.2. Diagrama de Entidad Relación

Un diagrama de entidad-relación es una herramienta visual utilizada en el campo del modelado de datos y la ingeniería de software para representar la estructura lógica de una base de datos. Este diagrama ilustra las entidades relevantes dentro del sistema y las relaciones entre ellas:

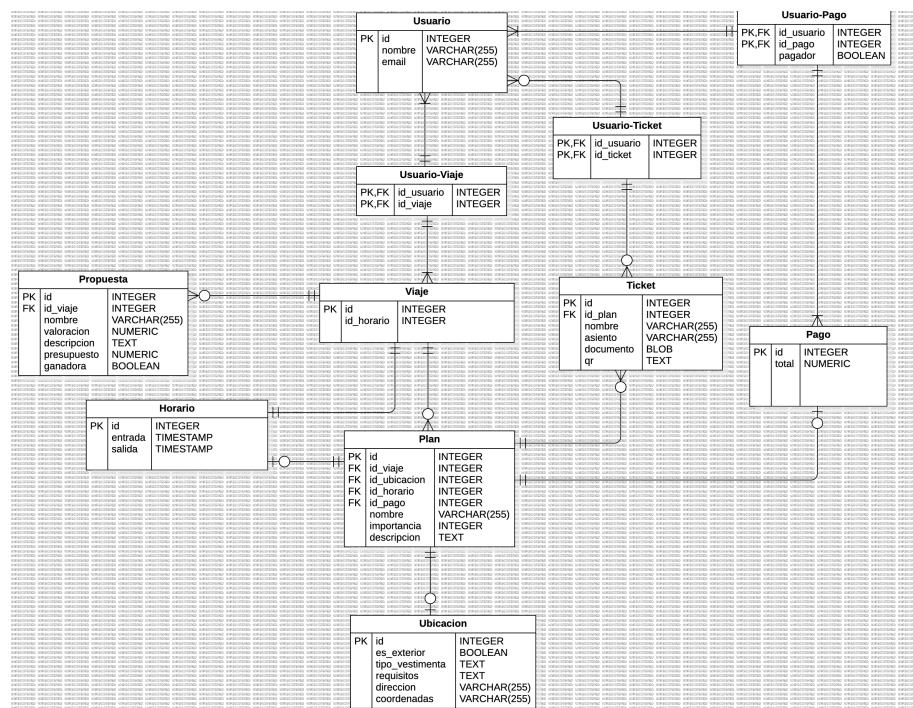


Ilustración 68 Diagrama de entidad relación

3.3. Diagrama de Secuencia

En este apartado se muestran los diagramas de secuencia de algunos de los casos de uso anteriores.

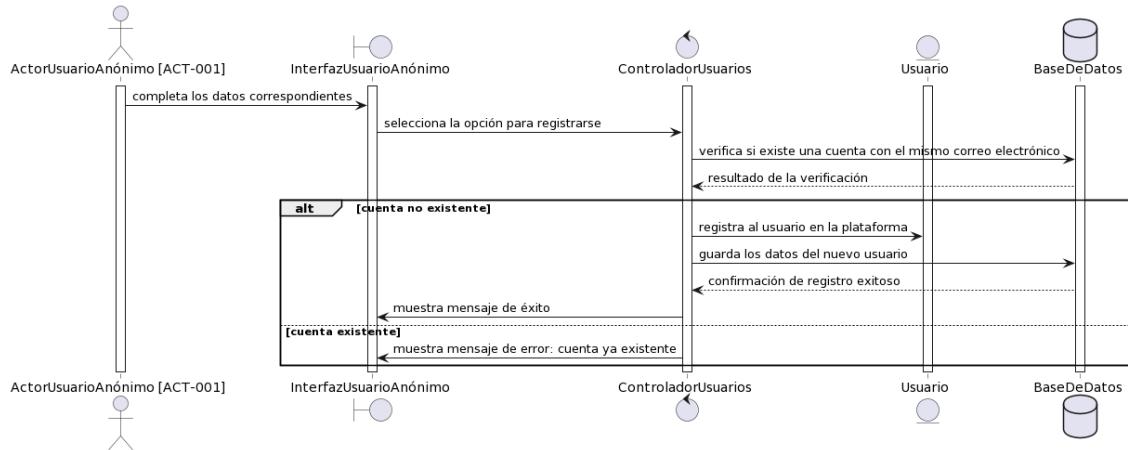


Ilustración 69: Diagrama de secuencia Registrarse

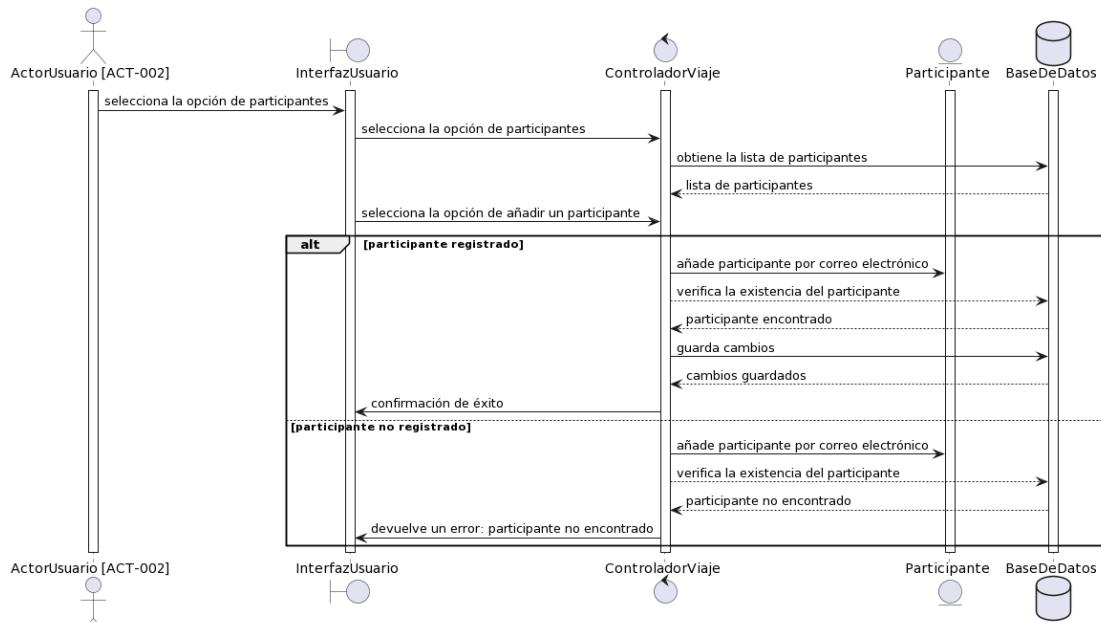


Ilustración 70: Diagrama de Secuencia Añadir participante al viaje

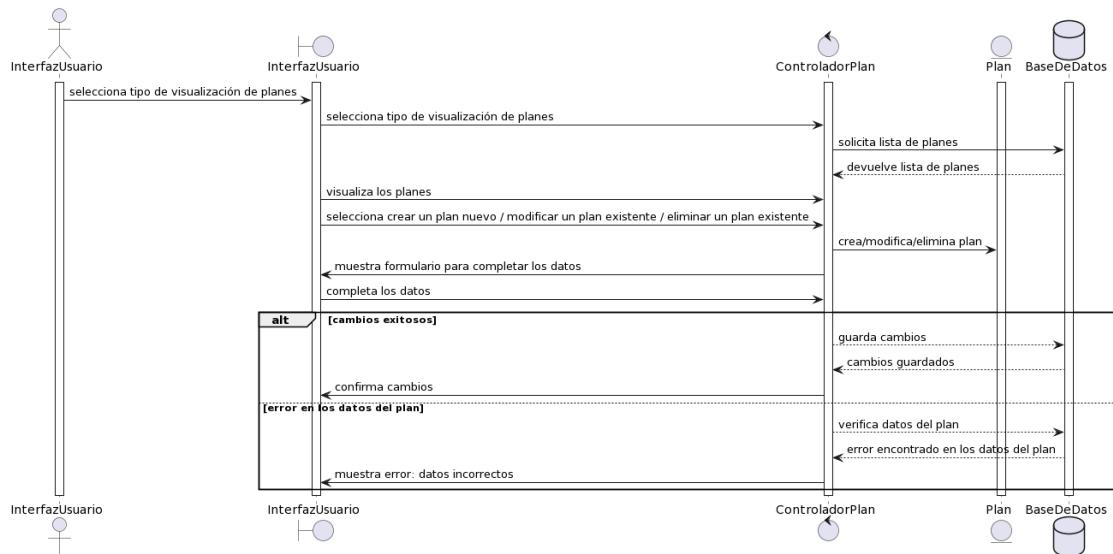


Ilustración 71: Diagrama de Secuencia CRUD Plan

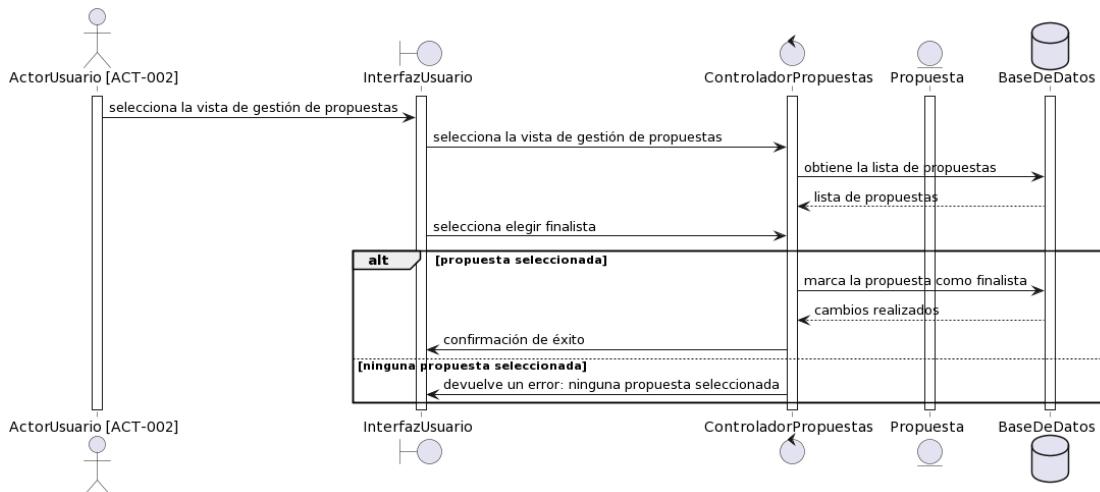


Ilustración 72 Diagrama de Secuencia Marcar Como Finalista

3.4. Prototipo de Interfaz de Usuario

En este apartado se pueden ver los primeros bocetos de las interfaces de usuario de la aplicación. Se muestran las distintas interfaces para visualizar los planes, gestionar propuestas y viajes.

Está diseñado en formato móvil porque, aunque sea web, está pensado para ser utilizado en móviles principalmente. Más adelante se añadirán un diseño *responsive* para navegadores web de escritorio.

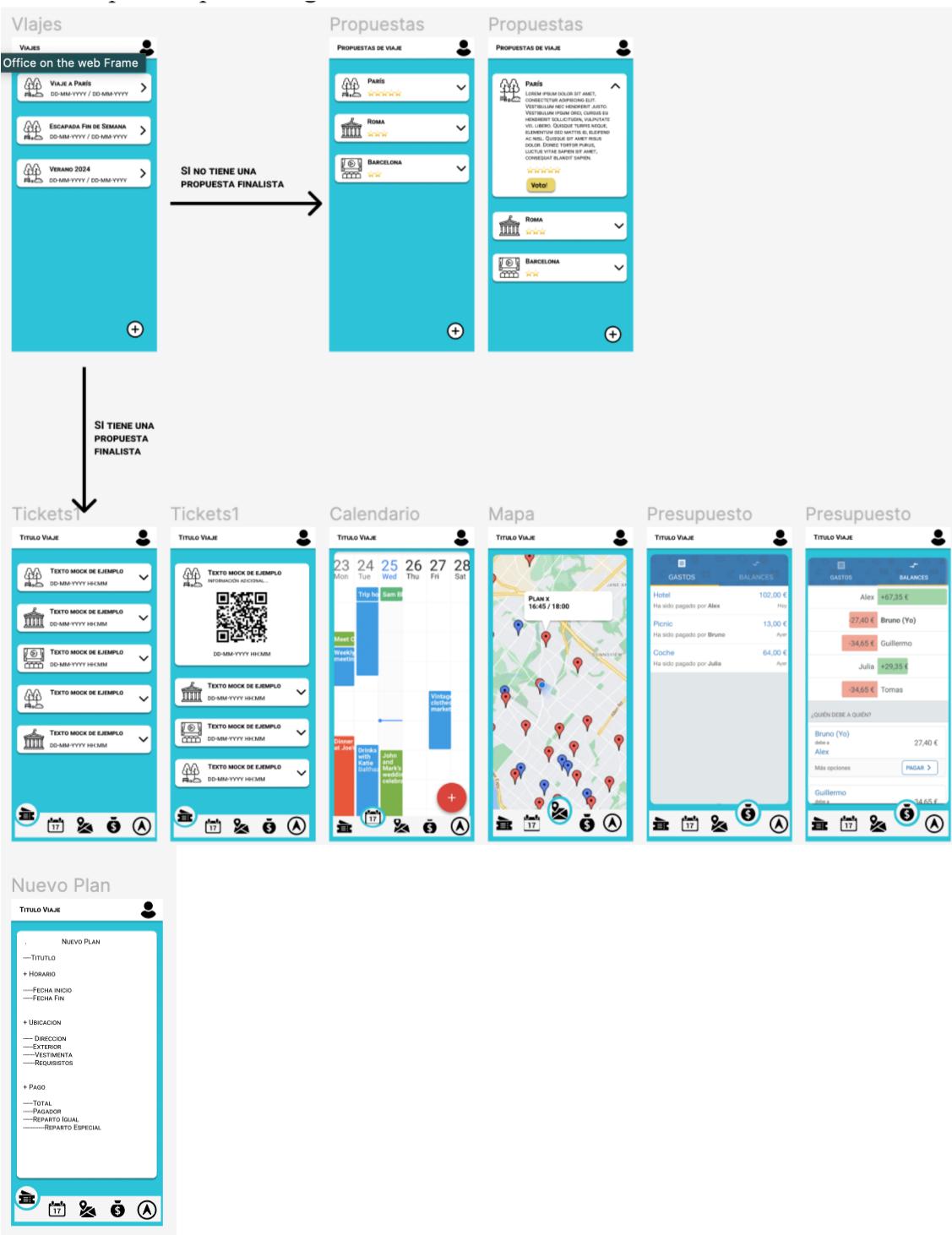


Ilustración 73: Prototipo de Interfaz de Usuario



3.5. Diseño UI/UX

El diseño de la interfaz de usuario se ha definido con el objetivo de promover la usabilidad y ofrecer una experiencia intuitiva y agradable. Se ha dado prioridad a una navegación fluida y a la inclusión de iconos fácilmente reconocibles.

La elección de la paleta de colores, mediante la herramienta Coolors, ha desembocado en una combinación vistosa y llamativa mostrada a continuación:



Ilustración 74: Selección de colores para la aplicación

Además, se ha optado por llevar a cabo un diseño responsive, incluyendo elementos diferentes en función de los tamaños de la pantalla: campos que se muestran o no según el dispositivo, menú lateral para ordenadores e inferior para móviles... Todos estos aspectos se podrán observar en el capítulo de implementación.



CAPÍTULO 4: IMPLEMENTACIÓN

En este capítulo se redactan los aspectos fundamentales del proceso de implementación, tales como las librerías, lenguajes, IDEs, hitos destacables...

Se puede acceder a la aplicación desplegada a través de este [enlace](#). Cabe destacar, que debido a la naturaleza del servidor gratuito, el back permanece en reposo si no hay solicitudes y por ello tarda unos dos minutos en levantarse.

4.1. Entorno de desarrollo

Para poder llevar a cabo el desarrollo de la forma más eficiente posible, se ha hecho uso de una serie de herramientas de trabajo para facilitar el día a día. Estas han sido:

- **IntelliJ Ultimate:** ha sido el IDE escogido para el desarrollo del *backend* [4] pues, para esta parte escogimos como lenguaje Java con Spring Boot [7]. Este entorno se adapta de forma excelente a las características de este Framework.
- **VSCode:** por otra parte, para la implementación del *frontend* [3], se ha decidido hacer uso de este IDE, pues es muy ligero y presenta muchos *plugins* de utilidad.
- **SourceTree:** para gestionar de forma más eficiente y visual los repositorios de GitHub [9], se ha optado por emplear esta herramienta.
- **Dbeaver y Neon Database:** en relación con la base de datos, se ha optado por alojarla en la nube para posteriormente llevar a cabo los despliegues de manera rápida. De este modo, se ha escogido DBeaver como Gestor de Base de datos para poder ver, editar y gestionar todos los aspectos relacionados con esta.
- **Slack:** para mantenernos informados del estado del código, se ha definido una acción de GitHub [9] para que envíe notificaciones vía Slack del estado de los commits:

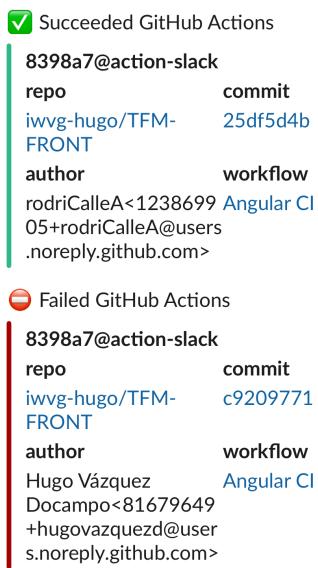


Ilustración 75: Ejemplo de acción de GitHub

- **Netlify y Render Dashboard:** son dos herramientas online para gestión de despliegues gratuitos. La primera de ellas permite alojar proyectos de *frontend* [3]. Por otro lado, la segunda, pese a requerir contenedORIZAR el proyecto, permite desplegar de forma gratuita el *backend* [4] de la aplicación.
- **Figma:** ha sido la herramienta escogida para realizar el prototipo de interfaz.

4.2. Semilla de la aplicación

Para poder facilitar el desarrollo, se han incluido una serie de herramientas en la semilla de la aplicación:

4.2.1. Liquibase

Para una rápida gestión de las BBDD, se hace uso de Liquibase que ejecuta los scripts recogidos en la carpeta **changelog** de forma automática. Esto facilita la migración entre entornos, pues al ejecutar la aplicación en el entorno deseado, cargará todos los changesets definidos, generando la base de datos desde cero.

Además, también aporta un valor añadido a los test-IT de la aplicación, pues genera una base de datos en tiempo real para la ejecución de las pruebas, permitiendo un mayor grado de fiabilidad a la hora de realizar las pruebas pues no se hace uso de los datos que se emplean en el entorno de desarrollo.

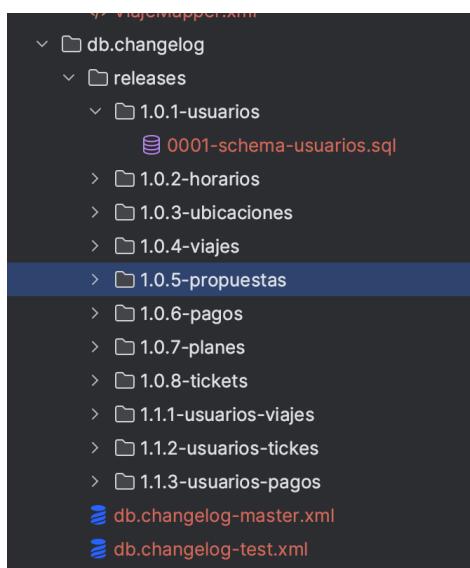


Ilustración 76: estructura scripts Liquibase

4.2.2. SonarCloud

Para facilitar la gestión de la calidad del software, se han incluido los proyectos de *frontend* [3] y de *backend* [4] en SonarCloud [10]; de forma que se generan, con cada subida, informes de la calidad del código desarrollado (*codesmells*, cobertura, security *hotspots*...)

4.2.3. GitHub Workflows

Para automatizar procesos, se han definido una serie de Jobs en GitHub [9] que se ejecutarán con cada *commit*:

- **Checkout:** para clonar el repositorio con una profundidad de *fetch* configurada a 0, permitiendo obtener todo el historial.
- **Set up Node.js:** configura el entorno de ejecución.
- **Install dependencies:** ejecuta npm install para instalar las dependencias necesarias para el proyecto.
- **Build:** realiza la construcción del proyecto ejecutando `npm run build -prod`, configurando el entorno para producción.



- **SonarCloud Scan [10]:** Utiliza SonarSource/sonarcloud-github-action@master para realizar un análisis estático del código con SonarCloud [10], configurado con variables de entorno para el token, clave del proyecto y organización, además de argumentos específicos para Sonar [10].
- **Slack WebHook:** Configura una notificación de Slack para ser enviada siempre que se complete el *job*, independientemente de su resultado.

```
Code Blame 46 lines (45 loc) · 1.18 KB
1 name: Angular CI
2
3 on:
4   push:
5     branches:
6       - main
7       - '*-{8-9}'
8   pull_request:
9     branches:
10      - main
11      - '*-{8-9}'
12
13 jobs:
14   build-angular:
15     runs-on: ubuntu-latest
16     steps:
17       - uses: actions/checkout@v2
18         with: {fetch-depth: 0}
19       - name: Set up Node.js
20         uses: actions/setup-node@v2
21         with:
22           node-version: 18
23       - name: Install dependencies
24         run: npm install
25         working-directory: ./TripPlanner/
26       - name: Build
27         run: npm run build --prod
28         working-directory: ./TripPlanner/
29       - name: SonarCloud Scan
30         uses: SonarSource/sonarcloud-github-action@master
31         env:
32           SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
33           SONAR_PROJECT_KEY: 'iwvg-hugo_FM-FRONT'
34           SONAR_ORGANIZATION: 'iwvg-hugo'
35         with:
36           projectBaseDir: './'
37           args:
38             -Dsonar.projectKey=${{ env.SONAR_PROJECT_KEY }}
39             -Dsonar.organization=${{ env.SONAR_ORGANIZATION }}
40       - name: Slack WebHook
41         if: always()
42         uses: 8398a7/action-slackv3
43         with:
44           status: ${{ job.status }}
45           fields: 'repo, author, commit, workflow'
46           env:
47             SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

Ilustración 77: WorkFlows de GitHub

4.2.4. MyBatis

Es utilizado en este proyecto para gestionar la interacción entre la aplicación y la base de datos de manera eficiente. Ofrece una ventaja significativa al permitir un mapeo personalizado de instrucciones SQL a objetos Java, lo que simplifica el desarrollo.

```
<select id="createViaje" parameterType="com.miw.tripplanner.dtos.ViajeDto">
    INSERT INTO viajes (id_horario)
    VALUES (#{idHorario})
    RETURNING id
</select>
```

Ilustración 78: Ejemplo de consulta con myBatis

4.2.5. Lombok

Se utiliza en el proyecto para minimizar el código repetitivo en Java, especialmente para modelos y datos de acceso. Proporciona una ventaja considerable al generar automáticamente métodos como *getters*, *setters* y constructores mediante anotaciones simples, lo que aumenta la legibilidad y reduce la posibilidad de errores.

```
@Data
@NoArgsConstructor
@EqualsAndHashCode(callSuper = false)
public class HorarioDto {
    private Integer id;
    private Timestamp inicio;
    private Timestamp fin;
}
```

Ilustración 79: Ejemplo de uso Lombok

4.2.6. Orval

Utilizado para generar automáticamente modelos y servicios en Angular [6] a partir de una especificación Swagger [11]. La ventaja de Orval [1] radica en su capacidad para sincronizar automáticamente los cambios en la API con el *frontend* [3], asegurando que las interfaces de usuario estén siempre actualizadas con las últimas definiciones de la API.

4.2.6. Swagger UI

Se emplea para crear una interfaz visual interactiva de la API del proyecto. Su principal ventaja es que permite interactuar con la API directamente desde el navegador, facilitando la comprensión y la prueba de los *endpoints* sin necesidad de herramientas adicionales.

4.4. Diseño MVC

El patrón Modelo Vista Controlador (MVC) es una arquitectura de software utilizada comúnmente para la creación de interfaces de usuario. Divide una aplicación en tres componentes interrelacionados: el modelo, la vista y el controlador. Esta separación permite el manejo eficiente de la lógica de la aplicación, la presentación de los datos y la interacción del usuario, mejorando la modularidad y facilitando la gestión del código.

Interacción en MVC:

1. **Usuario interactúa con la Vista:** La interacción del usuario con la vista (por ejemplo, presionar un botón) envía una solicitud al controlador.
2. **Controlador maneja la solicitud:** El controlador procesa la solicitud y decide la acción a tomar, como actualizar el modelo o cambiar la vista.
3. **Modelo se actualiza:** Si la acción del controlador involucra cambios en los datos, el modelo se actualiza en consecuencia.
4. **Vista refleja el modelo actualizado:** La vista se actualiza para reflejar los nuevos datos del modelo, proporcionando retroalimentación al usuario.

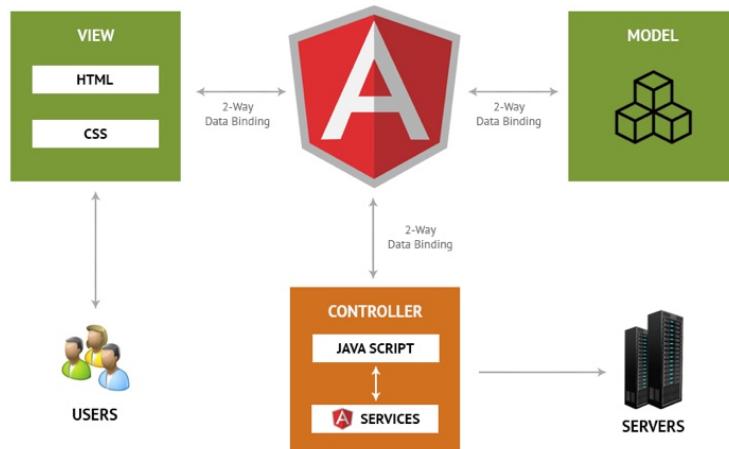


Ilustración 80 Modelo Vista Controlador [13]



4.5. Calidad de software

La integración de SonarCloud [10], ha permitido poder llevar a cabo un desarrollo que respeta los estándares generales de la programación tales como:

- Código bien formateado.
- Eliminar código duplicado.
- Brechas de seguridad tales como usuarios y contraseñas forzadas.
- Código no usado.
- Reducción de la complejidad del código (ciclomática y cognitiva)
- Formato correcto en el nombre de las variables (en este caso camelCase)
- Reducir el número de comentarios y espacios en blanco
- Métodos con baja cohesión, alto acoplamiento y de tamaño pequeño.
- Respetar el número recomendable de argumentos (2-5), líneas y métodos por clase
- Principio de única responsabilidad
- Versiones de librerías actualizadas.

4.6. Pantalla responsive

Como se ha mencionado en otras secciones del documento, se ha llevado a cabo un diseño *responsive* de la aplicación por lo que ha sido necesario añadir lógica adicional para validar que la visibilidad de las pantallas es correcta en todos los formatos de dispositivo.

A continuación, se reflejan algunos ejemplos:

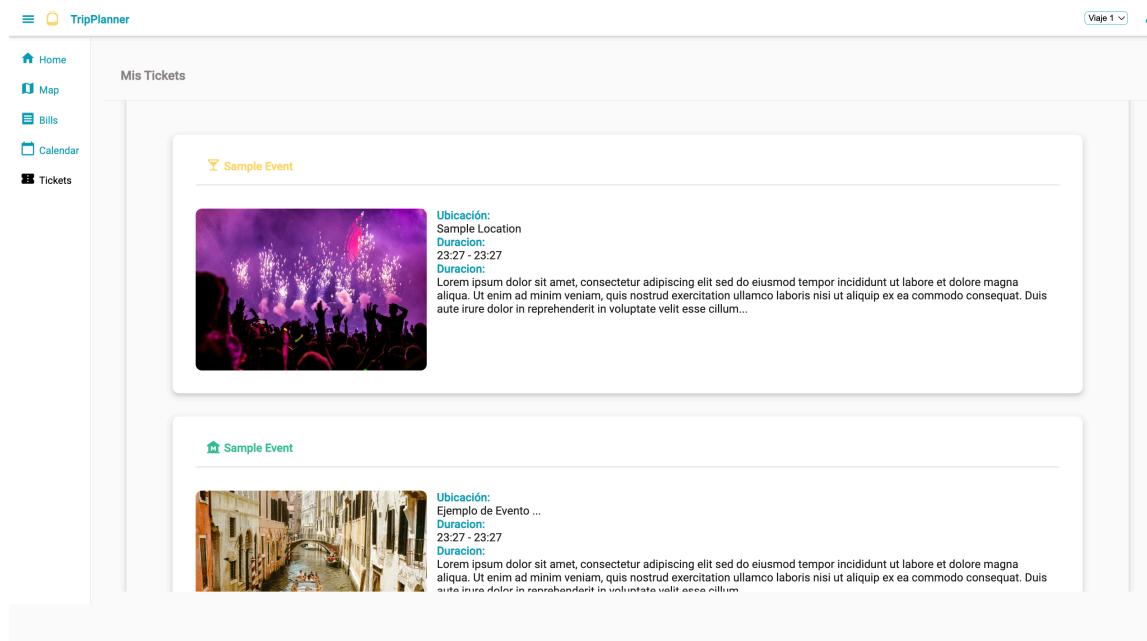


Ilustración 81: Pantalla de tickets (ordenador)

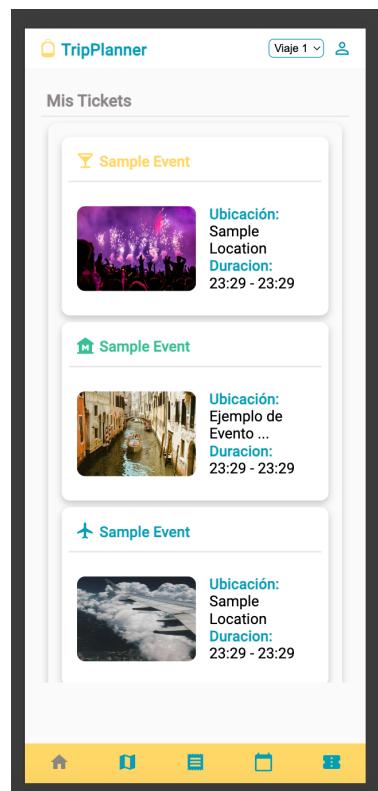


Ilustración 82: Pantalla de tickets (móvil)



En las imágenes anteriores se puede observar cómo según el dispositivo a utilizar se muestran unos datos u otros; cambiando el menú, los campos visibles...

A nivel de código, se hace uso de directivas como las siguientes:

```
</div>
<div class="apartado" *ngIf="isDesktop">
    <p style="font-weight: bold; color: #009fb7; margin: 0 !important;">Duracion:</p>
    <p style="margin: 0 !important;">{{ ticket.description.length >300 ? ticket.description.slice(0,300) + '...' }}</p>
</div>
```

```
constructor(private router: Router, private renderer: Renderer2) {
    this.isDesktop = window.innerWidth >= 768;
    window.onresize = () => {
        this.isDesktop = window.innerWidth >= 768;
    };
}
```

Ilustración 83: fragmento de código responsive

- Gracias al uso de ***ngIf** y de **Renderer2** se muestra el campo descripción solo en caso de que sea un dispositivo con pantalla grande.

```
@media screen and (max-width: 768px) {
    .foto {
        margin-top: 2vw;
        width: 35vw;
        border-radius: 10px;
        height: 25vw;
    }

    .info{
        padding: 20px 0 20px 10px;
    }

    .cabecera {
        padding: 2vw;
    }
}
```

Ilustración 84: Fragmento de estilos responsive

- Por otra parte, para sobrescribir ciertos estilos, se hace uso de **@media screen**.

Otro ejemplo de tratamiento responsivo se define en la pantalla de creación de viaje, en donde se establece la orientación del *stepper* en función del tipo de dispositivo (vertical en móviles y horizontal en ordenadores):

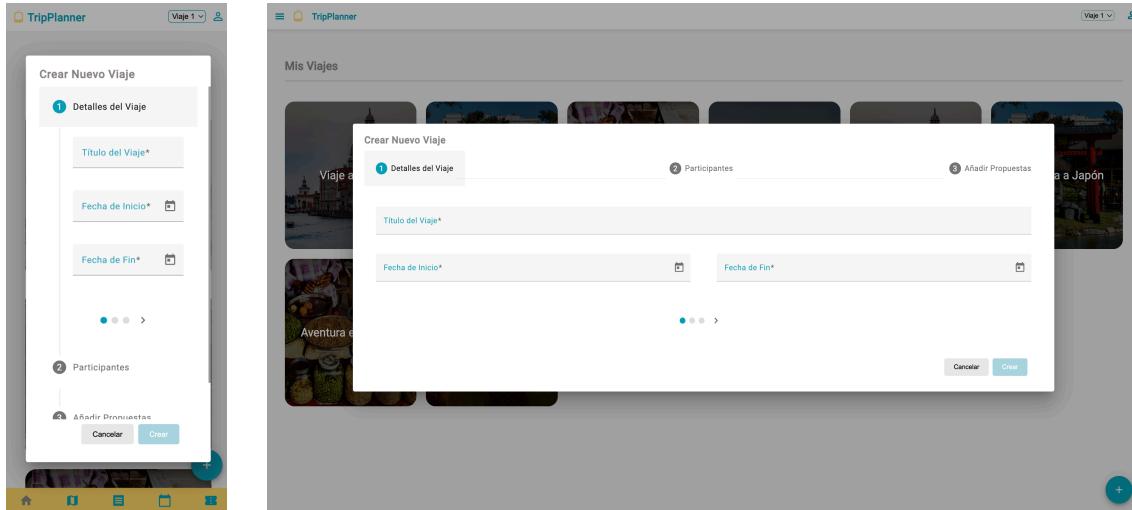


Ilustración 85: Pantalla responsive

```
<div mat-dialog-content>
  <mat-stepper #stepper [linear]="true" [orientation]="(stepperOrientation | async)!">
    <!-- Step 1: Detalles del Viaje -->
    <mat-step [stepControl]="firstFormGroup">
```

```
  this stepperOrientation = breakpointObserver
    .observe('min-width: 800px')
    .pipe(map({ matches }) => (matches ? 'horizontal' : 'vertical'));
```

Ilustración 86: Componente responsive



CAPÍTULO 5: PRUEBAS

Este capítulo se busca detallar y analizar las pruebas realizadas en el desarrollo del proyecto, enfocándose en asegurar la calidad y robustez del código. Las pruebas son esenciales para garantizar que las funcionalidades implementadas se comporten correctamente bajo diferentes escenarios y que la integración con otros componentes del sistema, como el *frontend* [3], sea exitosa y libre de errores.

5.1. Pruebas de DTOs

Una parte importante de las pruebas incluye el uso de Equals Verifier para testear los objetos de transferencia de datos (DTO). Equals Verifier es una herramienta que ayuda a garantizar que los métodos equals y hashCode en los DTOs estén correctamente implementados.

Esto es fundamental, ya que asegura que los objetos se comparan de manera adecuada, lo cual es vital para el procesamiento correcto de datos a través de las diferentes capas de la aplicación.

```
new *  
@Test  
void testEqualsAndHashCode() {  
    EqualsVerifier.simple().forClass(HorarioDto.class)  
        .withRedefinedSuperclass()  
        .verify();  
}
```

Ilustración 87: Ejemplo de test de DTO

5.2. Pruebas de integración con la base de datos

Se realizaron pruebas de integración que implican la base de datos, la cual se regenera en cada ejecución de prueba gracias al uso de Liquibase [5]. Este enfoque garantiza que el código interactúe correctamente con la base de datos bajo condiciones controladas, permitiendo debuggear cada prueba de manera efectiva.

Estas pruebas son fundamentales para verificar que la integración con el *frontend* [3] sea fiable, dado que el *backend* [4] está muy probado y se confirma que los datos recuperados y manipulados son los correctos para cada método del controlador.

A continuación, se incluye un ejemplo de prueba de integración, para mostrar cómo se llama al *endpoint* deseado y se valida que los datos recuperados son los esperados:



```
class ViajeControllerTestIT extends BaseTest {
    9 usages
    private ObjectMapper mapper = new ObjectMapper();

    @Autowired
    private ViajeController viajeController;

    @Test
    void testGetViajes() throws Exception {
        List<ViajeDto> response = new ArrayList<>();

        MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders
            .get(urlTemplate: "/viajes")
            .contentType(MediaType.APPLICATION_JSON).content(mapper.writeValueAsString(null));

        ResultActions ra = mockMvc.perform(requestBuilder);

        ra.andExpect(MockMvcResultMatchers.status().isOk());

        // Deserializar directamente a una lista de ViajeDto
        List<ViajeDto> viajes = mapper.readValue(ra.andReturn().getResponse().getContentAsString(),
            new TypeReference<List<ViajeDto>>() {
            });
        response = viajes;
        assertNotNull(response);
        assertEquals(expected: 1, response.size());
        assertEquals(expected: 9999, response.get(0).getId());
        assertEquals(expected: 9999, response.get(0).getIdHorario());
    }
}
```

Ilustración 88: Ejemplo Test-IT

Como se observa, solo se recupera un viaje en la llamada al método de recuperar todos los viajes porque en el Liquibase de inserción de datos de test solo existe una entrada de viajes:

```
--changeset tripPlanner:0003-data-viajes

insert into viajes (id, id_horario)
values (9999, 9999);
```

Ilustración 89: insert Liquibase viajes

5.3. Cobertura de pruebas y análisis de la calidad de código

La cobertura de pruebas ha sido ampliamente satisfactoria, alcanzando los objetivos de cobertura establecidos sin presentar *security hotspots*. De igual modo, el número de *Code Smells* se encuentra dentro de los parámetros permitidos.

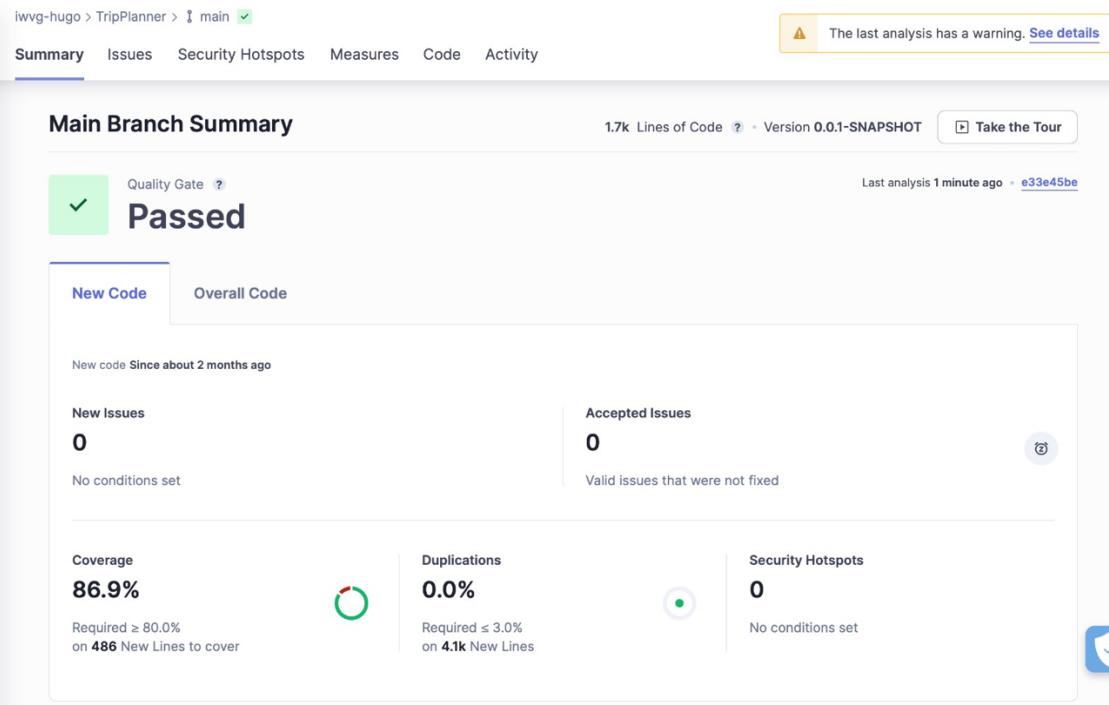


Ilustración 90: Análisis de Calidad - SonarCloud



CAPÍTULO 6: GESTIÓN DEL PROYECTO

Esta sección aborda la organización temporal de las tareas necesarias para el desarrollo del proyecto. Se proporcionará una visión global del tiempo estimado para cada fase.

Siguiendo la metodología RUP, se dividirá el proyecto en cuatro fases: Inicio, Elaboración, Construcción y Transición; cada una de ellas de una duración diferente como se observa en el cronograma recogido en la siguiente sección.

Para la fase de transición, consideraremos como entrega y problemas que mejorar aquellos aspectos observados por el otro miembro de la pareja de trabajo que considera mejorables y/o incompletos.

6.1. Cronograma

El cronograma es una herramienta esencial que detalla el calendario del proyecto, marcando las fechas de inicio y fin. A continuación, se recoge el cronograma asociado a este proyecto teniendo en cuenta que la metodología de trabajo empleada es RUP:

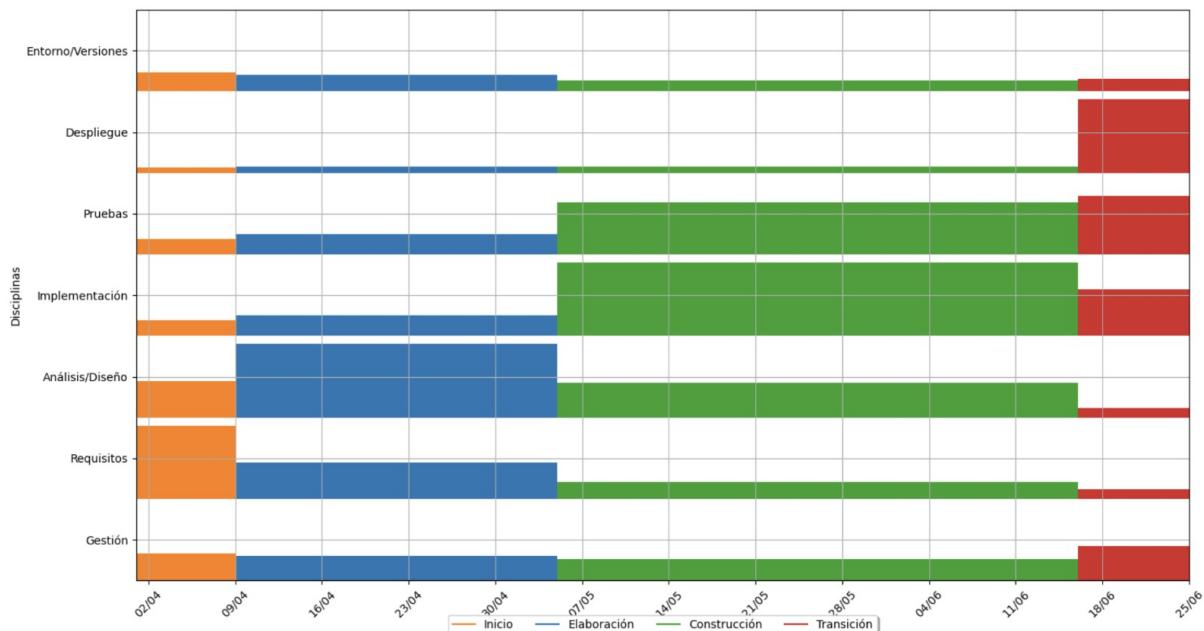


Ilustración 91: Cronograma del Proyecto



6.2. EDT (Estructura de Desglose de Trabajo)

La Estructura de Descomposición del Trabajo (EDT) es un desglose jerárquico del alcance del proyecto. Aquí se descompone el trabajo en partes más pequeñas y manejables, lo que facilita la organización, la programación y la asignación de responsabilidades. La EDT está muy relacionada con el cronograma anteriormente definido:

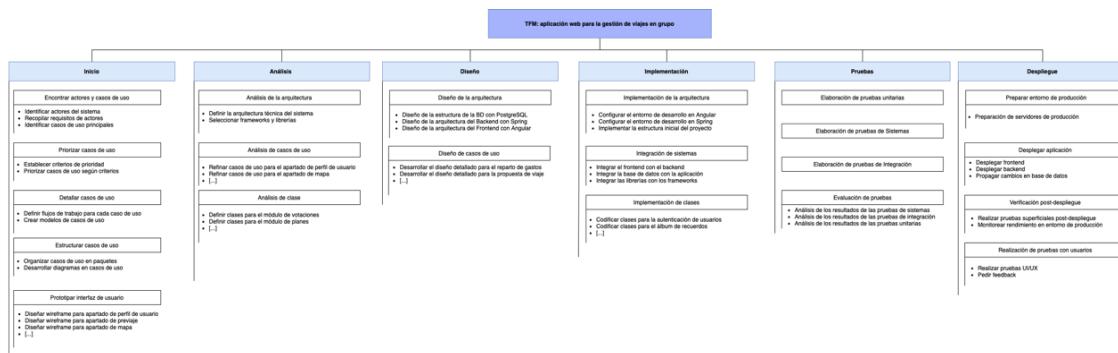


Ilustración 92: EDT del proyecto

CONCLUSIONES Y POSIBLES AMPLIACIONES

El desarrollo de la aplicación **TripPlanner** demuestra una vez más, como la digitalización mejora la eficiencia de ciertos procesos. A través del uso de tecnologías modernas como Angular [6] y Spring Boot [7], se ha logrado crear una plataforma *responsive* y funcional que facilita, desde la planificación inicial, hasta la gestión financiera de los viajes. La integración de herramientas como Liquibase [5], SonarCloud [10], GitHub Actions [9], y Docker [8] ha permitido una gestión eficiente y robusta del proceso de desarrollo.

Conclusiones:

1. **Adaptabilidad y funcionalidad:** La aplicación ha mostrado ser capaz de adaptarse a diferentes dispositivos y cumplir con los requisitos funcionales establecidos, proporcionando una experiencia de usuario consistente y eficiente.
2. **Optimización del proceso de desarrollo:** La utilización de metodologías avanzadas y herramientas permitiendo una integración continua y un despliegue eficiente. También se han aplicado los conocimientos de *DevOps* [2].
3. **Trabajo en equipo:** La colaboración ha sido clave, donde las fortalezas individuales se han complementado para lograr un producto final exitoso.
4. **Full-Stack:** Se han elaborado tareas de todas las disciplinas impartidas durante el Máster tanto *frontend* [3] como *backend* [4]. Y se han afianzado otras de la carrera tales como diseño y sistemas de bases de datos.

Posibles ampliaciones:

1. **Implementación de IA para recomendaciones personalizadas:** Integrar algoritmos de inteligencia artificial para ofrecer recomendaciones personalizadas basadas en el historial de viajes y preferencias del usuario.
2. **Expansión de funcionalidades sociales:** Añadir funciones de redes sociales para compartir itinerarios y experiencias, y así fomentar la interacción entre usuarios.
3. **Integración con plataformas de reserva:** Ampliar la funcionalidad para permitir la reserva directa de vuelos, hoteles y otros servicios a través de APIs de terceros.
4. **Optimización de la gestión financiera:** Desarrollar una herramienta más avanzada para la gestión de presupuestos y la distribución de gastos entre los participantes del viaje.



BIBLIOGRAFÍA

1. Orval. "Generate client with appropriate type-signatures", Victor Bury, 2024 [Online]. Available: <https://orval.dev/> [Accessed: 25-Jun-2024].
2. Devops. "What is DevOps?," Red Hat, 2023. [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-devops>. [Accessed: 25-Jun-2024].
3. Frontend. "What is Front-End Development?," Codecademy, 2023. [Online]. Available: <https://www.codecademy.com/articles/what-is-front-end>. [Accessed: 25-Jun-2024].
4. Backend. "What is Back-End Development?," Codecademy, 2023. [Online]. Available: <https://www.codecademy.com/articles/what-is-back-end>. [Accessed: 25-Jun-2024].
5. Liquibase. "Liquibase - Innovate faster with database DevOps," [Online]. Available: <https://www.liquibase.com>. [Accessed: 25-jun-2024].
6. Angular. "Angular - The modern web developer's platform," [Online]. Available: <https://angular.io>. [Accessed: 24-jun-2024].
7. Pivotal Software. "Spring Boot," [Online]. Available: <https://spring.io/projects/spring-boot>. [Accessed: 24-jun-2024].
8. Docker, Inc. "Docker Documentation," [Online]. Available: <https://docs.docker.com>. [Accessed: 24-jun-2024].
9. GitHub, Inc. "GitHub Actions," [Online]. Available: <https://github.com/features/actions>. [Accessed: 24-jun-2024].
10. SonarSource. "SonarCloud," [Online]. Available: <https://sonarcloud.io>. [Accessed: 24-jun-2024].
11. Swagger. "Swagger Codegen" [Online]. Available: <https://swagger.io/tools/swagger-codegen/> [Accessed: 25-jun-2024].
12. Anwar, A. (2014). A review of rup (rational unified process). International Journal of Software Engineering (IJSE), 5(2), 12-19.
13. Matrid Technologies, "5 Key Features of AngularJS That Make It the Best for Web Development," Accessed: Jun. 25, 2024. [Online]. Available: <https://www.matridtech.net/5-key-features-of-angularjs-that-make-it-the-best-for-web-development/>