Informe final

Integrantes:
Mathías Fernández
Eduardo Flores
Federico Acosta
Gabriel Suárez
Pablo Perdomo
Alexander Rodriguez

Tutor:
Fernando Arrieta

Queremos agradecer al Ing. Fernando Arrieta, tutor de este proyecto, por su guía constante, su disposición y por orientarnos con claridad durante cada instancia de tutoría.

Agradecemos también al tribunal evaluador por tomarse el tiempo para analizar nuestro trabajo y brindarnos una nueva oportunidad de evaluación.

A la carrera de Tecnólogo en Informática, por habernos formado con herramientas sólidas y por transmitirnos conocimientos que pudimos aplicar en este proyecto.

Y, por sobre todo, a nuestras familias, que nos acompañaron y apoyaron incondicionalmente durante incontables horas de esfuerzo, y que fueron un pilar fundamental para poder llegar hasta aquí.

Resumen

Este informe tiene como objetivo documentar el proceso completo de diseño, implementación y verificación de un sistema digital orientado a la venta de pasajes de ómnibus de larga distancia, desarrollado como parte de la asignatura Proyecto de la carrera Tecnólogo en Informática. La solución propuesta se construyó bajo un enfoque modular y multiplataforma, combinando tecnologías modernas y prácticas de desarrollo profesional para atender los requerimientos de distintos perfiles de usuario.

El sistema se compone de una aplicación web destinada a usuarios con roles de administrador, vendedor y clientes, así como una aplicación móvil dirigida exclusivamente al cliente final. Las funcionalidades incluyen la gestión de usuarios, ómnibus, localidades y viajes, así como el proceso completo de compra de pasajes, generación de comprobantes en PDF, calificación de viajes y recepción de notificaciones. La arquitectura se sustenta en una API REST desarrollada en Spring Boot, una interfaz web en React y una app mobile construida en React Native. La solución fue complementada con pruebas unitarias (JUnit, Mockito), E2E (Playwright) y validaciones manuales en entorno mobile, garantizando una cobertura adecuada y un funcionamiento robusto del sistema.

Palabras clave

Sistema de venta de pasajes – Spring Boot – Java – React – React Native – PostgreSQL – Docker – API REST – Aplicación móvil – Web app – Testing – JUnit – Mockito – Playwright – Gestión de viajes – Gestión de ómnibus – Notificaciones push – PDF

1. Introducción

Con el avance de la tecnología y la creciente digitalización de servicios, el sector del transporte de pasajeros enfrenta el desafío de modernizar sus canales de venta y gestión operativa. Tradicionalmente, la compra de pasajes de ómnibus en Uruguay ha estado asociada a procedimientos presenciales o soluciones fragmentadas, lo que genera demoras, errores y una experiencia de usuario poco optimizada. La necesidad de una plataforma integral que permita gestionar pasajes, usuarios, destinos y flotas en forma centralizada, accesible y segura es cada vez más relevante.

En este contexto se enmarca el desarrollo del sistema TECNOBUS, una solución orientada a la venta online de pasajes de ómnibus de larga distancia, creada en el marco de la asignatura “Proyecto” de la carrera Tecnólogo en Informática. Esta plataforma busca integrar funcionalidades clave para tres perfiles diferenciados: administradores, vendedores y clientes, permitiendo operar de forma eficiente tanto desde una aplicación web como desde una app móvil. El sistema contempla características modernas como autenticación con código OTP, generación de comprobantes PDF, notificaciones push, visualización de estadísticas y arquitectura basada en servicios.

Para lograr estos objetivos, el presente informe detalla las distintas etapas del desarrollo del sistema, siguiendo la estructura habitual de un proceso de ingeniería de software:

- Objetivos planteados y resultados obtenidos: se define el propósito general del proyecto y los resultados logrados al finalizar su implementación.
- Estado del arte: se analiza el panorama actual de soluciones similares en el mercado nacional e internacional.
- Análisis y diseño: se describen los elementos técnicos que sustentan la solución: casos de uso, modelo de dominio, arquitectura y diseño de interfaces.
- Gestión del proyecto: se expone la planificación grupal, la distribución de roles y la organización de tareas por iteraciones.

- Implementación: se documenta el proceso de desarrollo, tecnologías utilizadas y decisiones adoptadas durante la construcción del sistema.
- Testing: se explican las estrategias de verificación aplicadas: pruebas unitarias, E2E y validaciones manuales en mobile.
- Conclusiones y trabajo futuro: se presentan las reflexiones finales y posibles líneas de mejora o ampliación para futuras versiones del sistema.

2. Objetivos y resultados

El presente trabajo se enmarca en el curso “Proyecto” de la carrera Tecnólogo en Informática, dictado por la Facultad de Ingeniería de la Universidad de la República junto con la Dirección General de Educación Técnico Profesional (DGETP - ANEP). Esta carrera tiene como objetivo formar profesionales con conocimientos sólidos en análisis, desarrollo y mantenimiento de sistemas informáticos, fomentando el trabajo colaborativo y la aplicación de metodologías que reflejan la realidad de la industria.

En este contexto, el objetivo general del proyecto fue analizar, planificar, documentar y desarrollar un sistema online para la venta de pasajes de ómnibus de larga distancia. La solución debía contemplar las funcionalidades necesarias para permitir la operación eficiente de una empresa de transporte, incorporando una interfaz web para administradores, vendedores y clientes, y una aplicación móvil orientada exclusivamente al cliente final. Se buscó asegurar una experiencia de usuario satisfactoria, tanto en funcionalidad como en usabilidad, incorporando mecanismos modernos como autenticación por código, notificaciones push y descarga de comprobantes en PDF.

Para alcanzar este objetivo, se definieron y cumplieron las siguientes etapas:

- Análisis y estado del arte: Se investigaron soluciones existentes en el mercado y se identificaron sus puntos fuertes y débiles, con el fin de proponer una solución competitiva y ajustada a la realidad nacional.
- Selección y evaluación de tecnologías: Se eligieron tecnologías modernas y acordes a la experiencia del equipo: Java + Spring Boot para el backend, React para el frontend web, y React Native con Expo Go para mobile, todo con persistencia en PostgreSQL y despliegue en Docker sobre Digital Ocean.
- Diseño y planificación técnica: Se elaboraron modelos de dominio, arquitectura, diseño de interfaces, secuencias y casos de uso, asegurando una cobertura integral de los requerimientos funcionales.
- Implementación y validación: Se desarrolló la solución siguiendo las buenas prácticas aprendidas a lo largo de la carrera, incluyendo pruebas unitarias

con JUnit y Mockito, pruebas E2E con Playwright y verificaciones manuales en la app móvil.

Como resultado final, se logró construir un sistema funcional, documentado y verificado, capaz de gestionar usuarios, viajes, ventas y estadísticas, cumpliendo con los objetivos planteados tanto en términos técnicos como académicos.

3. Estado del arte

3.1 Marco teórico

La venta de pasajes de ómnibus en línea en Uruguay ha evolucionado significativamente, impulsada por la adopción tecnológica y las necesidades de los usuarios. Este documento analiza el desarrollo histórico, las plataformas clave (locales e internacionales), sus funcionalidades, ventajas, desventajas y tendencias emergentes en el sector.

3.2 Evolución histórica

Década de 2000: Primeros intentos de venta en línea por empresas locales, limitados por la baja penetración de internet y preferencia por métodos tradicionales.

2016: Lanzamiento de URUBUS, plataforma pionera que centralizó la oferta de múltiples empresas, facilitando la compra digital.

Expansión posterior: Empresas como *COT*, *Grupo Agencia* y *Turil* desarrollaron plataformas propias.

Auge móvil: Aplicaciones móviles mejoraron la accesibilidad, ofreciendo información en tiempo real y reservas instantáneas.

3.3 Plataformas web

3.3.1 COT (Compañía Oriental de Transporte)

COT [1] es una empresa uruguaya fundada en 1940 y dedicada al transporte de pasajeros. COT ofrece servicios de ómnibus que conectan Montevideo con diversos destinos, incluyendo Punta del Este, Maldonado, Piriápolis, La Paloma, La Pedrera, Punta del Diablo, Chuy y Colonia.

Ventajas:

- Accesibilidad: desde cualquier dispositivo y no requiere registro para compras.
- Compra libre: no es obligatorio registrarse para comprar pasajes.
- Información completa y actualizada: horarios, duración, paradas, etc.
- Atención al cliente: Múltiples canales de contacto (web, mail y teléfono).

Desventajas:

- Fallos ocasionales en el sistema de compra en línea obligando a compra telefónica.
- Limitaciones para usuarios extranjeros como tarjetas rechazadas y no tener PayPal.
- Falta de sección de preguntas frecuentes (FAQs).

Funcionalidad	**Descripción**
Búsqueda de pasajes	Los usuarios pueden buscar pasajes filtrando por: Origen, destino, cantidad de pasajeros, fecha de ida y fecha de vuelta
Tarifario y horarios	Tienen una sección exclusiva en donde presentan todos los días de salida, horarios y todos los destinos que alcanzan. Presentan destinos destacados acorde a los destinos que más viajes tienen.
Contacto	Facilitan en la web vías alternativas de contacto
Contrataciones particulares	Ofrecen un servicio de contratación de flota

	particular por fuera de los destinos ya ofrecidos por ellos
Bloqueo de asiento	Durante la compra, generan un bloqueo de asiento por 10 minutos aun así el mismo no haya sido comprado, pasado ese tiempo se libera.

3.3.2 Grupo Agencia

Grupo Agencia [2] es una empresa uruguaya conformada por: Agencia Central (1969), Chadre, Sabelin, COA y DAC, dedicada al transporte de pasajeros y encomiendas, brinda sus servicios en los departamentos de Montevideo, San José, Colonia, Soriano, Flores, Río Negro, Paysandú, Salto, Artigas, Rivera, Tacuarembó, Durazno y Maldonado.

Ventajas:

- Accesibilidad: desde cualquier dispositivo y no requiere registro para consultas.
- Ofrece la posibilidad de ver el historial de viajes del usuario.
- Contacto y soporte: Ofrece formulario para reclamos, bolsa de trabajo y postulaciones.

Desventajas:

- Registro obligatorio: el usuario debe registrarse para comprar pasajes.
- Acceso a Bot: Se reportan problemas de acceso al bot de respuestas automáticas autenticándose con la cuenta de Google.
- No se puede retomar una compra previamente iniciada.
- No presenta la posibilidad de filtrar destinos en la sección de consulta de horarios, solo la descarga de un PDF genérico.

Funcionalidad	Descripción
Búsqueda de pasajes	Los usuarios pueden buscar pasajes filtrando por: Origen, destino, cantidad de pasajeros, fecha de ida y fecha de vuelta, si ambos son abiertos o solamente el de vuelta.
Accesos rápidos	Ofrece accesos en la página principal al mapa con todas las agencias, link al procedimiento para acceder a descuentos a estudiantes y acceso a DAC (servicio de encomiendas)
Destinos	Se puede acceder a un PDF con todos los horarios y destinos, así como las agencias y links a web de las intendencias de cada departamento que ofrecen servicios.
Gestión de usuario	Ofrece historial de viajes y pasajes comprados, visualización de pasajes sin fecha asignada (abiertos) y modificación de información del perfil.
Contacto y soporte	En la página principal tiene link a chat de Whatsapp, opción de utilizar su bot o formulario de contacto.

3.3.3 Turil

Turil [3] es una empresa uruguaya fundada el 18 de agosto de 1975 (**Tu**rismo **Ri**verense **L**imitados), dedicada principalmente al transporte de pasajeros, conecta Montevideo con ciudades como Durazno, Paso de los Toros, Tacuarembó, Rivera, Tranqueras, Artigas, Libertad, Nueva Helvecia, Juan Lacaze y Colonia.

Ventajas:

- Compra libre: no es obligatorio registrarse para comprar pasajes.
- Horarios: Tabla comparativa de horarios, destinos y tarifas así como filtros (no PDF) y paradas intermedias.
- Acepta una variada gama de medios de pago.

Desventajas:

- Falta de sección de preguntas frecuentes (FAQs).
- No se puede retomar una compra previamente iniciada.
- Temporizador de reserva corto: Ofrece 5 minutos para completar la compra antes de liberar el asiento.

Funcionalidad	Descripción
Compra de pasajes	Permite al Usuario comprar pasajes seleccionando origen, destino y fecha de viaje
Consulta de horarios y precios	Se puede acceder a una sección en donde seleccionando origen, destino y día de la semana muestra los horarios, con paradas intermedias y también se puede acceder a todas las tarifas.
Información adicional	En la página principal ofrece variados enlaces a varios servicios e información: descuentos para estudiantes, encomiendas, traslado de animales, reglamentos de reclamos de URSEC.

3.3.4 CUT Corporación

CUT Corporación [4] es una empresa uruguaya de transporte de pasajeros que forma parte del Grupo Carminatti. Su historia se remonta hacia 1930 en la ciudad de Fray Bentos, donde Don Gualberto Carminatti funda la compañía dedicada al transporte (en ese entonces llamada *Plus Ultra*). A lo largo de los años, la empresa ha evolucionado y expandido sus operaciones, consolidándose como un referente en el sector. Hoy en día la empresa ofrece servicios hacia Artigas, Tacuarembó, Rivera, Paso de los Toros, Fray Bentos, Flores y Minas.

Ventajas:

- Pago diferido: Ofrece 20 minutos para completar el pago luego de reservar el pasaje.

Desventajas:

- Registro obligatorio: el usuario debe registrarse para comprar pasajes.
- Falta de sección de preguntas frecuentes (FAQs).
- Diseño en sección de compra de pasajes: no completamente responsive, textos que se solapan o menú que no completan la pantalla del celular.
- Gestión de usuario difícil de acceder: No tiene login en página principal, se debe acceder a la sección de compras.
- *Home link*: no se tiene link a la página principal una vez ingresado a la sección de compra de pasajes.

Funcionalidad	**Descripción**
Compra de pasajes	Permite al Usuario comprar pasajes seleccionando origen, destino y fecha de viaje
Gestión de usuario	Modificación de información del usuario.
Consulta de horarios	Ofrece accesos en la página principal al

	mapa con todas las agencias, horarios por línea y tarifas.
Otros Servicios	En la página principal ofrece su servicio de encomienda, recomendaciones de hoteles en Fray Bentos y su empresa de viajes.

3.3.5 Omio

Omio [5] es una sitio web alemán de comparación y reserva de viajes, con sede en Berlín, Alemania. Fue fundado en 2012 como GoEuro por Naren Shaam y actualmente reúne a más de 1,000 compañías de transporte entre trenes, autobuses, vuelos, ferries, coches y traslados al aeropuerto.

Si bien Omio no es una empresa de transporte per se, se considera su web en este documento por ciertas características que ofrece.

Ventajas:

- Proceso de compra optimizado (3 pasos máximo)
- Diseño 100% responsive (mobile-first)
- Disponible en 32 idiomas

Desventajas:

- Desde el precio que se muestra al precio final pueden haber cambios.

Funcionalidad	**Descripción**
Búsqueda de pasajes	Los usuarios pueden buscar pasajes, y organizarlos por medio de transporte (tren, avión, ómnibus) llamada búsqueda multimodal.
Recomendación de estadías	Contiene una integración con Booking que luego de una búsqueda de pasajes se abre una nueva pestaña con recomendación de hoteles, aptos., casas, etc. en el lugar de

	destino.
Tendencias	En la página principal muestran las conexiones, destinos y empresas de transportes más populares
Descuentos	Posibilidad de acceder a un descuento del 10% certificando ser estudiante.

3.4 Plataformas Mobile

3.4.1 Turil Móvil

Aplicación móvil de la empresa Turil [6] para compra de pasajes.

Ventajas:

- Horarios: Tabla comparativa de horarios, destinos y tarifas así como filtros (no PDF) y paradas intermedias.
- No requiere registro para consultas.

Desventajas:

- Falta de sección de preguntas frecuentes (FAQs).
- No se puede retomar una compra previamente iniciada.
- Temporizador de reserva corto: Ofrece 5 minutos para completar la compra antes de liberar el asiento.

Funcionalidad	**Descripción**
Compra de pasajes	Permite al Usuario comprar pasajes

	seleccionando origen, destino y fecha de viaje.
Destinos, Horarios y Tarifas	Se puede acceder a una sección en donde seleccionando origen, destino y día de la semana muestra los horarios, con paradas intermedias y también se puede acceder a todas las tarifas.

3.4.2 Urubus

Urubus [7] es una plataforma de comercio electrónico dedicada a la venta de pasajes de ómnibus. Fundada en 2018 por Juan Van de Kerchove. En sus inicios, la plataforma se enfocaba en brindar información sobre horarios y tarifas de ómnibus, pero siempre con la visión de evolucionar hacia la venta de pasajes en línea. Urubus ha revolucionado la forma en que los uruguayos acceden a los servicios de transporte interdepartamental, ofreciendo una plataforma fácil de usar que centraliza información y permite la compra de pasajes de manera rápida y segura.

Ventajas:

- Centralización: Compara precios y rutas de múltiples empresas sin cambiar de plataforma.
- Práctica post-compra: La aplicación envía un mail con el QR de la compra, que será escaneado en el ómnibus.
- Atención al público: Atención por varios medios (WhatsApp, correo, Instagram, Facebook, X [ex-twitter], LinkedIn y chat en línea)
- Mobile: Disponible tanto para Android como iOS

Desventajas:

- Sincronización: Ocasionalmente hay desfases en la disponibilidad real vs mostrada.

Funcionalidad	Descripción
Búsqueda y compras integradas	La aplicación agrega múltiples empresas de ómnibus en un solo lugar, filtrando por origen/destino, fecha y hora y mostrando asientos disponibles.
Información en tiempo real	Actualización dinámica de horarios, precios y disponibilidad de asientos.
Servicios adicionales	Adicionalmente ofrece información acerca de algunas terminales y empresas ómnibus así como recomendaciones para antes, durante y luego del viaje.

3.4.3 FlixBus

FlixBus [8] es una empresa alemana fundada en 2013, con presencia en Europa, Norteamérica y Latinoamérica que permite comprar pasajes en línea a través de su página web o app.

Ventajas:

- Proceso de compra eficiente: 3 clics < 2 minutos
- Mobile: Diseño 100% responsive
- Flexibilidad de compra: Cambios gratuitos hasta 15 minutos antes de la partida.
- Transparencia: Precios con tasas incluídas, política clara de equipaje.

Desventajas:

- Tarifas dinámicas: Los precios pueden aumentar o disminuir según

temporada.

- Atención al cliente: Se reportan respuestas entre 24 a 72hs

Funcionalidad	Descripción
Búsqueda y reserva/compra avanzada	Los usuarios pueden buscar pasajes ingresando: origen, destino, fecha ida, fecha vuelta y cantidad de pasajeros incluyendo transbordos.
Información en tiempo real	Actualización dinámica de horarios, precios y disponibilidad de asientos.
Mapa de exploración	Luego de ingresar una ciudad inicial muestra todas las ciudades cercanas a las que se puede comprar un pasaje.
Muestra de pasajes comprados	Muestra todos los pasajes comprados a través de la app o si es físico se puede subir a la aplicación insertando el número de reserva del boleto y correo electrónico del comprador.
Más - Ajustes generales	Ajustes de la divisa, unidad de distancia, configuración de privacidad, Términos y condiciones, política de privacidad, calificación de la app e Info sobre ella.

3.5 Análisis Comparativo

	COT	Agencia Central	Turil	CUT	Omio	Turil Móvil	Urubus	FlixBus
Requiere registrarse para	NO	SI	NO	SI	SI	NO	NO	NO

comprar								
Gestión de perfil	NO	SI	NO	SI	SI	NO	NO	NO
Historial de compras	NO	SI	NO	SI	SI	NO	NO	NO
Plataforma	Web	Web	Web	Web	Web	Mobile	Mobile	Mobile
Búsqueda horarios	SI	SI	SI	SI	SI	SI	SI	SI
Recomendaciones de Destino	SI	NO	NO	SI	SI	NO	SI	SI
Descarga de horarios en PDF	NO	SI	NO	NO	NO	NO	NO	NO
Listado de Agencias / Destinos	SI	SI	SI	SI	SI	SI	SI [3]	SI [4]
Seguimiento encomiendas	SI	SI[1]	SI[1]	NO	NO	NO	NO	NO [5]
Compra de pasajes	SI	SI	SI	SI	SI	SI	SI	SI
Compra de pasajes abiertos	NO	SI	NO	NO	NO	NO	NO	NO
Gestión de Descuentos	NO	SI	SI	NO	SI	SI	NO	SI [6]
Temporizador de reserva	5 min	10 min	5 min	20 min	5 - 15 min [2]	5 min	14 min	10 min
Formulario de contacto	SI	SI	SI	SI	SI	SI	SI	SI [7]
Sección FAQs	NO	NO	NO	NO	SI	NO	NO	SI [7]

Idiomas	ES	ES	ES, BR	ES	+30	ES	ES, BR, US, GR, FR	System [8]
Redes Sociales	NO	SI	NO	NO	SI	NO	SI	SI

[1] - Redirige a web de la división encargada de encomiendas.
[2] - Dependiendo de la empresa elegida.
[3] - Lista solamente 5 terminales.
[4] - Muestra todos los puntos donde hay una sucursal en un mapa.
[5] - No tienen servicios de encomienda solo transporte de pasajeros
[6] - A través de cupones
[7] - Redirige a web de ayuda de Flixbus
[8] - Establece como idioma el idioma del dispositivo.

3.6 Conclusiones

La venta de pasajes de ómnibus en línea en Uruguay ha experimentado una transformación significativa, impulsada por la digitalización y las demandas de los usuarios. Plataformas locales como COT, Grupo Agencia, Turil y CUT Corporación han logrado ampliar su cobertura y funcionalidades, destacando en accesibilidad, horarios detallados y servicios específicos (como pasajes abiertos o pagos diferidos). Sin embargo, hay espacio para mejoras como interfaces obsoletas, registro obligatorio, brindar la oportunidad de retomar una compra previamente iniciada y limitaciones en medios de pago, lo que afecta la experiencia del usuario.

URUBUS en cambio surge como líder innovador al centralizar múltiples empresas en una sola plataforma, ofreciendo comparación de precios y una app móvil eficiente. Por otra parte, actores internacionales como Omio y FlixBus establecen estándares globales en diseño responsive y multimodalidad, aunque aún no se encuentren en el mercado uruguayo.

El sector muestra tendencias hacia la optimización móvil y la demanda en integración de pagos digitales (Mercadopago, Paypal, Stripe, etc) así como la

transparencia en precios y políticas. Para consolidarse, las plataformas deben priorizar la modernización de interfaces, brindar incentivos a los usuarios que fomenten el uso de la herramienta y fortalecer herramientas de autoservicio.

4. Análisis y diseño

Esta sección se enfoca en la especificación de los requerimientos del sistema online para la venta de pasajes de ómnibus de larga distancia, abarcando tanto los aspectos funcionales como no funcionales. Se establecen claramente los límites del proyecto, definiendo qué funcionalidades serán abordadas y cuáles quedan fuera del alcance.

Además, se describen los diferentes tipos de usuarios que interactúan con la aplicación: administrador, vendedor y cliente, detallando sus respectivos permisos y las acciones que pueden realizar dentro del sistema. Esta segmentación permite comprender cómo se distribuyen las responsabilidades y cómo se estructura el acceso a las funcionalidades, de acuerdo al rol asignado a cada usuario.

4.1 Perfiles de usuario

A continuación, se detallan los tres tipos de usuarios que tendrá la aplicación.

4.1.1 Administrador

El administrador es el responsable de la supervisión y gestión general del sistema. Su principal función es administrar a los usuarios internos, lo que incluye el alta y baja de cuentas de vendedores y otros administradores, tanto de forma individual como masiva. Asimismo, tiene acceso a herramientas para la visualización de estadísticas de los usuarios, así como listados de usuarios con filtros avanzados. Todas sus actividades se realizan exclusivamente a través de la interfaz web destinada a la gestión operativa del sistema.

4.1.2 Vendedor

El vendedor es el usuario encargado de administrar la operativa diaria del sistema vinculada al transporte. Tiene la responsabilidad de registrar nuevas localidades y ómnibus (ya sea de forma manual o mediante carga masiva), crear y reasignar viajes, realizar venta de pasajes, así como efectuar devoluciones, activar o desactivar vehículos, consultar listados detallados de pasajes vendidos y acceder a estadísticas relacionadas con la flota, viajes y ventas. Su interacción se da

exclusivamente desde la aplicación web, con acceso restringido a funcionalidades específicas según su rol.

4.1.3 Cliente

El cliente representa al usuario final del sistema, y es el destinatario de la aplicación móvil, aunque también puede acceder desde la versión web. Puede crear su cuenta, iniciar sesión, editar su información personal y consultar viajes disponibles, visualizando los asientos libres al momento de la consulta. Tiene la posibilidad de comprar pasajes —tanto de ida como de ida y vuelta—, recibir notificaciones relevantes, consultar su historial de compras y calificar los viajes realizados. Su experiencia está centrada en la usabilidad y simplicidad, priorizando la accesibilidad desde dispositivos móviles.

4.2 Requisitos del sistema

4.2.1 Servidor (backend)

- Sistema operativo: Linux (Ubuntu 20.04 o superior recomendado)
- CPU: 2 núcleos mínimo (4 núcleos recomendados)
- RAM: 4 GB mínimo (8 GB recomendados)
- Almacenamiento: 80 GB libres mínimo
- Software requerido:
    - Java 17 (OpenJDK)
    - Spring Boot 3.x
    - PostgreSQL 15 o superior
    - Docker + Docker Compose
    - Git

4.2.2 Cliente Web (Front Office / Back Office)

- Navegador compatible: Google Chrome, Mozilla Firefox, Microsoft Edge (últimas versiones)
- Resolución mínima recomendada: 1366x768
- Conectividad: Conexión a Internet estable

- Requisitos técnicos:
    - JavaScript habilitado
    - Soporte para cookies y almacenamiento local.

4.3 Requerimientos específicos

En esta sección se detallan los requerimientos específicos necesarios para el funcionamiento de la aplicación *TecnoBus*. Estos requisitos abarcan tanto aspectos técnicos como operativos que deben ser considerados para asegurar el correcto desempeño de la solución en sus distintos entornos: servidor backend, cliente web y aplicación móvil.

La identificación y aplicación adecuada de estos requerimientos resulta fundamental para garantizar una experiencia estable, segura y eficiente para todos los perfiles de usuario involucrados, desde administradores y vendedores hasta clientes finales. Para una descripción detallada de los requerimientos funcionales y no funcionales, se recomienda consultar el *Documento de Requerimientos*.

4.3.1 Requerimientos funcionales

4.3.1.1 Funcionalidades Administrador (excluyentes)

- Gestión de usuarios:
    - Altas de usuarios (individuales o masivos).
    - Bajas de usuarios.
    - Listado y búsqueda de usuarios.
- Estadísticas de usuarios (con exportación PDF y CSV).

4.3.1.2 Funcionalidades Vendedor (excluyentes)

- Gestión de localidades:
    - Alta de localidades (individuales o masivas).
    - Cambio de estado de localidades (habilitado / deshabilitado)
    - Listado y búsqueda de localidades

- Gestión de Ómnibus:
    - Altas de ómnibus (individuales o masivos).
    - Listado y búsqueda de ómnibus.
    - Cambio de estado del ómnibus (habilitado / deshabilitado).
- Gestión de Viajes:
    - Alta de viajes.
    - Reasignación de ómnibus asignados.
    - Listado y búsqueda de viajes.
    - Cancelación de viajes.
- Operaciones de venta y post-ventas:
    - Venta de pasajes.
    - Devolución de pasajes.
    - Listado y búsqueda de pasajes vendidos.
- Estadísticas (viajes, ómnibus y pasajes con exportación PDF y CSV).

4.3.1.3 Funcionalidades Cliente (excluyentes)

- Creación de cuenta.
- Compra de pasajes.
- Calificación de viajes.

4.3.1.4 Funcionalidades comunes a Vendedor y Cliente

- Listado y búsqueda de viajes
- Historial de Compras / Ventas

4.3.2 Requerimientos no funcionales

4.3.2.1 Usabilidad

El sistema debe ser intuitivo y fácil de usar, con una interfaz clara y accesible, tanto en la aplicación web como en mobile. Asegurando contrastes adecuados entre texto y fondo, tamaños de fuente legibles en todos los dispositivos, con un tamaño

mínimo recomendado de 720p, evitar el uso exclusivo de colores e Interacciones accesibles desde teclado y compatibles con lectores de pantalla.

4.3.2.2 Compatibilidad

El sistema debe ser compatible con las últimas versiones de los navegadores web más populares (Chrome, Firefox, Edge). La aplicación móvil debe ser compatible con Android 10 o superior.

4.3.2.3 Rendimiento

El sistema deberá garantizar tiempos de respuesta menores a 5s en las operaciones de búsqueda y consulta incluso durante picos de tráfico para no comprometer la experiencia del usuario.

4.3.2.4 Seguridad

La información confidencial de los usuarios (especialmente las contraseñas) debe ser protegida. Las contraseñas deben ser encriptadas antes de almacenarse en la base de datos. La comunicación entre el cliente y el servidor debe ser cifrada mediante HTTPS. Además, se implementó un mecanismo de doble factor de autenticación (2FA) para el inicio de sesión de todos los usuarios, mediante el envío de una contraseña de un solo uso (OTP) al correo electrónico, la cual debe ser ingresada para completar el acceso al sistema.

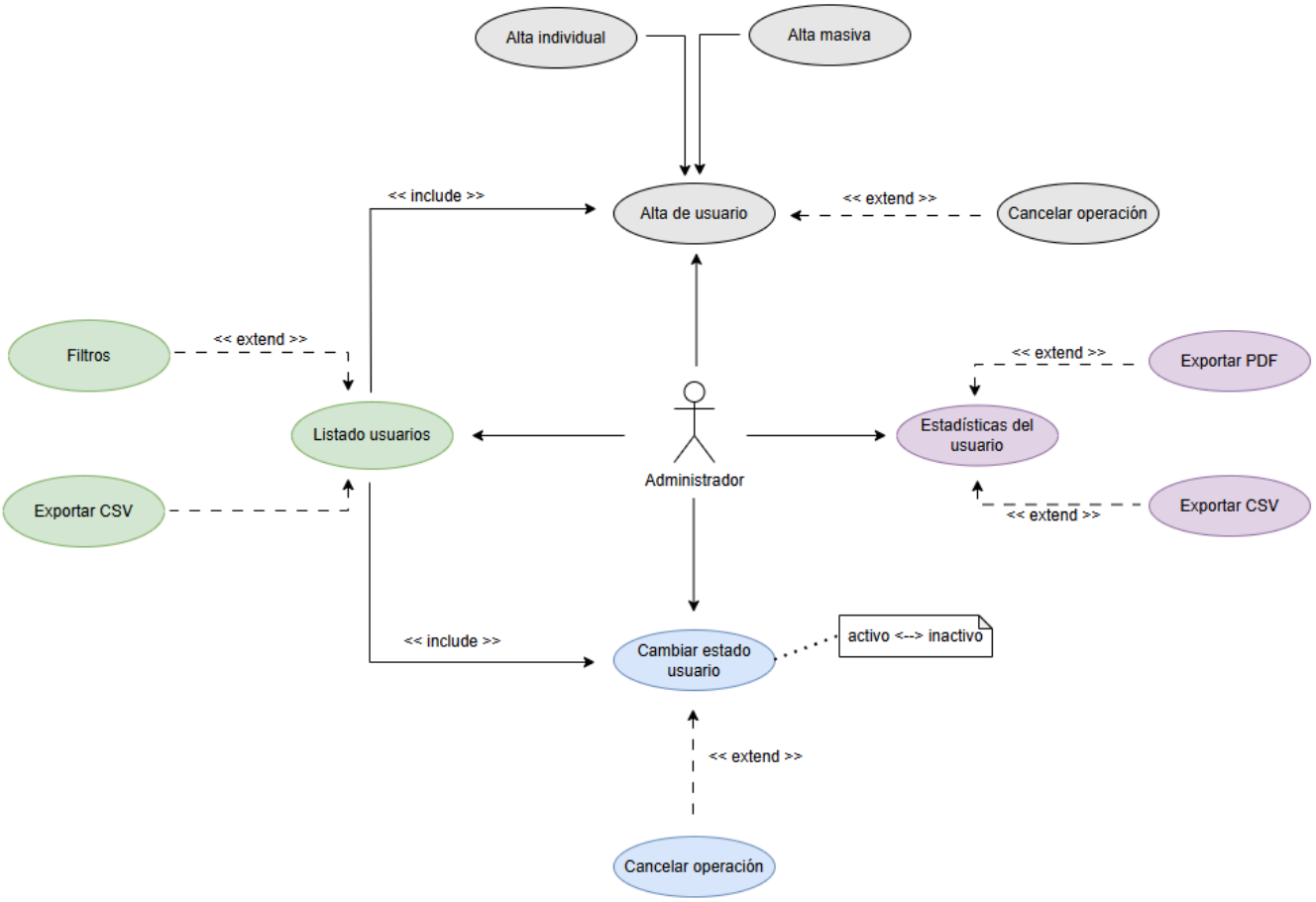
4.3.2.6 Interoperabilidad

El sistema se integrará con servicios externos para el envío de notificaciones por correo y notificaciones push, además se facilitará la exportación de documentos PDF empleando librerías. Estas decisiones apuntan a acelerar el desarrollo, simplificar la implementación de funcionalidades clave y garantizar una experiencia coherente para los usuarios, considerando tanto aspectos técnicos como objetivos del negocio.

4.4 Modelos de caso de uso

En esta sección se identifican los casos de uso que se consideran críticos dentro del sistema online para la venta de pasajes de ómnibus. Estos casos representan sólo una parte del total modelado, y han sido seleccionados por su relevancia para el funcionamiento esencial de la aplicación. Para una descripción completa de todos los casos de uso, se recomienda consultar el anexo "*Documento de Casos de Uso*".

Usuario Administrador



Alta de usuario:

Permite al administrador registrar nuevos usuarios con rol de vendedor o administrador dentro del sistema. Esta operación puede realizarse de forma individual, completando manualmente los datos requeridos, o de manera masiva, a través de la carga de un archivo CSV con múltiples registros; en ambos casos, el sistema valida que no existan duplicados.

Deshabilitar usuario:

Permite al administrador dar de baja usuarios existentes (de cualquier rol) del sistema. Esta operación implica la eliminación lógica del usuario, conservando su trazabilidad. Antes de concretarse, el sistema solicita confirmación, permitiendo también cancelar la operación en caso de ser necesario.

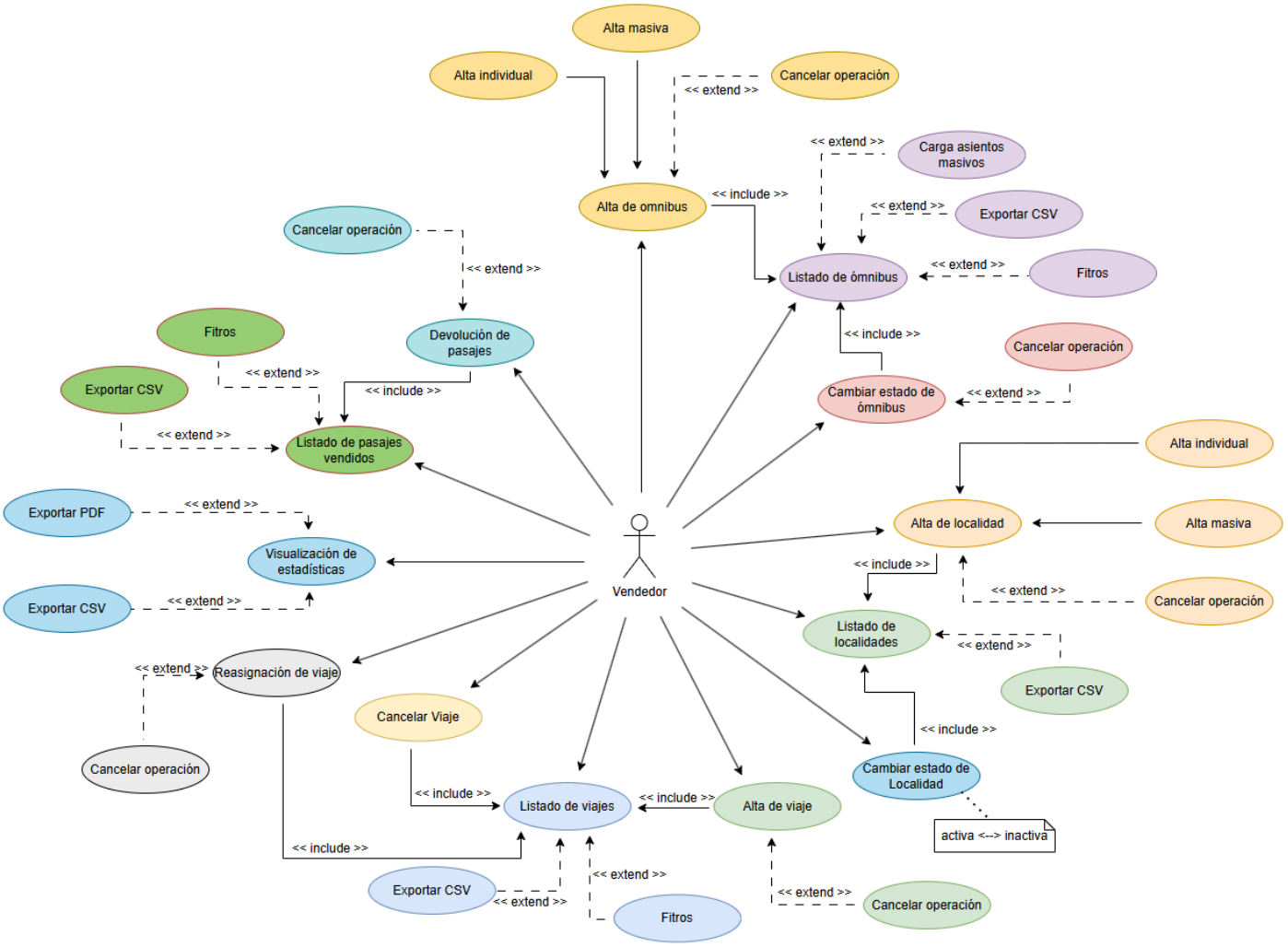
Listado usuarios:

Muestra al administrador un listado completo de los usuarios registrados en el sistema, con funcionalidades de búsqueda, ordenamiento y filtrado avanzado. Los filtros permiten refinar la consulta según criterios como nombre, cédula, correo, rol, estado (activo/inactivo), entre otros. Desde este listado, es posible navegar hacia acciones como eliminación lógica o edición de usuarios.

Estadísticas del usuario:

Ofrece al administrador una visualización detallada de distintas métricas relacionadas a los usuarios del sistema, segmentadas en categorías específicas. Además, el sistema brinda la opción de descargar los reportes en formato PDF o CSV extendiendo la funcionalidad de cada estadística.

Usuario Vendedor



Alta de ómnibus:

Permite al vendedor registrar nuevos vehículos en la flota. Esta operación puede realizarse de manera individual, ingresando los datos de un ómnibus en un formulario, o de forma masiva, mediante la carga de un archivo CSV. En ambos casos, el sistema valida la unicidad de la matrícula y ofrece la opción de cancelar la operación antes de finalizar.

Alta de localidad:

Faculta al vendedor a registrar nuevas localidades que formarán parte de los trayectos disponibles. El alta puede realizarse de forma individual o masiva, y el sistema garantiza que no se duplique una localidad existente.

Alta de viaje:

El vendedor puede crear un nuevo viaje especificando precio, localidad de origen y destino, fecha (inicio y posible fin), horario (inicio y posible fin) y ómnibus asignado. El sistema valida automáticamente la disponibilidad del vehículo y controla conflictos con viajes ya existentes. Se permite cancelar la operación si el vendedor desea abortar el registro.

Reasignación de viaje:

En caso de que un ómnibus no esté disponible, el vendedor puede reasignar un viaje a otro vehículo compatible. El sistema controla que el nuevo ómnibus tenga la capacidad suficiente y no tenga conflictos de agenda. Antes de confirmar, el vendedor puede cancelar la operación si lo desea.

Cambiar de estado de ómnibus:

El vendedor puede cambiar el estado de un ómnibus entre activo e inactivo (por ejemplo, por mantenimiento). Esta acción está disponible desde el listado de ómnibus, y se valida que el vehículo no tenga viajes asignados en curso antes de desactivarlo.

Devolución de pasajes:

Permite al vendedor procesar la devolución de un pasaje, siempre que esté dentro del plazo permitido antes de los 10 minutos previos de salida del ómnibus (modificable según lógica de negocio). El sistema verifica condiciones como fecha de viaje y estado del pasaje, y ofrece la posibilidad de cancelar la operación si no se desea continuar con el proceso.

Listado de pasajes vendidos:

El vendedor puede acceder a un listado completo de los pasajes vendidos para un determinado viaje. Este listado incluye filtros para facilitar la búsqueda por cliente, viaje, fecha o estado del pasaje.

Listado de ómnibus:

Muestra todos los ómnibus registrados en el sistema. Desde aquí se pueden aplicar filtros, ordenar por atributos (estado, matrícula, localidad), e iniciar acciones como cambiar el estado o editar datos.

Listado de viajes:

Visualiza todos los viajes cargados en el sistema. Incluye filtros por fecha, localidad o estado, etc, y permite acceder a acciones como reasignación de ómnibus o ver pasajes asociados. También puede ser el punto de entrada para editar o cancelar un viaje.

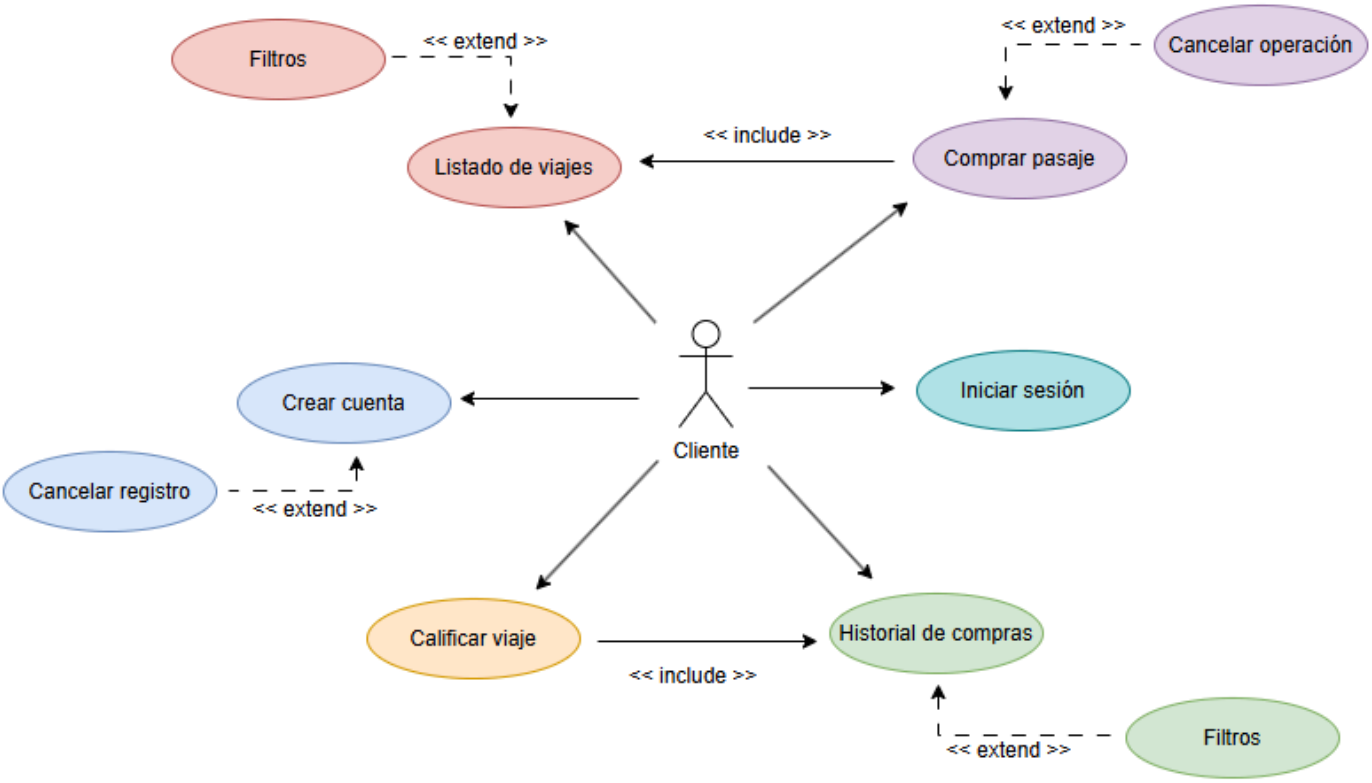
Visualización de estadísticas:

Brinda acceso a reportes sobre ventas por mes, calificación de viajes, cantidad de viajes por localidades, grupos de usuarios por edades, tipos de usuarios y su cantidad, crecimiento de la web por mes, etc. Estas estadísticas pueden ser exploradas en pantalla y también se pueden descargar en formato PDF o CSV, facilitando el análisis y la toma de decisiones.

Usuario Cliente

Listado de viajes:

Permite al cliente visualizar los viajes disponibles con detalles como origen, destino, fecha, horario, precio y cantidad de asientos. Incluye filtros por localidad y fecha para facilitar la búsqueda.



Comprar pasajes:

El cliente selecciona un viaje y puede adquirir un pasaje eligiendo asiento, aplicando descuentos y generando un comprobante en PDF. El sistema valida la disponibilidad del asiento y se conecta con una plataforma de pago electrónico para procesar la transacción, actualmente utilizando Stripe en entorno de pruebas (sandbox).

Iniciar sesión:

Permite al cliente autenticarse con su email y contraseña para acceder al sistema. Habilita el uso de funciones como la compra, el historial y la calificación de viajes.

Historial de compras:

Muestra todos los pasajes adquiridos por el cliente, con información del viaje, asiento y estado. Se puede filtrar y consultar el comprobante en PDF. Desde aquí se accede a la calificación.

Calificar viaje:

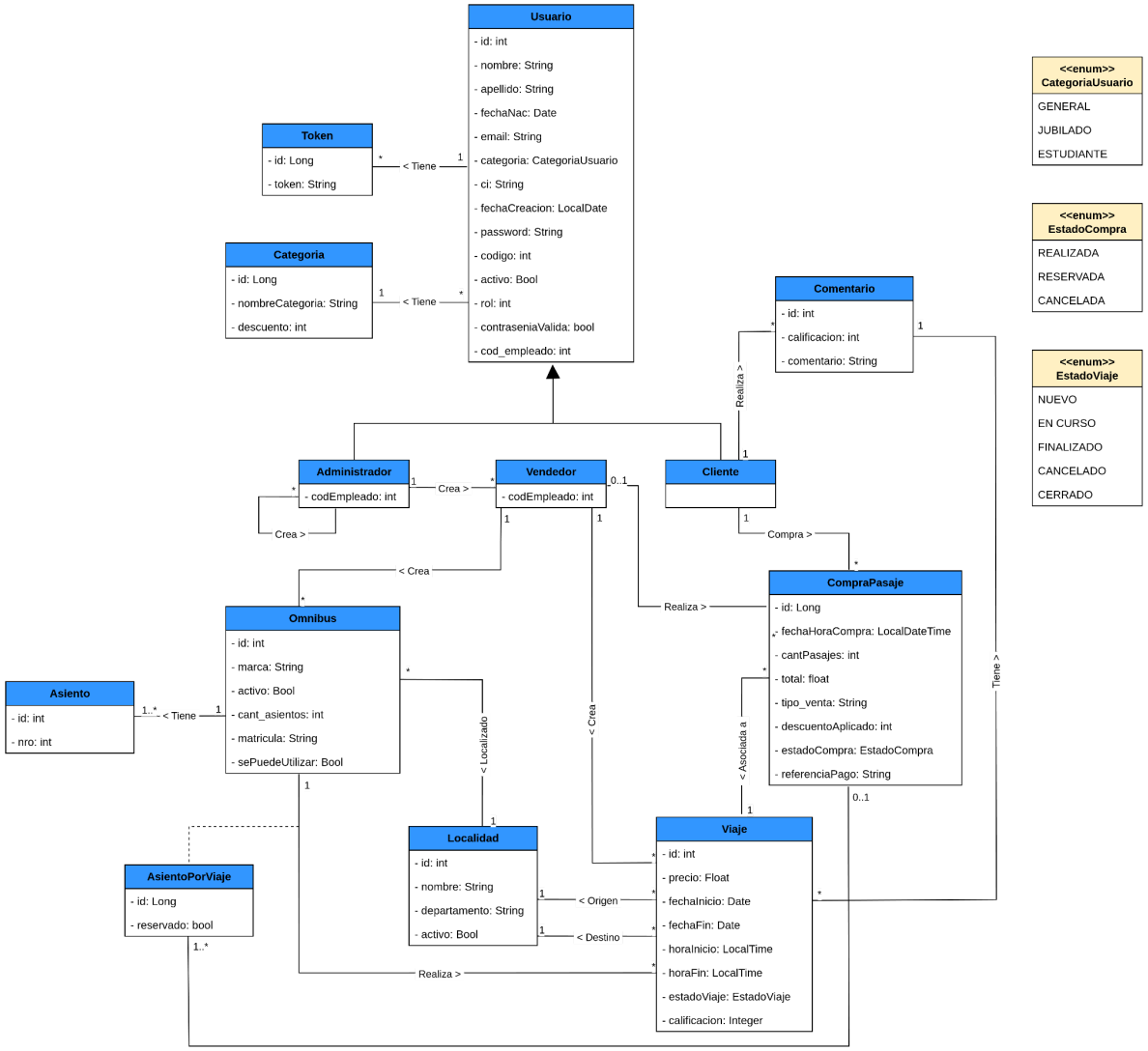
El cliente puede calificar con una puntuación del 1 al 5 un viaje que haya realizado. Esta acción se habilita solo si el viaje ya fue completado por ese usuario.

Crear cuenta:

Permite al cliente registrarse ingresando sus datos personales y contraseña. El sistema valida el email, la cédula y guarda el usuario para futuros accesos.

4.5 Modelo de dominio

En el siguiente diagrama se muestran en azul las diferentes clases que serán utilizadas para modelar la realidad y sus relaciones.



En la parte superior se encuentra la clase **Usuario** la cual cuenta con los datos básicos de todas las personas que utilizarán el sistema. Estos usuarios se van a dividir en 3 roles (**Administrador**, **Vendedor** y **Cliente**) cada uno de ellos con diferentes funcionalidades. Los atributos incluyen tanto los datos personales (nombre, apellido, fecha de nacimiento, cédula de identidad) como datos para el sistema (categoría, password). Los roles de **Administrador** y **Vendedor** además de los atributos básicos de usuario, poseen un atributo adicional, código de empleado, que permite identificarlos dentro de la empresa.

La clase **Localidad** modela a las diferentes ciudades, pueblos, balnearios, etc., a los que la empresa brinda sus servicios. Tiene como atributos el nombre de la localidad, el departamento en el que se encuentra y el atributo activo que indica si actualmente la empresa presta servicios en la localidad.

La clase **Omnibus** representa a los vehículos de la empresa. Sus atributos son su matrícula, la marca, la cantidad de asientos que posee y el estado en el que se encuentra (activo/mantenimiento). Se relaciona con la clase **Asiento**, la cual modela los diferentes asientos que posee el ómnibus, y con la clase **Localidad** para tener registro de la ubicación del ómnibus.

La clase **Viaje** representa los viajes que realiza la empresa uniendo las distintas localidades. Sus atributos incluyen la fecha y hora de inicio del viaje, fecha y hora de arribo a destino, el estado del viaje (Nuevo, En Curso, Finalizado, Cancelado o Cerrado) y el precio del pasaje. El origen y destino del viaje se modelan mediante sus relaciones con la clase **Localidad**.

De la relación entre la clase **Viaje** y la clase **Omnibus** se desprende la clase **AsientoPorViaje**, que modela los asientos de un viaje. Sus atributos indican el número de asiento y si el mismo se encuentra reservado.

La clase **CompraPasaje** representa la compra/venta de pasajes para los viajes de la empresa. Se relaciona con las clases **Usuario**, **Viaje** y **AsientoPorViaje**, y sus atributos son la fecha en la que se realizó la venta del pasaje, la cantidad de pasajes que se vendieron, el monto de la compra/venta, el descuento aplicado, la referencia

de pago de Stripe, la vía por la que se realizó la compra/venta y el estado de la compra (Realizada, Reservada o Cancelada).

La clase **Categoría** representa las categorías que tienen los clientes (General, Jubilado y Estudiante). Sus atributos son el nombre de la categoría y el descuento a aplicar en la compra/venta de pasajes.

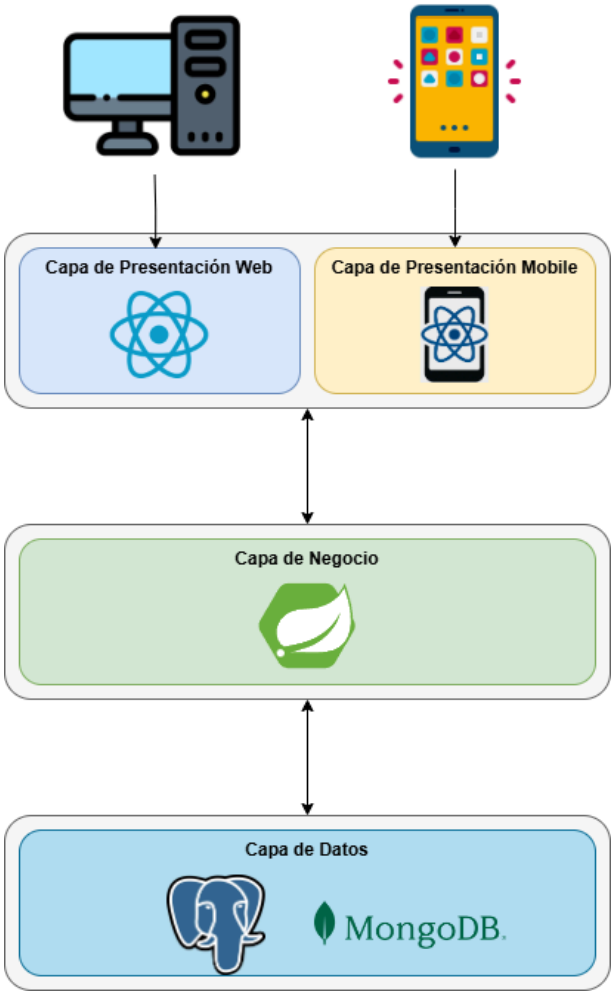
La clase **Token** representa el token que el usuario tiene en la web o en mobile.

Finalmente la clase **Comentario** representa la calificación que los clientes pueden dejar sobre los viajes que han realizado. Posee como atributos el valor numérico de la calificación otorgada por el cliente y el comentario que el mismo realizó. Está asociada tanto al **Viaje** como al **Cliente** que realizó la evaluación.

4.6 Arquitectura del sistema

4.6.1 Modelo de Implementación

En esta representación gráfica se ofrece una visión general del sistema en conjunto, siendo representadas las distintas capas que la componen.



4.6.1.1 Capa de Presentación

Esta capa es responsable de proporcionar las interfaces gráficas a los distintos tipos de usuarios.
Incluye tres componentes:

- Una aplicación **web**, desarrollada con **React** y servida con Node.js, orientada a los clientes finales (**FRONTOFFICE**).
- Una aplicación móvil, implementada con React Native, que se ejecuta en los dispositivos de los usuarios.
- Una interfaz administrativa, también desarrollada con React y Node.js, utilizada por administradores y vendedores (**BACKOFFICE**).

  Todas estas interfaces permiten la interacción con el sistema de forma intuitiva y adaptada a los diferentes perfiles de usuario.

4.6.1.2 Capa de Negocio

Esta capa se encarga de la lógica del sistema y de la exposición de servicios mediante APIs REST, que son consumidas por las aplicaciones de la capa de presentación.
Está compuesta por dos servicios desarrollados con Spring Boot:

- El servicio principal (**BACKEND**), que implementa toda la lógica de negocio de *Tecnobus*.
- El servicio de notificaciones (**NOTIFICACIONES**), que gestiona el envío de notificaciones push a clientes web y móviles.
  Ambos servicios están diseñados de forma modular y pueden escalar de forma independiente.

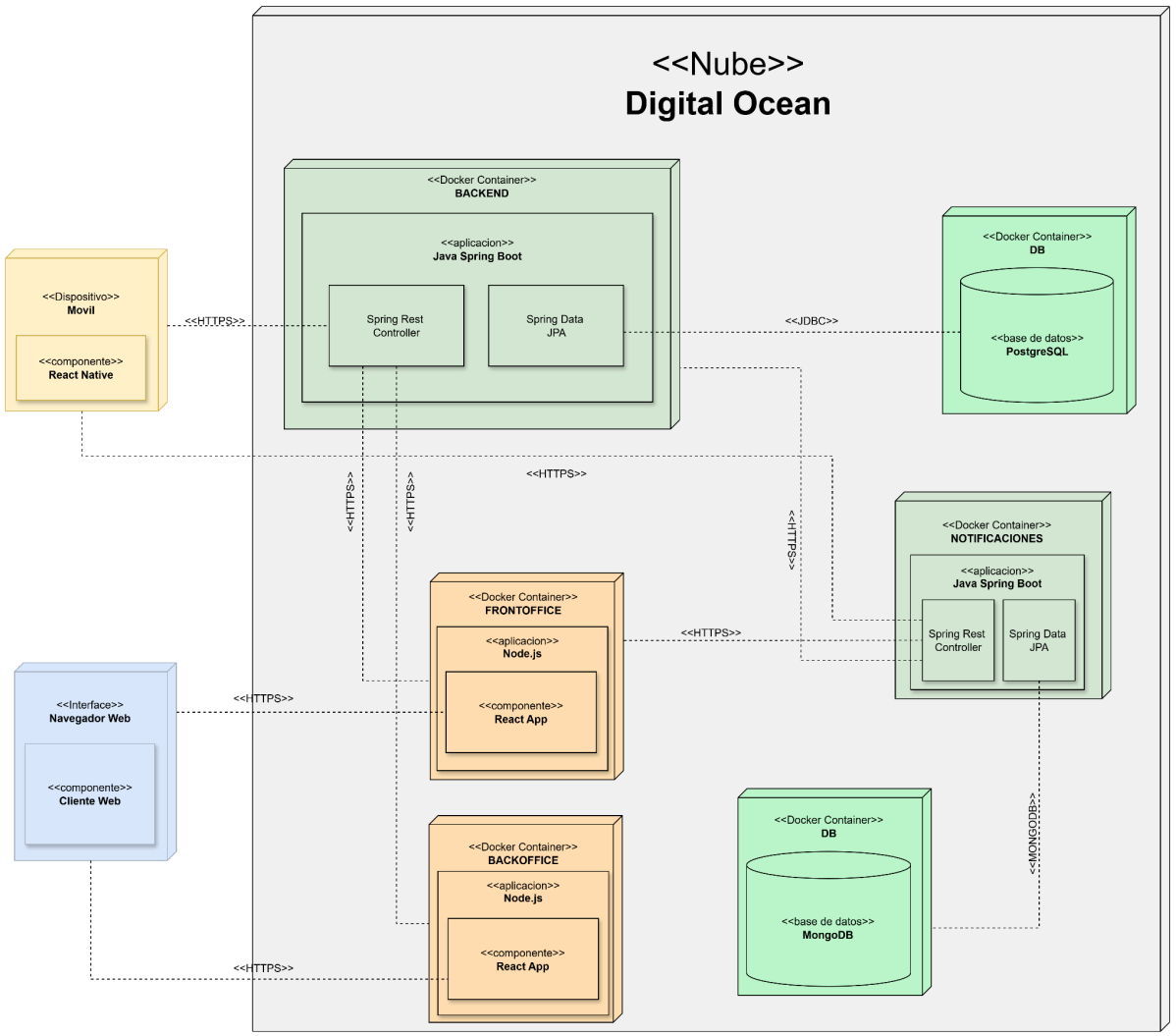
4.6.1.3 Capa de Datos

Esta capa gestiona el almacenamiento y acceso a la información utilizada por la capa de negocio.
Está compuesta por dos sistemas de gestión de bases de datos:

- **PostgreSQL**, utilizado como base de datos principal del sistema, responsable de almacenar la información estructurada gestionada por el backend.
- **MongoDB**, utilizado por el servicio de notificaciones para almacenar los tokens de dispositivos y los mensajes de cada notificación enviada.
  La capa de datos garantiza la persistencia, seguridad y eficiencia en el manejo de la información de todo el sistema.

4.6.2 Modelo de despliegue

A continuación se presenta un diagrama que describe el sistema en su totalidad y la arquitectura que lo compone, centrándose en la estructura interna, los componentes tanto internos como externos, y las conexiones que posibilitan el intercambio de datos entre ellos.



Todos los nodos de la aplicación se encuentran desplegados en la nube, específicamente en la infraestructura de Digital Ocean, con excepción de los clientes.

El cliente móvil, implementado con React Native, se ejecuta localmente en el dispositivo del usuario. De forma similar, el cliente web, accesible mediante un navegador, también se ejecuta en el entorno local del usuario y actúa como interfaz de interacción con la aplicación.

En el entorno cloud, la arquitectura consta de varios contenedores Docker, cada uno con un rol específico:

- El contenedor **BACKEND** encapsula la estructura principal de la aplicación, desarrollada con Spring Boot. Aquí se gestiona toda la lógica de negocio de

Tecnobus, mediante controladores REST (Spring Rest Controller) para manejar solicitudes HTTP y una capa de persistencia (Spring Data JPA) que interactúa con la base de datos principal.

- El contenedor **NOTIFICACIONES** alberga un segundo servicio basado también en Spring Boot, responsable de gestionar el sistema de notificaciones push que reciben los clientes, tanto en la aplicación web como en la móvil. Este servicio también sigue una arquitectura REST.
- El contenedor **FRONTOFFICE** aloja una aplicación construida con React, servida por un entorno Node.js, y representa la interfaz de usuario final, es decir, la utilizada por los clientes de Tecnobus.
- El contenedor **BACKOFFICE**, también construido con React y servido por Node.js, representa la interfaz utilizada por administradores y vendedores, proporcionando acceso a herramientas de gestión y administración del sistema.
- La infraestructura cuenta con dos contenedores de bases de datos:
    - El contenedor PostgreSQL, utilizado como base de datos principal para el backend, donde se almacenan los datos persistentes de la aplicación. La comunicación entre el backend y esta base se realiza mediante el protocolo JDBC.
    - El contenedor MongoDB, utilizado por el servicio de notificaciones para almacenar los tokens de los dispositivos y los mensajes asociados a cada notificación enviada. La comunicación entre el servicio de notificaciones y esta base de datos se realiza mediante el protocolo mongodb.

Toda la comunicación entre los distintos componentes desplegados en la nube se realiza mediante el protocolo HTTPS, garantizando la confidencialidad e integridad de los datos transmitidos.

4.7 Diseño del sistema

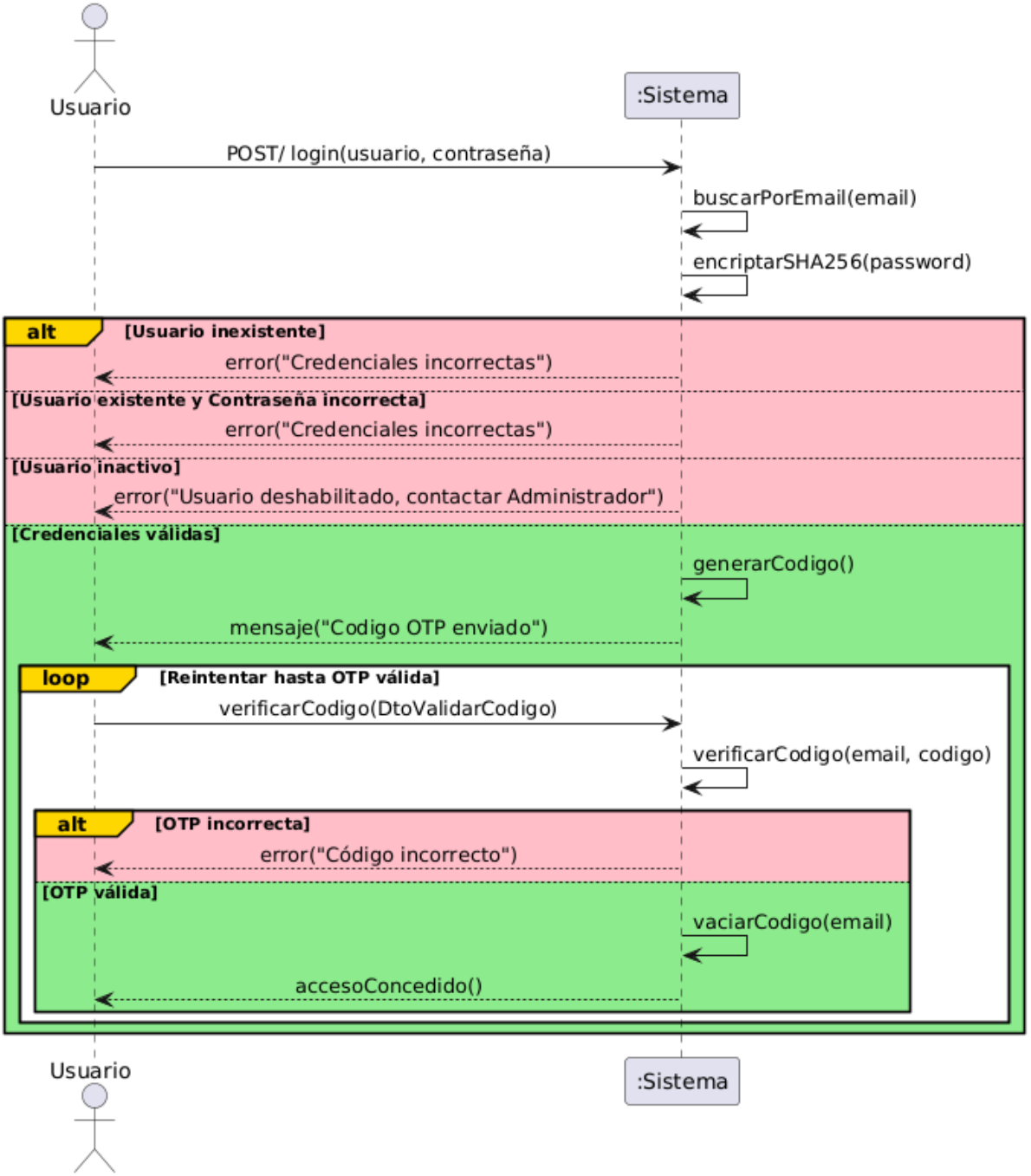
4.7.1 Diagrama de Clases

4.7.2 Diagramas de secuencia

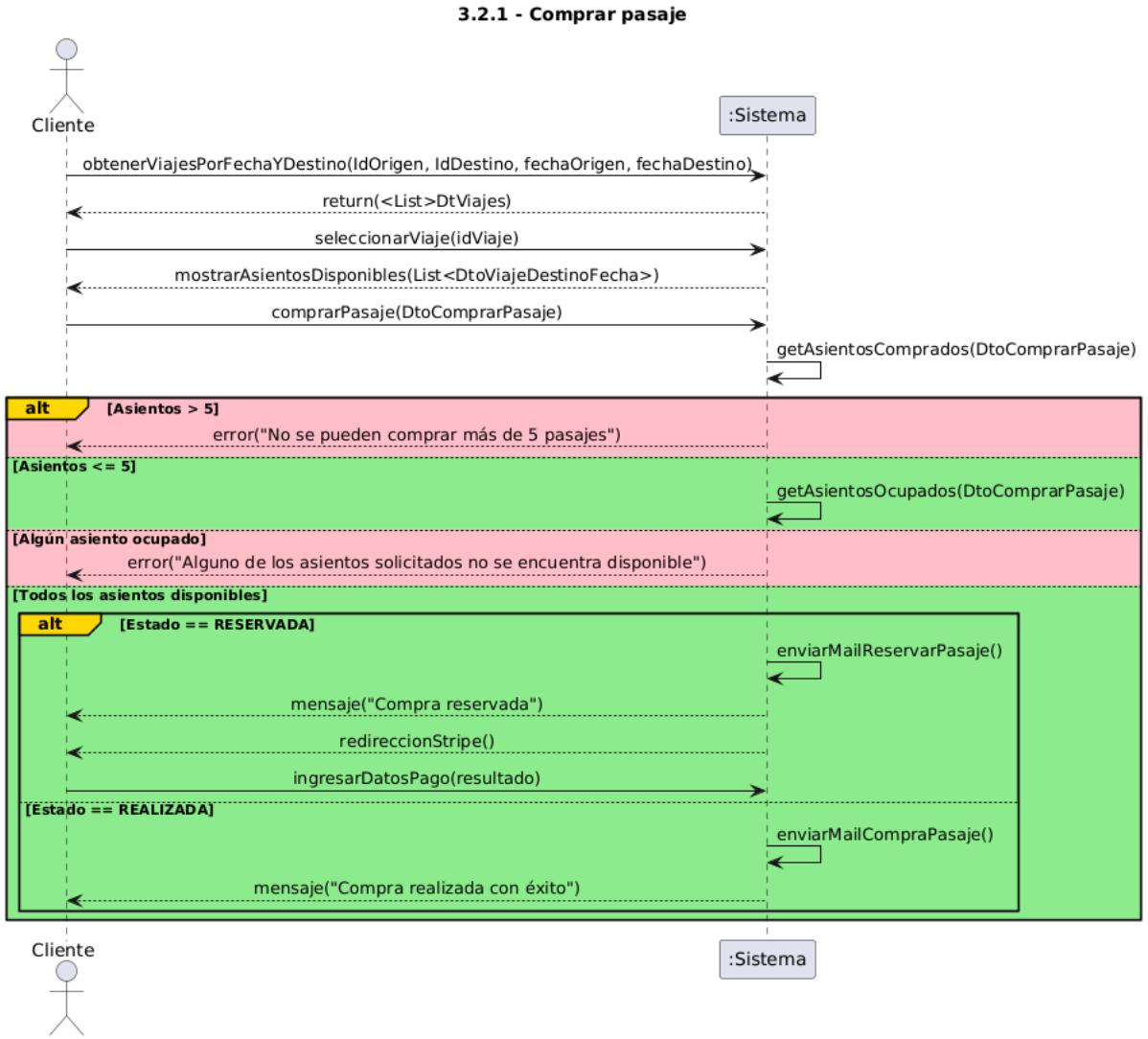
En la siguiente sección se detallan los diagramas de secuencia de algunos de los casos de uso considerados críticos al funcionamiento del prototipo presentado. Para

visualizar los diagramas correspondientes a los demás casos de uso críticos se puede consultar el documento anexo de diseño.

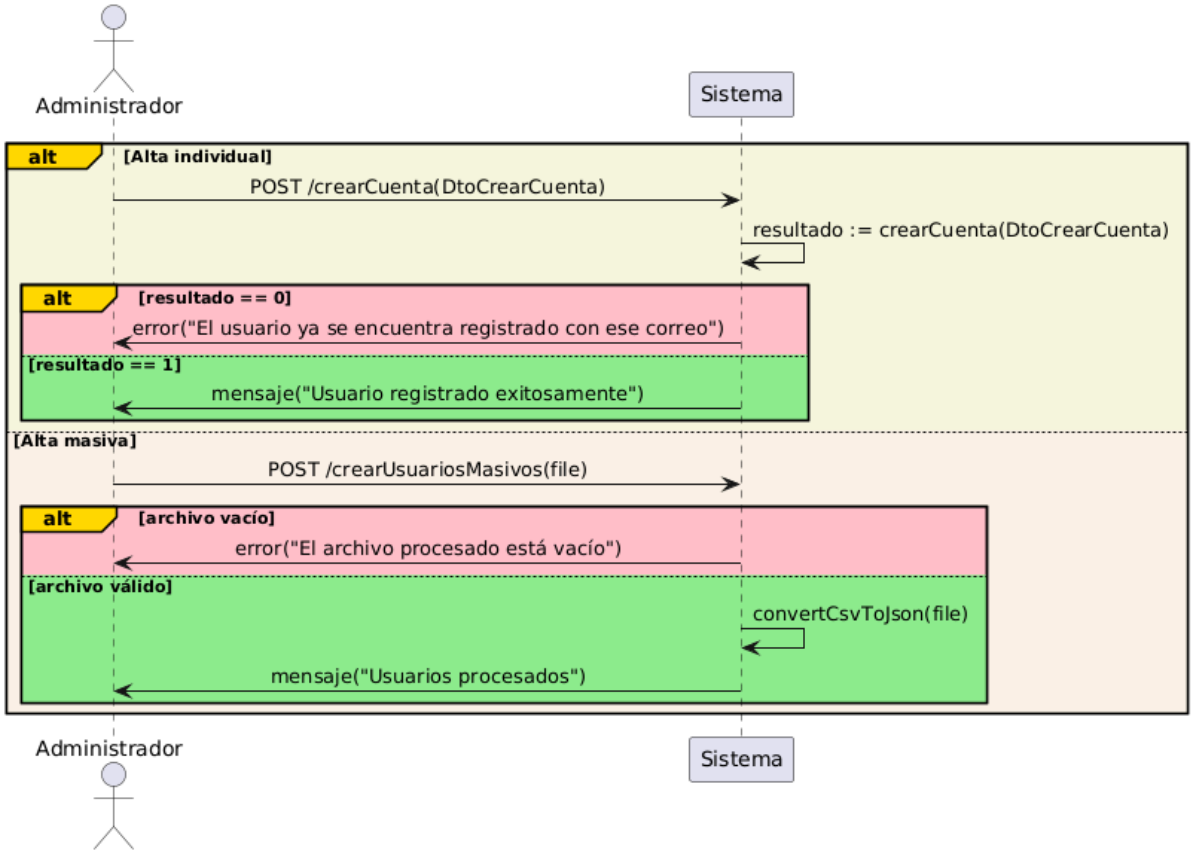
Iniciar sesión



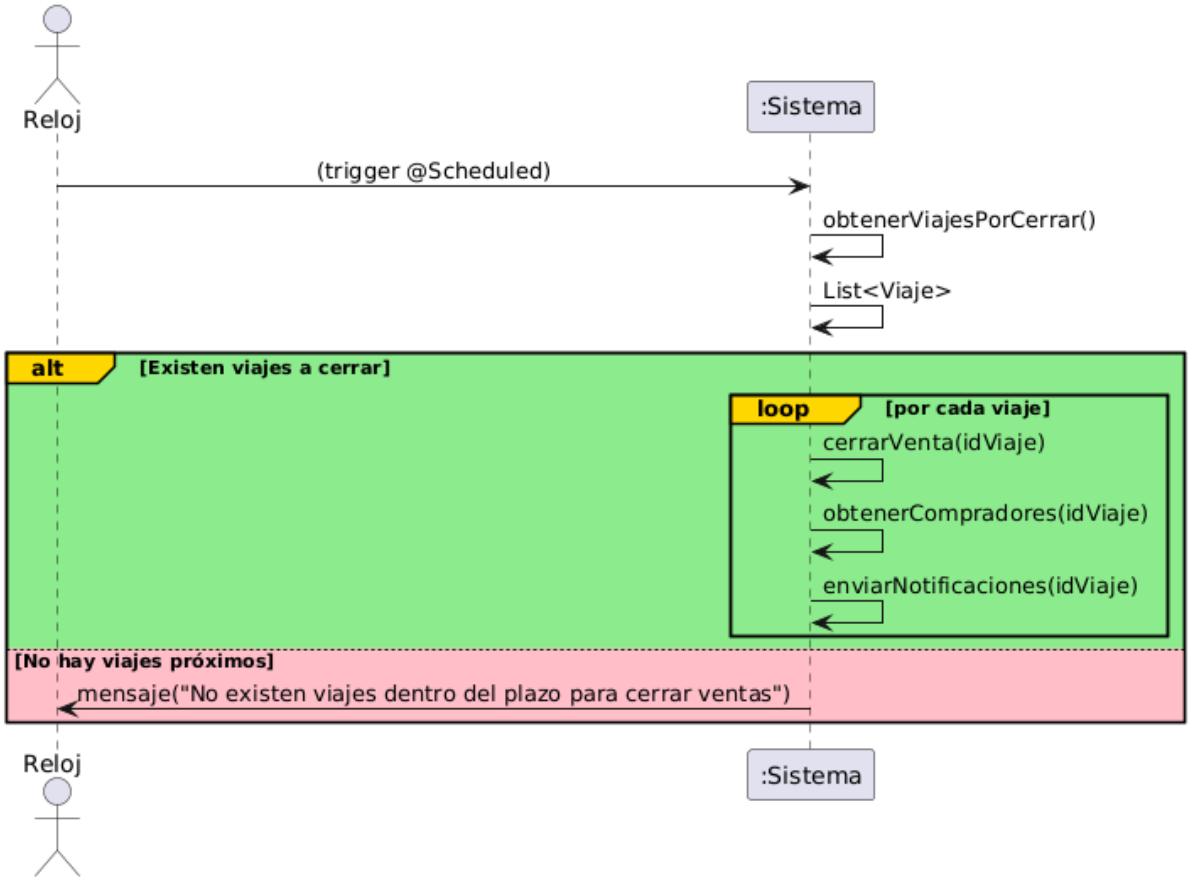
Compra de pasaje



Alta de usuario



Cierre de ventas de pasajes



5. Gestión del proyecto

En este capítulo se proporciona una visión detallada del proceso de trabajo llevado a cabo durante la creación de la aplicación TecnoBus. Se describen las metodologías empleadas, los roles asignados a los miembros del equipo y la planificación establecida para llevar a cabo el desarrollo del proyecto.

5.1 Metodología de trabajo

Durante la implementación del proyecto TecnoBus se adoptó un enfoque de trabajo estructurado por etapas. En el primer mes se llevaron a cabo pruebas de concepto, en las que se analizaron distintas tecnologías con el objetivo de determinar su pertinencia para el sistema. Los resultados de esta evaluación se documentan en detalle en el anexo "Documento de Pruebas de Concepto". El segundo mes estuvo dedicado al análisis y diseño del sistema. El tercer mes se orientó a la implementación, distribuyendo el trabajo en dos sprints de dos semanas cada uno. El primer sprint se centró en los casos de uso críticos, mientras que el segundo abordó el resto de las funcionalidades necesarias. Esta planificación permitió establecer prioridades claras y adaptarse de forma flexible a posibles ajustes durante el proceso de desarrollo. El cuarto mes se destinó a la incorporación de funcionalidades adicionales no requeridas explícitamente, así como a la realización de pruebas y a la elaboración de la documentación técnica del sistema. Esta etapa final permitió consolidar la calidad del producto y asegurar su correcto funcionamiento antes de su entrega.

Para la organización y comunicación interna del equipo se utilizaron herramientas como WhatsApp y Discord, que facilitaron la interacción ágil entre los integrantes. La coordinación y planificación de tareas se gestionó mediante Trello y Google Docs, aprovechando funciones como casillas de verificación y tablas que permitieron realizar un seguimiento en tiempo real del progreso.

En cuanto al control de versiones y la gestión del código fuente, se utilizó GitHub como plataforma central. Para la documentación del proyecto, se recurrió a Google Drive como repositorio de archivos, complementado con Google Docs para la

redacción técnica, Draw.io y PlantUML para la elaboración de diagramas, incluyendo el modelo de dominio y los diagramas de secuencia. Esta metodología contribuyó a establecer una base organizativa robusta y favoreció una colaboración eficiente entre los miembros del equipo durante todas las fases del desarrollo.

5.2 Distribución de roles del equipo y responsabilidades

5.2.1 Desarrollador Back-End

Responsable del desarrollo de la lógica de negocio y la implementación de los servicios del sistema TecnoBus. Encargado de diseñar e implementar las API REST, gestionar el acceso a la base de datos mediante JPA/Hibernate, y aplicar principios de seguridad y buenas prácticas en el manejo de la lógica del servidor. Además, colabora estrechamente con el equipo de Front-End y Mobile para asegurar la correcta integración entre capas.

5.2.2 Desarrollador Front-End

Encargado de implementar la interfaz de usuario web del sistema TecnoBus, para lo cual se definieron dos perfiles diferenciados: un frontoffice, destinado a la interacción del cliente final, y un backoffice, orientado a los usuarios con roles administrativos y de gestión (administrador y vendedor). Esta separación permitió diseñar experiencias específicas según las necesidades y permisos de cada tipo de usuario. Se emplearon tecnologías como HTML, CSS, JavaScript y React, garantizando una interfaz accesible y centrada en la usabilidad. Además, los desarrolladores fueron responsables de consumir los servicios provistos por el Back-End, realizar validaciones del lado del cliente y mantener la coherencia visual en ambas interfaces.

5.2.3 Desarrollador Mobile

Encargado del desarrollo de la aplicación móvil orientada al usuario cliente del sistema TecnoBus. Implementa funcionalidades clave como la compra de pasajes,

visualización de viajes disponibles, historial de compras y gestión de cuenta, utilizando tecnologías como React Native y Expo. Asegura la compatibilidad multiplataforma y una experiencia adaptada a dispositivos móviles.

5.2.4 Tester

Responsable de diseñar y ejecutar pruebas funcionales, unitarias y de integración. Verificó el cumplimiento de los requisitos definidos, detectando errores y validando la estabilidad de las funcionalidades implementadas. Utilizó herramientas como JUnit y Mockito para las pruebas del Back-End, Playwright para automatización en el Front-End, y pruebas de flujo para la parte mobile, contribuyendo así a garantizar la calidad del producto final.

5.2.5 DevOps

Encargado de la configuración del entorno de desarrollo, la integración continua, el control de versiones y la infraestructura de despliegue del sistema TecnoBus. Utilizó herramientas como GitHub, Docker y scripts de automatización para asegurar un desarrollo colaborativo eficiente y una implementación segura. También se ocupó de la puesta en producción del sistema en la nube, gestionando la adquisición de un dominio propio y configurando todos los elementos necesarios para disponer de URLs funcionales, asegurando así la disponibilidad del sistema para pruebas, presentaciones y uso final por parte de los distintos perfiles de usuario.

5.2.6 Analista Funcional

Responsable de relevar, documentar y analizar los requerimientos funcionales del sistema. Elabora los casos de uso, diagramas de interacción y documentos clave como el Documento de Requerimientos y el Documento de Diseño. Actúa como nexo entre los objetivos del proyecto y la implementación técnica, asegurando que las funcionalidades desarrolladas respondieran a las necesidades planteadas.

5.2.7 Administrador de Base de Datos (DBA)

Responsable del diseño, implementación y mantenimiento del esquema de base de datos del sistema TecnoBus. Se encarga de definir la estructura lógica de los datos, establecer relaciones entre entidades y optimizar consultas para garantizar un rendimiento eficiente. Además, supervisa la integridad y consistencia de la información, gestiona respaldos periódicos y colabora con el equipo de desarrollo Back-End para facilitar el acceso seguro y estructurado a los datos mediante JPA/Hibernate. Su labor es clave para asegurar la estabilidad, seguridad y escalabilidad del componente de persistencia del sistema.

5.2.8 Coordinador

Encargado de la planificación general del proyecto, distribución de tareas y seguimiento del avance. Facilita la comunicación entre los miembros del equipo y promueve el cumplimiento de los plazos establecidos en cada etapa. Además, supervisa la calidad de los entregables y se asegura de mantener una documentación actualizada, impulsando una gestión eficiente de tiempos y recursos.

Teniendo en cuenta la experiencia, preferencias, oportunidades de aprendizaje de los integrantes del grupo y los roles previamente descritos se dispuso la siguiente distribución de roles, siendo el primer rol de cada integrante su rol principal:

- Mathías Fernández
  - **Desarrollador Mobile**
  - Coordinador
- Eduardo Flores
  - **Desarrollador Back-End**
  - DBA
- Federico Acosta
  - **Desarrollador Front-End** (backoffice)
  - Tester
- Gabriel Suárez
  - **Desarrollador Front-End** (frontoffice)

- ➢ Analista Funcional
- ❖ Pablo Perdomo
  - ➢ **DevOps**
  - ➢ Desarrollador Back-End (notificaciones)
- ❖ Alexander Rodriguez
  - ➢ **Analista Funcional**
  - ➢ Tester

5.3 Planificación

En la etapa de implementación, se dividen las funcionalidades a desarrollar en dos *sprints* de dos semanas cada uno. El primer *sprint* está enfocado en la implementación de las funcionalidades críticas, mientras que en el segundo sprint se dedica a desarrollar funcionalidades secundarias.

5.3.1 Primer Sprint

Semana 1

- Inicio de sesión.
- Cierre de sesión.
- Alta administrador.
- Alta vendedor.
- Alta cliente.
- Alta ciudad.
- Alta ómnibus.
- Alta viaje.

Semana 2

- Edición perfil.
- Cambio contraseña.
- Recuperación de contraseña.
- Baja/desactivar usuarios
- Compra pasajes
- Venta pasajes
- Reasignación de ómnibus a un viaje

- Cierre de ventas de pasajes para un viaje.
- Marcado de un ómnibus como activo/inactivo.
- Nuevo ómnibus para viaje existente

5.3.2 Segundo Sprint

Semana 3

- Listado histórico de pasajes comprados por el cliente.
- Calificación de viajes por parte del cliente.
- Listado y búsqueda de viajes asignados a un ómnibus.
- Listado y búsqueda de usuarios.
- Listado y búsqueda de ómnibus.
- Listado y búsqueda de viajes.
- Listado de pasajes vendidos para un viaje.

Semana 4

- Devolución de un pasaje previamente comprado por un cliente.
- Estadísticas de viajes, ómnibus y pasajes.
- Estadísticas de usuarios.
- Baja de viaje.
- Baja de ómnibus.
- Baja de localidad.
- Push notification.

5.4 Distribución horaria

En esta sección se presenta una comparación entre las horas estimadas y las horas reales dedicadas a las actividades del proyecto.

Actividad	Horas Estimadas	Horas Reales	Diferencia
Monitoreos con tutor	6	6	0
Coordinaciones grupales	45	60	+15

Configuración de entornos de desarrollo	20	10	-10
Documento de Estado del Arte	35	25	-10
Documento de Pruebas de Concepto	95	120	+25
Documento de Alcance	20	25	+5
Documento de Requerimientos	30	20	-10
Documento de Casos de Usos	80	110	+30
Documento de Modelos de Dominio	15	20	+5
Glosario	5	5	0
Documento de Arquitectura	20	20	0
Documento de Diseño	50	75	+25
Documento de Modelo de Datos	40	45	+5
Documento de Verificación	20	35	+15
Implementación del Prototipo Funcional	450	550	+100
Testing	100	130	+30
Informe Final	50	60	+10
Preparación y Defensa final	5	10	+5
TOTAL	**1086**	**1326**	**+240**

La estimación inicial del esfuerzo requerido para el desarrollo del proyecto TecnoBus se basó en una dedicación semanal promedio de entre 11 y 12 horas por cada uno de los seis integrantes del equipo, lo cual representa una capacidad total aproximada de 72 horas semanales. Esta estimación se mantuvo como base para todas las actividades planificadas, incluyendo análisis, diseño, desarrollo, pruebas y documentación.

Durante el desarrollo, se registraron algunas desviaciones respecto a la planificación original. Una parte importante de la diferencia horaria se debió a la decisión estratégica de incrementar la frecuencia y duración de las reuniones internas. Esta medida fue adoptada con el objetivo de fortalecer la comunicación entre los

integrantes, compartir avances en tiempo real y facilitar una mayor coordinación en la toma de decisiones.

En el caso de las pruebas de concepto, si bien algunas tecnologías fueron descartadas tempranamente, se optó por continuar realizando pruebas complementarias para confirmar de forma definitiva su viabilidad o descarte, minimizando así el riesgo de tomar decisiones prematuras basadas en errores puntuales.

El documento de Casos de Uso experimentó múltiples revisiones a lo largo del proceso. Estas modificaciones respondieron a ajustes en el diseño de los flujos funcionales con el objetivo de mejorar la experiencia del usuario final. Dichos cambios en la concepción de la interfaz y en la lógica de navegación implicaron la necesidad de alinear constantemente los casos de uso con las decisiones de implementación.

La etapa de implementación fue la que presentó la mayor desviación en horas reales frente a lo estimado. Diversos obstáculos técnicos, detallados en la sección correspondiente, requirieron esfuerzos adicionales y provocaron que el equipo debiera superar las 12 horas semanales previstas por integrante. A su vez, para evitar una concentración excesiva de tareas hacia el final del cronograma, las actividades de testing comenzaron en paralelo durante la última semana de desarrollo. Esto implicó la necesidad de rehacer o ajustar algunos tests unitarios y funcionales, debido a cambios recientes en los servicios evaluados.

Esta dinámica adaptativa permitió mantener el control del proyecto y garantizar la entrega de un producto funcional, priorizando siempre la calidad técnica y la coherencia con los objetivos establecidos.

5.5 Problemas encontrados

5.5.1 Backend

Cross-Origin Resource Sharing (CORS)

En las fases iniciales del desarrollo del backend, se presentó un problema relacionado con CORS (Cross-Origin Resource Sharing), que impedía al frontend realizar solicitudes HTTP de modificación, como POST, PUT o DELETE. La causa radicaba en la falta de los encabezados CORS necesarios en las respuestas del servidor. Tras detectar la raíz del inconveniente, se implementó la configuración adecuada utilizando la interfaz WebMvcConfigurer de Spring Boot, definiendo los orígenes autorizados, los métodos HTTP permitidos y los encabezados requeridos.

5.5.2 Frontend

Separación del Frontend

Durante el desarrollo, se identificó que un único frontend dificultaba la experiencia de uso debido a la diversidad de roles y funcionalidades. Por esta razón, se optó por dividir la interfaz en dos aplicaciones diferenciadas: una de backoffice (web) destinada a administradores y vendedores, enfocada en la gestión operativa; y una de frontoffice (mobile), dirigida exclusivamente al cliente final, optimizada para la compra de pasajes y consulta de viajes. Esta separación permitió simplificar la navegación, mejorar la usabilidad y adaptar cada interfaz a las necesidades específicas de sus usuarios.

Simulación de pagos

Inicialmente se planificó la integración de PayPal como pasarela de pago, dado su amplio reconocimiento y facilidad de uso para los clientes. Sin embargo, durante la etapa de pruebas surgieron dificultades técnicas relacionadas con la autenticación y configuración de entorno en modo sandbox, lo que ralentizaba el desarrollo. Como solución, se decidió utilizar Stripe, que ofreció una integración más sencilla, mejor documentación y compatibilidad directa con las librerías utilizadas en el frontend.

5.5.3 Mobile

Generación de APK

En la etapa mobile, uno de los principales inconvenientes fue el tiempo excesivo que insumía la generación del archivo APK mediante herramientas gratuitas. Esto impactaba negativamente en el ciclo de pruebas y ajustes. Para mitigar este problema, se optó por suscribirse a un plan de pago que permitiera generar el APK de forma más ágil y estable, acelerando significativamente la validación de cambios y mejorando la eficiencia del desarrollo mobile.

Simulación de pagos

Inicialmente se planificó la integración de PayPal como pasarela de pago, dado su amplio reconocimiento y facilidad de uso para los clientes. Sin embargo, durante la etapa de pruebas surgieron dificultades técnicas derivados de limitaciones de Expo, específicamente por incompatibilidades con las librerías requeridas para la conexión con PayPal.

6. Implementación

6.1 Tecnologias aplicadas

En esta sección se describen las tecnologías utilizadas durante la implementación de la aplicación TecnoBus. La elección de cada herramienta se fundamentó en la experiencia del equipo, los requerimientos técnicos del proyecto y los resultados obtenidos en la fase de pruebas de concepto. Para un análisis detallado de las evaluaciones previas, se recomienda consultar el anexo "Documento de Pruebas de Concepto".

6.1.1 Backend

El lenguaje de programación seleccionado para el backend de la aplicación es Java, cumpliendo así con uno de los requisitos no funcionales del proyecto. Específicamente, se utilizará la versión 17. El sistema será implementado utilizando el framework Spring Boot, decisión basada tanto en la familiaridad del equipo con esta tecnología como en su capacidad para facilitar un despliegue rápido de la aplicación.

6.1.2 Frontend

React fue seleccionado debido a la experiencia previa del equipo, la disponibilidad de componentes reutilizables y el amplio soporte de la comunidad, factores que permiten acelerar el desarrollo. Sumado a lo anterior, React es una tecnología que se adapta bien tanto a web como a mobile, simplificando y ahorrando problemas causados por la mezcla de tecnologías. Si bien también se realizaron Pruebas de Concepto en Angular ésta tecnología se descartó según el análisis explicitado en el Documento de Pruebas de Conceptos.

6.1.3 Mobile

Se seleccionó React Native como tecnología móvil por su integración natural con React, permitiendo aprovechar el conocimiento existente del equipo y facilitar la reutilización de lógica entre plataformas. Debido a que el equipo está familiarizado con los conceptos de React, la sintaxis JSX, la gestión del estado y los patrones de

componentes se puede transferir en gran parte el conocimiento directamente a React Native, reduciendo significativamente la curva de aprendizaje en comparación con otro tipo de desarrollo (ej.: Swift/Kotlin). Además, esto abre la puerta a la reutilización de código, especialmente en la lógica de negocio, lo que acelera el desarrollo de la aplicación móvil y facilita la coherencia entre la web y mobile.

6.1.4 Base de datos y persistencia

Para la persistencia de datos se utilizó PostgreSQL. La elección de postgreSQL se debe a su facilidad de instalación y estabilidad, además, los integrantes del equipo contaban con experiencia utilizando esta tecnología.

6.1.5 Servidor de aplicaciones

La arquitectura fue desplegada utilizando contenedores Docker en la nube de DigitalOcean, permitiendo segmentar los entornos de backend, frontend y base de datos en instancias independientes. El backend se ejecuta en un contenedor con Spring Boot, el frontend es servido a través de Node.js, y la base de datos principal se gestiona en un contenedor PostgreSQL y MongoDB para las notificaciones. Las conexiones se realizan sobre protocolo HTTPS, y el backend se comunica con la base de datos principal mediante JDBC y las notificaciones mediante el protocolo de Mongo.

6.1.6 Testing

Las pruebas unitarias y de integración fueron desarrolladas utilizando JUnit y Mockito, herramientas estándar en el ecosistema Java. Estas pruebas permitieron validar la lógica de negocio de los servicios y la correcta respuesta de los controladores REST. Para los tests de extremo a extremo (E2E) sobre la interfaz web, se utilizó Playwright lo que permitió automatizar flujos funcionales completos. En el entorno mobile, se realizaron pruebas funcionales manuales desde dispositivos físicos, ejecutando los principales flujos del cliente, como el inicio de sesión, la compra de pasajes y la consulta del historial. Estas pruebas permitieron detectar errores visuales, validar la navegación entre pantallas y comprobar la experiencia general del usuario en dispositivos reales.

6.1.7 Gestión de versiones y DevOps

Para la gestión del código fuente se utilizó Git como sistema de control de versiones distribuido, con repositorios alojados en GitHub, lo que permitió mantener un flujo de trabajo colaborativo, trazabilidad de los cambios y control de ramas. En cuanto a prácticas de DevOps, se emplearon contenedores Docker para facilitar la portabilidad y consistencia de los entornos, especialmente en la etapa de despliegue en la nube. Además, se configuraron scripts de automatización básicos para la ejecución local de servicios y pruebas, asegurando coherencia entre los entornos de desarrollo y producción. Esta estrategia permitió reducir errores de integración, acelerar el ciclo de entrega y asegurar una mayor estabilidad del sistema en cada iteración.

6.1.8 Entorno de Desarrollo

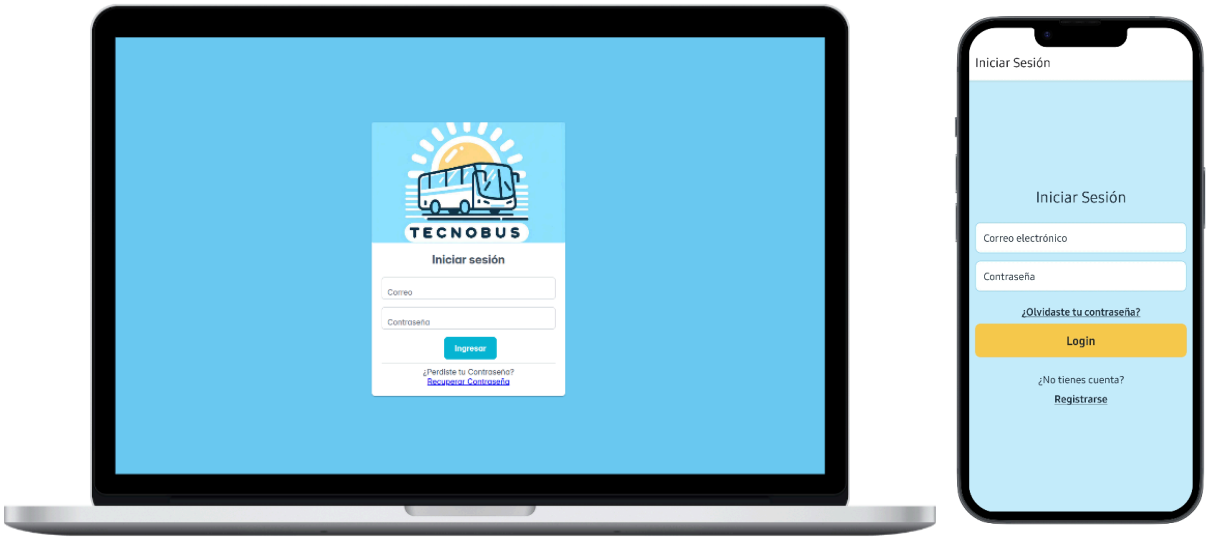
El entorno de desarrollo utilizado para el backend fue Eclipse IDE, aprovechando su integración con proyectos Java y herramientas como Maven y Spring Boot. Para el desarrollo frontend (React) se utilizó Docker Desktop para la puesta en marcha de los contenedores y se utilizó Visual Studio Code, debido a su ligereza, extensibilidad y compatibilidad con entornos JavaScript así como también para mobile (React Native). La validación de la experiencia móvil se realizó mediante emuladores y dispositivos físicos. Asimismo, se empleó Postman para testear endpoints REST y validar la interacción con la API de backend.

6.2 Aplicación desarrollada

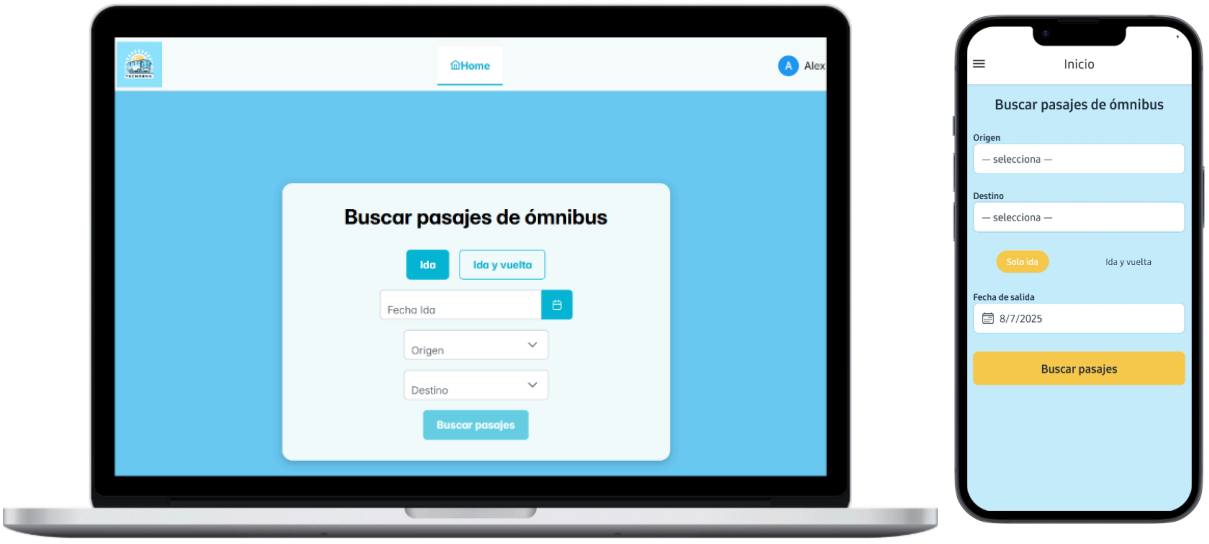
A continuación se presentan, a modo ilustrativo, algunas de las pantallas (mockups) diseñadas durante la etapa previa a la implementación del sistema TecnoBus. Estos diseños sirvieron como referencia para el desarrollo de las interfaces gráficas en los entornos web y mobile, asegurando coherencia visual y funcional entre ambos. El conjunto completo de pantallas será exhibido durante la demostración funcional de la aplicación en la instancia de defensa del proyecto.

6.2.1 Pantallas principales del sistema

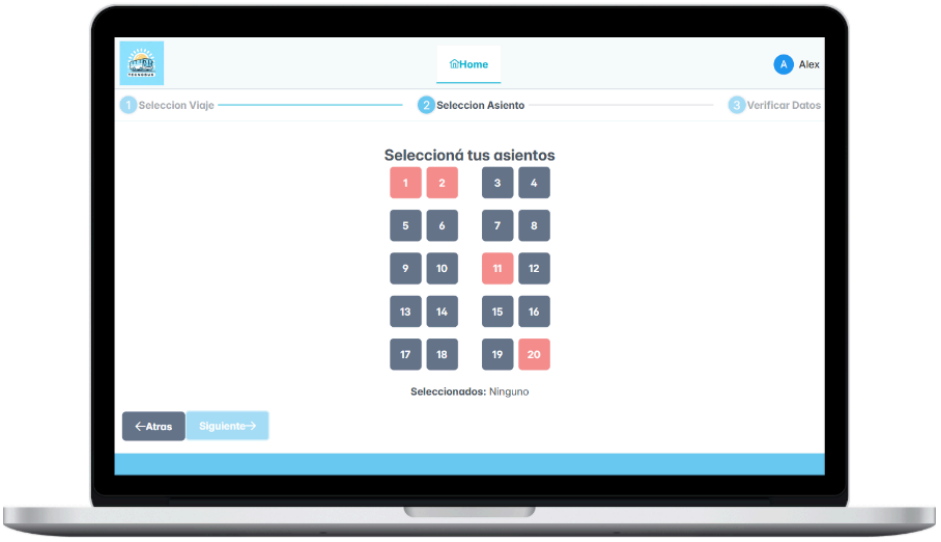
Inicio de sesión



Pantalla principal cliente



Selección de asientos


Home
Alex
Seleccion Viaje
Seleccion Asiento
Verificar Datos
Seleccioná tus asientos
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
Seleccionados: Ninguno
←Atras
Siguiente→


Seleccionar Bus
Seleccioná tu asiento
Disponible
Seleccionado
Ocupado
Asientos Ida
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
Ir a pagar

7. Testing

Entre los documentos anexos se incluyó el Documento de Verificación, en el cual se profundiza en los conceptos tratados en esta sección. Se sugiere consultar dicho documento para ampliar información con relación al testing de la aplicación.

7.1 Pruebas unitarias

Las pruebas unitarias se enfocan en validar el comportamiento de métodos o clases individuales del backend, en particular aquellos que implementan la lógica de negocio en servicios desarrollados con Java y Spring Boot. Estas pruebas se ejecutan de forma aislada y sin involucrar capas externas del sistema, utilizando simulaciones (mocks) de dependencias a través de frameworks como JUnit y Mockito. Permiten verificar la consistencia lógica de cada componente y detectar errores de forma rápida y localizada.

7.2 Pruebas E2E

Simulan la experiencia de un usuario real al interactuar con la aplicación web desde el navegador, reproduciendo de forma automatizada los pasos clave de los flujos más relevantes, como el proceso de compra de pasajes o la calificación de viajes. Estas pruebas permiten validar tanto la respuesta visual de la interfaz como la correcta interacción con los servicios backend. Además de verificar que los flujos completos funcionen de extremo a extremo, ayudan a identificar errores que solo se presentan en escenarios integrados. En este proyecto, su uso fue limitado a recorridos funcionales críticos, pero resultaron esenciales para comprobar la coherencia general del sistema desde la perspectiva del usuario.

7.3 Pruebas funcionales de interfaz

Se realizaron de forma exploratoria tanto sobre la interfaz web como en la aplicación móvil. En el entorno mobile, las pruebas se llevaron a cabo manualmente desde dispositivos físicos, ejecutando flujos funcionales clave como el inicio de sesión, la compra de pasajes o la consulta del historial. Estas pruebas permitieron detectar problemas visuales, errores de navegación y validar la experiencia general del usuario.

7.4 Resultados obtenidos

Finalizada la etapa de verificación, los resultados obtenidos muestran que se logró un comportamiento sólido y estable en sus principales funcionalidades. A lo largo del proceso se aplicaron distintos tipos de pruebas, tanto automatizadas como manuales, que permitieron validar los flujos críticos del sistema, identificar fallos puntuales y asegurar que las funcionalidades implementadas respondieran correctamente a lo definido en las fases de análisis y diseño. El sistema demostró una respuesta consistente frente a escenarios reales de uso, tanto en la interfaz web como en la aplicación móvil.

A partir de los resultados obtenidos, se destacan los siguientes puntos:

- **Cobertura de código alta:** Se logró una cobertura del 97% de los 366 métodos identificados entre servicios y controladores. Este resultado incluye una verificación completa (100%) en todos los componentes considerados *críticos* para el funcionamiento del sistema.
- **Casos de uso críticos altamente verificados:** El 95% de los casos de uso críticos fueron validados mediante pruebas unitarias y E2E automatizadas, lo que confirma el funcionamiento correcto de los flujos esenciales tanto en frontend como backend.
- **Estabilidad funcional en Web y Mobile:** Las pruebas E2E ejecutadas con Playwright y las pruebas funcionales manuales realizadas en la aplicación mobile permitieron validar los flujos más relevantes del sistema, como la compra de pasajes, el historial de compras, la autenticación y la navegación

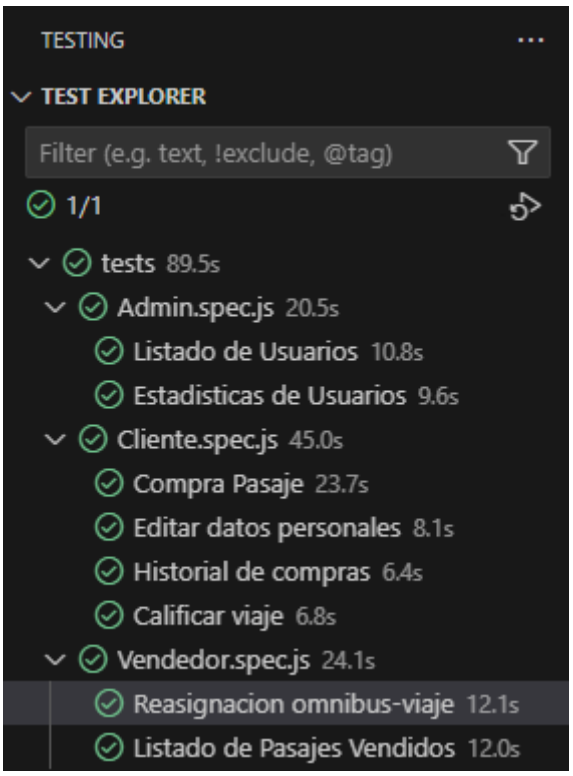
entre pantallas. Se comprobó la coherencia de comportamiento entre plataformas y se detectaron y corrigieron errores puntuales.

Element	Class, %	Method, %
com.example.Login.service	100% (15/15)	95% (105/110)
AsientoService	100% (1/1)	100% (3/3)
CategoriaService	100% (1/1)	100% (1/1)
CerrarViajeService	100% (1/1)	100% (1/1)
CompraPasajeService	100% (2/2)	100% (9/9)
EmailService	100% (1/1)	100% (1/1)
GenerarContraseniaService	100% (1/1)	100% (3/3)
JwtService	100% (1/1)	100% (6/6)
LocalidadService	100% (1/1)	100% (7/7)
OmnibusService	100% (1/1)	100% (11/11)
TokenService	100% (1/1)	100% (3/3)
TransactionalService	100% (1/1)	100% (1/1)
UsuarioService	100% (2/2)	100% (38/38)
ViajeService	100% (1/1)	80% (21/26)

Cobertura tests unitarios de servicios

Element	Class, %	Method, %
com.example.Login.controller	100% (10/10)	100% (73/73)
AsientoController	100% (1/1)	100% (2/2)
CategoriaController	100% (1/1)	100% (1/1)
CerrarViajeController	100% (1/1)	100% (1/1)
ComprasController	100% (1/1)	100% (3/3)
LocalidadController	100% (1/1)	100% (6/6)
OmnibusController	100% (1/1)	100% (10/10)
TokenController	100% (1/1)	100% (2/2)
UsuarioController	100% (2/2)	100% (30/30)
ViajeController	100% (1/1)	100% (18/18)

Cobertura tests unitarios de controladores



Tests E2E con Playwright

8. Conclusiones y trabajo a futuro

En este capítulo, se exponen las conclusiones derivadas del proceso de desarrollo, así como las posibles mejoras para futuras versiones.

8.1 Conclusiones finales

Desde el comienzo, la temática del proyecto resultó estimulante para el equipo de trabajo. La familiaridad previa de los integrantes con plataformas de compra de pasajes de ómnibus facilitó la comprensión del dominio y aportó claridad sobre las funcionalidades esperadas, lo cual fue especialmente valioso considerando los tiempos acotados para el desarrollo.

Durante la elaboración del estado del arte, se analizaron diversas soluciones disponibles en el mercado, lo que permitió identificar aspectos fundamentales en este tipo de plataformas: una experiencia de compra sencilla, beneficios adicionales para el usuario y una interfaz intuitiva. También se detectaron debilidades recurrentes en materia de seguridad, por lo que se decidió incorporar mecanismos de protección más robustos, incluso cuando no formaban parte de los requisitos iniciales.

Una vez consolidada una visión compartida del producto, se definieron las tecnologías a utilizar, priorizando criterios como el conocimiento previo del equipo, la facilidad de integración entre componentes y la curva de aprendizaje asociada a las herramientas nuevas. Finalmente, se optó por una arquitectura basada en tecnologías modernas y robustas: Java con Spring Boot para el backend, React para el frontend web y React Native para el desarrollo mobile.

La etapa de diseño resultó más extensa y detallada de lo previsto, requiriendo múltiples iteraciones para ajustar aspectos técnicos y funcionales. Sin embargo, este esfuerzo permitió establecer una base sólida para el desarrollo posterior, generando una trazabilidad clara entre las fases de análisis, diseño e implementación.

En balance, a pesar de los desafíos técnicos y organizativos, el equipo logró avanzar de forma ordenada gracias a la distribución efectiva de roles, la planificación compartida y la colaboración constante. La experiencia dejó aprendizajes significativos no solo en el plano técnico, sino también en el fortalecimiento de habilidades transversales como la comunicación, la resolución conjunta de problemas y la adaptación al cambio. El proceso de desarrollo de **TecnoBus** representó una instancia valiosa de crecimiento tanto profesional como humano para todos los involucrados.

8.2 Trabajo a futuro

En esta sección se exponen propuestas de trabajo futuro que incluyen mejoras y funcionalidades adicionales, identificadas desde las etapas iniciales como posibles ampliaciones a considerar una vez finalizado el desarrollo del prototipo funcional. Estas ideas surgen tanto de la planificación original como del análisis realizado durante el proceso de implementación y validación del sistema.

8.2.1 Sistema de Fidelización

Implementación de un sistema de fidelización que recompense la recurrencia de los clientes mediante la acumulación de puntos o beneficios.

8.2.2 Notificaciones SMS

Incorporación de notificaciones por SMS como canal complementario a los actuales medios de notificación (correo electrónico y notificaciones push).

8.2.3 Precios por Temporadas

Desarrollo de una lógica de precios dinámicos basada en variables como demanda, estacionalidad u otros criterios definidos por la empresa.

8.2.4 Gestión de convenios institucionales

Funcionalidad para la gestión de convenios institucionales con entidades educativas o empresas, que permita configurar condiciones especiales para compras grupales o masivas.

8.2.5 Soporte para operación multiempresa

Integración de múltiples empresas de transporte dentro de una misma plataforma, posibilitando una gestión multiempresa con control independiente de flotas, destinos y tarifas.

8.2.6 Módulo de gestión de encomiendas

Adición de un módulo completo para la gestión de encomiendas, con el fin de extender el alcance comercial del sistema más allá del transporte de pasajeros.

8.2.7 Alta automatizada de usuarios institucionales

Automatización del proceso de alta de usuarios institucionales bajo el dominio “@tecnobus.uy”, mediante la búsqueda inteligente en la base de datos de combinaciones posibles entre nombre y apellido.

8.2.8 Sistema inteligente de soporte y reclamos

Desarrollo de un sistema avanzado de soporte y gestión de reclamos, apoyado en tecnologías de inteligencia artificial, que permita registrar incidencias, realizar seguimientos y brindar respuestas multicanal de forma eficiente.

8.2.9 API pública para integración con terceros

Creación de una API pública que permita a otras empresas del sector integrar el sistema de Tecnobus en sus propias plataformas, facilitando la interoperabilidad comercial y la ampliación del alcance del servicio.

8.2.10 Inicio de sesión simplificado para perfiles internos

Simplificación del inicio de sesión para administradores y vendedores, permitiendo autenticar únicamente con el nombre de usuario en lugar de ingresar el correo electrónico completo.

8.2.11 No requerir registro obligatorio para comprar

Incorporación de una funcionalidad que permita a los usuarios realizar la compra de pasajes sin necesidad de crear una cuenta o iniciar sesión. Esta mejora apunta a optimizar la experiencia de usuarios ocasionales, reduciendo barreras de entrada y agilizando el proceso de compra. Previendo que esta opción conviva con el registro tradicional, permitiendo luego asociar la compra a una cuenta si el usuario lo desea.

9. Referencias

[1] - https://www.cot.com.uy

[2] - https://www.grupoagencia.com.uy

[3] - https://www.turil.com.uy

[4] - https://www.cutcorporacion.com.uy

[5] - https://www.omio.com

[6] - https://play.google.com/store/apps/details?id=uy.com.turil.twa&hl=es_UY

[7] - https://play.google.com/store/apps/details?id=uy.com.urubus&hl=es_UY

[8] - https://play.google.com/store/apps/details?id=de.flixbus.app&hl=es_UY

10. Anexos

En esta última sección se describe brevemente el propósito de cada uno de los documentos anexos a este informe, los cuales contienen información complementaria y detallada sobre los distintos capítulos del proyecto.

10.1 Documento de alcance

Define los límites del proyecto, especificando qué funcionalidades fueron incluidas, en qué fases se planificó su desarrollo y qué objetivos se buscó alcanzar. También presenta la planificación general del trabajo y su vinculación con las etapas del ciclo de vida del sistema.

10.2 Documento de requerimientos

Contiene el relevamiento de requerimientos funcionales y no funcionales, identificados a partir de las necesidades de los usuarios y los objetivos del proyecto. Sirve como base para las decisiones de diseño y validación del sistema.

10.3 Documento de casos de uso

El documento de casos de uso especifica, en formato estructurado, los distintos casos de uso del sistema TecnoBus. Incluye los actores involucrados, los flujos de eventos y las condiciones para el inicio y finalización de cada operación, facilitando la trazabilidad funcional.

10.4 Documento de modelo de dominio

Representa las entidades principales del sistema y sus relaciones mediante diagramas UML. Este modelo conceptual sirve de puente entre los requerimientos y la implementación concreta, y fue clave para definir la estructura de datos.

10.5 Documento de arquitectura

El documento de arquitectura describe la arquitectura general del sistema, incluyendo el paradigma adoptado (en capas), los componentes principales y su interacción. También incluye las decisiones arquitectónicas que guiaron la implementación de la solución.

10.6 Documento de diseño

Presenta el diseño detallado del sistema, incluyendo los diagramas de clases, diagramas de secuencia y las decisiones tomadas en relación a la estructura interna de los módulos. Refleja cómo se implementaron los casos de uso críticos y cómo se organizó el código.

10.7 Documento de modelo de datos

Especifica la estructura de la base de datos utilizada en el sistema, incluyendo las tablas, campos, claves primarias y foráneas. Define cómo persisten las entidades del dominio en la capa de datos.

10.8 Documento de verificación

Describe el enfoque y los resultados del proceso de testing aplicado al sistema. Incluye pruebas unitarias, pruebas de integración y pruebas end-to-end, con métricas de cobertura y detalle de las incidencias encontradas y corregidas.

10.9 Glosario

Incluye definiciones breves de los principales términos técnicos y funcionales utilizados en los documentos del proyecto, con el fin de facilitar la comprensión de los conceptos empleados por parte de lectores no especializados.