

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE



**“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de
Transporte Escolar”**

CARLA ROMINA GONZÁLEZ CORREA

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA**

Profesor Guía: Cristian Antonio Lara Valenzuela

Junio - 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Backend Furgo Finder: Solución integral para la gestión y contratación de transporte escolar.

Nombre del candidato(a): Carla Romina González Correa

Carrera / Grado: Ingeniería en Informática

Campus: Concepcion; Departamento: Electrónica e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, , Cristian Antonio Lara Valenzuela, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO** contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 30/06/2025

; Firma:

Estudiante o Candidato(a):

Fecha: 30/06/2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

“Dedico esta memoria con todo mi cariño a mi querida madre y a mi padrino, quienes fueron un pilar fundamental y un gran apoyo durante todo este viaje académico. Su amor, paciencia y confianza en mí me dieron la fuerza para superar cada desafío. También a mis amigos, que estuvieron a mi lado en todo el transcurso de mi carrera, ofreciéndome su apoyo incondicional, compañía y motivación. Gracias por ser parte de este importante capítulo de mi vida.

Y, por último, a todas las personas que me ayudaron a crecer como persona durante todos estos años, dejando una huella significativa en mi vida. Gracias por ser parte de este camino que me ha llevado hasta aquí”.

AGRADECIMIENTOS

Quisiera expresar mi mas profundo agradecimiento a mi madre Rosa Correa, y a mi padrino Manuel Correa. Durante todos estos años, han sido el motor que me impulso a seguir adelante y mi refugio en algunos momentos difíciles. Las palabras de aliento, su amor incondicional y el apoyo constante no solo me motivó a alcanzar mis metas, sino que también me enseñó el verdadero valor de la perseverancia y la dedicación. A ellos les debo no solo mis logros, sino también la persona que soy hoy.

A mis amigos y compañeros, quienes siempre estuvieron a mi lado ofreciendo su apoyo incondicional, les dedico un profundo agradecimiento. Han sido una fuente de inspiración y confianza, ayudándome a superar mis dudas e inquietudes con su paciencia y generosidad. Su compañía durante este proceso no solo enriqueció mi experiencia académica, sino que también transformó los momentos difíciles en oportunidades de aprendizaje y crecimiento. Cada conversación, cada consejo y cada gesto de apoyo quedarán grabados en mi corazón como un recordatorio de la importancia de la amistad y el trabajo en equipo.

Quiero también extender mi sincero reconocimiento a los funcionarios y docentes de la Universidad Técnica Federico Santa María, quienes con su compromiso y dedicación contribuyeron de manera significativa a mi formación profesional. En especial, agradezco a Cristian Lara, Viviana Delgado y Eduardo Urrutia, quienes confiaron en mis capacidades y me guiaron con sabiduría y empatía. Su orientación fue un pilar fundamental para mi desarrollo académico y personal, y siempre llevaré conmigo las valiosas lecciones que aprendí.

Finalmente, quiero agradecer a todas aquellas personas que, de manera directa o indirecta, dejaron una huella en este camino. A mi familia, por ser mi ancla en los momentos de incertidumbre; a mis amigos, por las risas y los recuerdos compartidos; y a cada persona que, con un gesto de apoyo o una palabra amable, contribuyó a que hoy pueda estar escribiendo estas palabras. Este logro no es solo mío, sino también de todos aquellos que creyeron en mí y me acompañaron en esta travesía.

A todos ustedes, gracias. Este es solo el comienzo de un nuevo capítulo, y cada paso que dé estará marcado por el amor, el apoyo y las enseñanzas que recibí de todos ustedes.

RESUMEN

FurgoFinder es una plataforma web y móvil que conecta a apoderados con conductores de transporte escolar, facilitando la contratación segura y la validación documental. Con un backend en Flask y una interfaz desarrollada en Spring Boot y Flutter, digitaliza procesos clave como registro, gestión de contratos y seguimiento, promoviendo la seguridad, eficiencia y formalización del sector.

Palabras clave: Transporte Escolar, Plataforma Web, Aplicación Móvil, Backend, Flask, Spring Boot, Contratación, Seguridad, Validación Documental, API REST.

ABSTRACT

Abstract— **FurgoFinder** is a web and mobile platform that connects guardians with school transport drivers, enabling secure hiring, document validation, and efficient service management. It features a Flask-based backend, a Spring Boot web frontend, and a Flutter mobile app, digitalizing a traditionally informal process and improving the safety and traceability of school transportation.

Keywords— School Transportation, Web Platform, Mobile Application, Backend, Flask, Spring Boot, Hiring, Security, Document Validation, REST API.

GLOSARIO

AWS (Amazon Web Services): Plataforma de servicios en la nube que ofrece soluciones de almacenamiento, cálculo y despliegue de aplicaciones.

API REST: Interfaz de programación de aplicaciones que utiliza el protocolo HTTP para la comunicación entre sistemas, permitiendo operaciones como creación, lectura, actualización y eliminación de datos.

Back-End: Parte del desarrollo web que se encarga de la lógica del negocio y la interacción con la base de datos.

BCrypt: Algoritmo de hashing utilizado para encriptar contraseñas de manera segura.

CRUD: Acrónimo de Create, Read, Update, Delete, que define las operaciones básicas sobre bases de datos.

DynamoDB: Base de datos NoSQL de AWS para almacenamiento de datos no estructurados.

FurgoFinder: Plataforma diseñada para facilitar la contratación y gestión de transporte escolar de forma segura y eficiente.

JWT (JSON Web Token): Estándar para la autenticación y autorización mediante el uso de tokens seguros.

MySQL: Sistema de gestión de bases de datos relacional utilizado para almacenar información estructurada.

Spring Boot: Framework de desarrollo en Java que facilita la creación de aplicaciones web robustas y escalables.

Thymeleaf: Motor de plantillas para aplicaciones Java que permite generar contenido dinámico en vistas web.

UI/UX: Diseño de interfaz y experiencia de usuario, centrado en la facilidad de uso y satisfacción del cliente.

Flask: Microframework de Python utilizado para desarrollar aplicaciones web y APIs de manera rápida y sencilla. En FurgoFinder se utilizó para construir el backend RESTful.

Flutter: Framework de desarrollo de aplicaciones móviles creado por Google, basado en el lenguaje Dart, que permite crear interfaces nativas para Android e iOS desde una única base de código.

Tabla de contenido

RESUMEN	4
ABSTRACT	4
INDICE DE FIGURAS	10
INTRODUCCIÓN.....	12
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	13
1.1. Contexto y problemática	13
1.2 Situación Actual.....	14
1.3 Actores involucrados	14
1.3.1 Apoderados:	14
1.3.2 Conductores:	14
1.3.3 Estudiantes:	15
1.4. Dificultades Actuales y Justificación	15
1.5 Estado del arte y soluciones existentes	15
1.5.1 HopSkipDrive	16
1.5.2 KiddoRide	16
1.6 Motivación	17
1.7 Objetivos del proyecto.....	18
1.7.1 Objetivo general	18
1.7.2 Objetivos específicos	18
1.8 Objetivo General y Objetivos Específicos del backend	18
1.8.1 Objetivo general	18
1.8.2 Objetivos Específicos.....	19
1.9 Alcance.....	19
1.9.1 Problemáticas detectadas	19

“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de Transporte Escolar”

1.9.2	Alcance Inicial	20
1.9.3	Expansión Futura	20
CAPÍTULO 2: MARCO CONCEPTUAL		21
2.1	Conceptos fundamentales.....	21
2.1.1	Transporte Escolar	21
2.1.2	Seguridad Digital.....	21
2.1.4	Dispositivo GPS	21
2.1.5	Desarrollo Ágil	21
2.1.6	Test-Driven Development (TDD).....	22
2.2	Relación con Soluciones Similares	22
2.3	HERRAMIENTAS TECNOLÓGICAS	22
2.3.1	Spring Boot	22
2.3.2	Thymeleaf.....	22
2.3.3	Flutter	22
2.3.4	MySQL	23
2.3.5	API REST	23
2.3.6	Postman	23
2.4	Metodologías de Desarrollo y Organización del Proyecto	23
2.4.1	Metodologías Ágiles.....	23
2.4.2	Control de Versiones Colaborativo	23
2.4.3	Procesos de Desarrollo.....	24
2.4.4	Documentación del Código	24
2.5	Tendencias Actuales en Diseño de Interfaces y Experiencia de Usuario	24
2.5.1	Diseño Responsivo	24
2.5.2	Microinteracciones	24
2.5.3	Diseño Minimalista	24
2.5.4	Interfaces de Voz y Conversacionales	24

“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de Transporte Escolar”

2.5.5	Accesibilidad.....	24
2.6	Arquitectura y Metodología del Backend	25
2.6.1	Frameworks Backend: Alternativas y Selección	25
2.6.2	Elección del Framework	25
2.6.3	Metodología de Desarrollo Backend.....	25
CAPÍTULO 3: PROPUESTA DE SOLUCION		27
3.1	DESCRIPCIÓN DEL PRODUCTO O SERVICIO	27
3.1.1	Características claves	27
3.2	Implementación técnica.....	27
3.3	Tabla historia de usuario.....	29
3.4	Diagramas de casos de usos	34
3.5	Modelo de arquitectura	43
3.6	Modelo de datos.....	45
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		58
4.1	Test de Usabilidad	58
4.2	Encuestas de Aceptación.....	58
4.3.	Análisis de Competencia	59
4.4	Plan de prueba	60
CAPÍTULO 5: CONCLUSIÓN		60
REFERENCIAS BIBLIOGRÁFICAS.....		63

INDICE DE FIGURAS

1. Figura 2: Diagrama de Casos de Uso: Inicio de Sesión. Fuente: Elaboración Propia.
2. Figura 3: Diagrama de Casos de Uso: Gestión de Furgones. Fuente: Elaboración Propia.
3. Figura 4: Diagrama de Casos de Uso: Gestión de Contratos. Fuente: Elaboración Propia.
4. Figura 5: Diagrama de Casos de Uso: Listado de Estudiantes. Fuente: Elaboración Propia.
5. Figura 6: Diagrama de Casos de Uso: Gestión de Solicitudes. Fuente: Elaboración Propia.
6. Figura 7: Arquitectura de Servicio: Proyecto FurgoFinder. Fuente: Elaboración Propia.
7. Figura 8: Modelo de Base de Datos del Proyecto. Fuente: Elaboración Propia

ÍNDICE DE TABLAS

1. Tabla 1: Historia de Usuario: Registro de Apoderado. Fuente: Elaboración Propia.
2. Tabla 2: Historia de Usuario: Gestión de Estudiantes. Fuente: Elaboración Propia.
3. Tabla 3: Historia de Usuario: Solicitud de Contratación. Fuente: Elaboración Propia.
4. Tabla 4: Historia de Usuario: Gestión de Furgones. Fuente: Elaboración Propia.
5. Tabla 5: Historia de Usuario: Revisión de Solicitudes. Fuente: Elaboración Propia.
6. Tabla 6: Historia de Usuario: Edición del Perfil del Conductor. Fuente: Elaboración Propia.
7. Tabla 7: Historia de Usuario: Subida de Documentos del Furgón. Fuente: Elaboración Propia.
8. Tabla 8: Historia de Usuario: Listado de Furgones Disponibles. Fuente: Elaboración Propia.
9. Tabla 9: Resultados de Encuestas de Aceptación de Precios. Fuente: Elaboración Propia.
10. Tabla 10: Resultados de Aceptación de Servicios Ofrecidos. Fuente: Elaboración Propia.

INTRODUCCIÓN

El transporte escolar es un servicio esencial para muchas familias, ya que garantiza que los estudiantes puedan desplazarse de manera segura y puntual a sus establecimientos educativos. Sin embargo, el proceso de contratación de este servicio puede ser complejo, especialmente al buscar opciones confiables y que cumplan con normativas legales. Esta problemática afecta tanto a los apoderados, quienes enfrentan dificultades para encontrar servicios seguros, como a los conductores, que necesitan herramientas eficientes para gestionar sus operaciones y conectar con potenciales clientes.

En este contexto, surge **FurgoFinder**, una plataforma diseñada para simplificar y optimizar la gestión del transporte escolar. Este proyecto combina tecnología web y móvil para conectar a apoderados con conductores de furgones escolares de manera eficiente, transparente y segura. La plataforma no solo facilita la contratación del servicio, sino que también prioriza la seguridad de los estudiantes mediante la verificación de documentos y el cumplimiento de estándares legales por parte de los conductores y sus vehículos.

FurgoFinder propone una solución innovadora que integra funcionalidades clave como la búsqueda de servicios disponibles, la gestión de solicitudes y contratos, y herramientas de comunicación directa entre las partes involucradas. Además, destaca la importancia de la seguridad y la confianza como pilares fundamentales en la implementación de esta plataforma, respondiendo así a una necesidad evidente en el sector del transporte escolar. La propuesta no solo optimiza la experiencia de apoderados y conductores, sino que también promueve la eficiencia, la transparencia y la tranquilidad de las familias usuarias.

El backend de la plataforma es el motor principal que respalda todas las operaciones de FurgoFinder. Desarrollado con **FLASK**, este componente robusto permite construir aplicaciones seguras, escalables y eficientes. Desde la gestión de registros y login hasta la administración de contratos y precontratos, el backend garantiza el funcionamiento óptimo de las funcionalidades críticas, priorizando tanto la experiencia del usuario como el cumplimiento de las normativas vigentes.

A través de este proyecto de memoria, se aborda la problemática actual del transporte escolar, presentando a FurgoFinder como una alternativa tecnológica innovadora que simplifica el acceso a servicios seguros y eficientes. Este trabajo enfatiza cómo la tecnología puede transformar un sector tradicional, resolviendo desafíos logísticos y fomentando la confianza entre apoderados y conductores.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

El transporte escolar es un servicio esencial para garantizar la educación de los estudiantes, ya que permite su traslado diario de manera segura y eficiente. Sin embargo, la gestión y contratación de estos servicios presenta múltiples desafíos tanto para las familias como para los conductores. Actualmente, este proceso es mayoritariamente manual, lo que genera problemas de confianza, falta de información, y dificultades logísticas. Este proyecto se basa en la línea de desarrollo de sistemas informáticos que buscan automatizar y optimizar procesos esenciales.

El proceso tradicional de contratación de transporte escolar suele llevarse a cabo a través de contactos informales, recomendaciones o anuncios locales. Los apoderados deben confiar en conductores sin garantías formales sobre su experiencia o documentación, mientras que los conductores enfrentan dificultades para administrar contratos, gestionar sus rutas y mantener la ocupación de sus vehículos. Además, no existe un sistema centralizado que facilite la validación de documentos, la búsqueda de servicios o la comunicación entre apoderados y conductores.

1.1. Contexto y problemática

El transporte escolar es una necesidad vital para muchas familias, especialmente en comunidades urbanas y suburbanas, donde el acceso seguro y confiable a los colegios puede ser un desafío logístico. A pesar de su importancia, la contratación de este servicio enfrenta varias problemáticas:

- **Falta de confianza:** La mayoría de los apoderados contratan servicios a través de recomendaciones informales o anuncios locales, careciendo de garantías sobre la experiencia, antecedentes y documentación de los conductores.
- **Procesos manuales e ineficientes:** La administración de contratos, solicitudes y pagos se realiza sin herramientas tecnológicas, dificultando la organización y seguimiento.
- **Inseguridad:** La falta de un sistema centralizado de validación de documentos genera incertidumbre y posibles riesgos para los estudiantes.
- **Acceso desigual:** En áreas rurales y suburbanas, la oferta de transporte escolar es limitada y poco confiable, dejando a muchas familias sin opciones seguras.

Estos problemas limitan el acceso a transporte escolar seguro y confiable, incrementando el estrés de los apoderados y reduciendo las oportunidades laborales de los conductores. En el mediano plazo, esto podría limitar el acceso a transporte seguro para estudiantes en regiones vulnerables.

1.2 Situación Actual

Actualmente, la contratación de servicios de transporte escolar se realiza principalmente mediante métodos tradicionales e informales, como:

- **Recomendaciones personales:** Apoderados que buscan opciones a través de redes sociales, círculos comunitarios o referencias de otros padres.
- **Anuncios locales:** Publicaciones en plataformas como Facebook, tablones comunitarios o grupos de mensajería.
- **Contactos directos:** Conductores que promocionan sus servicios de manera independiente, sin intermediación formal.

Estas prácticas presentan limitaciones importantes:

- **Falta de validación de conductores:** Los apoderados no tienen garantías sobre la experiencia, antecedentes o documentación de los conductores.
- **Inseguridad:** La falta de un sistema centralizado para verificar vehículos y conductores incrementa los riesgos para los estudiantes.
- **Gestión ineficiente:** Conductores administran sus contratos y rutas de manera manual, lo que resulta en problemas organizativos.
- **Acceso desigual:** En áreas rurales o suburbanas, encontrar opciones confiables de transporte escolar es especialmente difícil.

1.3 Actores involucrados

Los principales actores afectados por esta problemática son:

1.3.1 Apoderados:

- Padres o tutores que buscan transporte seguro y confiable para sus hijos.
- Dificultades actuales: Falta de opciones validadas, acceso limitado a información sobre rutas, precios y conductores.

1.3.2 Conductores:

- Proveedores del servicio, responsables del traslado diario de los estudiantes.
- Dificultades actuales: Limitaciones para captar nuevos clientes, gestionar contratos, optimizar rutas y mantener la ocupación de sus vehículos.

1.3.3 Estudiantes:

- Usuarios finales que dependen de un transporte seguro para asistir a clases.
- Dificultades actuales: Riesgo de seguridad al usar servicios no validados.

1.3.4 Administradores:

- Supervisores de la solución propuesta, encargados de validar documentos y supervisar el cumplimiento de normativas.

1.4. Dificultades Actuales y Justificación

Las principales dificultades que enfrentan los actores son:

- Falta de confianza y transparencia: Los apoderados no cuentan con herramientas para verificar la documentación de los conductores ni el estado de los vehículos.
- Ineficiencia en la gestión: Conductores y apoderados invierten tiempo y recursos en procesos manuales para buscar y gestionar servicios.
- Falta de acceso: En comunidades rurales o alejadas, las familias tienen opciones limitadas y deben depender de servicios no regulados.

Impacto si no se resuelve el problema:

- En el corto plazo, los apoderados seguirán enfrentando incertidumbre al contratar servicios no validados, exponiendo a los estudiantes a posibles riesgos de seguridad.
- En el mediano plazo, los conductores independientes perderán oportunidades de negocio por falta de herramientas para profesionalizar su servicio.
- En términos generales, el sector del transporte escolar continuará operando de manera desorganizada, afectando a familias y estudiantes.

1.5 Estado del arte y soluciones existentes

Existen plataformas internacionales que ofrecen servicios relacionados con el transporte escolar, principalmente en mercados desarrollados. Estas soluciones presentan características avanzadas, pero no siempre se adaptan a las necesidades y restricciones de nuestro país

1.5.1 HopSkipDrive

Descripción: Plataforma en Estados Unidos que conecta a padres con conductores para el transporte de niños.

Características principales:

- Conductores certificados con verificaciones de antecedentes.
- Seguimiento en tiempo real durante los traslados.
- Precios premium adaptados a familias de ingresos medios y altos.

Limitaciones: Altos costos y dependencia de tecnologías avanzadas que dificultan su adopción en mercados con restricciones económicas.

1.5.2 KiddoRide

Descripción: Plataforma diseñada para ofrecer servicios de transporte escolar en áreas urbanas de Estados Unidos.

Características principales:

- Filtros para personalizar rutas y horarios.
- Validación de conductores y vehículos mediante estándares estrictos.
- Aplicación móvil con interfaz intuitiva.

Limitaciones: Enfocada en contextos urbanos desarrollados, con poca flexibilidad para mercados emergentes.

En Chile las soluciones actuales para el transporte escolar son en su mayoría informales y carecen de tecnología avanzada. Entre los enfoques más comunes se encuentran los contactos informales, donde los apoderados recomiendan servicios basados en experiencias personales, generando acuerdos fundamentados en la confianza, pero sin procesos de validación. También es frecuente el uso de anuncios locales a través de publicaciones en redes sociales, grupos comunitarios o tableros de anuncios, lo que facilita el acceso a información, pero no garantiza la calidad ni la seguridad del servicio. Además, muchas familias optan por la contratación directa con conductores, sin intermediarios ni regulaciones formales, lo que, aunque práctico, puede conllevar riesgos debido a la falta de supervisión y estándares. Estas prácticas evidencian la necesidad urgente de soluciones tecnológicas que profesionalicen el sector, refuercen la seguridad y brinden mayor confianza a las familias.

Criterio	HopSkipDrive	KiddoRide	Soluciones Locales	FurgoFinder
Costo	Alto	Alto	Bajo	Accesible
Validación	Estricta	Estricta	Nula	Exhaustiva
Tecnología	Avanzada	Avanzada	Básica	Intermedia
Accesibilidad	Limitada	Limitada	Alta	Alta
Enfoque Regional	No	No	Parcial	Total

1.6 Motivación

El desarrollo de FurgoFinder surge como respuesta a problemas críticos en el transporte escolar, centrados en seguridad, eficiencia y accesibilidad. Actualmente, la falta de validación de conductores y vehículos deja a los estudiantes expuestos a riesgos, mientras los apoderados enfrentan incertidumbre al contratar servicios sin garantías sobre calidad o confiabilidad.

Por otro lado, los conductores carecen de herramientas para gestionar eficientemente rutas, contratos y clientes, limitando su capacidad de profesionalizar su servicio. Además, en áreas rurales y suburbanas, las familias tienen pocas opciones de transporte formal, enfrentándose a servicios informales y desorganizados.

FurgoFinder busca:

- **Mejorar la seguridad:** Validando exhaustivamente documentos, antecedentes y vehículos.
- **Reducir la incertidumbre:** Centralizando la búsqueda de opciones con información confiable sobre precios, rutas y disponibilidad.
- **Profesionalizar el sector:** Ofreciendo a los conductores herramientas tecnológicas para captar más clientes y administrar sus servicios.
- **Ampliar el acceso:** Beneficiando a comunidades con opciones limitadas y precios accesibles.

Esta solución no solo moderniza un sector desatendido, sino que también genera un impacto positivo a largo plazo al mejorar la calidad del transporte escolar y formalizar la actividad de los conductores.

1.7 Objetivos del proyecto

1.7.1 Objetivo general

Desarrollar una plataforma tecnológica integral que conecte a apoderados y conductores de transporte escolar, proporcionando herramientas para facilitar la contratación, gestión y validación de servicios, garantizando un enfoque centrado en la seguridad, la accesibilidad y la transparencia.

1.7.2 Objetivos específicos

- Diseñar un sistema que permita validar documentos clave de conductores y vehículos, asegurando el cumplimiento de normativas legales y estándares de seguridad.
- Crear filtros avanzados para que los apoderados puedan buscar y contratar servicios de transporte escolar según ubicación, colegio y disponibilidad, mejorando la experiencia de usuario.
- Proveer a los conductores herramientas digitales para administrar solicitudes, rutas y contratos de manera eficiente, optimizando su operación diaria.
- Implementar medidas de seguridad, como encriptación de contraseñas y autenticación mediante JWT, para proteger la información de los usuarios y evitar accesos no autorizados.
- Diseñar un sistema de calificaciones y comentarios que permita a los apoderados evaluar la calidad del servicio de los conductores, fomentando la transparencia y la confianza en la plataforma.
- Validar la plataforma a través de pruebas funcionales y recolección de feedback de los usuarios finales para garantizar su usabilidad y efectividad.
- Establecer una arquitectura escalable que facilite la integración de futuras funcionalidades, como seguimiento en tiempo real y aplicaciones móviles, y que permita su expansión a otras regiones.

1.8 Objetivo General y Objetivos Específicos del backend

1.8.1 Objetivo general

Desarrollar el backend y la API RESTful del proyecto FurgoFinder, permitiendo la gestión centralizada, segura y eficiente de los procesos de contratación, validación documental y administración de usuarios, con soporte tanto para la plataforma web y aplicaciones móviles.

1.8.2 Objetivos Específicos

- Analizar los procesos funcionales del sistema FurgoFinder (contratación, validación de documentos, gestión de usuarios, pagos y notificaciones), con el fin de definir los requerimientos técnicos necesarios para el desarrollo del backend y su API.
- Diseñar un modelo de datos relacional y una arquitectura modular basada en servicios REST, que represente de forma estructurada las entidades clave del sistema, tales como apoderados, alumnos, conductores, furgones y contratos.
- Implementar una API RESTful utilizando Flask como microframework, que permita realizar operaciones CRUD y exponer servicios consumibles tanto por la plataforma web como por una futura aplicación móvil, incorporando mecanismos de autenticación segura mediante JWT y cifrado de contraseñas con bcrypt.
- Validar el correcto funcionamiento del backend y su API mediante pruebas unitarias, integradas y funcionales utilizando herramientas como Postman, garantizando estabilidad, seguridad y compatibilidad multiplataforma.

1.9 Alcance

El alcance de **FurgoFinder** está enfocado en resolver las problemáticas actuales del transporte escolar mediante una plataforma tecnológica diseñada para conectar a apoderados y conductores de manera eficiente y segura. Este sistema se estructura en etapas y funcionalidades específicas para maximizar su impacto en los usuarios y garantizar su escalabilidad en el futuro.

1.9.1 Problemáticas detectadas

- **Falta de centralización de la información:** En Chile no existe plataformas que conecten de manera directa a los apoderados con los conductores validados.
- **Gestión manual y fragmentada:** La gestión de contratos, documentos y comunicaciones suele realizarse de manera informal, ya sea verbalmente o mediante medios no validados por terceros.
- **Inexistencia de una API robusta:** El backend tradicional no contemplaba una arquitectura orientada a servicios que permitiera escalar fácilmente el sistema ni integrarse con otras plataformas o aplicaciones externas. Esto limitaba significativamente la interoperabilidad, dificultaba la expansión del sistema

hacia nuevas interfaces (como aplicaciones móviles) y generaba cuellos de botella al intentar automatizar procesos o compartir datos con otros sistemas. Para abordar este desafío, se desarrolló una API RESTful utilizando Flask, un microframework ligero y flexible que permite una rápida implementación sin sacrificar la escalabilidad. Esta API expone de forma segura y estructurada los principales recursos del sistema, incluyendo la gestión de alumnos, apoderados, conductores, contratos, pagos, rutas y otros componentes clave del negocio. La implementación de esta API no solo facilita la comunicación entre el sistema web actual y futuros desarrollos móviles o de escritorio, sino que también establece una base sólida para la integración con servicios de terceros, tales como pasarelas de pago, sistemas de gestión escolar o plataformas de análisis de datos. Asimismo, se incorporaron mecanismos de autenticación y autorización para asegurar el acceso controlado a la información, mejorando considerablemente la seguridad y la trazabilidad de las operaciones.

1.9.2 Alcance Inicial

- **Búsqueda y Contratación:** Los apoderados podrán filtrar servicios por colegio, comuna y población, visualizando detalles del conductor, furgón y estado de validación.
- **Validación:** Los documentos de conductores y vehículos serán revisados por administradores para garantizar la seguridad.
- **Gestión de Contratos:** Vinculación de apoderados, conductores, estudiantes y furgones mediante contratos electrónicos.
- **Seguridad:** Uso de encriptación y autenticación JWT para proteger datos.

1.9.3 Expansión Futura

- Ampliación a regiones rurales y nuevos mercados.
- Implementación de funcionalidades como seguimiento en tiempo real, calificación de servicios y métodos de pago digitales.
- Desarrollo de una aplicación móvil para mejorar la accesibilidad.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1 Conceptos fundamentales

2.1.1 Transporte Escolar

El transporte escolar garantiza la movilidad segura de los estudiantes entre sus hogares y sus centros educativos. Este servicio, regulado por normativas legales en la mayoría de los países, busca asegurar estándares como el mantenimiento adecuado de los vehículos y la validación de antecedentes de los conductores. En el contexto del proyecto, este concepto se aborda como una necesidad crítica que puede optimizarse mediante el uso de tecnología para satisfacer las expectativas de seguridad y eficiencia de los apoderados.

2.1.2 Seguridad Digital

La protección de datos es esencial en plataformas que manejan información personal. FurgoFinder integra estándares de seguridad como:

- **Encriptación de contraseñas:** Uso de algoritmos de hashing para proteger credenciales.
- **Gestión de roles y permisos:** Control granular de acceso para apoderados, conductores y administradores.
- **Protocolo HTTPS:** Asegura la transmisión de datos en todas las comunicaciones.

2.1.4 Dispositivo GPS

Un dispositivo GPS utiliza el sistema de posicionamiento global para determinar su ubicación geográfica. Estos dispositivos son fundamentales para el seguimiento de las rutas de transporte escolar en FurgoFinder, garantizando que los apoderados puedan monitorizar en todo momento el trayecto de sus hijos.

2.1.5 Desarrollo Ágil

El proyecto FurgoFinder se desarrolló bajo una metodología ágil, iterativa e incremental, que permitió adaptarse a cambios durante el ciclo de vida del software. La adopción de reuniones diarias y sprints semanales facilitó la colaboración entre los integrantes del equipo y la priorización de tareas, asegurando una entrega constante de valor a través de incrementos funcionales.

2.1.6 Test-Driven Development (TDD)

Se utilizó TDD para garantizar la calidad del código, desarrollando pruebas unitarias antes de implementar la lógica del negocio. Este enfoque permitió detectar y corregir errores desde etapas tempranas del desarrollo, asegurando que cada funcionalidad cumpliera con los requisitos establecidos.

2.2 Relación con Soluciones Similares

Existen aplicaciones internacionales como KiddoRide en Estados Unidos, que ofrecen servicios de transporte escolar adaptados a su contexto. Sin embargo, estas plataformas están diseñadas para satisfacer las necesidades de mercados con mayor capacidad económica y diferentes requisitos legales. En contraste, FurgoFinder se enfoca en abordar las particularidades de los países latinoamericanos, considerando:

- **Restricciones presupuestarias:** Ofreciendo una solución accesible y rentable.
- **Validación exhaustiva de documentos:** Garantizando la seguridad y confiabilidad del servicio mediante la verificación de conductores y vehículos.
- **Enfoque cultural:** Adaptándose a las necesidades locales, como rutas escolares específicas y métodos de pago adecuados.

2.3 HERRAMIENTAS TECNOLÓGICAS

2.3.1 Spring Boot

Spring Boot es un framework basado en Java que facilita el desarrollo de aplicaciones web robustas y escalables. En el caso de **FurgoFinder**, se utilizó exclusivamente para construir el **frontend de la versión web** de la plataforma. Gracias a su integración con **Thymeleaf** y herramientas como **Spring Security**, fue posible implementar una interfaz visual dinámica y segura que interactúa con el backend mediante peticiones HTTP.

2.3.2 Thymeleaf

Thymeleaf es un motor de plantillas para Java que permite generar páginas HTML dinámicas en tiempo de ejecución. En FurgoFinder, se utilizó junto con Spring Boot para renderizar las vistas de la plataforma web, facilitando la visualización de datos obtenidos desde el backend y mejorando la experiencia de usuario.

2.3.3 Flutter

Para el desarrollo de la aplicación móvil de FurgoFinder se utilizó **Flutter**, un framework multiplataforma de Google basado en Dart. Flutter permitió crear una interfaz moderna, rápida y responsiva para dispositivos móviles, conectándose al backend a través de peticiones HTTP hacia la API desarrollada en Flask.

2.3.4MySQL

MySQL fue utilizado como sistema de gestión de bases de datos relacional (RDBMS) en FurgoFinder, permitiendo almacenar de forma estructurada la información crítica de la plataforma. Esta base de datos se conecta directamente con el backend desarrollado en Flask, lo que facilita la ejecución eficiente de operaciones como inserciones, consultas y actualizaciones. Entre los principales datos gestionados se incluyen los perfiles de apoderados y conductores, las solicitudes de transporte realizadas por los usuarios, los contratos establecidos entre las partes, los detalles técnicos de los furgones registrados y las reseñas del servicio. Gracias a su fiabilidad, facilidad de uso y compatibilidad con diversos entornos, MySQL resultó una solución adecuada para garantizar la integridad y escalabilidad de la información manejada en la aplicación.

2.3.5API REST

Una API REST es una interfaz que utiliza el protocolo HTTP para la transferencia de datos entre sistemas. En FurgoFinder, se implementó una API RESTful mediante el microframework **Flask** en Python. Esta API permite la comunicación entre los frontends (web y móvil) y el backend, gestionando recursos como usuarios, vehículos y reseñas de forma eficiente.

2.3.6Postman

Postman se utilizó como herramienta de prueba para los endpoints REST del backend. Gracias a su interfaz gráfica, fue posible simular peticiones GET, POST, PUT y DELETE hacia la API desarrollada en Flask, validando su funcionalidad, estructura de respuesta y manejo de errores antes de integrarla con los frontends.

2.4 Metodologías de Desarrollo y Organización del Proyecto

2.4.1Metodologías Ágiles

Las metodologías ágiles, como Scrum, promueven la entrega iterativa e incremental, la colaboración en equipo y la adaptación al cambio. Estas metodologías permiten a los equipos responder de manera flexible a los requisitos cambiantes del proyecto y centrarse en la entrega de valor continuo.

2.4.2 Control de Versiones Colaborativo

El control de versiones colaborativo es fundamental en el desarrollo de software. Herramientas como GitHub proporcionan un entorno centralizado para alojar repositorios de código, realizar seguimiento de cambios y gestionar problemas, facilitando la colaboración entre desarrolladores.

2.4.3 Procesos de Desarrollo

Los procesos de desarrollo abarcan desde la planificación y el diseño hasta la implementación, las pruebas y el despliegue continuo del software. Establecer procesos claros y eficientes garantiza la calidad, la coherencia y la entrega oportuna del producto final. La adopción de prácticas como la integración y la entrega continuas ayuda a automatizar el desarrollo y a minimizar errores.

2.4.4 Documentación del Código

La documentación del código es esencial para garantizar la comprensión y el mantenimiento efectivo del software. Mantener una documentación clara y completa, incluyendo comentarios, guías de estilo y explicaciones de la lógica de negocio, facilita la colaboración y permite a nuevos miembros del equipo incorporarse rápidamente.

2.5 Tendencias Actuales en Diseño de Interfaces y Experiencia de Usuario

2.5.1 Diseño Responsivo

Con el aumento del uso de dispositivos móviles, el diseño responsivo se ha vuelto fundamental. Las interfaces deben adaptarse dinámicamente a diferentes tamaños de pantalla y dispositivos para ofrecer una experiencia coherente y agradable.

2.5.2 Microinteracciones

Las microinteracciones son pequeñas animaciones o cambios visuales que ocurren en respuesta a las acciones del usuario. Estas agregan feedback y respuesta que mejoran la usabilidad y la satisfacción del usuario.

2.5.3 Diseño Minimalista

La simplicidad y la claridad son cada vez más valoradas en el diseño de interfaces. Los diseños minimalistas reducen el desorden visual y hacen que la información sea más fácil de entender y asimilar para el usuario.

2.5.4 Interfaces de Voz y Conversacionales

Con la popularidad de los asistentes virtuales como Siri, Alexa y Google Assistant, las interfaces de voz y conversacionales están en aumento. Estas permiten a los usuarios interactuar con el software utilizando comandos de voz naturales.

2.5.5 Accesibilidad

La accesibilidad web está ganando cada vez más atención, con un enfoque en hacer que el software sea accesible para personas con discapacidades. Esto incluye la implementación de prácticas de diseño inclusivo y el cumplimiento de estándares de accesibilidad.

2.6 Arquitectura y Metodología del Backend

2.6.1 Frameworks Backend: Alternativas y Selección

Durante la fase de diseño del backend para FurgoFinder, se evaluaron múltiples tecnologías ampliamente utilizadas en el desarrollo de aplicaciones web modernas. A continuación, se resumen algunas de las principales alternativas consideradas:

- **Spring Boot (Java):** Ofrece una arquitectura robusta y herramientas avanzadas como Spring Security y Spring Data. Es ideal para sistemas empresariales de gran escala, aunque su curva de aprendizaje y configuración es más elevada.
- **Django (Python):** Framework completo y maduro que incluye ORM, panel de administración y sistema de autenticación integrados. Sin embargo, su estructura rígida y complejidad inicial fueron menos adecuadas para un desarrollo ágil y modular.
- **Express.js (Node.js):** Ligero y altamente personalizable, basado en JavaScript. Muy útil para aplicaciones en tiempo real, pero requiere un mayor control manual de validaciones y seguridad.
- **Flask (Python):** Microframework ligero y flexible, ideal para proyectos que requieren rapidez en el desarrollo, control sobre cada componente y una arquitectura limpia.

2.6.2 Elección del Framework

FurgoFinder adoptó Flask como framework para el desarrollo del backend debido a sus siguientes ventajas:

- Bajo peso y facilidad de uso para proyectos medianos.
- Integración nativa con bibliotecas como SQLAlchemy, Flask-JWT y Flask-CORS.
- Permite desarrollar APIs RESTful limpias y bien organizadas.
- Soporte activo de la comunidad y documentación clara.

Esta elección permitió a los desarrolladores tener **mayor control sobre la arquitectura** del backend, facilitando su integración con diferentes interfaces (web y móvil) y adaptándolo a necesidades futuras.

2.6.3 Metodología de Desarrollo Backend

El desarrollo del backend se realizó siguiendo prácticas de ingeniería de software modernas, orientadas a mantener calidad, escalabilidad y mantenibilidad. Se utilizaron los siguientes enfoques:

- **Arquitectura en capas:** Separación lógica en componentes como modelos, controladores, servicios, seguridad y persistencia, lo que facilita el mantenimiento y las pruebas unitarias.
- **Test-Driven Development (TDD):** Se aplicó TDD para funcionalidades críticas, asegurando que el código cumpliera con los requisitos desde etapas tempranas mediante la escritura de pruebas antes de la implementación.
- **Desarrollo API-First:** Se diseñó primero la interfaz RESTful de la API, permitiendo que frontend y backend trabajaran en paralelo y se garantizara coherencia en la comunicación entre ambos.
- **Despliegue continuo:** El backend fue implementado en Heroku, lo cual permitió realizar pruebas funcionales de manera iterativa, obtener feedback constante y garantizar la disponibilidad inmediata en un entorno controlado.

CAPÍTULO 3: PROPUESTA DE SOLUCION

3.1 DESCRIPCIÓN DEL PRODUCTO O SERVICIO

FurgoFinder es una plataforma tecnológica diseñada para facilitar la gestión y contratación de servicios de transporte escolar. La solución conecta de manera eficiente y segura a apoderados con conductores, asegurando el cumplimiento de estándares legales y proporcionando herramientas intuitivas para ambas partes.

3.1.1 Características claves

1. Gestión de apoderados y estudiantes:

- Los apoderados pueden registrar a sus hijos (estudiantes) y gestionar la información asociada a ellos, como colegio, dirección, y necesidades específicas.
- Acceso a un listado de furgones validados que operan en sus comunas y colegios.

2. Validación de conductores y furgones:

- Todos los conductores deben cargar documentos obligatorios, como licencias, seguros y permisos de circulación, que son revisados antes de ser visibles en la plataforma.
- Cada furgón cuenta con un perfil detallado que incluye información técnica, documentos y fotografías.

3. Flujo de contratación:

- Apoderados pueden enviar solicitudes de contratación a conductores.
- Los conductores revisan estas solicitudes, asignan precios y confirman o rechazan los contratos.

4. Notificaciones y seguimiento:

- Notificaciones en tiempo real para apoderados y conductores sobre el estado de las solicitudes, contratos y pagos.

5. Historial y seguridad:

- Historial de contratos y pagos accesible para ambas partes.
- Implementación de encriptación para proteger los datos personales.

3.2 Implementación técnica

Durante la implementación del proyecto FurgoFinder, se utilizaron diversas herramientas tecnológicas que facilitaron el desarrollo del software y la integración de sus diferentes componentes. A continuación, se describen las principales herramientas empleadas y su función dentro del proceso de implementación:

- **Spring Boot:** Se utilizó como marco de trabajo principal para desarrollar el backend de la aplicación. Spring Boot facilitó la creación rápida de una arquitectura robusta y escalable, incluyendo funcionalidades para la gestión de usuarios, contratos y solicitudes.
- **HTML, CSS y Bootstrap:** Estas tecnologías fueron utilizadas para diseñar y dar formato a la interfaz de usuario de la plataforma web. Bootstrap permitió crear una experiencia visual atractiva y responsiva, optimizada tanto para dispositivos móviles como para navegadores de escritorio.
- **Thymeleaf:** Este motor de plantillas se empleó para generar páginas web dinámicas en el frontend, facilitando la integración con los datos proporcionados por el backend.
- **MySQL:** Se seleccionó como sistema de gestión de bases de datos relacional para almacenar y gestionar la información del sistema, como usuarios, furgones, estudiantes, contratos y solicitudes. MySQL se integró con Spring Boot para garantizar consultas rápidas y seguras.
- **RESTful API:** Se implementó una arquitectura RESTful para la comunicación entre el frontend y el backend. Esto permitió la integración eficiente de los componentes del sistema y la conexión con aplicaciones móviles.
- **Spring Security y JWT:** Estas herramientas se usaron para la autenticación y autorización de usuarios, garantizando la protección de los datos sensibles y el acceso seguro a las funcionalidades según el rol del usuario (apoderado, conductor, administrador).
- **Postman:** Fue empleado para realizar pruebas exhaustivas de los endpoints de la API RESTful, verificando su comportamiento y asegurando que las solicitudes y respuestas cumplieran con los requisitos definidos.
- **GitHub:** Se utilizó como repositorio para el control de versiones del código fuente del proyecto. GitHub permitió gestionar de manera eficiente las colaboraciones y mantener un historial organizado de cambios durante el desarrollo.
- **AWS EC2:** Se configuró una instancia de Linux en AWS EC2 para alojar y desplegar la aplicación en un entorno de producción, garantizando escalabilidad y disponibilidad.
- **Lombok:** Se incorporó a través de Spring Boot para simplificar el código en las clases del modelo, reduciendo el uso de métodos repetitivos como getters, setters y constructores.

3.3 Tabla historia de usuario

Las tablas de historias de usuario presentan los requisitos funcionales más importantes para la aplicación FurgoFinder, expresados en forma de historias de usuario. Estas historias fueron recopiladas y diseñadas tras un análisis de las necesidades de apoderados y conductores, usuarios clave de la plataforma. Cada historia describe una funcionalidad específica con detalles como su descripción, prioridad y criterios de aceptación.

Estas tablas ayudan a planificar, priorizar y estructurar el desarrollo del software, permitiendo construir un prototipo mínimo viable (MVP) que responda a las necesidades básicas de los usuarios.

Propuesta del prototipo mínimo viable

Para cada funcionalidad se definen los siguientes campos:

- **Usuario:** ¿Quién utilizará la funcionalidad?
- **Criterio:** Requisitos específicos que deben cumplirse para considerar la funcionalidad como completa.
- **Dependencias:** Casos previos de los que depende la funcionalidad.

Prioridad: Indica la importancia de la funcionalidad para el usuario (Alta, Media o Baja).

- **Riesgo:** Nivel de riesgo asociado a la implementación (Alto, Medio o Bajo).
- **Estimación:** Escala del 1 al 10 que representa la complejidad técnica de la funcionalidad.
- **Responsable:** Persona o equipo responsable de la implementación.
- **Descripción:** Qué desea el usuario y cómo se implementará la funcionalidad.
- **Pruebas de Aceptación:** Criterios que deben cumplirse para considerar que la funcionalidad está lista.
- **Observaciones:** Consideraciones adicionales relacionadas con la funcionalidad.

Tabla 1: Historia de Usuario: Registro de Apoderado Fuente: Elaboración Propia

HU001		Registro de Apoderado			
Usuario	Apoderado	Criterio	Usabilidad	Dependencias	Ninguna
Prioridad	Alta	Riesgo	Medio	Estimación	5
Responsable	XX				
Descripción	Como apoderado, quiero registrarme en la plataforma ingresando mi nombre, correo, contraseña y dirección, para poder acceder a las funciones como la gestión de estudiantes y la contratación de servicios.				
Pruebas de Aceptación	1. Validar campos obligatorios como nombre, correo y contraseña. 2. Verificar que el correo electrónico no esté duplicado. 3. Recibir correo de confirmación tras completar el registro.				
Observaciones	Es importante que la validación de datos sea eficiente y rápida.				

Tabla 2: Historia de Usuario: Gestión de Estudiantes Fuente: Elaboración Propia

HU002		Gestión de Estudiantes			
Usuario	Apoderado	Criterio	Usabilidad	Dependencias	HU001
Prioridad	Alta	Riesgo	Medio	Estimación	4
Responsable	XX				
Descripción	Como apoderado, quiero registrar a mis hijos en la plataforma para asociarlos a solicitudes de transporte escolar, asegurando que la información esté completa y actualizada.				
Pruebas de Aceptación	1. Permitir registrar múltiples estudiantes con nombre, edad, colegio y dirección. 2. Solo el apoderado que creó el registro puede visualizar o editar los datos. 3. Mostrar mensaje de confirmación al guardar.				
Observaciones	Es fundamental garantizar la privacidad de los datos del estudiante.				

Tabla 3: Historia de Usuario: Solicitud de Contratación Fuente: Elaboración Propia

HU003		Solicitud de Contratación			
Usuario	Apoderado	Criterio	Usabilidad	Dependencias	HU002
Prioridad	Alta	Riesgo	Alto	Estimación	6
Responsable	XX				
Descripción	Como apoderado, quiero enviar una solicitud a un conductor para contratar un servicio de transporte para mis hijos, especificando detalles como horario y punto de recogida.				
Pruebas de Aceptación	1. Seleccionar un furgón disponible y asignar estudiantes registrados. 2. Notificar al conductor tras enviar la solicitud. 3. Revisar el estado de la solicitud (pendiente, aceptada, rechazada).				
Observaciones	La experiencia de usuario debe ser fluida y confiable.				

Tabla 4: Historia de Usuario: Gestión de Furgones Fuente: Elaboración Propia

HU004		Gestión de Furgones			
Usuario	Conductor	Criterio	Usabilidad	Dependencias	HU001 (Registro del conductor)
Prioridad	Alta	Riesgo	Medio	Estimación	5
Responsable	XX				
Descripción	Como conductor, quiero gestionar la información de mis furgones en la plataforma, pudiendo registrar, editar o eliminar detalles como capacidad, documentos y fotos.				
Pruebas de Aceptación	1. Registrar un nuevo furgón con marca, modelo, capacidad y documentos. 2. Reflejar cambios inmediatamente tras editar un furgón. 3. Permitir que solo el conductor elimine sus furgones.				
Observaciones	El proceso de carga de documentos debe ser sencillo y soportar formatos estándar.				

Tabla 5: Historia de Usuario: Revisión de Solicitudes Fuente: Elaboración Propia

HU05		Revisión de Solicitudes			
Usuario	Conductor	Criterio	Usabilidad	Dependencias	HU003
Prioridad	Alta	Riesgo	Alto	Estimación	6
Responsable	XX				

“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de Transporte Escolar”

Descripción	Como conductor, quiero revisar las solicitudes recibidas de apoderados para aceptar o rechazar según mi disponibilidad y condiciones del servicio.
Pruebas de Aceptación	1. Mostrar detalles de las solicitudes como horario, estudiantes y puntos de recogida. 2. Permitir asignar un precio antes de aceptar. 3. Notificar al apoderado tras aceptar o rechazar la solicitud.
Observaciones	El flujo de decisiones debe ser claro para evitar demoras en la contratación.

Tabla 6: Historia de Usuario: Edición del Perfil del Conductor Fuente: Elaboración Propia

HU006		Edición del Perfil del Conductor			
Usuario	Conductor	Criterio	Usabilidad	Dependencias	HU001
Prioridad	Alta	Riesgo	Medio	Estimación	4
Responsable	XX				
Descripción	Como conductor, quiero editar mi perfil para actualizar información personal como correo, teléfono y datos de contacto.				
Pruebas de Aceptación	1. El sistema debe permitir actualizar todos los campos excepto el RUT. 2. Mostrar un mensaje de confirmación tras guardar los cambios. 3. Validar la integridad de los datos ingresados antes de guardar.				
Observaciones	La edición debe incluir la opción de cambiar la contraseña con validación de seguridad.				

Tabla 7: Historia de Usuario: Edición del Perfil del Conductor Fuente: Elaboración Propia

HU007		Subida de Documentos del Furgón			
Usuario	Conductor	Criterio	Usabilidad	Dependencias	HU004
Prioridad	Alta	Riesgo	Medio	Estimación	5

“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de Transporte Escolar”

Responsable	XX
Descripción	Como conductor, quiero subir los documentos obligatorios de mis furgones, como permisos de circulación y seguros, para cumplir con los requisitos legales de la plataforma.
Pruebas de Aceptación	1. Validar el formato de los documentos subidos (PDF, JPG). 2. Mostrar un mensaje de confirmación al completar la carga. 3. Asociar los documentos al furgón correspondiente.
Observaciones	Implementar una notificación al administrador para validar los documentos subidos.

Tabla 8: Historia de Usuario: Listado de Furgones Disponibles Fuente: Elaboración Propia

HU008		Listado de Furgones Disponibles			
Usuario	Apoderado	Criterio	Usabilidad	Dependencias	Ninguna
Prioridad	Alta	Riesgo	Bajo	Estimación	4
Responsable	XX				
Descripción	Como apoderado, quiero visualizar un listado de furgones disponibles, para evaluar las opciones y seleccionar el servicio más adecuado.				
Pruebas de Aceptación	1. Mostrar todos los furgones registrados en la plataforma. 2. Permitir filtrar por ubicación, colegio y capacidad. 3. Incluir detalles como nombre del conductor, estado de validación, y documentos del furgón.				
Observaciones	El diseño debe ser responsivo				

Tabla 9: Historia de Usuario: Listado de Furgones Disponibles Fuente: Elaboración Propia

HU009		Subida de Documentos del Furgón			
Usuario	Apoderado	Criterio	Seguridad	Dependencias	HU003
Prioridad	Alta	Riesgo	Alto	Estimación	6
Responsable	XX				

Descripción	Como apoderado, quiero confirmar un contrato con un conductor después de revisar los términos, para asegurar el transporte de mis hijos.
Pruebas de Aceptación	1. Mostrar un resumen del contrato (precio, horario, estudiantes incluidos). 2. Requerir confirmación explícita del apoderado antes de cerrar el contrato. 3. Enviar una notificación al conductor tras la confirmación.
Observaciones	La confirmación debe estar protegida con autenticación activa del usuario.

3.4 Diagramas de casos de usos

Conductor

Persona que ofrece el servicio de transporte escolar y gestiona sus furgones, contratos y estudiantes asignados.

Apoderado

Cliente que busca contratar el servicio de transporte escolar para sus hijos o estudiantes a su cargo.

Administrador

Usuario encargado de supervisar y gestionar el funcionamiento general de la plataforma, incluyendo validaciones y administración de usuarios.

3.3.1 Gestion de perfil

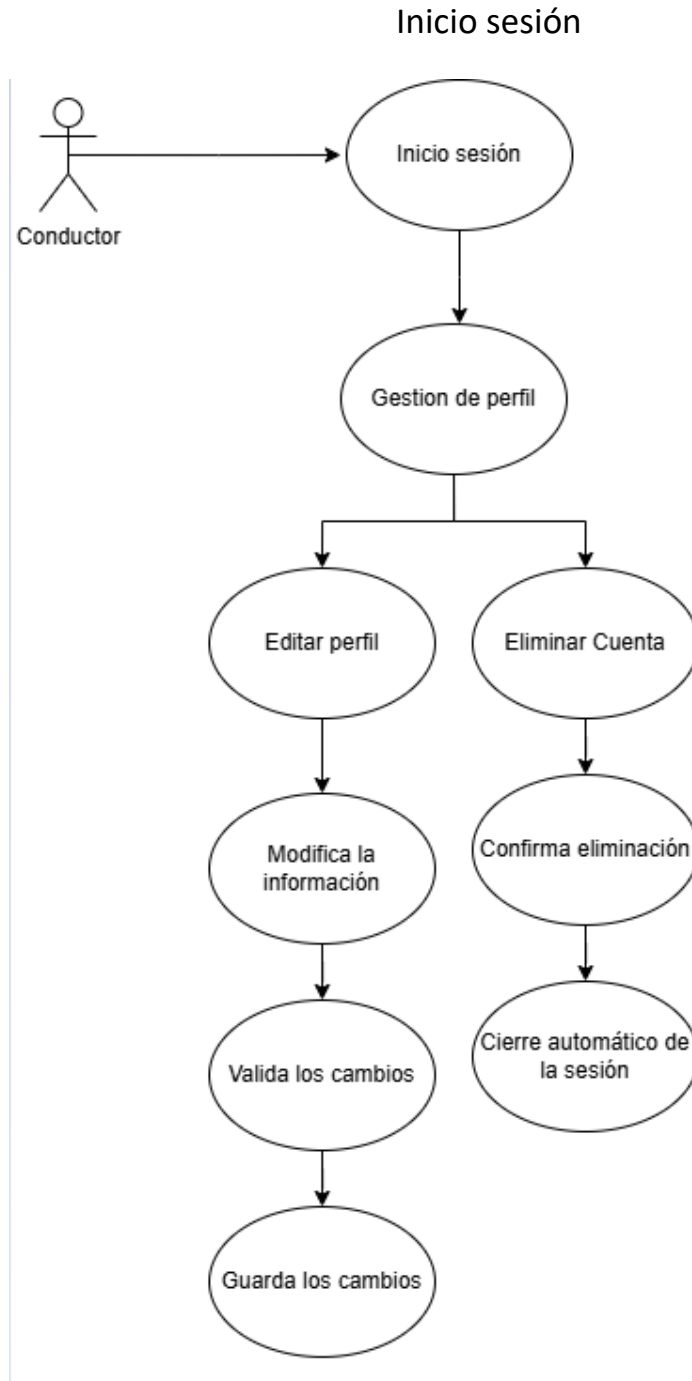


Figura 1: Diagrama de Casos de Usos: Inicio Sesión. Fuente: Elaboración Propia.

Actor: Conductor

Descripción: Este diagrama representa el proceso que realiza un conductor al gestionar su perfil después de iniciar sesión en el sistema. Primero, el conductor accede al sistema

proporcionando sus credenciales. Una vez autenticado, tiene la opción de gestionar su perfil, lo que incluye dos funcionalidades principales: editar sus datos personales o eliminar su cuenta.

Si decide editar su perfil, puede modificar su información, que luego es validada y guardada por el sistema. Si opta por eliminar su cuenta, el sistema solicita confirmación antes de proceder, eliminando toda la información asociada (como contratos y furgones) y cerrando automáticamente la sesión.

3.3.2 GESTIÓN DE FURGONES

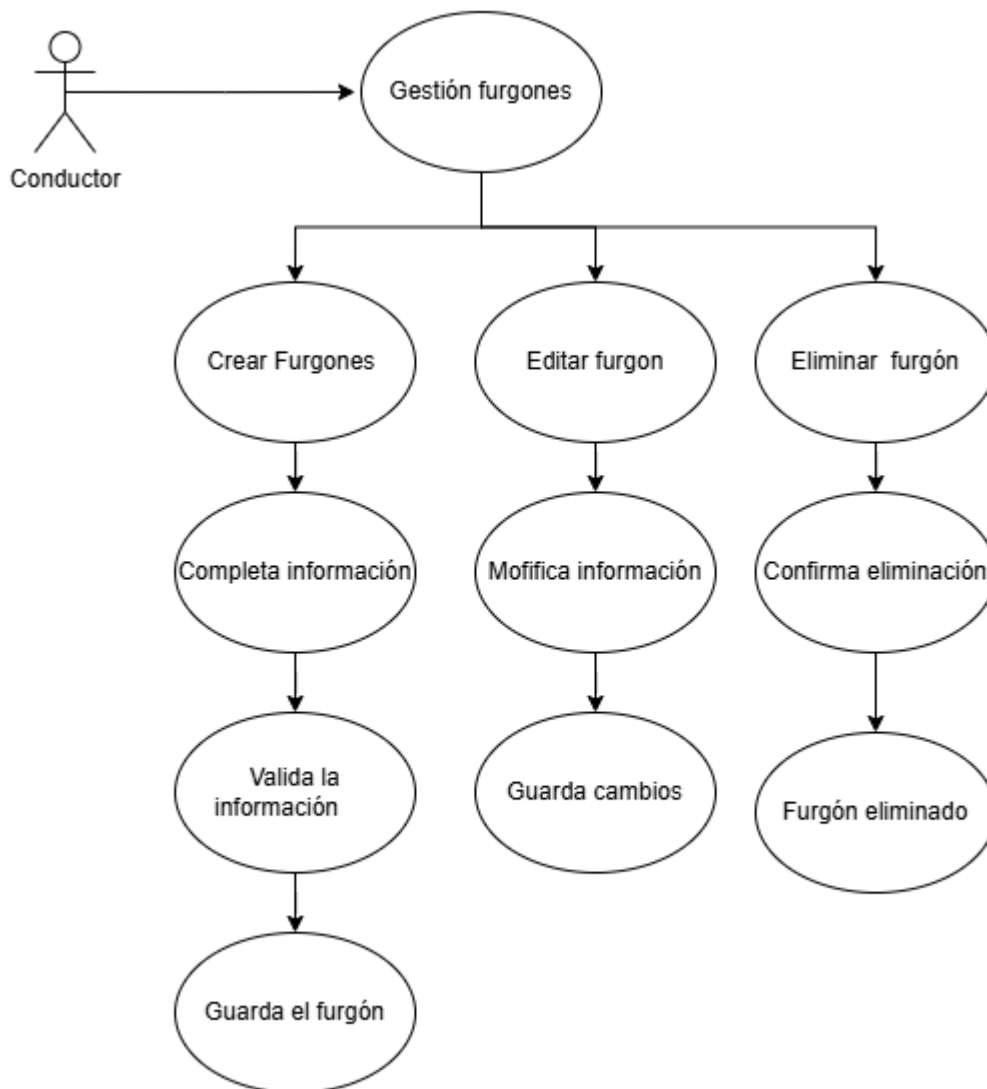


Figura 2: Diagrama de Casos de Usos: Gestión furgones. Fuente: Elaboración Propia.

Actor: Conductor

Descripción: Este diagrama describe el proceso de gestión de furgones que realiza un conductor en el sistema. El flujo principal incluye tres acciones: crear, editar y eliminar furgones.

En la creación de un furgón, el conductor completa la información requerida, que es validada por el sistema antes de guardarla. Si elige editar un furgón existente, el conductor modifica la información, guarda los cambios, y el sistema actualiza los datos correspondientes. En el caso de eliminar un furgón, el sistema solicita confirmación antes de proceder, eliminando definitivamente el furgón del registro.

El diagrama refleja cómo el conductor puede administrar los furgones asociados a su cuenta, asegurando que los datos ingresados sean consistentes y válidos en cada acción.

3.3.3 Gestión de contratos

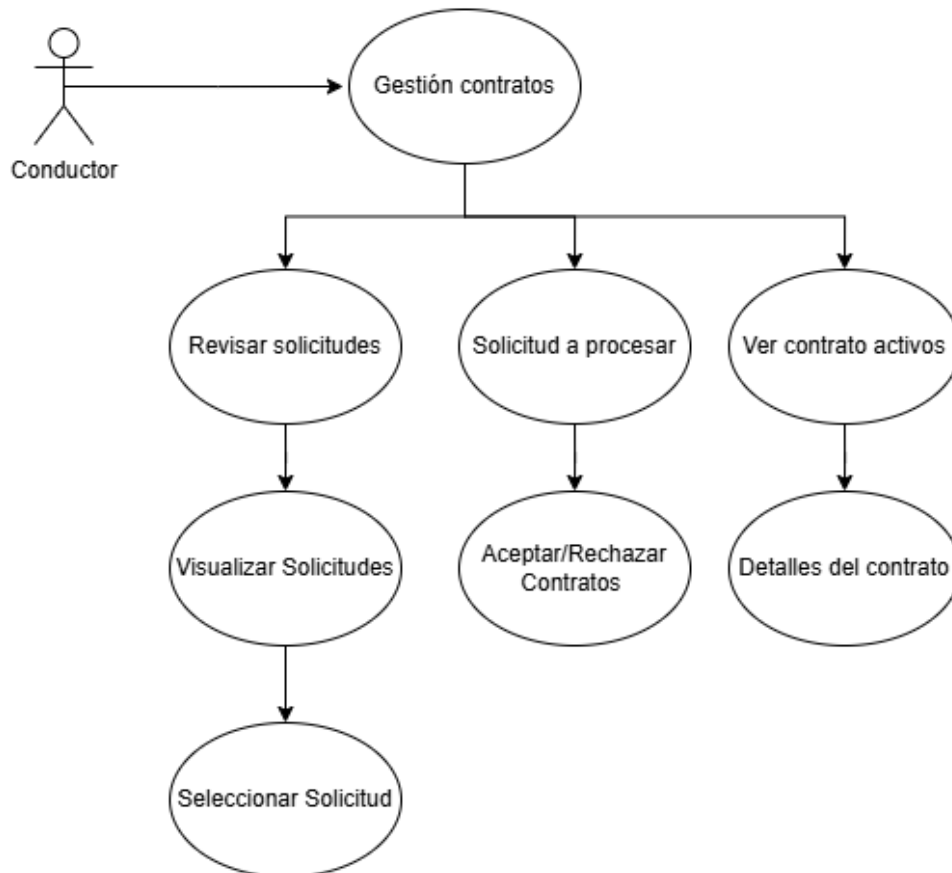


Figura 3: Diagrama de Casos de Usos: Gestión Contratos. Fuente: Elaboración Propia.

Actor: Conductor

Descripción: Este diagrama describe cómo el conductor gestiona los contratos en el sistema, incluyendo la revisión de solicitudes enviadas por los apoderados, el procesamiento de solicitudes mediante la aceptación o rechazo, y la visualización de los contratos activos. Cada etapa permite al conductor interactuar con las solicitudes y contratos asociados a su servicio, garantizando un flujo claro y eficiente para administrar las relaciones con los apoderados.

3.3.4 Ver listado de estudiantes

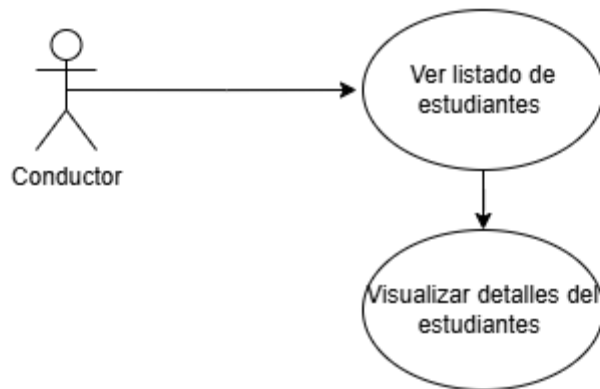


Figura 4: Diagrama de Casos de Usos: Listado de estudiantes. Fuente: Elaboración Propia.

Actor: Conductor

Descripción: Este diagrama muestra cómo el conductor puede gestionar la información de los estudiantes asignados a su furgón. Inicialmente, tiene acceso a un listado de estudiantes vinculados a los contratos activos. Desde este listado, puede seleccionar a un estudiante específico para visualizar detalles adicionales, como nombre, colegio y otros datos relevantes. Este proceso facilita al conductor la organización y el seguimiento de los estudiantes en su ruta.

3.3.4Gestion de solicitudes

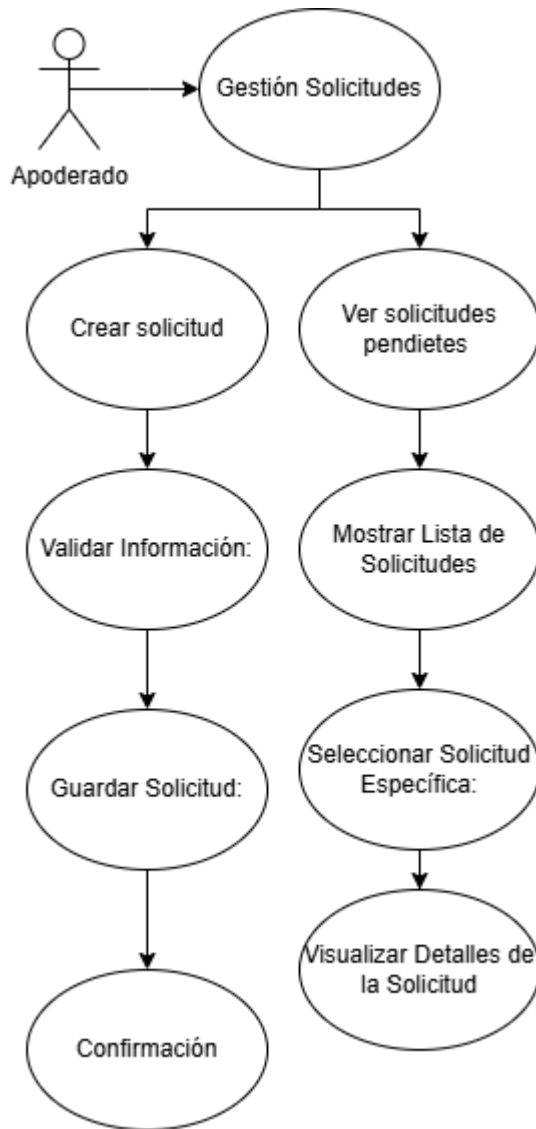


Figura 5: Diagrama de Casos de Usos: Gestión de solicitudes. Fuente: Elaboración Propia.

Actor: Apoderado

Descripción: El diagrama representa el proceso de gestión de solicitudes por parte del apoderado, permitiéndole crear nuevas solicitudes ingresando y validando información, que posteriormente se guarda en el sistema y se confirma. Además, incluye la opción de visualizar solicitudes pendientes, donde el apoderado puede consultar una lista, seleccionar una solicitud específica y revisar sus detalles para dar seguimiento al estado del proceso.

3.3.5 Inicio sesión

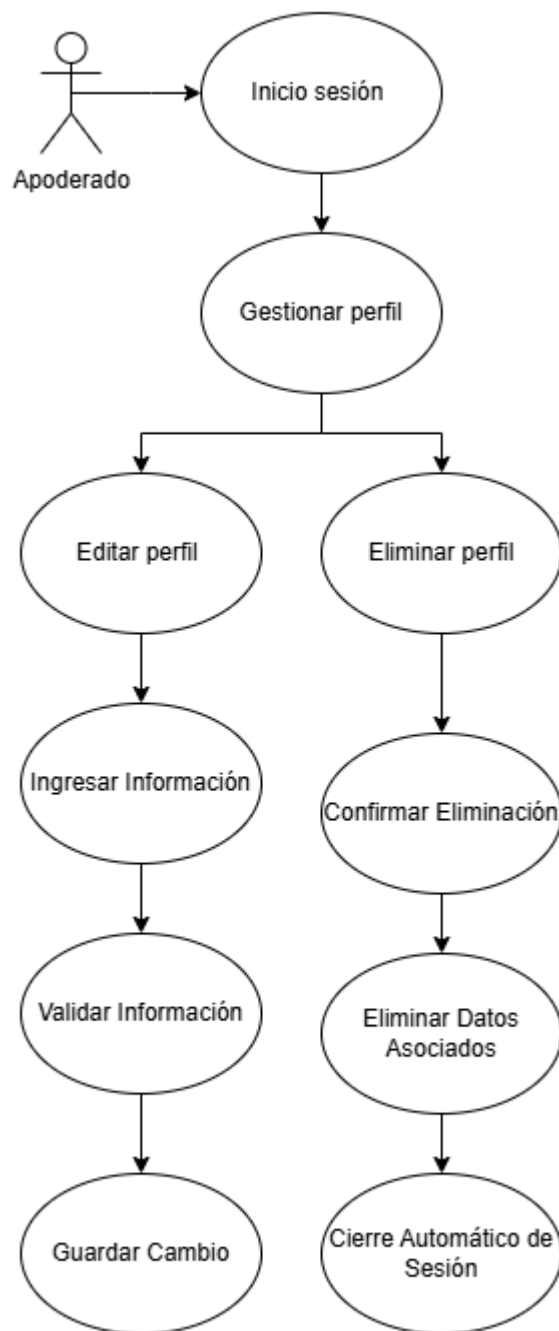


Figura 6: Diagrama de Casos de Usos: Gestión de contrato . Fuente: Elaboración Propia.

Actor: Apoderado

Descripción: El diagrama describe el proceso de gestión de perfil del apoderado en el sistema, que incluye dos flujos principales: editar perfil y eliminar perfil. En el caso de editar perfil, el apoderado puede modificar su información personal ingresando nuevos datos, los cuales son validados por el sistema antes de ser guardados y confirmados. En el flujo de eliminar perfil, el apoderado confirma la acción, tras lo cual el sistema elimina todos los datos asociados, como estudiantes, solicitudes y contratos, cerrando automáticamente la sesión como medida de seguridad. Este flujo asegura que el apoderado tenga control total sobre su información, permitiendo la gestión eficiente y segura de sus datos en el sistema.

3.3.6 Gestión de contrato

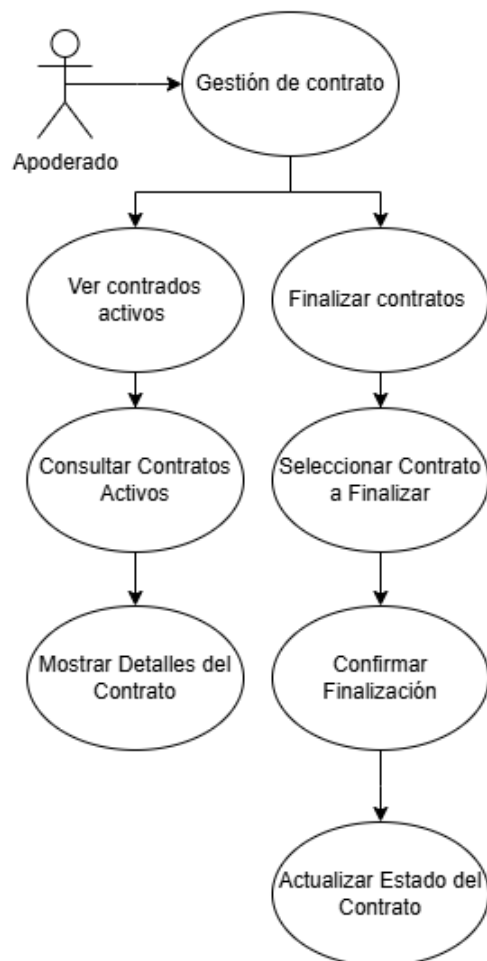


Figura 7: Diagrama de Casos de Usos: Gestión de contrato. Fuente: Elaboración Propia.

Descripción: Apoderado

Descripción: El diagrama muestra el proceso de **Gestión de Contrato** para el apoderado, dividido en dos flujos principales: **Ver Contratos Activos y Finalizar Contratos**. En el primer flujo, el apoderado puede consultar todos los contratos en curso y visualizar detalles como el conductor asignado, el furgón, los estudiantes vinculados y el estado del contrato. En el segundo flujo, el apoderado tiene la opción de finalizar un contrato específico, seleccionándolo de la lista de contratos activos, confirmando la acción, y permitiendo que el sistema actualice el estado del contrato a "finalizado", registrándolo en el historial. Este proceso facilita al apoderado la administración completa de sus contratos.

3.3.7 Gestión de estudiantes

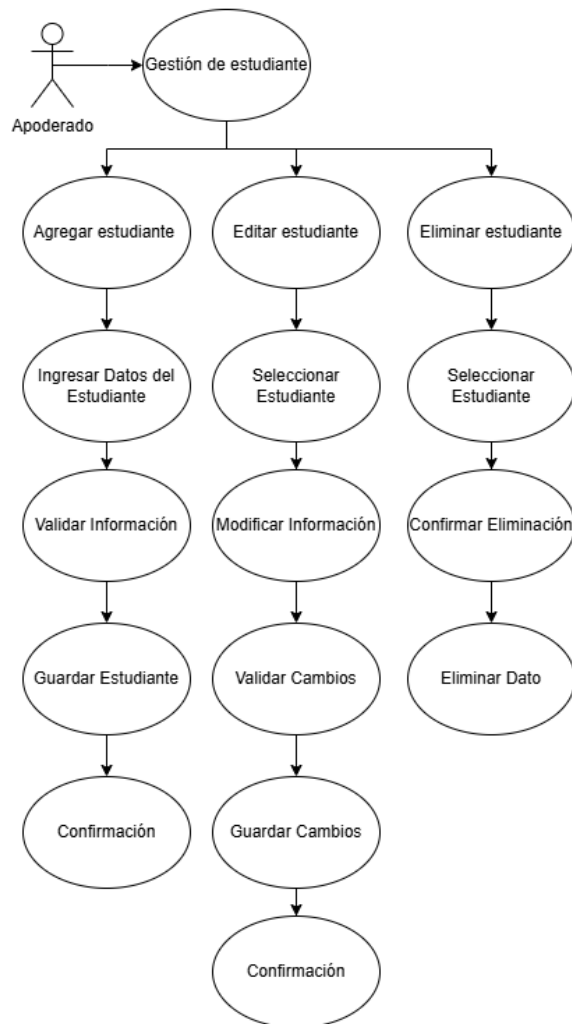


Figura 8: Diagrama de Casos de Usos: Gestión de contrato. Fuente: Elaboración Propia.

Actor: Apoderado

Descripción: El diagrama representa el proceso de **Gestión de Estudiantes** por parte del apoderado, permitiéndole agregar, editar y eliminar estudiantes asociados a su cuenta. En el flujo de agregar, el apoderado ingresa la información del estudiante, la cual es validada y guardada en el sistema, confirmando su registro. Para editar, el apoderado selecciona un estudiante, modifica los datos, y guarda los cambios tras la validación del sistema. Finalmente, en el flujo de eliminar, el apoderado selecciona un estudiante, confirma la acción, y el sistema elimina todos los datos asociados, notificando la eliminación exitosa. Este proceso proporciona una gestión completa y eficiente de los estudiantes.

3.5 Modelo de arquitectura

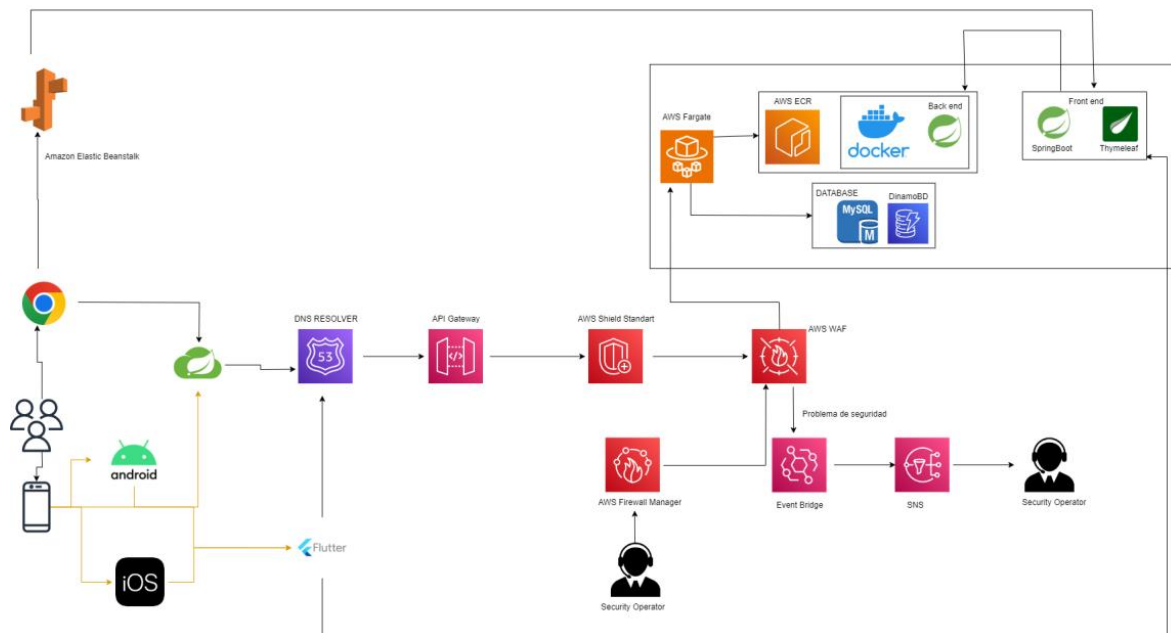


Figura 9: Arquitectura de Servicio: Proyecto FurgoFinder. Fuente: Elaboración Propia

La figura representa una arquitectura tecnológica basada en servicios de AWS (Amazon Web Services) para el desarrollo y despliegue de una aplicación que integra múltiples capas y tecnologías. A continuación, se describe el flujo y los componentes principales de la arquitectura:

Front-End:

- Los usuarios interactúan con la aplicación mediante navegadores web o dispositivos móviles (Android e iOS). El front-end está desarrollado con

SpringBoot y Thymeleaf, ofreciendo una interfaz de usuario dinámica y funcional.

DNS Resolver:

- Route 53 de AWS actúa como el sistema de resolución de nombres, facilitando el enrutamiento del tráfico hacia los servicios alojados.

API Gateway:

- Gestiona las solicitudes desde los clientes hacia los servicios backend, asegurando una comunicación eficiente y segura.

Seguridad:

- AWS Shield Standard protege contra ataques de denegación de servicio (DDoS).
- AWS WAF (Web Application Firewall) mitiga amenazas a nivel de aplicación.
- Si se detecta un problema de seguridad, este se maneja mediante EventBridge, AWS Firewall Manager y SNS (Simple Notification Service), notificando a los operadores de seguridad para una intervención inmediata.

Backend:

- El backend utiliza Spring Boot desplegado en contenedores Docker. Estos contenedores son gestionados por AWS Fargate, que asegura un entorno escalable y eficiente.
- Los datos se almacenan en bases de datos MySQL (relacional) y DynamoDB (NoSQL), asegurando flexibilidad en el manejo de la información.

Despliegue:

- AWS Elastic Beanstalk facilita el despliegue y la gestión del entorno para las aplicaciones web.

Flujo General:

- Los usuarios envían solicitudes desde sus dispositivos al DNS Resolver. Desde allí, las solicitudes pasan al API Gateway y son dirigidas al backend.

- El backend interactúa con las bases de datos para procesar y devolver la información solicitada.
- Todas las interacciones están protegidas mediante las capas de seguridad proporcionadas por AWS, incluyendo WAF y Shield.

Esta arquitectura demuestra un enfoque robusto y escalable para el desarrollo de aplicaciones modernas, asegurando tanto el rendimiento como la seguridad de la información en todo momento.

3.6 Modelo de datos

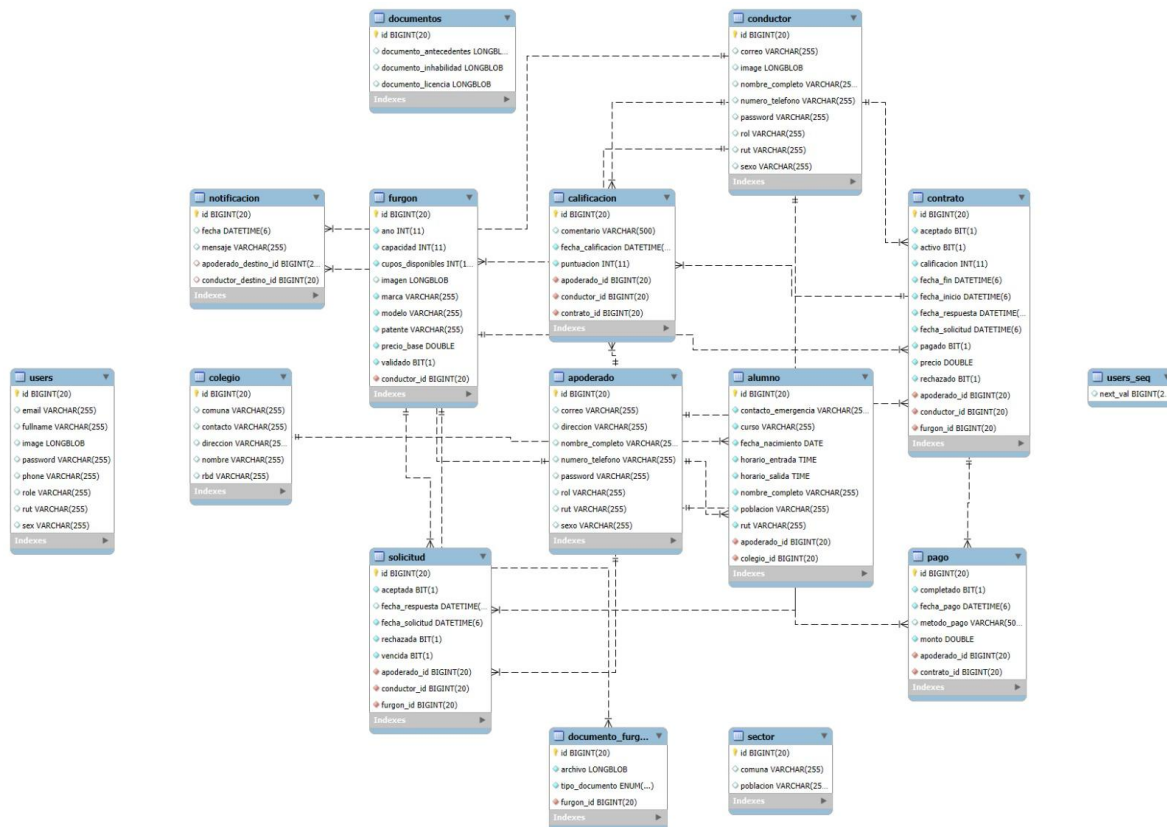


Figura 10: Base de Datos Proyecto FurgoFinder. Fuente: Elaboración Propia

Esta base de datos está diseñada para un sistema de gestión de transporte escolar, posiblemente relacionado con tu proyecto **FurgoFinder**. A continuación, te doy un resumen de las tablas y sus relaciones principales:

Tablas principales y su propósito:

usuarios:

- Gestiona información básica de todos los usuarios del sistema.
- Contiene datos comunes como correo, nombre completo, contraseña, rol, RUT y género.

conductor:

- Detalla información específica de los conductores.
- Incluye datos como imagen, nombre, correo, teléfono, y relaciones con otras tablas como documentos, furgones y contratos.

apoderado:

- Registra información sobre los apoderados (tutores).
- Contiene datos como correo, dirección, teléfono, y relaciones con estudiantes, contratos y solicitudes.

alumno:

- Contiene información de los estudiantes.
- Se vincula con apoderados, colegios y, a través de contratos, con furgones.

furgon:

- Representa los vehículos de transporte escolar.
- Incluye detalles como capacidad, modelo, marca, patente, validación y relación con conductores, sectores y contratos.

colegio:

- Almacena información de los colegios asociados al sistema.
- Se relaciona con alumnos y sectores.

documentos:

- Almacena documentos necesarios, como antecedentes y licencias de los conductores.
- Está vinculado con los conductores.

contrato:

- Gestiona la relación entre apoderados, conductores y furgones para la prestación del servicio.
- Incluye información como fechas, precios, estado del contrato (aceptado, rechazado, pagado) y evaluaciones.

solicitud:

- Registra solicitudes realizadas por apoderados para contratar un servicio de transporte.
- Contiene estados como aceptada, rechazada o vencida.

pago:

- Registra los pagos realizados por los apoderados.
- Incluye información como monto, método de pago y estado (completado).

notificacion:

- Administra mensajes y notificaciones entre apoderados y conductores.
- Se relaciona con los usuarios (apoderados y conductores).

documento_furgon:

- Gestiona los documentos relacionados con los furgones, como seguros o permisos.
- Está vinculado con la tabla **furgon**.

calificacion:

- Registra las calificaciones que apoderados otorgan a los conductores.
- Se relaciona con los contratos.

sector:

- Define sectores de operación de los furgones.
- Está vinculado con las tablas de furgón y colegio.

Relaciones clave:

- **Usuarios y roles:** Los usuarios tienen roles específicos (apoderado o conductor), y esto se extiende a las tablas de apoderado y conductor.
- **Apoderado - Alumno - Colegio:** Cada apoderado puede registrar varios alumnos, y cada alumno está vinculado a un colegio.
- **Conductor - Furgón:** Cada conductor puede gestionar uno o más furgones, con documentos asociados.
- **Solicitud - Contrato:** Las solicitudes pueden convertirse en contratos una vez aceptadas.
- **Pago y contrato:** Los pagos están vinculados directamente a contratos específicos.
- **Documentos:** Tanto los conductores como los furgones requieren documentos validados.

Propósito:

Esta base de datos permite gestionar un sistema integral de transporte escolar, abarcando desde la inscripción de estudiantes hasta la asignación de conductores y la gestión de solicitudes, contratos y pagos. Las relaciones están diseñadas para garantizar que todas las operaciones estén conectadas y que se pueda realizar un seguimiento completo de las actividades del sistema.

3.8 Arquitectura del Backend

El desarrollo del backend de **FurgoFinder** fue una de las etapas más importantes del proyecto, ya que implicó diseñar la lógica central del sistema que conecta las distintas interfaces (móvil y web) con la base de datos, garantizando la seguridad, el control de acceso y la gestión de información de forma estructurada.

3.8.1 Estructura General

El backend se basa en una arquitectura en capas que separa claramente los componentes lógicos:

- **Modelos (models.py):** Las entidades definidas en el sistema utilizando SQLAlchemy como ORM. Cada clase (como Alumno, Apoderado, Conductor, Contrato, Furgon, entre otras) representa una tabla en la base de datos. Esta estructura permite trabajar con objetos Python en lugar de escribir consultas SQL manualmente.

“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de Transporte Escolar”

```
class Alumno(db.Model):
    __tablename__ = 'alumno'

    id = db.Column(db.Integer, primary_key=True)
    nombre_completo = db.Column(db.String(255), nullable=False)
    rut = db.Column(db.String(20), unique=True, nullable=False)
    fecha_nacimiento = db.Column(db.Date, nullable=False)
    horario_entrada = db.Column(db.Time, nullable=False)
    horario_salida = db.Column(db.Time, nullable=False)
    curso = db.Column(db.String(50), nullable=False)
    direccion_hogar = db.Column(db.String(255), nullable=True)
    nombre_contacto_emergencia = db.Column(db.String(255), nullable=True)
    contacto_emergencia = db.Column(db.String(15), nullable=False)

    # Relaciones ManyToOne
    colegio_id = db.Column(db.Integer, db.ForeignKey('colegio.id'), nullable=False)
    colegio = db.relationship('Colegio', backref='alumnos_colegio') # Cambiado el backref
    apoderado_id = db.Column(db.Integer, db.ForeignKey('apoderado.id'), nullable=False)
    sector_id = db.Column(db.Integer, db.ForeignKey('sector.id'), nullable=False)
    sector = db.relationship('Sector', backref='alumnos_sector') # Cambiado el backref
```

```
class Conductor(db.Model):
    __tablename__ = 'conductor'
    id = db.Column(db.Integer, primary_key=True)
    nombre_completo = db.Column(db.String(255), nullable=False)
    rut = db.Column(db.String(20), unique=True, nullable=False) # Ejemplo: largo máximo de 20 para el RUT
    correo = db.Column(db.String(255), unique=True, nullable=False)
    password = db.Column(db.String(255), nullable=False)
    numero_telefono = db.Column(db.String(15), nullable=False) # Ejemplo: largo máximo de 15 para el número de teléfono
    sexo = db.Column(db.String(10), nullable=True) # Ejemplo: largo máximo de 10 para 'M'/'F' o 'Masculino'/'Femenino'
    rol = db.Column(db.String(50), nullable=True) # Ejemplo: largo máximo de 50 para roles como 'Admin', 'User', etc.
    # Campo LOB (Large Object Binary) para la imagen
    image = db.Column(db.LargeBinary, nullable=True)
    # Relación OneToMany con Asistente
    asistentes = db.relationship('Asistente', backref='conductor', cascade='all, delete-orphan', lazy=True)
```

```
class Furgon(db.Model):
    __tablename__ = 'furgon'

    id = db.Column(db.Integer, primary_key=True)
    patente = db.Column(db.String(20), unique=True, nullable=False) # Ejemplo: largo máximo de 20 para la patente
    marca = db.Column(db.String(50), nullable=False) # Ejemplo: largo máximo de 50 para la marca
    modelo = db.Column(db.String(50), nullable=False) # Ejemplo: largo máximo de 50 para el modelo
    ano = db.Column(db.Integer, nullable=False)
    capacidad = db.Column(db.Integer, nullable=False)
    cupos_disponibles = db.Column(db.Integer, nullable=False)
    precio_base = db.Column(db.Float, nullable=False)
    validado = db.Column(db.Boolean, nullable=False)

    # Relaciones ManyToOne
    conductor_id = db.Column(db.Integer, db.ForeignKey('conductor.id'), nullable=False)
    #conductor = db.relationship('Conductor', backref='furgones')

    sector_id = db.Column(db.Integer, db.ForeignKey('sector.id'), nullable=False)
    #sector = db.relationship('Sector', backref='furgones')

    colegio_id = db.Column(db.Integer, db.ForeignKey('colegio.id'), nullable=False)
    #colegio = db.relationship('Colegio', backref='furgones')

    # Relaciones OneToMany
    contratos = db.relationship('Contrato', backref='furgon', cascade='all, delete-orphan', lazy=True)
    imagenes = db.relationship('ImagenFurgon', backref='furgon', cascade='all, delete-orphan', lazy=True)
```

- **Repositorios (/repositories):** Cada entidad cuenta con un repositorio propio encargado de encapsular las operaciones de acceso a la base de datos. Estos módulos realizan consultas, inserciones, actualizaciones y eliminaciones, y permiten reutilizar la lógica de persistencia en distintos servicios o controladores.

```
#Alumno Repository
#Este módulo contiene las funciones para interactuar con la base de datos relacionadas con el modelo Alumno
from models import Alumno, db

def get_all_alumnos():
    return Alumno.query.all()

def get_alumno_by_id(alumno_id):
    return Alumno.query.get(alumno_id)

def create_alumno(data):
    alumno = Alumno(**data)
    db.session.add(alumno)
    db.session.commit()
    return alumno

def update_alumno(alumno_id, data):
    alumno = Alumno.query.get(alumno_id)
    if not alumno:
        return None
    for key, value in data.items():
        setattr(alumno, key, value)
    db.session.commit()
    return alumno

def delete_alumno(alumno_id):
    alumno = Alumno.query.get(alumno_id)
    if not alumno:
        return False
    db.session.delete(alumno)
    db.session.commit()
    return True
```

- **Servicios:** Los servicios encapsulan la lógica de negocio, incluyendo validaciones, reglas específicas y transformaciones de datos. Se diseñaron servicios como alumno_service.py, apoderado_service.py, contrato_service.py, entre otros. Esta capa permite mantener los controladores livianos y organizados, y promueve la reutilización de código.

```
def obtener_alumno_por_id(alumno_id):
    alumno = alumno.query.get(alumno_id)
    if not alumno:
        raise NotFound(f"Alumno con ID {alumno_id} no encontrado")
    return alumno

def obtener_alumno_por_rut(rut):
    alumno = get_alumno_by_rut(rut)
    if not alumno:
        raise NotFound(f"No existe un alumno registrado con el RUT {rut}")
    return alumno
```

- **Controladores:** Aquí se definen los endpoints de la API RESTful, encargados de recibir peticiones HTTP, delegar la lógica al servicio correspondiente y retornar una respuesta estructurada en formato JSON.

```
@app.route('/alumnos', methods=['POST'])
def create_alumno():
    try:
        data = request.get_json()
        if not data:
            return jsonify({'error': 'Solicitud vacía o malformada'}), 400

        nuevo = registrar_nuevo_alumno(data)
        return jsonify({'mensaje': 'Alumno creado exitosamente', 'id': nuevo.id}), 201

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

```
@app.route('/contrato', methods=['POST'])
def create_contrato():
    try:
        data = request.get_json()
        if not data:
            return jsonify({'error': 'El cuerpo de la solicitud está vacío o no es JSON válido'}), 400

        contrato = crear_contrato_completo(data)
        return jsonify({'mensaje': 'Contrato creado exitosamente', 'id': contrato.id}), 201

    except ValueError as ve:
        return jsonify({'error': str(ve)}), 400
    except Exception as e:
        db.session.rollback()
        return jsonify({'error': 'Error interno al crear contrato', 'detalles': str(e)}), 500
```

- **Seguridad (auth.py):** Se implementó una capa de autenticación utilizando bcrypt para el cifrado de contraseñas y Flask-Session para el manejo de sesiones. Esta capa gestiona el inicio de sesión, el almacenamiento temporal de los datos del usuario autenticado y el control de acceso basado en roles (apoderado, conductor). También se dejó preparada la arquitectura para una futura integración con JWT (JSON Web Tokens).

```
# Verificar token
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None

        # El token se pasa en el header Authorization: Bearer <token>
        if 'Authorization' in request.headers:
            auth_header = request.headers['Authorization']
            if auth_header.startswith("Bearer "):
                token = auth_header.split(" ")[1]

        if not token:
            return jsonify({'message': 'Token faltante'}), 401

        try:
            data = jwt.decode(token, SECRET_KEY, algorithms=['HS256'])
            request.user = data
        except jwt.ExpiredSignatureError:
            return jsonify({'message': 'Token expirado'}), 401
        except jwt.InvalidTokenError:
            return jsonify({'message': 'Token inválido'}), 401

        return f(*args, **kwargs)
    return decorated
```

3.8.2 Endpoints RESTful

Todos los recursos se exponen mediante métodos HTTP semánticamente adecuados y responden en formato JSON, permitiendo su consumo tanto desde aplicaciones web como móviles. Conjunto de endpoints estructurados conforme a los principios de diseño REST. Entre los más relevantes se encuentran

- **Autenticación:**

POST /login: Verifica las credenciales del usuario utilizando bcrypt, y retorna sus datos si la autenticación es exitosa

- **Gestión de entidades:** CRUD completo para entidades como alumno, apoderado, conductor, asistente, colegio y furgón.

- **Contratos y relaciones:**

POST /contrato: Permite registrar un contrato asociado a un alumno, apoderado y conductor, validando previamente la existencia de relaciones consistentes entre

estas entidades. La lógica correspondiente se encuentra encapsulada en el servicio de contratos.

3.8.3 Seguridad y Autenticación

El backend contempla una arquitectura de seguridad basada en:

JWT (JSON Web Tokens): La arquitectura contempla la futura implementación de JSON Web Tokens para ofrecer autenticación sin estado y mayor compatibilidad con entornos distribuidos.

Sesiones de usuario: Se empleó Flask-Session para gestionar sesiones activas. Durante la autenticación, se almacenan datos como el ID del usuario, su rol y su nombre completo, los cuales se utilizan para personalizar y proteger el uso del sistema.

Bcrypt: Utilizado para el hashing de contraseñas, eliminando el almacenamiento de credenciales en texto plano.

Control de roles: Se implementó un sistema de roles para restringir el acceso a determinadas rutas según el tipo de usuario autenticado (apoderado, conductor, asistente).

3.8.4 Persistencia de Datos

Para la persistencia de datos se utilizó el sistema de gestión de bases de datos MySQL, accedido mediante el conector mysql.connector. Se definió un modelo de datos relacional normalizado, incluyendo entidades como usuarios, contratos, furgones, colegios, y más.

El acceso a los datos se encuentra desacoplado gracias a la capa de repositorios. Estos módulos ejecutan directamente las operaciones CRUD sobre las entidades, mientras que los servicios actúan sobre los resultados obtenidos, aplicando lógica de negocio y devolviendo datos listos para ser serializados por los controladores

3.8.5 Integración Cliente–Servidor y Despliegue

El backend fue diseñado bajo el principio **API-First**, lo que permitió exponer de manera anticipada los endpoints requeridos por los frontends del sistema: El backend fue desplegado en un entorno PaaS (Platform-as-a-Service) utilizando Heroku, lo que aporta:

- La aplicación web fue desarrollada con **Spring Boot** y **Thymeleaf**.
- La aplicación móvil fue construida utilizando **Flutter**, y se conecta al backend mediante llamadas HTTP.

Para el despliegue, se utilizó la plataforma como servicio Heroku, lo que permitió:

- Exponer la API mediante protocolo HTTPS en un entorno de producción.
- Configurar integración continua con GitHub para despliegue automático.
- Garantizar alta disponibilidad y escalabilidad básica.
- Validar pruebas funcionales completas durante cada iteración de desarrollo.

3.9 Diseño de la interfaz de usuario

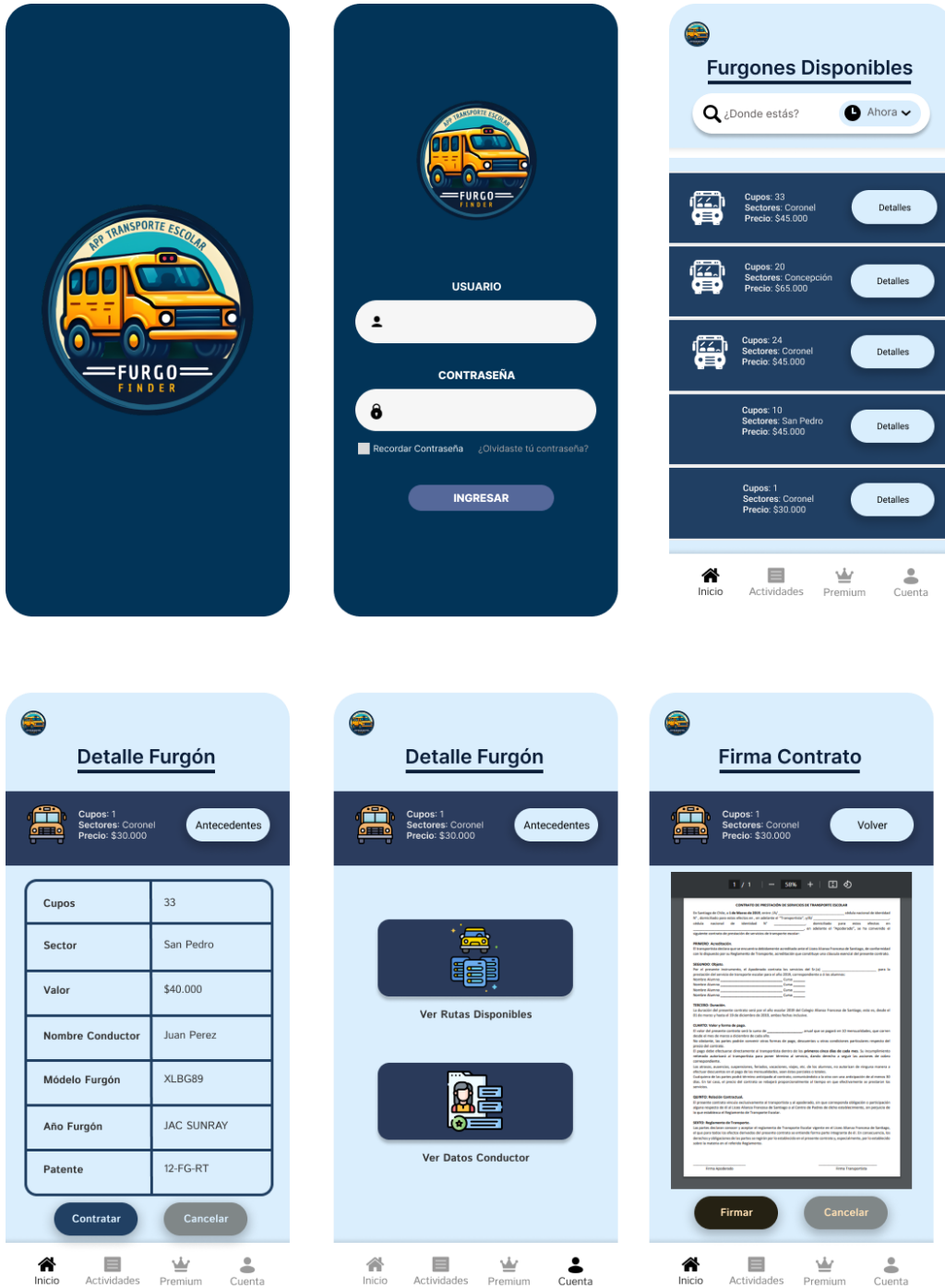
FurgoFinder - Solución Integral para el Transporte Escolar

La imagen presenta un diseño conceptual de la aplicación móvil FurgoFinder, una plataforma que conecta apoderados con conductores de transporte escolar. La interfaz se enfoca en la experiencia del usuario, priorizando la seguridad y comodidad en cada paso del proceso.

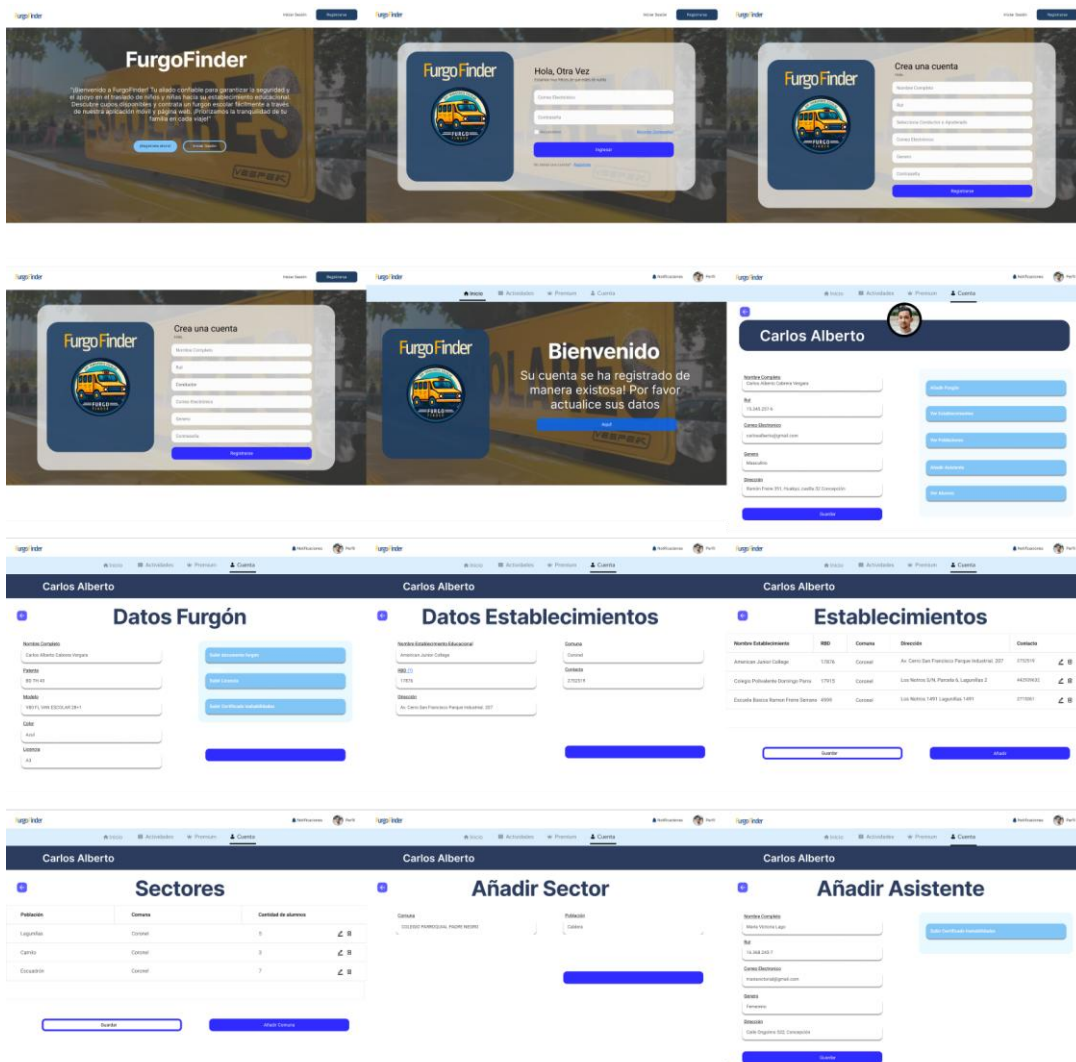
1. Pantalla de Inicio: Una bienvenida visual con el logo distintivo del servicio.
2. Pantalla de Login: Permite a los usuarios ingresar con su cuenta para acceder a las funcionalidades personalizadas.
3. Listado de Furgones Disponibles: Una sección que muestra los furgones según ubicación, capacidad y precio, con opciones para ver detalles.
4. Detalle del Furgón: Información completa del furgón, incluyendo cupos disponibles, sector, precio, datos del conductor, y detalles del vehículo. También ofrece opciones para cotizar o cancelar la selección.
5. Firma de Contrato: Una interfaz que permite la confirmación digital del servicio mediante la firma del contrato, garantizando formalidad y seguridad.

El diseño incluye un menú de navegación intuitivo en la parte inferior, con acceso rápido a las secciones de Inicio, Actividades, Premium y Cuenta, asegurando una experiencia de usuario fluida y eficiente.

**“BACKEND FURGO FINDER:
Solución Integral para la Gestión y Contratación de Transporte Escolar”**



“BACKEND FURGO FINDER: Solución Integral para la Gestión y Contratación de Transporte Escolar”



FurgoFinder - Plataforma de Gestión de Transporte Escolar

La imagen muestra un diseño más amplio de la plataforma **FurgoFinder**, incluyendo pantallas tanto para apoderados como para conductores, con un enfoque en la administración y funcionalidad del sistema.

Página Principal y Registro:

- **Inicio:** Interfaz atractiva con una llamada a la acción para registrarse o iniciar sesión.
- **Registro de Cuenta:** Formulario amigable que permite a los usuarios crear una cuenta ingresando datos básicos como correo y contraseña.
- **Pantalla de Bienvenida:** Notificación para nuevos usuarios destacando la necesidad de completar su perfil.

Panel de Usuario (Carlos Alberto):

- Muestra un diseño específico para conductores, con datos personales y acceso a funcionalidades clave, como gestión de furgones, establecimientos, sectores y asistentes.

Gestión de Datos:

- **Datos Furgón:** Formulario detallado para agregar información de los furgones, incluyendo capacidad, modelo y patente.
- **Datos de Establecimientos:** Permite registrar y visualizar los colegios asociados a las rutas.
- **Sectores:** Gestión de zonas de servicio con opciones para agregar o editar sectores específicos.

Funcionalidades de Administración:

- **Añadir Sector:** Herramienta para extender las áreas de cobertura del conductor.
- **Añadir Asistente:** Permite incluir asistentes de transporte para mejorar la calidad del servicio.

El diseño refleja una interfaz intuitiva y profesional, con un enfoque en la personalización para cada tipo de usuario, mejorando la experiencia tanto para apoderados como para conductores en la gestión de transporte escolar.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

4.1 Test de Usabilidad

Se llevaron a cabo pruebas de interacción con apoderados y conductores reales, donde se evaluaron aspectos como:

- **Facilidad de uso:** Navegación intuitiva en la aplicación.
- **Satisfacción:** Opiniones generales sobre la experiencia al utilizar FurgoFinder.
- **Identificación de mejoras:** Revisión de posibles puntos de fricción en la interfaz.

Resultados:

- **90%** de los usuarios encontraron la aplicación fácil de usar.
- Los usuarios destacaron la claridad en la información presentada y la utilidad de las funciones de seguimiento en tiempo real y alertas.

4.2 Encuestas de Aceptación

Se realizaron encuestas con usuarios potenciales para medir:

- **Aceptación de precios propuestos.**
- **Aceptación de los servicios ofrecidos.**

Tabla 10: Resultados de la Encuesta de Aceptación de Precios

Precio Propuesto	Porcentaje de Aceptación
\$ 1.500,00	10%
\$ 3.000,00	90%

Interpretación: El precio de \$3,000 fue ampliamente aceptado, validando su alineación con las expectativas del mercado.

Tabla 11: Aceptación de Servicios Ofrecidos

Servicio	Porcentaje de Aceptación
Seguimiento en Tiempo Real	85.7%
Aviso de Emergencias	85.7%
Compartir la Ruta en Tiempo Real	71.4%
Cambios de Ruta	75%

Interpretación: Las funcionalidades clave fueron bien recibidas, consolidando su relevancia para los usuarios.

4.3. Análisis de Competencia

Se comparó FurgoFinder con aplicaciones existentes como Bibip, Hakuna y Aveeza, destacando:

- **Ventajas Competitivas:**
 - Mayor enfoque en la validación de documentos y seguridad.
 - Funcionalidades adaptadas específicamente al transporte escolar.
 - Experiencia de usuario optimizada para apoderados y conductores.

Resultados:

FurgoFinder se posiciona como una solución integral que aborda las necesidades críticas que otras aplicaciones no cubren de manera específica.

4.4 Plan de prueba

Categoría	Caso de prueba	Endpoint	Entrada	Resultado Esperado
Funcional	Crear alumno con datos válidos	POST /alumnos	JSON correcto	201 Created
Funcional	Crear alumno con datos faltantes	POST /alumnos	JSON incompleto	400 Bad Request
Funcional	Obtener lista de alumnos	GET /alumnos	N/A	200 OK + lista
Funcional	Crear contrato válido	POST /contrato	JSON con IDs válidos	201 Created
Funcional	Login con credenciales correctas	POST /login	Correo + contraseña	200 OK + datos
Categoría	Caso de prueba	Endpoint	Entrada	Resultado Esperado
Seguridad	Contraseña incorrecta	POST /login	Correo correcto, contraseña errónea	401 Unauthorized
Seguridad	Usuario no encontrado	POST /login	Correo no existente	404 Not Found
Seguridad	Validación de rol no permitido	POST /login	Usuario con rol inválido	403 Forbidden

CAPÍTULO 5: CONCLUSIÓN

El desarrollo de FurgoFinder marcó un avance significativo tanto desde el punto de vista técnico como social, logrando digitalizar y profesionalizar el transporte escolar mediante la implementación de una plataforma robusta, segura y centrada en la experiencia del usuario. La construcción del backend se realizó utilizando Flask como microframework en Python, bajo una arquitectura modular y orientada a servicios que permitió separar de forma clara los modelos, repositorios, servicios, controladores y mecanismos de seguridad.

Desde el punto de vista funcional, se cumplió satisfactoriamente con los objetivos definidos: exponer una API RESTful limpia, implementar operaciones CRUD sobre entidades clave (como alumnos, apoderados, contratos y furgones), y establecer mecanismos de autenticación mediante cifrado con bcrypt, validación de roles y control de sesiones. Esta organización favoreció la escalabilidad del sistema y su preparación para integraciones futuras.

En paralelo, FurgoFinder logró abordar de manera efectiva problemáticas críticas del sector del transporte escolar, como la informalidad en la contratación, la falta de trazabilidad y la escasa conexión entre apoderados y conductores. Se integraron funcionalidades avanzadas como filtros inteligentes, validación de documentos y un sistema de precontratos, mejorando significativamente la usabilidad y transparencia del sistema.

El despliegue en Heroku bajo un esquema PaaS permitió validar la conectividad en distintos entornos, incluyendo aplicaciones móviles desarrolladas en Flutter y frontends web. Esto garantizó una arquitectura interoperable y fácilmente accesible para los usuarios finales. El enfoque API-First adoptado durante el desarrollo facilitó además la integración ágil entre equipos y tecnologías.

A nivel estratégico, FurgoFinder no solo representa una herramienta tecnológica innovadora, sino que también promueve la formalización y profesionalización de un sector tradicionalmente informal. Las funcionalidades de control, seguridad y trazabilidad impactan directamente en la confianza de los usuarios, especialmente apoderados, aportando así a la mejora de la calidad de vida de las comunidades escolares.

Las limitaciones detectadas, como la necesidad de mayores recursos para implementar funciones avanzadas (por ejemplo, seguimiento en tiempo real con IoT o algoritmos de optimización de rutas con IA), abren interesantes líneas de mejora para el futuro del sistema. Estas oportunidades, junto con la validación positiva obtenida en pruebas de usabilidad, consolidan a FurgoFinder como una base sólida para su evolución continua.

Desde una perspectiva académica y técnica, el proyecto destaca por su uso de herramientas modernas, principios de desarrollo ágil y una fuerte orientación al usuario. Esto lo convierte en un caso de estudio replicable para el desarrollo de soluciones

tecnológicas en sectores similares, contribuyendo tanto al campo del software como al impacto social mediante la inclusión digital.

En resumen, FurgoFinder establece un precedente valioso para la transformación digital del transporte escolar, integrando tecnología, seguridad y experiencia de usuario en una solución escalable y adaptable a nuevas demandas. Se recomienda continuar con su evolución funcional y técnica, profundizando en el desarrollo de nuevas características que respondan a las necesidades cambiantes del entorno y fomenten la innovación en servicios públicos.

Durante el desarrollo del proyecto se logró implementar exitosamente un backend completo utilizando Flask y SQLAlchemy, estructurado como una API RESTful modular que permite gestionar todas las entidades clave del sistema mediante operaciones CRUD. Este backend fue diseñado bajo una arquitectura desacoplada, con capas bien definidas para modelos, repositorios, servicios, controladores y seguridad, lo cual facilita la mantenibilidad y la posibilidad de escalar o modificar componentes sin afectar la estructura general.

Asimismo, se desarrollaron y consumieron correctamente los endpoints desde ambas interfaces cliente: la aplicación móvil creada con Flutter y la versión web desarrollada con Spring Boot y Thymeleaf, lo que validó la interoperabilidad del sistema. El backend fue desplegado con éxito en Heroku, exponiéndose al entorno productivo mediante HTTPS. Además, se realizaron pruebas funcionales manuales con Postman, cubriendo distintos escenarios y verificando la estabilidad y correcto funcionamiento de los servicios expuestos.

REFERENCIAS BIBLIOGRÁFICAS

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change* (2nd Edition). Addison-Wesley.

Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code* (2nd Edition). Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Hartley, M. (2022). *Spring Boot in Practice*. Manning Publications.

Richards, M. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media.

Amazon Web Services. (2023). *Amazon EC2 Documentation*. Disponible en: <https://aws.amazon.com/documentation/ec2/>

Oracle. (2023). *MySQL Documentation*. Disponible en: <https://dev.mysql.com/doc/>

Spring. (2023). *Spring Boot Documentation*. Disponible en: <https://spring.io/projects/spring-boot>

Jakob Nielsen. (1993). *Usability Engineering*. Academic Press.

Thymeleaf. (2023). *Thymeleaf Documentation*. Disponible en: <https://www.thymeleaf.org/documentation.html>

Diario El Mercurio. (2023). *Transporte escolar en Chile: riesgos y soluciones tecnológicas*. Disponible en: <https://www.elmercurio.com/>

La Tercera. (2023). *El transporte escolar en cifras: la necesidad de modernización en el país*. Disponible en: <https://www.latercera.com/>

TVU(2024) Tres mil vehículos inscritos: llaman a denunciar a furgones escolares ilegales <https://www.tvu.cl/>

Flask Documentation. (2024). Welcome to Flask — Flask Documentation.

Richardson, L., & Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media.