

Sistematización de la Gestión Administrativa de Información y Lógica de Negocio para la Recolección de Desechos en la Empresa Aguas de la Sabana

Juan Camilo Varela Ocoró
Juan.varela.ocoro@correounivalle.edu.co

Modalidad:
Pasantía

Director:
Héctor Fabio Ocampo Arbeláez
Profesor de la Escuela de Ingeniería de Sistemas y Computación
hector.ocampo@correounivalle.edu.co

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá - Valle del Cauca
2024



Sistematización de la Gestión Administrativa de Información y Lógica de Negocio para la Recolección de Desechos en la Empresa Aguas de la Sabana

Juan Camilo Varela Ocoró
Juan.varela.ocoro@correounivalle.edu.co

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá - Valle del Cauca
2024

Tabla de Contenido

Resumen **6**

1. Formulación del Problema. **7**

2. Descripción del Problema. **8**

3. Definición del problema **8**

4. Marco referencial **8**
4.1. Marco teórico . . . 8
4.1.1. Procesos de gestión de residuos . . . 8
4.1.2. Arquitecturas de software: monolíticas y microservicios . . . 9
4.1.3. Antecedentes de aplicaciones similares . . . 11
4.2. Marco conceptual . . . 13
4.2.1. Estado del arte . . . 15

5. Objetivos **16**
5.1. Objetivo general . . . 16
5.2. Objetivos específicos . . . 16

6. Análisis de procesos previos al sistema **18**
6.1. Investigación según lógica de negocio . . . 18
6.2. Resultados del análisis . . . 18
6.3. CASO 1: Vinculación de un cliente. . . . 18
6.3.1. **H.U-1:** . . . 18
6.3.2. **H.U-2:** . . . 19
6.4. **CASO 2:** Programación de rutas de recolección. . . . 20
6.4.1. H.U-1: . . . 20
6.4.2. **H.U-2:** . . . 21
6.5. **CASO 3:** Proceso de recolección en terreno (App). . . . 22
6.5.1. **H.U-1:** . . . 22
6.5.2. **H.U-2:** . . . 22
6.5.3. **H.U-3:** . . . 23
6.5.4. **H.U-4:** . . . 24
6.5.5. **H.U-5:** . . . 26
6.5.6. **H.U-6:** . . . 26
6.5.7. H.U-7: . . . 27
6.5.8. H.U-8: . . . 28
6.5.9. H.U-9: . . . 29
6.6. **CASO 4:** Proceso de cotización de residuos especiales . . . 29
6.6.1. H.U-1: . . . 29
6.6.2. H.U-2: . . . 30

7. Arquitectura global del proyecto: Oracle, SQL Developer y microservicios **32**
7.1. Esquema de proyecto . . . 32
7.2. Frontend . . . 32
7.2.1. Aquitectura de diseño: **App web** . . . 33
7.2.2. Aquitectura de diseño: **App móvil** . . . 35
7.3. Backend . . . 37
7.3.1. Aquitectura de diseño: **Backend** . . . 37
7.4. Base de datos . . . 38
7.5. Esquema de base de datos . . . 40
7.5.1. Micro Route Business (Usuarios, Rutas y Micro-rutas): . . . 41
7.5.2. Micro Route Config (Rutas y Micro-rutas): . . . 42
7.5.3. Aforo (Recolecciones y recipientes): . . . 43
7.5.4. User (Control de acceso y datos de usuario): . . . 44
7.5.5. Client: . . . 44
7.5.6. Truck: . . . 45
7.5.7. Refuel: . . . 46
7.5.8. Incident: . . . 47
7.5.9. Provision: . . . 48

8. Configuración y características **49**
8.1. Alojamiento de repositorio: BitBucket . . . 49
8.1.1. Despliegues: Docker . . . 49

9. Implementación de prototipos **50**
9.1. Aplicación web: React, Mantine y TanStack . . . 50
9.1.1. Configuración inicial y desarrollo básico . . . 50
9.1.2. Creación del proyecto: . . . 50
9.1.3. Mockups de interfaz . . . 51
9.1.4. Uso de Mantine para componentes prefabricados: . . . 51
9.1.5. Exploración de Tan Stack Table: . . . 52
9.2. Implementación avanzada de tablas y optimización de componentes . . . 53
9.2.1. Desarrollo de la tabla con datos de clientes: . . . 53
9.2.2. Funcionalidad drag-and-drop: . . . 54
9.3. Edición y creación in-row: . . . 54
9.3.1. Creación: . . . 54
9.3.2. Edición: . . . 55
9.4. Integración con el backend y estabilidad del sistema . . . 55
9.4.1. Depuración de tablas y extras: . . . 55
9.4.2. Conexión al backend: . . . 55
9.4.3. Manejo de errores y validación: . . . 55
9.5. Aplicación móvil: Android Studio & Kotlin . . . 57
9.5.1. Elección de tecnologías y herramientas . . . 57
9.5.2. Mockups de interfaz . . . 57

9.5.3. Implementación del diseño de la interfaz . . . 58
9.5.4. Gestión de la comunicación y datos . . . 58
9.5.5. Pruebas y aseguramiento de calidad . . . 58
9.5.6. Optimización y rendimiento: Tareas asíncronas . . . 58

10. Adición al trabajo movil terminado **59**
10.1. Semana 1: Análisis y propuesta inicial . . . 59
10.1.1. Análisis del código existente: . . . 59
10.1.2. Propuesta de base de datos local: . . . 59
10.1.3. Requerimientos funcionales: . . . 59
10.2. Semana 2: Reestimación y cambio de enfoque . . . 59
10.2.1. Reestimación de la complejidad técnica: . . . 59
10.2.2. Segunda etapa de la implementación: . . . 60
10.3. Semanas 3 y 4: Implementación definitiva y refinamiento . . . 60
10.3.1. Cambio en la estrategia de persistencia: . . . 60
10.3.2. Flujo de sincronización: . . . 60

11. Capactaciones **62**
11.1. Mecanismos de recolección de feedback . . . 62
11.2. Resultados de validación . . . 62

12. Metodología **63**
12.1. Metodología de desarrollo . . . 63
12.2. Organización del trabajo . . . 63
12.3. Seguimiento y control . . . 63
12.4. Evaluación y retroalimentación . . . 63
12.5. Resultados . . . 64
12.6. Proceso de capacitación al personal . . . 64
12.7. Métricas de adopción y mejora del proceso . . . 65

13. Conclusiones **66**
13.1. ¿Por qué estas tecnologías? . . . 66
13.2. Contribución académica . . . 67
13.3. Trabajos futuros... . . . 68

14. Anexos **69**
14.1. Anexo 1: Esquema completo de base de datos . . . 69
14.2. Anexo 2: Credenciales y primera vista . . . 70
14.3. Anexo 3: Validación mecanica del vehiculo . . . 71
14.4. Anexo 4: Toma de un aforo y reportes . . . 72
14.5. Anexo 5: Inicio de recorrido . . . 73
14.6. Anexo 6: Pausa de ruta . . . 74
14.7. Anexo 6: Repostaje . . . 75
14.8. Anexo 7: Registro de incidencias . . . 76
14.9. Anexo 8: Finalizacion de ruta . . . 77

14.10Anexo 9: Inicio de viaje a relleno sanitario 78
14.11Anexo 10: Registro de ingreso a relleno 79
14.12Anexo 11: Finalizacion y retorno a base 80

15.Referencias **81**

Glosario

Tabla 1: Glosario de términos técnicos

Término	Definición
Stack tecnológico	Conjunto de tecnologías de software utilizadas en el desarrollo de una solución (frameworks, lenguajes, bases de datos, etc.).
Microservicio	Unidad funcional independiente que ejecuta una tarea específica y se comunica con otros servicios mediante APIs.
CRUD	Acrónimo de Create, Read, Update, Delete. Representa las operaciones básicas sobre datos.
Base de datos relacional	Sistema de almacenamiento de datos organizado en tablas relacionadas entre sí mediante llaves primarias y foráneas.
REST API	Interfaz de programación basada en el protocolo HTTP que permite la comunicación entre sistemas de forma estructurada.
Aforo	Medición del volumen o cantidad de residuos recolectados en un punto determinado.
Checklist	Lista de verificación que permite asegurar el cumplimiento de pasos o requisitos durante una operación.
Frontend	Parte visual de una aplicación, con la cual interactúa el usuario final.
Backend	Parte lógica de la aplicación que gestiona reglas de negocio, procesamiento y acceso a datos.
Arquitectura MVC	Modelo de arquitectura que separa los componentes de una aplicación en Modelo, Vista y Controlador.
Arquitectura MVVC	Variante del patrón MVC que introduce una capa de modelo de vista para desacoplar la interfaz del modelo.
Docker	Herramienta de virtualización ligera para empaquetar aplicaciones junto a sus dependencias en contenedores portables.
Despliegue continuo (CI/CD)	Proceso automatizado que permite la integración, pruebas y entrega rápida de cambios en un sistema.
Punto crítico de control	Etapa del proceso en la que se debe ejercer control específico para prevenir errores o desviaciones operativas.

Resumen

En este documento se aborda la profundización respecto a la necesidad de un sistema integral alrededor del gestionamiento y la administración de la información de las operaciones de negocio en la empresa Aguas de la Sabana de Bogotá S.A E.S.P., una empresa privada prestadora de los servicios de acueducto, alcantarillado y aseo, que contribuye al desarrollo de los municipios de Cota, Funza y Tenjo [1].

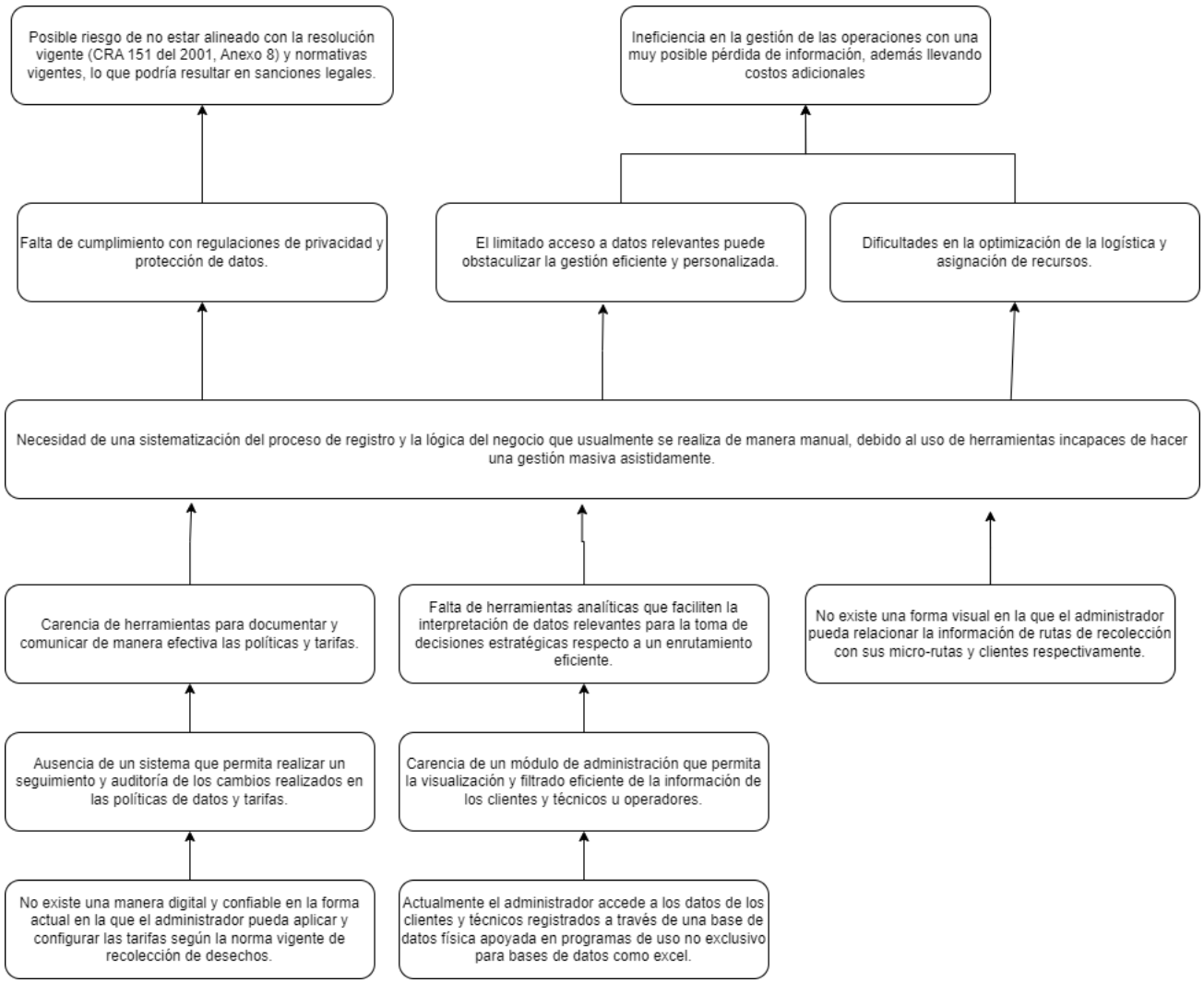
El objetivo es mejorar la administración de acuerdos y tarifas según norma CRA 151 del 2001, Anexo 8 [2], la gestión de clientes y técnicos/operarios registrados, la relación de información de rutas, micro rutas y clientes correspondientes a las mismas, y la gestión de los recursos operacionales como los camiones de recolección, entre otros. Contribuyendo así a la mejora de la experiencia tanto del administrador como de los operarios y al cumplimiento normativo de la empresa.

Palabras Clave: Application software, Client-server systems, Enterprise resource planning, Data aggregation

1. Formulación del Problema.

En el contexto actual de las empresas de servicios públicos, la gestión de la información relacionada con los procesos operativos y administrativos es un factor determinante para garantizar eficiencia, trazabilidad y cumplimiento normativo. Aguas de la Sabana, como prestadora del servicio de recolección de residuos sólidos en Bogotá, enfrenta desafíos operativos derivados del uso de herramientas manuales y sistemas desconectados que limitan la toma de decisiones y la transparencia del servicio.

Para entender de forma estructurada las causas, efectos y dimensiones del problema identificado, se construyó un árbol de problemas que permite visualizar la situación central que motiva este proyecto: la necesidad de sistematizar el proceso de registro y la lógica del negocio de recolección, con el fin de superar limitaciones tecnológicas, operativas y normativas.



2. Descripción del Problema.

La empresa Aguas de la Sabana de Bogotá S.A E.S.P., encargada de la prestación de servicios de acueducto, alcantarillado y aseo en los municipios de Cota, Funza y Tenjo, enfrenta diversos problemas que afectan la eficiencia y optimización de sus operaciones de recolección de residuos, especialmente hacia los grandes generadores y zonas agroindustriales. Actualmente, la gestión de los procesos de registro y seguimiento de las rutas de recolección, las micro rutas, y los datos de los clientes se realizan de manera manual. Esto conlleva varios inconvenientes, entre los cuales destacan:

- **Pérdida de información:** Al carecer de un sistema digital especializado, existe un riesgo significativo de pérdida de datos relevantes durante el proceso de recolección y registro, lo que puede obstaculizar la gestión eficiente y personalizada de las operaciones.
- **Costos adicionales:** La falta de automatización y la necesidad de realizar tareas manuales, repetitivas y amplias implican costos adicionales para la empresa, impactando su rentabilidad.
- **Dificultades logísticas:** La asignación y optimización de recursos, como los camiones de recolección, la tripulación operadora, entre otros, se ve dificultada debido a la falta de información centralizada y en tiempo real sobre las rutas y clientes.
- **Cumplimiento normativo:** La empresa enfrenta el riesgo de incumplir con la normatividad legal vigente de Comisión de Regulación de Agua Potable y Saneamiento Básico (CRA) 151 del 2001, Anexo 8 con la que se encuentran definidas las tarifas del servicio y otras normativas relacionadas con la privacidad y protección de datos, lo que podría acarrear sanciones legales.

3. Definición del problema

¿Cómo sistematizar el proceso de registro y gestión de recolección de residuos por medio de un producto de software?

4. Marco referencial

4.1. Marco teórico

4.1.1. Procesos de gestión de residuos

La gestión de residuos sólidos es un proceso fundamental para mantener un entorno sostenible y saludable. Este proceso implica la recolección, transporte, tratamiento y disposición final de los desechos generados por actividades humanas. La gestión eficiente de

residuos requiere la implementación de sistemas de información que permitan la automatización y optimización de los procesos involucrados, por ende los procesos deben mapearse operativamente para identificar cuellos de botella, tiempos muertos y oportunidades de automatización [3].

El mapeo de flujos operativos (Business Process Mapping) permite representar visualmente cómo circula la información y los recursos a lo largo de las distintas etapas del proceso de recolección, desde la solicitud del servicio hasta la disposición final en el relleno sanitario. Esto es fundamental para identificar tareas repetitivas, redundantes o manuales susceptibles de sistematización [3], [4].

Existen varios modelos de recolección: domiciliaria, selectiva, grandes generadores y residuos especiales. Cada modelo tiene requisitos logísticos y normativos que deben ser considerados en el diseño de un sistema de información, en este caso el enfoque del proyecto se encuentra en los últimos dos modelos mencionados.

Los sistemas de información son conjuntos de componentes interrelacionados que recopilan, procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control en una organización (Laudon & Laudon, 2012) [5]. En el contexto de la gestión de residuos, estos sistemas pueden incluir aplicaciones web, aplicaciones móviles, bases de datos y herramientas de análisis de datos.

4.1.2. Arquitecturas de software: monolíticas y microservicios

La arquitectura de microservicios se ha convertido en una alternativa robusta y escalable frente a las arquitecturas monolíticas tradicionales [6]. Un microservicio es una unidad independiente que ejecuta una función específica y se comunica con otros servicios a través de interfaces ligeras como REST APIs.

Entre las ventajas clave se encuentran: escalabilidad horizontal, despliegue independiente, tolerancia a fallos y modularidad del código [6], [7]. En este proyecto, el uso de microservicios permite aislar funcionalidades críticas como la gestión de rutas, clientes o camiones, facilitando el mantenimiento y evolución del sistema.

Tabla 2: Comparación entre arquitectura monolítica y arquitectura de microservicios

Criterio	Arquitectura de Monolítica	Arquitectura Microservicios
Despliegue	Una sola unidad, todo el sistema junto.	Cada servicio puede desplegarse de forma independiente.
Escalabilidad	Escalamiento de toda la aplicación.	Escalamiento por servicio específico.
Mantenimiento	Dificultad para mantener el sistema al crecer.	Fácil mantenimiento y evolución modular.
Fallas	Una falla puede comprometer todo el sistema.	Aislamiento de fallas, un servicio puede fallar sin afectar todo.
Tecnología	Limitada a un único stack tecnológico.	Permite usar diferentes tecnologías por servicio.
Curva de aprendizaje	Menor complejidad inicial.	Mayor complejidad en diseño y comunicación.
Comunicación entre módulos	Llamadas internas dentro del mismo proceso.	API REST, eventos, mensajería asincrónica.
Ejemplo típico	Aplicaciones CRUD tradicionales.	Plataformas SaaS, apps de gran escala y alta disponibilidad

Una vez que se elige la arquitectura de microservicios, es importante entender sus componentes principales aplicados al concepto del presente proyecto:

Tabla 3: Componentes de una arquitectura de microservicios aplicada al proyecto

Componente	Descripción
Service Registry	Registro de todos los servicios disponibles.
Gateway/API	Punto de entrada a la aplicación. Maneja rutas, autenticación, y balanceo.
Configuración central	Permite gestionar la configuración de servicios de forma centralizada.
Tabla por servicio	Cada entidad se considera una pseudo base de datos, cada microservicio gestiona sus propios datos y así evita acoplamiento.
Orquestador/Contenedor	Herramientas como Docker o Kubernetes que permiten desplegar servicios aislados.
Servicio de monitoreo	Registra logs, métricas, y errores. Ej.: Google Cloud Logging.

Esta arquitectura permite que la empresa escale su sistema sin rehacer toda la aplicación, lo que es crucial en flujos de negocio como el de Aguas de la Sabana, donde el

volumen de datos y operaciones puede aumentar con el crecimiento de la cobertura y los servicios ofrecidos [6], [7].

4.1.3. Antecedentes de aplicaciones similares

A lo largo de los últimos años, múltiples plataformas tecnológicas han sido desarrolladas para abordar los retos de la gestión de residuos sólidos urbanos, industriales y especiales. Estos sistemas han evolucionado desde simples herramientas de registro hasta soluciones completas de trazabilidad, monitoreo, optimización logística y cumplimiento regulatorio.

Uno de los ejemplos más relevantes a nivel europeo es **Ecoembes**, una organización en España que opera un sistema de reciclaje apoyado por plataformas digitales. Su solución combina bases de datos centralizadas con visualización en tiempo real de los flujos de reciclaje, optimización de rutas mediante SIG (Sistemas de Información Geográfica) y gestión de indicadores clave de sostenibilidad. Esta iniciativa ha demostrado cómo el análisis de datos y la automatización pueden mejorar significativamente la eficiencia operativa y la trazabilidad de residuos [8].

Por su parte, **SmartWaste** es una plataforma desarrollada en el Reino Unido, especializada en el monitoreo ambiental de proyectos de construcción. Su enfoque se centra en el seguimiento de residuos peligrosos, emisiones de CO2 y cumplimiento normativo en entornos industriales. Aunque su dominio es distinto al de la recolección domiciliaria o empresarial, destaca por la integración con sensores y paneles de control en línea para mejorar la toma de decisiones [9].

Estas soluciones, si bien operan en contextos regulatorios y territoriales distintos, comparten una serie de funcionalidades: digitalización de registros operativos, uso de tecnologías móviles en campo, integración de datos con visualización en tiempo real y generación de reportes administrativos o regulatorios.

El presente proyecto toma como base estos principios para adaptarlos al contexto colombiano, con énfasis en la normatividad local (CRA 151 del 2001), y en la necesidad de sistematizar procesos que aún hoy se realizan de forma manual o parcial.

A diferencia de muchas de estas plataformas, la solución propuesta no solo contempla el componente administrativo, sino también la interacción directa en campo, incluyendo funcionalidades específicas como control de aforos, incidencias técnicas, validación mecánica de vehículos y sincronización asincrónica. Esto hace que la solución sea integral y modular, y aplicable a empresas de aseo intermunicipales con cobertura heterogénea como **Aguas de la Sabana**.

Tabla 4: Análisis comparativo: Soluciones existentes y propuesta para Aguas de la Sabana

Criterio	**Ecoembes (España)**	**SmartWaste (UK)**	**Propuesta Aguas de la Sabana**
Contexto geográfico y legal	Europa – Enfoque en reciclaje y sostenibilidad	Reino Unido – Construcción y monitoreo ambiental	Colombia – Normatividad CRA y enfoque multicliente urbano/rural
Tipo de residuos gestionados	Reciclables (plásticos, papel, metales)	Construcción, peligrosos, emisiones	Ordinarios, reciclables y especiales (verdes, escombros, etc.)
Plataforma web para administración	Sí	Sí	Sí (React + Mantine)
Aplicación móvil de recolección	Limitada o indirecta	No (interfaz web adaptativa)	Sí, diseñada específicamente para recolección en campo (Kotlin)
Georreferenciación / rutas SIG	Sí	Parcial	Parcial, el uso de servicios de mapa se contempla para una segunda versión
Automatización de cotizaciones	No	No	Sí, para residuos especiales, validación y aprobación integrada
Manejo de aforos y evidencia	No	No	Sí, con firma del cliente y evidencia fotográfica
Escalabilidad / arquitectura	Centralizada, adaptable	Centralizada	Distribuida: Microservicios con Spring Boot + Docker
Cumplimiento normativo local	Adaptado a normativas europeas	Foco ambiental general	CRA 151 de 2001 + normas locales

4.2. Marco conceptual

Se presentan las definiciones y conceptos clave necesarios para entender y contextualizar el desarrollo de la solución propuesta. Se abordarán los siguientes temas: gestión de residuos, consola de administración web, aplicación móvil, optimización de procesos y cumplimiento normativo.

1. Gestion de residuos

La gestión de residuos es un conjunto de actividades que busca manejar adecuadamente los desechos sólidos desde su generación hasta su disposición final. Este proceso incluye la recolección, transporte, tratamiento y disposición de los residuos. La correcta gestión de residuos es fundamental para mantener un entorno sostenible y saludable, y requiere la implementación de sistemas eficientes que permitan su control y optimización. Según la Organización Mundial de la Salud (OMS), una gestión de residuos ineficiente puede llevar a problemas ambientales y de salud pública graves [10].

Componentes de la Gestión de Residuos:

- **Recolección:** Proceso de recogida de residuos en puntos generadores y su transporte a lugares de tratamiento o disposición.
- **Transporte:** Movimiento de los residuos desde los puntos de recolección hasta los centros de tratamiento o disposición final.
- **Tratamiento:** Proceso de transformación de los residuos para reducir su volumen o peligrosidad, incluyendo reciclaje, compostaje y tratamientos químicos.
- **Disposición Final:** Colocación definitiva de los residuos tratados en lugares diseñados para tal fin, como rellenos sanitarios.

2. Consola de Administración Web

Una consola de administración web es una aplicación accesible a través de navegadores de internet, que permite a los administradores gestionar y monitorear datos y procesos de manera centralizada. Esta herramienta es esencial para la toma de decisiones estratégicas y para la optimización de operaciones.

Funcionalidades Clave:

- **Gestión de Datos de Clientes:** Permite registrar, actualizar y consultar información relevante de los clientes, facilitando la personalización de servicios y mejorando la atención al cliente.
- **Gestión de Rutas y Micro Rutas:** Facilita la planificación, asignación y optimización de las rutas de recolección de residuos, mejorando la eficiencia operativa.

- **Gestión de Tarifas:** Permite administrar las tarifas de los servicios de recolección, asegurando el cumplimiento de las normativas vigentes y optimizando los ingresos de la empresa.
- **Generación de Reportes:** Proporciona herramientas para la creación de informes detallados sobre el rendimiento operativo, permitiendo una evaluación continua y la implementación de mejoras.

3. Aplicación Móvil

Una aplicación móvil es un software diseñado para dispositivos móviles, como smartphones y tabletas, que facilita la captura y acceso a información en el campo. En el contexto de la gestión de residuos, una aplicación móvil para los operadores de camiones de recolección es fundamental para la eficiencia y la calidad del servicio.

Funcionalidades Clave:

- **Registro de Datos en Tiempo Real:** Permite a los operadores registrar información sobre las recolecciones, incidentes y otros eventos en tiempo real, asegurando la precisión y actualidad de los datos.
- **Visualización de Rutas:** Proporciona a los operadores un mapa interactivo con las rutas y micro rutas asignadas, optimizando el tiempo y los recursos.
- **Asignación de Recursos:** Facilita la asignación y redistribución de recursos operativos como camiones y personal, en respuesta a las necesidades y condiciones del momento.
- **Reporte de Incidentes:** Permite a los operadores reportar percances o irregularidades durante la recolección, asegurando una respuesta rápida y eficiente.

4. Optimización de Procesos

La optimización de procesos es una estrategia que busca mejorar continuamente los procesos operativos para aumentar la eficiencia, reducir costos y maximizar beneficios. En la gestión de residuos, la optimización de procesos puede involucrar la automatización de tareas manuales, la mejora en la asignación de recursos y la implementación de sistemas de información avanzados.

Estrategias de Optimización:

- **Automatización de Tareas:** Reducción de la intervención manual en procesos repetitivos mediante el uso de tecnologías avanzadas, disminuyendo el riesgo de errores y reduciendo costos operativos.
- **Mejora Continua:** Implementación de metodologías como Kaizen o Six Sigma para identificar y eliminar ineficiencias, mejorando constantemente los procesos.

- **Tecnologías de Información:** Uso de sistemas de gestión de información (MIS), bases de datos, y herramientas de análisis de datos para apoyar la toma de decisiones informadas y estratégicas.

5. Cumplimiento Normativo

El cumplimiento normativo se refiere a la adhesión a las leyes, regulaciones y estándares establecidos por organismos reguladores. Para la empresa Aguas de la Sabana, es crucial cumplir con las normativas establecidas por la Comisión de Regulación de Agua Potable y Saneamiento Básico (CRA) [2] y otras entidades reguladoras, asegurando que todas las operaciones se realicen de manera legal y ética.

Principales Normativas:

- **CRA 151 del 2001 Anexo 8:** Normativa que define las tarifas del servicio de recolección de residuos y establece estándares de operación y gestión.
- **Protección de Datos:** Legislación que regula el manejo y protección de la información personal de los clientes, asegurando su privacidad y seguridad.

4.2.1. Estado del arte

En los últimos años, la gestión de residuos ha visto un avance significativo gracias a la implementación de soluciones tecnológicas por parte de diversas empresas y organizaciones. Estas tecnologías buscan mejorar la eficiencia y efectividad en la recolección y tratamiento de residuos. Un ejemplo notable es el desarrollo de aplicaciones móviles que permiten a los operadores de camiones de recolección registrar datos en tiempo real, optimizando así las rutas de recogida y reduciendo costos operativos.

Además, el uso de Sistemas de Información Geográfica (SIG) ha facilitado el monitoreo y la planificación de la recolección de residuos en diferentes zonas urbanas. Por ejemplo, en ciertas ciudades, se han implementado SIG para mapear y analizar patrones de generación de residuos, lo que ha permitido a los administradores urbanos planificar mejor las rutas de recolección y asignar recursos de manera más eficiente [11].

Estos avances tecnológicos no solo mejoran la eficiencia operativa, sino que también contribuyen a la sostenibilidad ambiental al reducir las emisiones de carbono asociadas con la recolección de residuos. La integración de tecnologías avanzadas en la gestión de residuos demuestra cómo la innovación puede transformar sectores tradicionales y contribuir a la creación de ciudades más inteligentes y sostenibles [11].

5. Objetivos

5.1. Objetivo general

Implementar una solución tecnológica que permita automatizar cortas tareas repetitivas y sistematizar los procesos de gestión de rutas, micro rutas, clientes y recursos operacionales, cumpliendo con las regulaciones vigentes y facilitando la toma de decisiones estratégicas.

5.2. Objetivos específicos

1. Analizar los procesos actuales de gestión de residuos para identificar áreas de mejora y oportunidades de automatización.

2. Diseñar una arquitectura escalable y segura que cumpla con los requisitos de privacidad y protección de datos.

3. Desarrollar una consola de administración web que permita a los administradores gestionar de manera eficiente los datos de clientes, rutas, micro rutas, tarifas y demás información relevante.

4. Implementar una aplicación móvil para operadores de camiones de recolección que facilite el registro de datos en tiempo real, la visualización de rutas y la asignación óptima de recursos con posibilidad de mejoras modulares dentro de la misma según necesidad de la empresa.

5. Capacitar al personal de la empresa en el uso de la nueva solución para garantizar una adopción efectiva.

Tabla 5: Resultados esperados por objetivo específico

Objetivo específico	**Resultados esperados**	**Evidencia del objetivo**
1. Analizar los procesos actuales de gestión de residuos para identificar áreas de mejora y oportunidades de automatización.	Informe detallado que documente las áreas de mejora identificadas y las oportunidades de automatización, con recomendaciones específicas para cada proceso.	Sección 6: Análisis de procesos previos al sistema
2. Diseñar una arquitectura escalable y segura que cumpla con los requisitos de privacidad y protección de datos.	Documento de diseño de arquitectura que incluya diagramas, especificaciones técnicas y un plan de implementación.	Sección 7: Arquitectura global del proyecto
3. Desarrollar una consola de administración web que permita a los administradores gestionar de manera eficiente los datos de clientes, rutas, micro rutas, tarifas y demás información relevante.	Consola de administración web funcional y accesible, con una interfaz de usuario intuitiva y capacidades de gestión completas con una muestra de código según permisos. Manual de usuario y guía de administración para la consola, incluyendo tutoriales y mejores prácticas.	Sección 9.2–9.6: Implementación de prototipos (Aplicación Web)
4. Implementar una aplicación móvil para operadores de camiones de recolección que facilite el registro de datos en tiempo real, la visualización de rutas y la asignación óptima de recursos con posibilidad de mejoras modulares dentro de la misma según necesidad de la empresa.	Aplicación móvil operativa en dispositivos tipo tablet para los operadores, con funcionalidades clave implementadas y en funcionamiento con una muestra de código fuente según permisos. Resultados de pruebas de usuario realizadas con los operadores para asegurar la usabilidad y efectividad de la aplicación, con feedback documentado y aplicado.	Sección 6.5 y 9.7: Historias de usuario y App móvil
5. Capacitar al personal de la empresa en el uso de la nueva solución para garantizar una adopción efectiva.	Programas de capacitación realizados y reproducibles con documentación detallada como manual guía, con evidencia de participación y satisfacción en el aprendizaje del personal.	Sección 11: Capacitaciones

6. Análisis de procesos previos al sistema

Correspondiente al desarrollo del primer y segundo objetivo del proyecto de Aguas de la Sabana.

6.1. Investigación según lógica de negocio

Para identificar las áreas de mejora y las oportunidades de automatización en los procesos de gestión de residuos de la empresa Aguas de la Sabana, se llevó a cabo un análisis exhaustivo mediante reuniones y encuestas con el cliente. Estas sesiones permitieron recoger información directa sobre los desafíos operativos, las ineficiencias, y las necesidades de los usuarios del sistema.

El análisis se estructuró en torno a cinco casos representativos, cada uno asociado a diferentes aspectos de la gestión de residuos. Las historias de usuario correspondientes a estos casos fueron desarrolladas a partir de las actividades propuestas y ya mencionadas.

6.2. Resultados del análisis

Como resultado de este proceso, se identificaron cuatro casos principales, que agrupan las áreas críticas en las que se requiere intervención. Cada caso fue detallado en unas historias de usuario **(H.U)**, que describen las funcionalidades necesarias para mejorar la eficiencia y la automatización de los procesos, así mismo dando cumplimiento a los objetivos del proyecto.

Para cada una se tiene una descripción general de la funcionalidad, un resultado esperado, los criterios de aceptación inferidos por el cliente, y unas **"dev tasks"** correspondientes a unas anotaciones relevantes realizadas por el equipo de desarrollo.

6.3. CASO 1: Vinculación de un cliente.

6.3.1. H.U-1:

- **Funcionalidad:** Registro y validación de un cliente ante la dirección comercial de la empresa Y el tipo de medición por **AFORO** o por **PESO**.
- **Resultado:** Determinar si el usuario cumple con los parámetros establecidos para una vinculación.
- **Criterios de aceptación:**

  1. El líder del proceso deberá registrar el cliente que solicita el servicio
  2. El líder del proceso deberá validar que el cliente que solicita el servicio , ya lo había solicitado previamente y actualmente se le está atendiendo.
  3. El administrador deberá validar la documentación requerida y solicitada.

4. Si el cliente cumple con la documentación requerida, el líder del proceso deberá ingresar los siguientes datos; datos personales, datos de contacto, información adicional, datos de la empresa (si aplica).
5. Con los datos del cliente registrados y validados correctamente, el cliente estará disponible para la vinculación en el sistema comercial, lo que permitirá continuar con la visita de verificación de cumplimiento de condiciones técnicas.
6. El sistema deberá permitir elegir el tipo de medición de cada cliente por medio de una lista de comprobación (Check-List).
7. El líder del proceso deberá asignar un técnico para que ejecute la visita donde se corrobore las condiciones de entrega de los residuos.
8. El líder del proceso deberá validar si el cliente que solicita el servicio ya cuenta con el servicio de acueducto que presta la empresa para determinar el tipo de facturación a realizar (Conjunta o Directa).

- **Dev tasks:**

1. Diseñar y desarrollar formularios de registro claros y fáciles de usar que recojan todos los datos necesarios del cliente. mensajes de error claros y descriptivos para informar al usuario sobre cualquier problema encontrado durante el proceso de registro y validación.
2. Enviar notificaciones de éxito al usuario cuando el proceso se complete satisfactoriamente.
3. Registrar eventos importantes durante el proceso de registro y validación, como la fecha y hora de registro, el usuario responsable y cualquier error encontrado durante la validación.
4. Implementar lógica de validación de los campos en el backend: Realizar consultas a la base de datos para verificar si el nombre de usuario proporcionado por el recolector existe en la base de datos como también verificar las contraseñas asignadas.

6.3.2. H.U-2:

- **Funcionalidad:** Desvinculación del cliente a una ruta y micro -ruta
- **Resultado:** Se espera que el cliente se encuentre a paz y salvo con la empresa, previo a la desvinculación.
- **Criterios de aceptación:**

1. El sistema deberá validar que el cliente esté a paz y salvo en conceptos monetarios con la empresa, a través de estados.
2. El sistema deberá registrar la documentación básica para la desvinculación de la empresa.

- **Dev tasks:**

    1. La desvinculación del cliente se realiza de manera efectiva y se actualizan correctamente los registros en la base de datos del sistema.
    2. Se manejan adecuadamente las excepciones y situaciones de error, proporcionando mensajes claros y útiles al usuario.
    3. La funcionalidad de desvinculación del cliente de una ruta y micro-ruta funciona sin problemas y cumple con los requisitos del sistema

6.4. CASO 2: Programación de rutas de recolección.

6.4.1. H.U-1:

- **Funcionalidad:** Gestión de una ruta y su respectiva micro-ruta a un equipo de trabajo (vehículo, conductor y operarios).
- **Resultado:** Se espera que se realice el agendamiento correctamente de la ruta y la micro-ruta.
- **Criterios de aceptación:**

    1. El líder del proceso deberá asignar micro-rutas a equipos de trabajo de manera eficiente, considerando la disponibilidad de vehículos, conductores y operarios.
    2. El sistema de agendamiento deberá permitir adicionar a las micro-rutas las solicitudes de recolección de residuos especiales que previamente hayan sido requeridas por los clientes, considerando que el tipo de residuo especial pueda ser transportado bajo las mismas condiciones del residuo ordinario.
    3. El sistema deberá permitir monitorear las incidencias reportadas por los técnicos en las frecuencias de recolección.
    4. El sistema permitirá verificar si la ruta asignada se encuentra en ejecución o finalizada.
    5. El sistema permitirá que se puedan editar tanto rutas como micro-ruta.
    6. El sistema permitirá realizar la asignación de la micro-ruta sobre el conductor.
    7. El sistema permitirá que una vez organizada la agenda de recorrido (micro-ruta asignadas a vehículos) se genere el proceso que permite mostrar las rutas en los dispositivos móviles para cada técnico.

- **Dev tasks:**

    1. El sistema permite planificar y asignar rutas de recolección de manera eficiente, optimizando la utilización de recursos y minimizando los tiempos de viaje.
    2. La interfaz de usuario es intuitiva y fácil de usar, permitiendo a los usuarios gestionar las rutas de recolección de manera rápida y efectiva.

3. El sistema se integra correctamente con otras tecnologías existentes, mejorando la precisión y la eficiencia en la gestión de rutas.
4. El sistema es estable, confiable y escalable, funcionando sin problemas en un entorno de producción.

6.4.2. H.U-2:

- **Funcionalidad:** Reprogramar la micro-ruta a los clientes que, por condiciones extraordinarias, no se pudieron atender en su ruta original.
- **Resultado:** Se espera que el cliente ya se encuentre en una ruta y micro-ruta asignada para poder ser reprogramado en caso de que no se puedan atender.
- **Criterios de aceptación:**

  1. El administrador podrá editar la micro-ruta reprogramada.
  2. El sistema deberá permitir registrar la causal de no recolección de cada cliente que generó la reprogramación.
  3. El sistema deberá permitir verificar la capacidad de almacenamiento del vehículo.
  4. El líder del proceso deberá seleccionar la causal de recolección de la siguiente manera:
     - **code:** 00001 **msj:** Recolección sin inconveniente.
     - **code: 00002 msj:** Cerrado.
     - **code: 00003 msj:** No permite la recolección.
     - **code: 00004 msj:**Imposibilidad acceso.
     - **code: 00005 msj:** Incidente técnico.

- **Dev tasks:**

  1. Se deben respetar las restricciones y limitaciones establecidas, como la capacidad de los vehículos y la disponibilidad del personal, al asignar clientes reprogramados a rutas alternativas.
  2. El sistema debe permitir re-programar clientes en diferentes momentos del día o en días alternativos, para adaptarse a las necesidades cambiantes de los clientes y a las condiciones operativas.
  3. La funcionalidad se integra correctamente con el sistema de gestión de rutas existente, garantizando una coordinación efectiva entre las asignaciones regulares y las re-programaciones.

6.5. CASO 3: Proceso de recolección en terreno (App).

6.5.1. H.U-1:

- **Funcionalidad:** El recolector ingresa la APP y proporcional credenciales validas para poder estar activo en el sistema.
- **Resultado:** Se espera un login exitoso con las credenciales proporcionadas.
- **Criterios de aceptación:**
  1. El usuario podrá iniciar sesión correctamente utilizando usuario y contraseña válida.
  2. En caso de que las credenciales sean incorrectas el usuario podrá visualizar mensajes claros donde se muestre "Datos incorrectos".
  3. El usuario podrá recuperar clave de usuario mediante una opción de recuperación clave, dicha opción será enviada al correo del técnico registrado.
  4. El usuario podrá visualizar el panel principal para iniciar su recorrido.
- **Dev tasks:**
  1. Implementar lógica de validación de los campos en el backend: Realizar consultas a la base de datos para verificar si el nombre de usuario proporcionado por el recolector existe en la base de datos como también verificar las contraseñas asignadas.
  2. Diseñar la Interfaz de Usuario (UI): Crear campos de texto de inicio de sesión en la interfaz de usuario donde los recolectores puedan ingresar su nombre de usuario y contraseña.
  3. Manejar Errores y Excepciones: Manejar posibles errores y excepciones que puedan ocurrir durante el proceso de inicio de sesión, como credenciales incorrectas, nombre de usuario no encontrado, errores de conexión a la base de datos, etc.
  4. Pruebas y Depuración: Realizar pruebas exhaustivas para asegurarse de que el proceso de inicio de sesión funcione correctamente en diferentes escenarios y condiciones.

6.5.2. H.U-2:

- **Funcionalidad:** Realizar el proceso de validación de condiciones técnicas del vehículo antes de iniciar el recorrido y registrar datos de la tripulación.
- **Resultado:** El recolector deberá registrar el estado del vehículo con todos los criterios definidos por el sistema.
- **Criterios de aceptación:**

1. El vehículo debe tener en buen estado las llantas al igual que su presión del aire.
2. El vehículo debe tener suficiente combustible para completar el recorrido planificado, teniendo en cuenta la distancia y las condiciones del camino.
3. Todas las luces del vehículo, incluyendo faros, luces traseras, luces de freno, luces intermitentes y luces de marcha atrás, deben funcionar correctamente.
4. El vehículo debe de tener en funcionamiento la bocina.
5. El nivel de líquido de frenos debe estar dentro del rango óptimo, y el líquido mismo debe estar libre de contaminantes o signos de deterioro al igual que su funcionamiento.
6. Verificar la vigencia de los documentos del vehículo.
7. El técnico podrá generar observaciones adicionales que no estén dentro de los criterios establecidos.
8. El técnico podrá tener la opción de cambiar de vehículo en caso de que lo requiera ya sea por fallas técnicas u otros criterios que lo ameriten.
9. El técnico podrá ingresar una nueva placa del vehículo al momento del cambio si este lo requiere, se abrirá nuevamente la interfaz de verificación técnica.
10. El técnico deberá ingresar el nombre de los tripulantes con su número de cédula mediante un botón con el llamado a dicha función.

- **Dev tasks:**

1. Diseño de la Interfaz de Usuario (UI): Desarrollar una interfaz de usuario intuitiva y fácil de usar que permita al usuario ingresar y visualizar los datos de manera clara y precisa.
2. Validación de Entradas: Implementar mecanismos de validación de entradas para garantizar que los datos ingresados por el usuario sean correctos y estén dentro de los rangos aceptables.
3. Generación de Informes y Alertas: Desarrollar funcionalidades para generar informes detallados sobre el estado del vehículo y alertas en caso de que se detecten problemas o condiciones fuera de los límites aceptables.

6.5.3. H.U-3:

- **Funcionalidad:** Tomar los aforos con sus respectivos soportes.
- **Resultado:** Los aforos de los clientes deben de quedar registrados en la ruta asignada.
- **Criterios de aceptación:**

1. Los aforos deben ser tomados de manera clara y precisa, sin ambigüedades ni errores significativos en la medición.

2. Los aforos deben ser consistentes con las normas y procedimientos establecidos para la toma de aforos de la empresa Aguas de la Sabana.
3. Los aforos deben reflejar con precisión la cantidad de bolsas presentes en el recipiente en el momento de la medición, evitando subestimar o sobreestimar la cifra real.
4. Al momento de terminar el aforo de un cliente se debe de generar una alerta que indique que el cliente ya está atendido.
5. La sección del aforo deberá mostrar los detalles del aforo y deberá permitir eliminar detalles que no estén contemplados en la información técnica del cliente.
6. La sección de recolección de aforo deberá mostrar un checklist especificando si se realizó el aforo.
7. La sección aforo deberá mostrar la cantidad recogida durante el recorrido tanto en metros cúbicos como en toneladas.
8. La sección aforo deberá permitir agregar un nuevo recipiente en caso de que el cliente lo requiera.
9. La sección aforo deberá permitir tener la opción para recolectar la evidencia fotográfica del aforo del cliente.
10. La sección aforo deberá permitir adjuntar la firma del cliente al momento de recoger su aforo.

- **Dev tasks:**

1. Recopilación de Evidencia Fotográfica: Desarrollar la funcionalidad que permita a los operadores capturar fotografías de los lugares donde se realiza la recogida del aforo, integrando la cámara del dispositivo móvil o una cámara externa si es necesario.
2. Obtención de la Firma del Cliente: Implementar una función que permita al cliente firmar electrónicamente para confirmar la recogida del servicio. Esto podría incluir la integración de un sistema de firma digital.
3. Gestión de Servicios (Guardar y Cancelar): Desarrollar la lógica de negocio para permitir a los operadores guardar los detalles del servicio completado y cancelar servicios si es necesario, actualizando la base de datos correspondiente.
4. Integración con la Base de Datos: Integrar la aplicación con una base de datos para almacenar de forma segura la información recopilada, incluyendo las fotografías, las firmas electrónicas y los detalles de los servicios.

6.5.4. H.U-4:

- **Funcionalidad:** Iniciar el proceso de recorrido.
- **Resultado:** Se espera que la ruta previamente asignada al recolector esté lista para ser iniciada.

- **Criterios de aceptación:**

  1. El usuario podrá visualizar el estado del recorrido, la ruta a la que esté asignada, los avances del recorrido.
  2. Después de que el usuario ingrese los datos solicitados se inicia el recorrido y a su vez este deberá mostrar un mensaje "has iniciado tu recorrido".
  3. El Usuario podrá registrar detalles como la fecha y la hora exacta, y cualquier otro dato relevante para el seguimiento y la supervisión.
  4. Los clientes visitados y a los cuales se les registró el aforo se podrán visualizar con un checklist para reflejar si ya fueron atendidos.
  5. El Usuario debe visualizar la información de la ruta en tiempo real, reflejando cualquier cambio o actualización realizada por el administrador o cualquier otro usuario autorizado mientras la ruta está en curso.
  6. El sistema deberá permitir validar la capacidad actual del vehículo dependiendo los aforos recogidos.

- **Dev tasks:**

  1. Implementación del formulario de inicio de recorrido:
     - El sistema debe permitir al usuario realizar un seguimiento del progreso mientras realiza la ruta, mostrando información como la distancia recorrida, el tiempo transcurrido y los hitos alcanzados.
     - Desarrollar el formulario de inicio de recorrido en la interfaz de usuario, utilizando las herramientas y componentes de desarrollo adecuados para la plataforma.
     - Establecer la lógica para capturar y validar los datos ingresados por el usuario, asegurando que se proporcionen todos los campos requeridos y que los datos sean del tipo correcto.
  2. Almacenamiento de información en la base de datos:
     - Desarrollar la lógica en el backend para almacenar la información del recorrido en la base de datos de manera segura.
     - Utilizar consultas para interactuar con la base de datos y realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los datos del recorrido.
  3. Manejo de errores y excepciones:
     - Configurar el sistema para manejar adecuadamente los errores y excepciones que puedan ocurrir durante el proceso de inicio de recorrido, como errores de validación de datos o problemas de conexión con el servidor.
     - Proporcionar mensajes de error claros y descriptivos para informar al usuario sobre cualquier problema que surja y orientarlo sobre cómo proceder.

6.5.5. H.U-5:

- **Funcionalidad:** Se requiere que el técnico registre un motivo o una nota para justificar la pausa de la ruta. Esto puede ser útil para mantener un registro de eventos y proporcionar contexto sobre la interrupción en la ruta.
- **Resultado:** Se espera que el técnico tenga un motivo prioritario para pausar su ruta.
- **Criterios de aceptación:**
  1. El usuario podrá registrar su motivo por el cual pausa la ruta.
  2. El usuario tendrá visible un botón donde podrá pausar su ruta.
  3. El usuario entre las opciones podrá pausar su ruta por ÏNCIDENCIAçorrespondiente a eventos específicos, como repostaje.
- **Dev tasks:**
  1. Diseñar una interfaz clara y accesible que permita a los conductores o usuarios pausar una ruta de manera intuitiva.
  2. Incluir un botón o elemento visual prominente en la interfaz de usuario que indique claramente la opción para pausar la ruta.

6.5.6. H.U-6:

- **Funcionalidad:** Registro de repostajes.
- **Resultado:** Se espera que se puedan registrar datos relevantes del vehículo respecto al repostaje de combustible.
- **Criterios de aceptación:**
  1. El usuario deberá tener la potestad de pausar su ruta para realizar el repostaje del vehículo.
  2. El usuario podrá registrar un nuevo repostaje con los siguientes datos: lugar, cantidad de galones y valor.
  3. El usuario podrá visualizar y editar la lista de reportes de reportajes creados.
  4. El usuario podrá tomar evidencia fotográfica de la factura del repostaje y podrá visualizarla sobre el repostaje creado.
  5. El usuario podrá eliminar la foto del repostaje, también podrá subir varias fotos.
  6. Los usuarios pueden acceder fácilmente a la funcionalidad de registro de repostaje desde la interfaz del sistema de gestión de flotas.
  7. El usuario podrá visualizar las notificaciones automáticas sobre los repostajes realizados y se registra un historial completo de repostajes en el sistema.

8. El usuario podrá finalizar o cancelar el repostaje.

- **Dev tasks:**

1. Diseñar una interfaz intuitiva que permita a los usuarios ingresar los detalles del repostaje, como la cantidad de combustible repostado, el precio por litro, la fecha y la ubicación de la estación de servicio.
2. Incluir campos de validación para garantizar que se ingresen datos válidos y requeridos antes de enviar el formulario de registro.

6.5.7. H.U-7:

- **Funcionalidad:** Registro de incidencias.
- **Resultado:**

  - Se espera que el sistema muestre una confirmación al usuario de que la incidencia se ha registrado correctamente.
  - Los responsables designados revisan la incidencia, investigan las causas subyacentes y toman las medidas apropiadas para resolverla. Esto puede implicar la programación de reparaciones, la comunicación con partes interesadas relevantes o la implementación de medidas preventivas para evitar incidentes similares en el futuro.
  - Si es necesario, se comunica con el usuario que reportó la incidencia para obtener más detalles o proporcionar actualizaciones sobre el progreso de la resolución.

- **Criterios de aceptación:**

1. El usuario deberá tener la potestad registrar una incidencia con los siguientes datos: tipo de incidencia, lugar de incidencia, descripción de la incidencia.
2. El usuario podrá validar si el tipo de incidencia creada amerita el abandono de la ruta.
3. El usuario podrá registrar una fotografía de la incidencia presentada.
4. El usuario podrá visualizar los tipos de incidencias creados.
5. Los usuarios pueden acceder fácilmente a la funcionalidad de registro de incidencias desde la interfaz del sistema de gestión.
6. Las incidencias registradas se almacenan correctamente en la base de datos y están asociadas con el vehículo correspondiente.
7. La integración con el sistema de seguimiento de vehículos se realiza de manera efectiva, actualizando el estado del vehículo para reflejar la incidencia registrada.
8. El usuario visualiza las notificaciones automáticas sobre las incidencias registradas y se proporciona una interfaz de gestión de incidencias para su seguimiento y resolución.

9. El usuario podrá editar y finalizar una incidencia.

- **Dev tasks:**

1. Diseñar una interfaz intuitiva que permita a los usuarios ingresar los detalles de la incidencia, como la descripción del problema, la fecha y hora de ocurrencia, y la ubicación del incidente.
2. Configurar notificaciones automáticas para informar a los usuarios relevantes (como administradores o supervisores) sobre las incidencias registradas.
3. Implementar una interfaz de gestión de incidencias que permita a los usuarios revisar, editar y resolver las incidencias registradas, así como agregar comentarios adicionales o adjuntar archivos relevantes.

6.5.8. H.U-8:

- **Funcionalidad:** Finalizar ruta.
- **Resultado:**
  - Se espera que el sistema muestre una confirmación al conductor o al usuario de que la ruta se ha finalizado correctamente.
  - Se espera que el estado del vehículo en el sistema se actualice para reflejar que la ruta ha sido finalizada.
  - Una vez finalizada la ruta, el vehículo está disponible para ser asignado a nuevas tareas o rutas dentro del sistema de gestión.
- **Criterios de aceptación:**

1. El usuario podrá visualizar la notificación de confirmación de finalización de la ruta.
2. El usuarios puede acceder fácilmente a la funcionalidad de finalización de ruta desde la interfaz del sistema de gestión.
3. El usuario al finalizar la ruta lo llevará a una notificación de "si/no" desea iniciar el viaje al relleno sanitario.

- **Dev tasks:**

1. El sistema debe registrar la hora exacta en que se finalizó la ruta para futuras referencias y seguimiento de eventos.
2. Se debe registrar la distancia total recorrida durante la ruta para mantener un registro preciso de la actividad del vehículo.
3. Diseñar una interfaz clara y fácil de usar que permita a los conductores o usuarios finalizar una ruta.

4. Incluir elementos visuales que indiquen claramente la opción para finalizar la ruta y proporcionar retroalimentación visual una vez que la acción se haya completado.
5. Registrar la hora exacta de finalización de la ruta junto con cualquier otra información relevante, como la distancia recorrida y el tiempo total empleado.

6.5.9. H.U-9:

- **Funcionalidad:** Inicio de viaje al relleno sanitario.
- **Resultado:** El estado del recorrido de recolección en el sistema debe ser actualizado a "finalizado", indicando que la fase de recolección ha concluido satisfactoriamente.
- **Criterios de aceptación:**
  1. El usuario podrá visualizar el cambio de estado de ïniciado.ª "finalizado".
  2. El usuario deberá agregar el kilometraje inicial, es decir, con el que se encuentra al momento del inicio del viaje al relleno sanitario.
  3. El usuario deberá ingresar la cantidad de peajes que cruzó.
  4. El usuario cuando llega al sitio de relleno deberá dar clic en el botón .ªCEPTAR", para visualizar los kilometrajes recorridos y cantidad de peajes para iniciar con su disposición final.
- **Dev tasks:**
  1. El sistema debe tener registrada la ubicación del relleno al que se dirigirá el vehículo para depositar los residuos recolectados.
  2. El sistema debe confirmar al usuario que el recorrido de recolección ha sido finalizado con éxito y que el viaje al relleno ha sido iniciado.
  3. El sistema debe registrar la hora exacta en que se finalizó el recorrido de recolección y se inició el viaje al relleno.

6.6. CASO 4: Proceso de cotización de residuos especiales

6.6.1. H.U-1:

- **Funcionalidad:** Solicitud del servicio especial donde se especifique el tipo de residuo especial y se adjunte la foto.
- **Resultado:** Se espera la cotización emitida por la empresa y la respectiva aprobación por parte del cliente.
- **Criterios de aceptación:**

1. El líder del proceso debe validar si el cliente que solicita el servicio especial ya se encuentra vinculado como suscriptor de la empresa (cliente vinculado), en caso de que no esté vinculado se debe realizar el proceso de vinculación (Caso 1) para poder realizar la cotización.
2. El administrador deberá registrar la información del servicio especial con los siguientes datos: código identificación del cliente (contrato), tipo de residuo especial (escombro, verdes, peligrosos), volumen estimado (fotografía enviada por el cliente) y fecha pactada de recolección.
3. El administrador deberá definir las tarifas por cada tipo de residuo especial.
4. El administrador podrá generar una cotización con la información suministrada del servicio especial y según las tarifas definidas.
5. El administrador podrá registrar la aceptación de la cotización por parte del cliente.
6. El sistema debe permitir manejar los siguientes estados para las cotizaciones: entregada al cliente, aprobada por el cliente y trabajo ejecutado.

- **Dev tasks:**

1. Deben existir políticas y procedimientos claros y establecidos para el manejo de las solicitudes de servicio especial, incluyendo criterios de priorización, plazos de respuesta y protocolos de comunicación con los usuarios.
2. La funcionalidad permite adjuntar fotos u otros archivos relevantes a las solicitudes de servicio especial.
3. La interfaz de usuario mejorada proporciona una vista clara y detallada del estado de las solicitudes de servicio especial.

6.6.2. H.U-2:

- **Funcionalidad:** Agendamiento en la recolección del residuo especial con el cliente para luego incluirlo en la ruta.
- **Resultado:** Se espera que los datos del cliente estén registrados correctamente.
- **Criterios de aceptación:**

1. El administrador podrá definir la frecuencia de la recolección de los residuos especiales según la condición pactada con el cliente.
2. El administrador podrá agendar la recolección de residuos especial en la micro-ruta que cumpla con las condiciones pactadas por el cliente.

- **Dev tasks:**

1. Se envía automáticamente una confirmación al cliente una vez que se ha programado la recolección del residuo especial.

2. La recolección agendada del residuo especial se incluye automáticamente en las rutas de recolección correspondientes.

7. Arquitectura global del proyecto: Oracle, SQL Developer y microservicios

7.1. Esquema de proyecto

Este diagrama representa la arquitectura general del sistema, que consta de un front-end, un backend basado en microservicios y una base de datos.

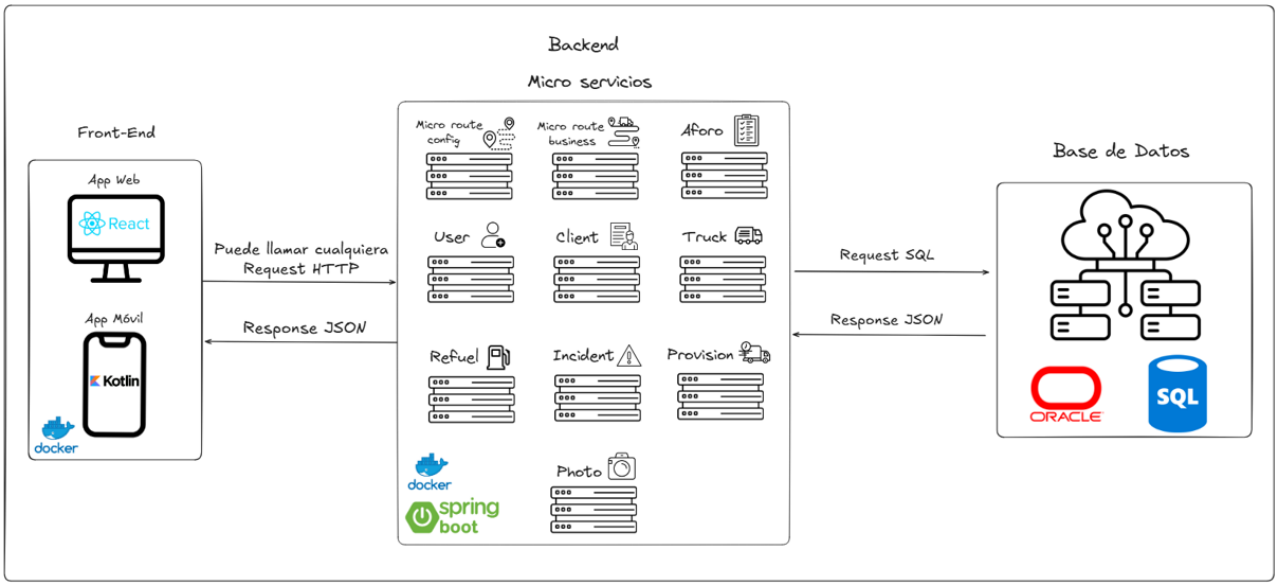


Figura 1: Esquema general del proyecto

7.2. Frontend

En el Front-End, se desarrollaron dos aplicaciones: una app web para administración interna usando React y una app móvil para los operadores de recolección en Kotlin. Ambas aplicaciones interactúan con el backend mediante solicitudes HTTP, enviando y recibiendo datos en formato JSON. El despliegue de estas aplicaciones se realiza con **Docker**, asegurando consistencia y facilidad de mantenimiento.

7.2.1. Aquitectura de diseño: App web

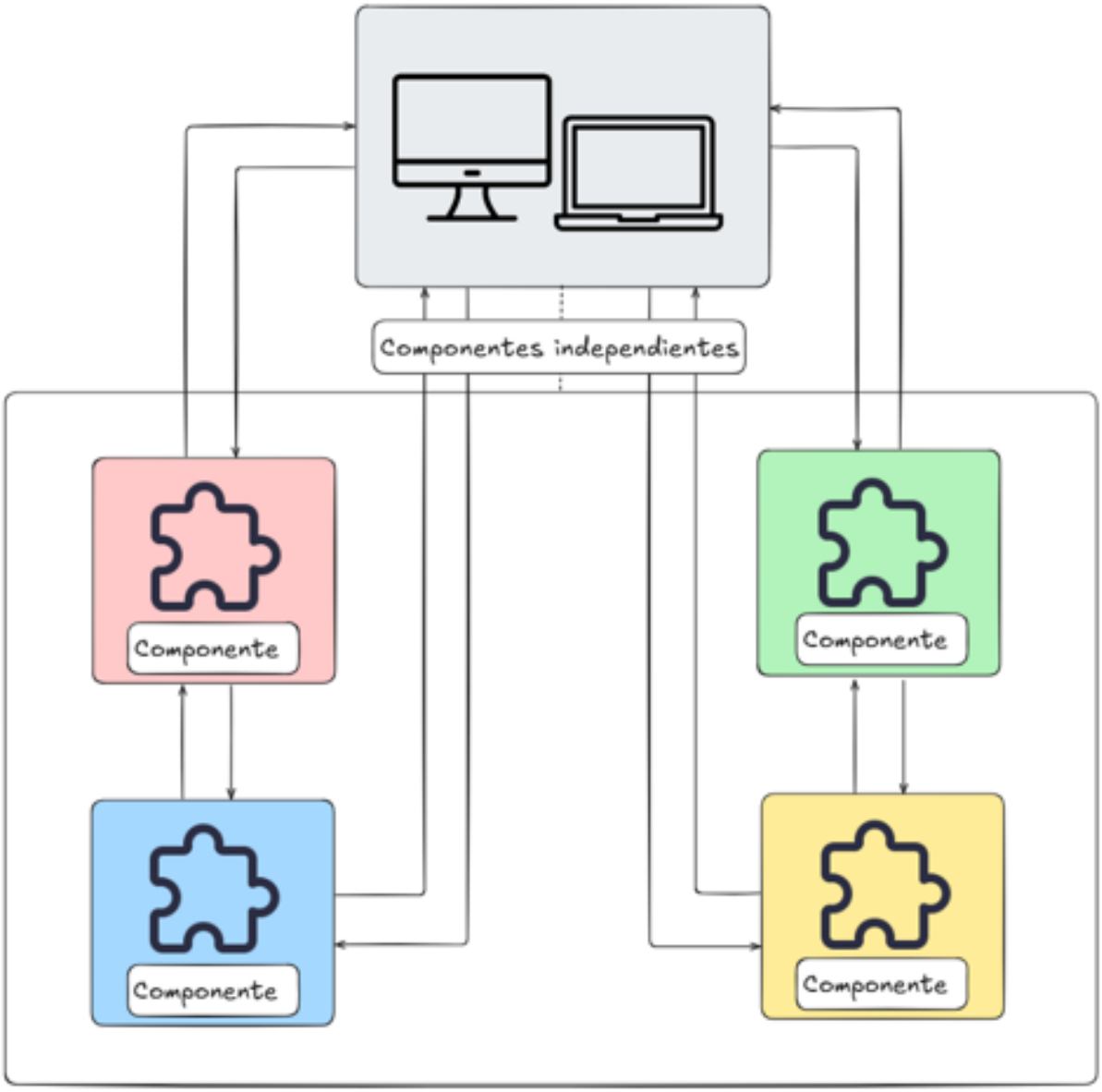


Figura 2: Arquitectura de app web

El diagrama muestra una arquitectura de aplicación web en **React**, como se mencionó anteriormente, estructurada con una **arquitectura basada en componentes**

- **Interfaz de usuario:** Representa la capa visible para el usuario, donde interactúa con la aplicación.
- **Componentes modulares:** Cada bloque de color es un componente React que cumple una función específica (por ejemplo, formularios, navbar o las tablas). Son independientes y reutilizables, facilitando el mantenimiento y la escalabilidad.

- **Comunicación:** Las flechas muestran la interacción entre la interfaz y los componentes, que comparten datos mediante props o un estado global, una característica particular de React.

7.2.2. Aquitectura de diseño: App móvil

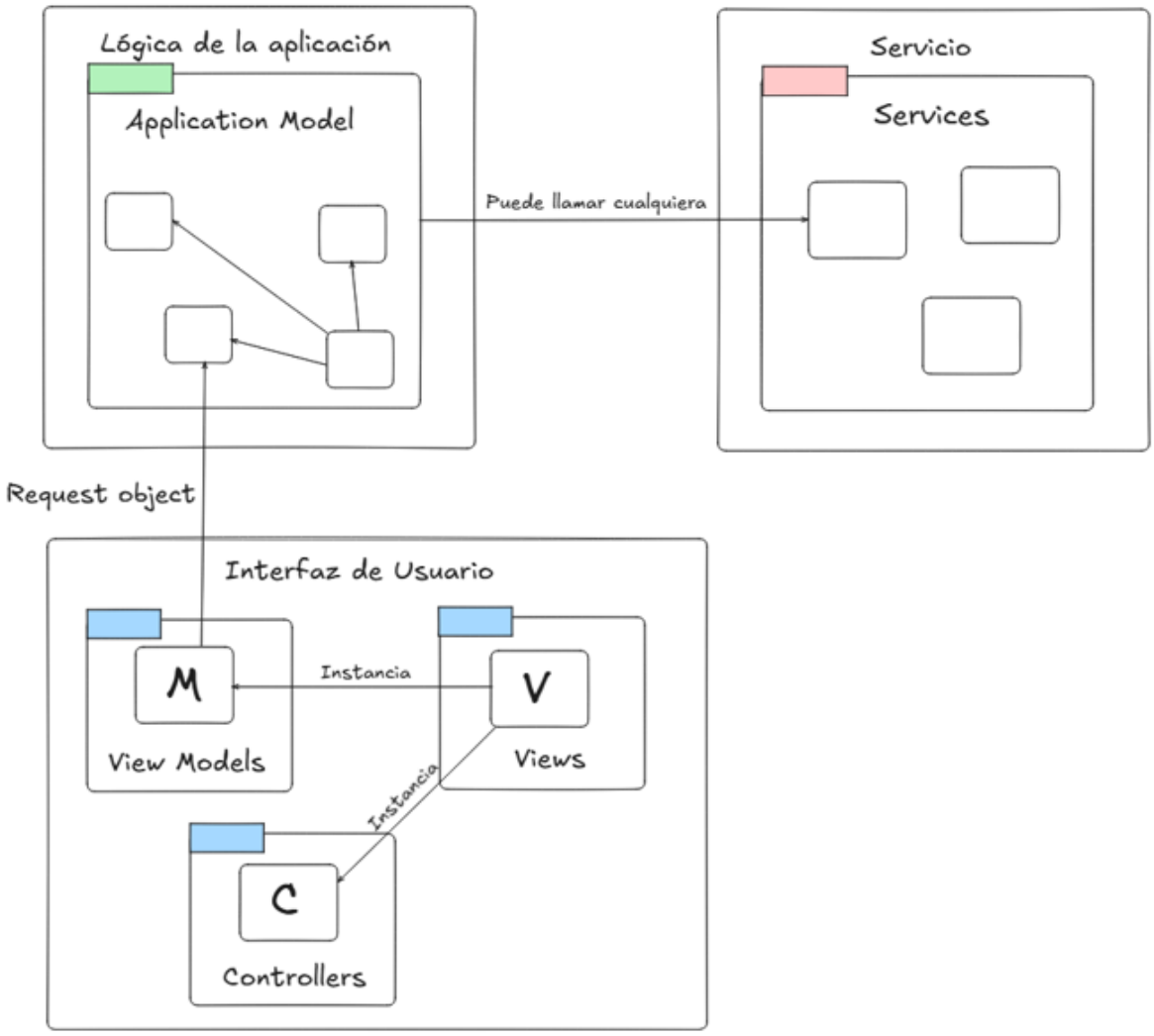


Figura 3: Arquitectura de app móvil

La arquitectura de diseño de la app móvil que se ha implementado sigue el patrón **MVVC** (**Model-View-ViewModel-Controller**), el cual organiza los componentes en capas claramente definidas para mejorar la separación de responsabilidades y la mantenibilidad del código.

- **Lógica de la Aplicación:**
  - **Application Model:** Representa la lógica y los datos de la aplicación. En esta capa se gestionan los modelos de datos y las relaciones entre ellos. Aquí es donde se centraliza la lógica que no depende de la interfaz de usuario y se manejan las operaciones necesarias para procesar los datos que la aplicación necesita.

- **Servicio:**
    - **Services:** Esta sección es donde se centralizan los servicios que interactúan con recursos externos, como las llamadas a la API o la gestión de datos remotos. La capa de servicios se conecta a los modelos de datos y proporciona a la lógica de la aplicación la información necesaria para que funcione correctamente. El Application Model puede invocar cualquier servicio necesario, como una llamada a una API o una base de datos externa.
- **Interfaz de Usuario:**
    - **View Models (M):** Contiene los modelos de vista, que funcionan como intermediarios entre los modelos de la aplicación y las vistas. El ViewModel transforma y adapta los datos del modelo para que se presenten de manera adecuada en la interfaz de usuario. Esta capa permite desacoplar la lógica de la interfaz del control directo de los datos, lo cual facilita la prueba y reutilización de componentes.
    - **Views (V):** Representa la interfaz gráfica con la que interactúa el usuario. La vista se encarga únicamente de mostrar la información y notificar al controlador (Controller) sobre las interacciones del usuario. En este diseño, la vista se actualiza automáticamente cuando el ViewModel cambia, gracias al uso de observables o Live Data.
    - **Controllers (C):** Maneja la lógica de navegación e interacción de la interfaz. Es responsable de conectar las interacciones del usuario (como toques o gestos) con las acciones que deben realizarse, delegando al ViewModel o a otras partes del sistema para procesar las solicitudes.

7.3. Backend

El backend está construido con **Spring Boot** y organizado en **microservicios independientes**, cada uno manejando una parte específica del sistema, como la gestión de usuarios, clientes, rutas, camiones, incidentes y abastecimiento. Estos microservicios permiten modularidad y escalabilidad, facilitando actualizaciones y mantenimiento.

7.3.1. Aquitectura de diseño: Backend

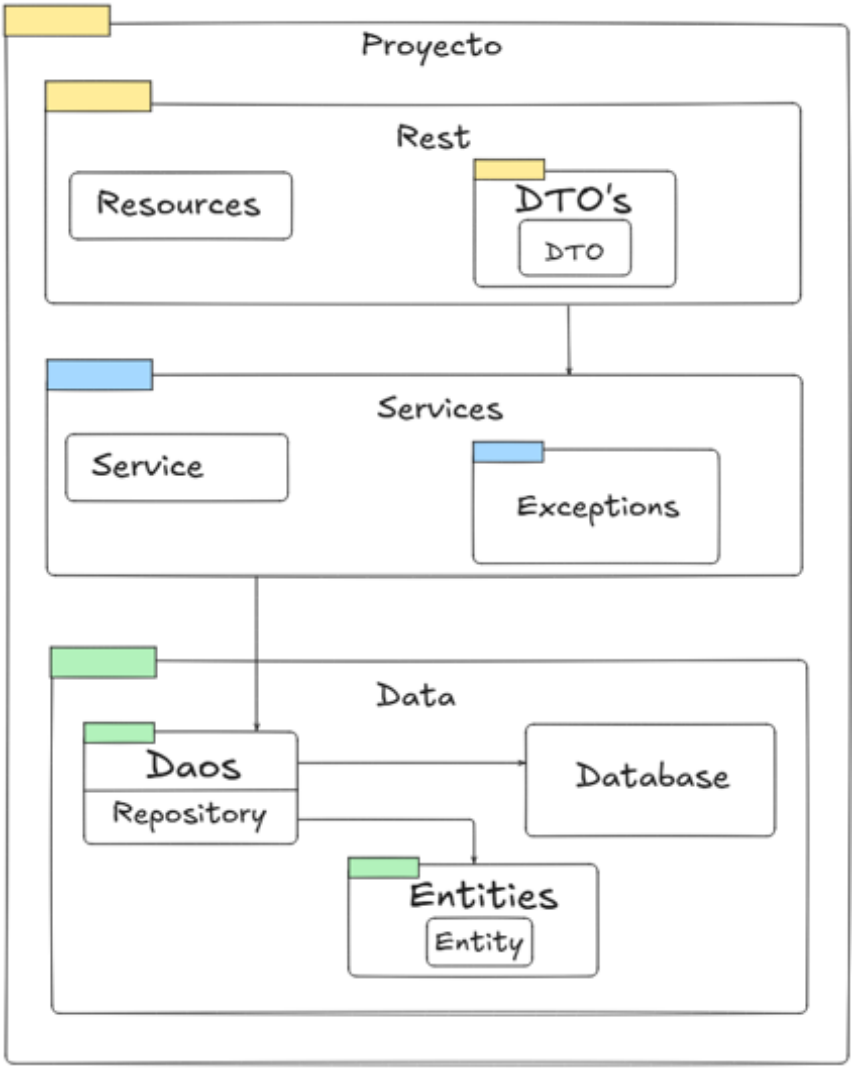


Figura 4: Arquitectura del backend

Este diagrama representa la arquitectura interna del backend, estructurada en varias capas para mantener una separación de responsabilidades y modularidad en el diseño.

- **Rest:** Es la capa de presentación y comunicación externa. Aquí se encuentran:

    - **Resources:** Controladores que manejan las solicitudes HTTP que llegan al sistema, actuando como el punto de entrada para los diferentes servicios.
    - **DTOs (Data Transfer Objects):** Objetos de transferencia de datos, usados para enviar datos entre la capa de presentación y la lógica de negocio, asegurando que solo la información necesaria sea expuesta.
- **Services:** Capa de lógica de negocio. Aquí se define la funcionalidad principal de la aplicación:
    - **Service:** Contiene la lógica de negocio, realizando las operaciones principales y coordinando la interacción entre los diferentes componentes.
    - **Exceptions:** Manejo de excepciones, que permite capturar y gestionar errores en la lógica de negocio de manera controlada.
- **Data:** Capa de acceso a datos. Su función es interactuar directamente con la base de datos:
    - **DAOs (Data Access Objects) / Repository:** Objetos que permiten acceder a la base de datos, encapsulando las operaciones de lectura y escritura.
    - **Entities:** Representaciones de las tablas de la base de datos en formato de objetos (normalmente usando una herramienta ORM).
    - **Database:** Representa la conexión y gestión de la base de datos misma.

Esta estructura se conoce comúnmente como **arquitectura en capas** (o **arquitectura multicapa**), y en el contexto de aplicaciones Java, se refiere a una variación de la **arquitectura hexagonal** o **arquitectura limpia** cuando se aplican principios de separación de responsabilidades.

En particular, la arquitectura presentada sigue una aproximación basada en el patrón **Modelo-Vista-Controlador (MVC)** en combinación con principios de **DAO (Data Access Object)** para el acceso a datos y **DTO (Data Transfer Object)** para la transferencia de datos. En el desarrollo de aplicaciones empresariales en Java, este tipo de arquitectura es una práctica común porque permite una clara separación entre la lógica de presentación, la lógica de negocio y el acceso a datos.

7.4. Base de datos

La base de datos está soportada en **Oracle** y administrada con **SQL Developer** (herramienta brindada por el mismo Oracle), y cada microservicio interactúa con ella a través de solicitudes SQL. La respuesta de la base de datos, en JSON, es procesada por los microservicios y devuelta al front-end. Esta estructura asegura un flujo de datos eficiente y adaptable, optimizando la recolección y gestión de información en tiempo real.

El proyecto, como bien se menciona, está diseñado siguiendo una arquitectura de microservicios, en este caso compuesto de 10 microservicios dedicados a cada entidad en la base de datos que se consideró pertinente:

Service	Description
Micro route business	Lógica del negocio que sostiene las operaciones principales durante los recorridos de recolección. Es el micro-servicio principal.
Micro route config	Encargado de las configuraciones de cada micro ruta, como la fecha de programación o la frecuencia de recolección, a través de su sistema CRUD.
Aforo	Encargado de las operaciones CRUD correspondientes a los aforos, es decir, la recolección en cada punto de la ruta.
User	Encargado de las operaciones CRUD correspondientes a los usuarios, incluyendo sesiones web y móvil, autenticaciones y obtención de datos de usuario.
Client	Encargado de las operaciones CRUD correspondientes a los clientes a los que se les realizan los aforos.
Truck	Encargado de las operaciones CRUD correspondientes a los camiones de recolección operados por los usuarios, llevando registro de datos como la capacidad de carga, placa de cada camión, documentación, combustible y demás datos técnico-mecánicos.
Refuel	Encargado de las operaciones CRUD correspondientes a los repostajes de combustible y la evidencia del mismo.
Incident	Encargado de las operaciones CRUD correspondientes a los incidentes, estas son las diferentes situaciones extraordinarias que podrían suceder fuera durante un recorrido de recolección.
Provision	Encargado de las operaciones CRUD correspondientes a los registros de ida a los rellenos sanitarios, maneja datos de traza como posibles peajes y lugares de provision exclusivamente durante el camino de ida al relleno sanitario.
Photo	Encargado de las operaciones CRUD correspondientes a las fotos de evidencias, pues la mayoría de operaciones como los aforos y repostajes requieren evidencias fotográficas.

Tabla 6: Descripción de micro-servicios

7.5. Esquema de base de datos

Se establece la imagen completa del esquema de la base de datos en el primer anexo.

En el esquema se hace notable que una tabla como **"nodo central"** de toda la base de datos, es esta la entidad de micro ruta, es básicamente la razón principal del negocio, pues una micro ruta es la traza de puntos de recolección de residuos; esta se define como:

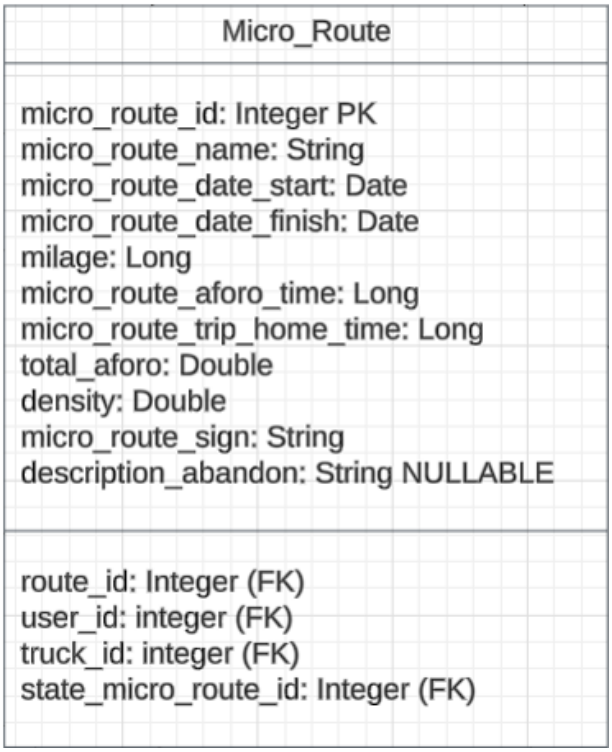
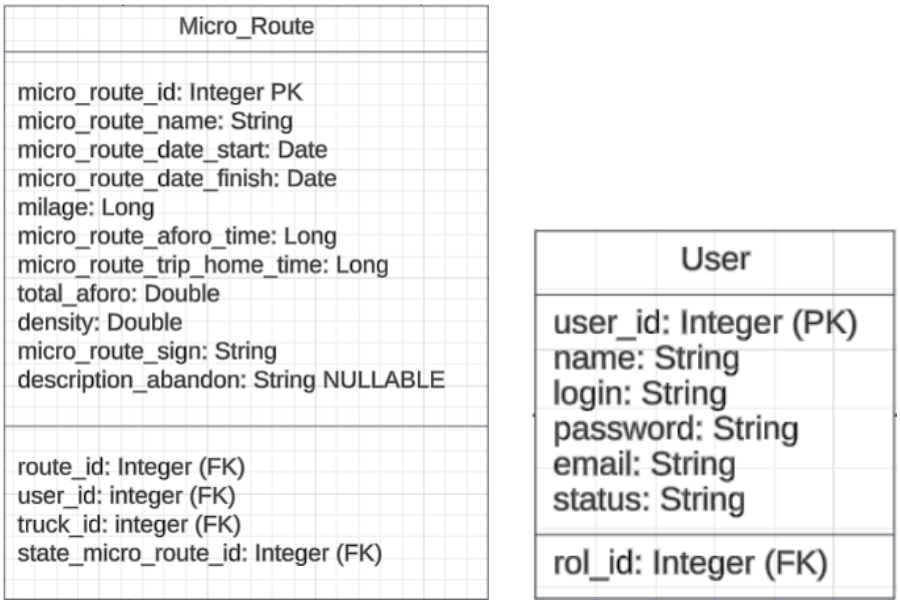


Figura 5: Esquema micro ruta

Conteniendo datos como: identificador único, nombre de la micro ruta, fecha de inicio y fin, kilometraje de esa micro ruta, el tiempo que toma volver a la base después de finalizada la micro ruta, esta con fines de control, el tiempo que tomó todo el recorrido, total aforado correspondiente a la cantidad de aforos (recolecciones) realizadas en la micro ruta, densidad (en peso, espacio o contenedor), firma del cliente después de la recolección, y descripción de abandono en caso en que el conductor lo haya decidido bajo una de diversas y puntuales razones. La base de datos está diseñada para gestionar la logística y rutas de recolección. La estructura se centra en varias entidades clave interconectadas, **se explican de acuerdo a los micro servicios que las contienen**:

7.5.1. Micro Route Business (Usuarios, Rutas y Micro-rutas):



(a) Micro-route (b) User

(c) Route (d) Crew

Figura 6: Micro route business

Este microservicio maneja las operaciones principales relacionadas con las micro rutas. Gestiona las tablas **Micro Route (presentada en la Figura 2), Route, Client**, y auxiliarmente **User** y **Crew**.

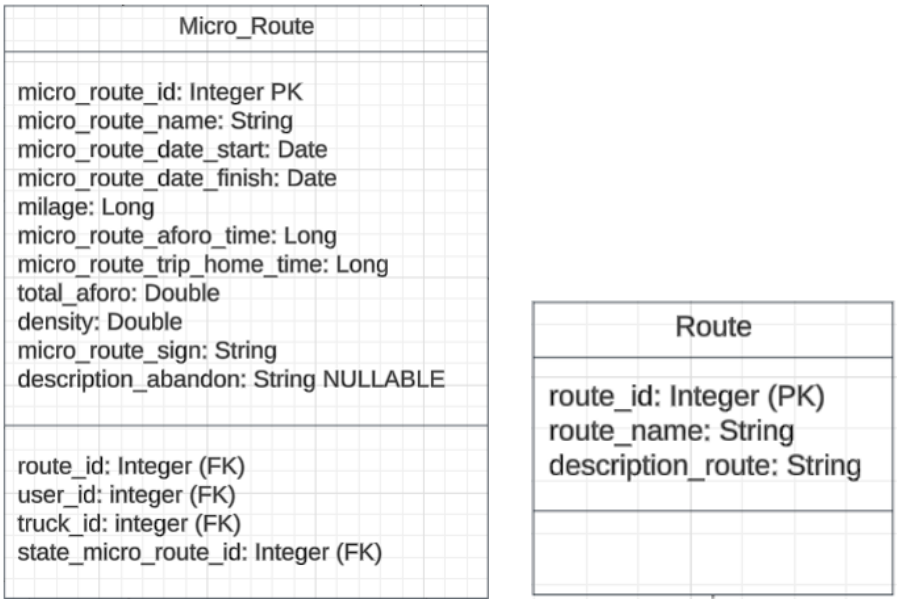
La función principal de este servicio es obtener y actualizar información básica de las micro rutas. Maneja la creación de rutas en relación con la tripulación **(crew)**, lo que enlaza la integración entre las rutas y el personal asignado.

- La tabla **"User"** contiene información sobre los usuarios del sistema.
- **"Route"** representa las rutas principales, se definen éstas como pequeñas zonas urbanas, podría abarcar varios barrios de la ciudad.
- Las rutas se dividen en **"Micro Route"**, que son segmentos más pequeños, una traza punto a punto de cada lugar de recolección.

- **"Crew"** contiene información sobre el personal que opera los vehículos y la tripulación recolectora que acompañan al conductor.

7.5.2. Micro Route Config (Rutas y Micro-rutas):

Este microservicio se centra en la configuración y gestión del ciclo de vida de las micro rutas. Maneja principalmente las tablas **Route** y **Micro Route**.



(a) Micro-route (b) Route

Figura 7: Micro route config

Existen operaciones de CRUD **(get list of route**, **create micro route**, **update micro route**, **delete micro route**, entre muchas operaciones específicas) que indican un control completo sobre el ciclo de vida de las micro rutas. Este servicio es crucial para la planificación y previo ajuste de las rutas de recolección.

7.5.3. Aforo (Recolecciones y recipientes):

Este microservicio gestiona la capacidad y los recipientes de recolección. Probablemente maneja las tablas **Aforo**, **Recipient** y auxiliarmente **Not Collected**.



(a) Aforo

Figura 8: Aforo

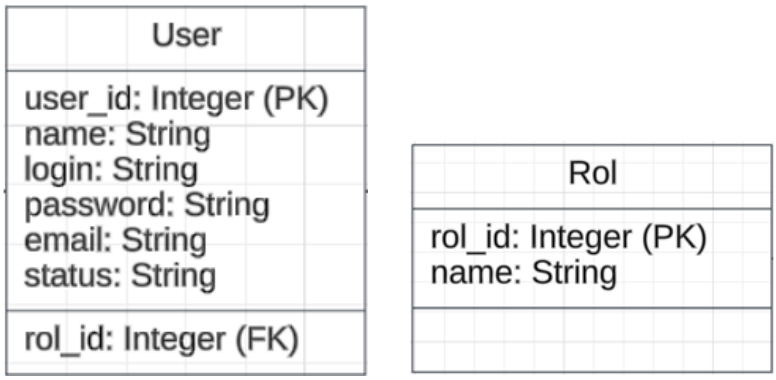
- **"Client"** guarda los datos de los clientes, refiriéndose a aquellos a los que se les hace la recolección.
- **"Aforo"** se define como el servicio proporcionado a los clientes, es decir, la recolección como tal.

La creación de aforo y recipientes, junto con la gestión de fotos y listas de no recolección, hacen de este servicio el responsable de monitorear y registrar la capacidad de recolección en cada punto de la ruta.

7.5.4. User (Control de acceso y datos de usuario):

Este microservicio maneja la gestión de usuarios, interactuando principalmente con la tabla **User** y auxiliarmente con **Rol**.

Existen funciones de CRUD (**create users**, **get list of users**, **authentication**) para hacer de este servicio el responsable de la seguridad y el control de acceso al sistema.



(a) User (b) Route

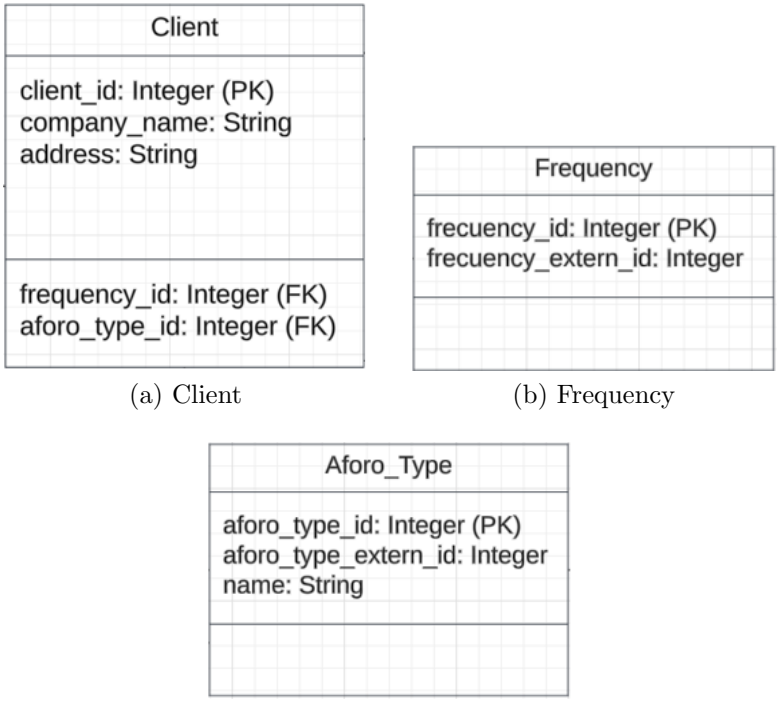
Figura 9: User service

7.5.5. Client:

Este microservicio gestiona la información de los clientes y la frecuencia de recolección. Maneja las tablas **Client**, **Frequency**, y auxiliarmente **Atoro Type**.

La creación de clientes y la asociación con tipos de frecuencia y aforo denotan que este servicio es crucial para personalizar los servicios de recolección según las necesidades específicas de cada cliente.

- **"Frequency"** hace referencia a la frecuencia en días con la que se hará visita para recolección.
- **"Aforo Type"** hace referencia al tipo de aforo que se le realizará al cliente, este puede ser diferente dependiendo de la forma de medida del total de desechos.



(a) Client (b) Frequency

(c) Aforo Type

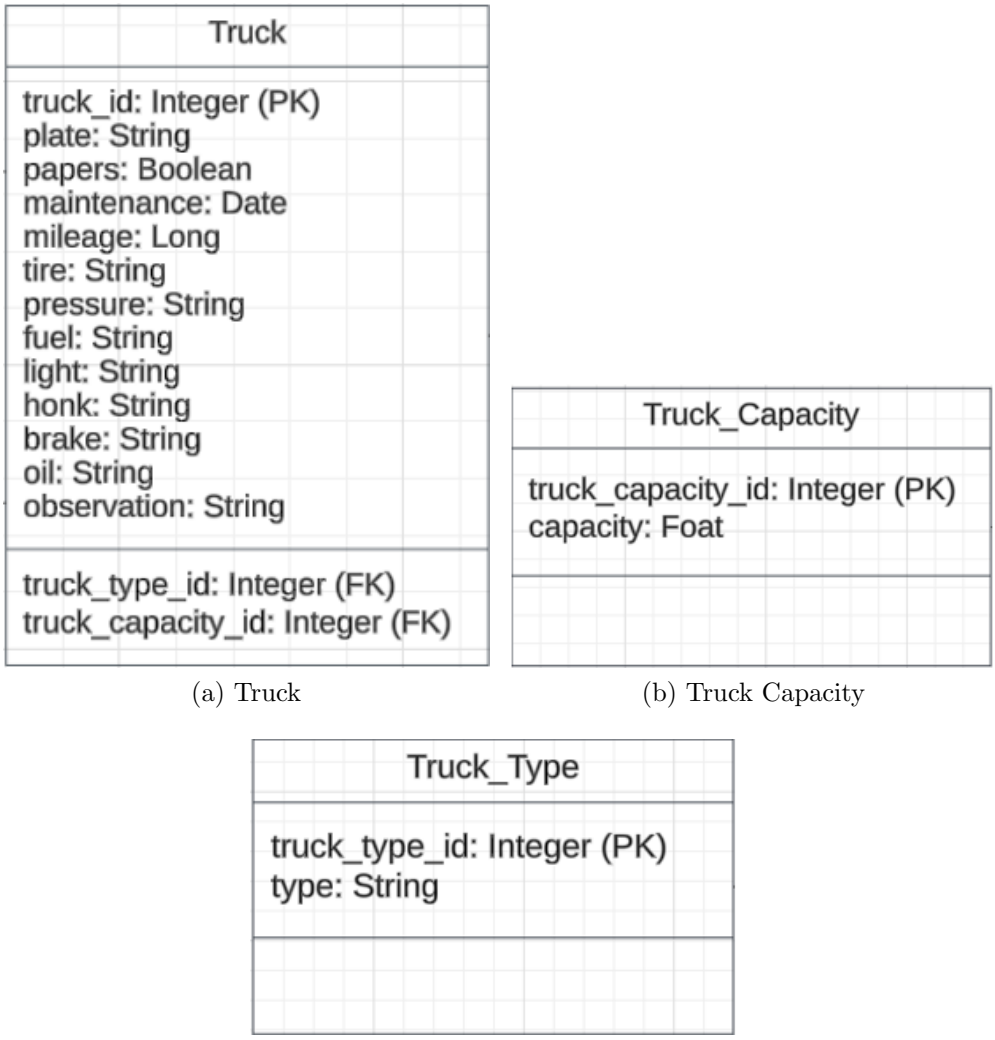
Figura 10: Client service

7.5.6. Truck:

Este microservicio maneja la información relacionada con los vehículos de recolección. Gestiona las tablas **Truck**, **Truck Type**, y de manera auxiliar **Truck Capacity**.

Las operaciones para este microservicio realizan un seguimiento detallado del estado y las características de los camiones, lo que es esencial para la planificación eficiente de las rutas y el mantenimiento de la flota.

- **"Truck"** almacena datos sobre los vehículos a disposición de la empresa, desde papeleo hasta estado del vehículo.
- **"Truck Capacity"** guarda datos auxiliares sobre la capacidad de cada vehículo.
- **"Truck Type"** guarda datos auxiliares sobre el tipo de camión, pues existen variantes de camiones de recolección.



(a) Truck

(b) Truck Capacity

(c) Truck Type

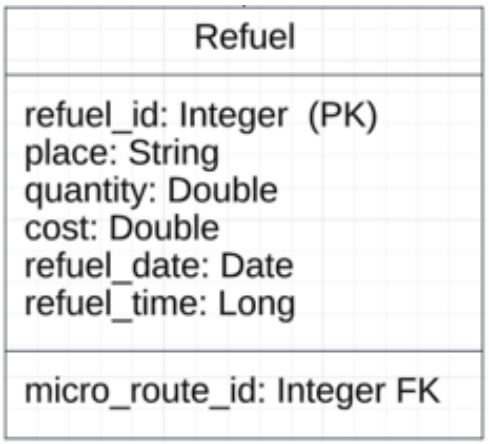
Figura 11: Truck Service

7.5.7. Refuel:

Este microservicio gestiona el reabastecimiento de combustible. Maneja la tabla **Refuel**.

Aunque tiene funcionalidades limitadas, la gestión de reabastecimiento es importante para la operación continua de la flota y la planificación de costos.

- **"Refuel"** hace referencia a datos relevantes sobre un repostaje de combustible, como el lugar, la cantidad y el costo.

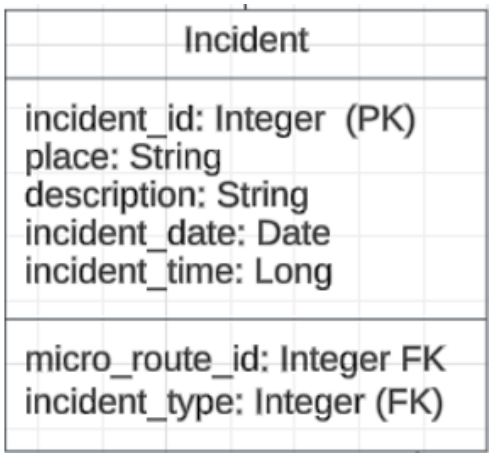


(a) Refuel

Figura 12: Refuel service

7.5.8. Incident:

Este microservicio maneja los incidentes durante las operaciones de recolección. Gestiona las tablas **Incident** e **Incident Type**.



(a) Incident

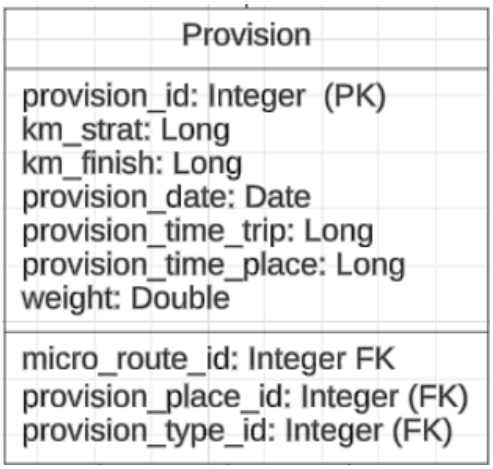
Figura 13: Incident service

La capacidad de crear y categorizar incidentes, así como adjuntar fotos, sugiere que este servicio es esencial para el manejo de problemas en tiempo real y la mejora continua de las operaciones.

- **"Incident"** permite el registro de eventos o problemas durante las operaciones.

7.5.9. Provision:

Este microservicio gestiona las provisiones necesarias para las operaciones. Maneja las tablas **Provision**, **Provision Type**, y **Provision Place**.



(a) Provision

Figura 14: Provision service

La creación de provisiones y la gestión de tipos y lugares de provisión hacen de este servicio el responsable de la logística y el abastecimiento de los recursos necesarios para las operaciones de recolección.

- **"Provision"** referente a datos de reposición de combustible de los camiones recolectores.

Esta estructura permite un seguimiento detallado de las operaciones, desde la planificación de rutas hasta la ejecución y el servicio al cliente, facilitando la gestión de la logística de la recolección.

8. Configuración y características

8.1. Alojamiento de repositorio: BitBucket

La decisión de utilizar Bitbucket para alojar el repositorio del proyecto se basa en varias ventajas clave que ofrece la plataforma, especialmente para un entorno de desarrollo colaborativo en el ámbito empresarial.

- **Integración con herramientas de Atlassian:** Bitbucket forma parte del ecosistema de Atlassian, lo que facilita la integración con otras herramientas. Esto permite una gestión más eficiente del ciclo de vida del proyecto. Además de ser este el ecosistema de uso principal dentro de la empresa.
- **Soporte para Git:** Bitbucket admite Git como sistema de control de versiones.
- **Control de acceso y seguridad:** Para un proyecto interno, Bitbucket ofrece control granular sobre los permisos de acceso al repositorio. Esto asegura que solo los desarrolladores autorizados puedan acceder al código fuente y realizar cambios.
- **Repositorios privados gratuitos para equipos pequeños:** Bitbucket permite la creación de repositorios privados de forma gratuita para equipos pequeños, lo que lo convierte en una opción rentable para proyectos internos que requieren colaboración controlada sin costos adicionales significativos

8.1.1. Despliegues: Docker

- **App web**

Para el despliegue tanto del back-end como del front-end web de usó Docker, este funciona creando contenedores, que son entornos ligeros y aislados donde se empaqueta la aplicación junto con todas sus dependencias. Esto permite que tanto el frontend como el backend se ejecuten de manera consistente en cualquier servidor, sin importar el entorno subyacente. Para el despliegue, Docker utiliza un archivo de configuración (como un **Dockerfile** o **docker-compose.yml**) que define cómo se debe construir y ejecutar cada contenedor. Una vez configurado, los contenedores pueden ser iniciados, detenidos o escalados fácilmente, facilitando el despliegue y la gestión de la aplicación en un entorno de producción.

Docker impulsa la **escalabilidad** horizontal al permitir la duplicación rápida de instancias para balancear carga, y vertical al ajustar los recursos asignados a cada contenedor según la demanda. Además, su integración con orquestadores como Kubernetes facilita una gestión dinámica y eficiente, optimizando el rendimiento sin modificar la infraestructura base. De esta manera se garantiza un apoyo al cumplimiento del **segundo punto en los objetivos planteados** para el **proyecto**.

Importante mencionar que el dominio web utilizado durante los desarrollos, pruebas y producción es un dominio propio adquirido por la empresa.

- **App móvil**

La aplicación móvil NO será de uso comercial, será de uso interno en la empresa, como ya se mencionó, por ende, al finalizar el desarrollo se empaquetará y se generará un **APK** (**Android Application Package**), es decir, un archivo ejecutable que contiene todos los datos que se necesitan para instalar y hacer funcionar la aplicación, para distribuirlo en los dispositivos que estarán utilizando los operadores de la recolección.

9. Implementación de prototipos

Respecto al funcionamiento back-end del proyecto, tanto para la app móvil como para el panel web se han desarrollado varios microservicios encargados de áreas/elementos específicos de acuerdo con las entidades en la base de datos, tal como se demostró en el esquema del proyecto **(Figura 1)**.

De la misma manera como se demuestra en el esquema de la base de datos **(Figura 1)**, esta es la misma para ambos ambientes, tanto app móvil como panel administrativo web; es así necesario para poder ver en tiempo real lo que se modifique desde el panel directamente en la app.

9.1. Aplicación web: React, Mantine y TanStack

9.1.1. Configuración inicial y desarrollo básico

9.1.2. Creación del proyecto:

El desarrollo de la aplicación web se llevó a cabo utilizando **React** con **TypeScript**, aprovechando la eficiencia de **Vite** como herramienta de construcción, siendo esta la opción más rápida en cuanto a tiempos de compilación y la capacidad de ver cambios en el servidor en vivo durante el desarrollo. Para el proyecto se usó la versión (actual en su fecha - mayo) **21 de Node.js** para garantizar compatibilidad y soporte a largo plazo.

La elección de TypeScript añadió robustez al proyecto al proporcionar el tipado estático característico de este, que minimizó errores comunes, aprendidos en proyectos de empresa pasados, durante el desarrollo.

9.1.3. Mockups de interfaz

Bajo el mismo concepto explicado en la app móvil, para el desarrollo de la parte front-end de la app web, es decir, el panel administrativo, se tomó como referencia un módulo desarrollado previamente por la empresa, debido a que este encaja perfectamente con lo que se necesita según requerimientos, compuesto de una navbar vertical que gestiona las diferentes vistas en la parte derecha según selección, tal como se evidencia a continuación:

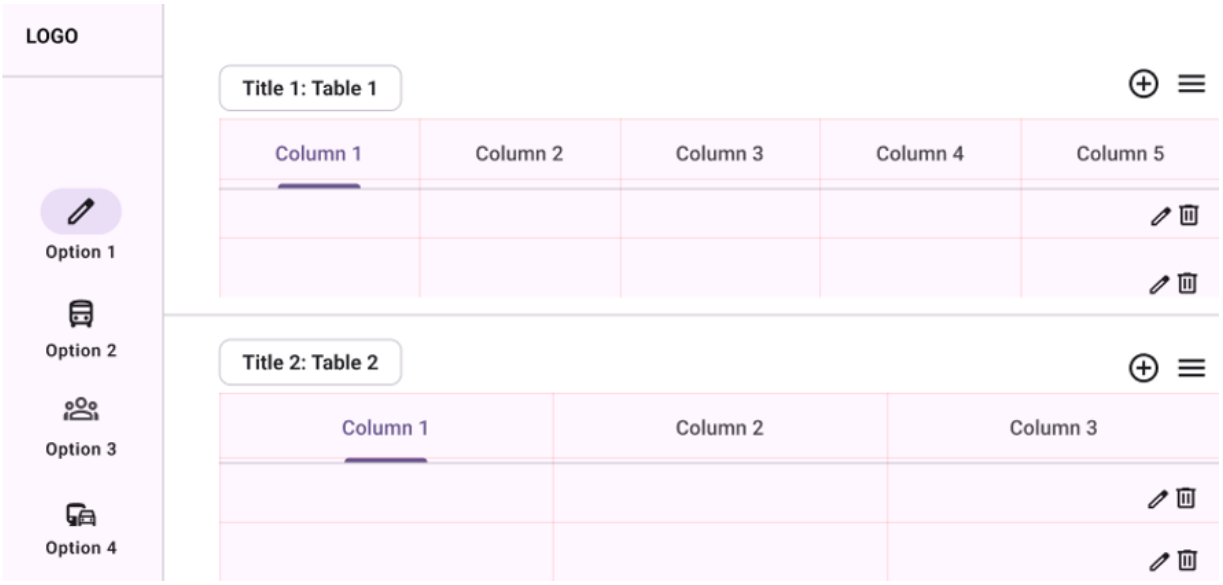


Figura 15: Mockup general de vista con tabla

- Todas las **secciones en la navbar** contienen tablas de diferentes categorías, algunas editables otras solo informativas
- Se pretende que las tablas sean **editables "In-Row"**, es decir, que al seleccionar editar en una fila, esta pondrá todos sus campos como editables, luego se guardar, esta realizara la petición de edición, actualizará y mostrará los nuevos datos
- El **filtro de búsqueda** se hará de manera que: al seleccionar el botón de filtro, se despliegue una fila más por debajo de las cabeceras de las tablas con campos de entrada de texto, la búsqueda será instantánea, conforme se escribe, se realizan peticiones de búsqueda, por lo que será bastante rápida y agradable a la vista

9.1.4. Uso de Mantine para componentes prefabricados:

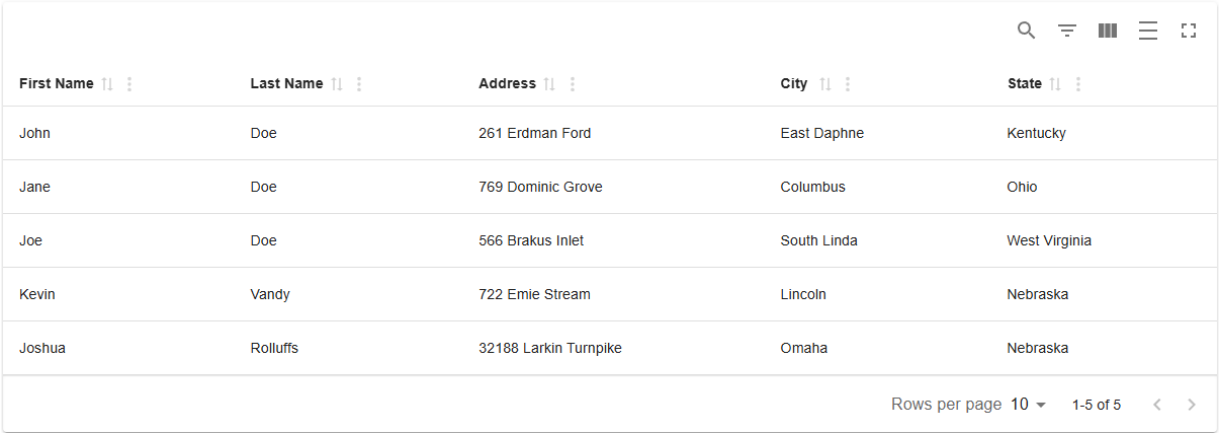
Mantine Dev proporcionó un conjunto de componentes preconstruidos y personalizables que aceleraron el desarrollo del diseño de interfaz.

Elementos como **formularios, modales y barra de navegación** se adaptaron muy fácilmente a los requisitos del proyecto.

El sistema de tematización de Mantine permite personalizar colores, tipografía y espaciados para alinearlos con la identidad visual que requiere el cliente.

9.1.5. Exploración de Tan Stack Table:

Inicialmente, se utilizaron las tablas dinámicas que ofrecía TanStack Table para gestionar los datos de rutas y micro-rutas. Sin embargo, surgieron problemas de compatibilidad con otras dependencias y con la versión 21 de Node.js, debido a la falta de actualizaciones recientes en la librería de TanStack.



First Name	Last Name	Address	City	State
John	Doe	261 Erdman Ford	East Daphne	Kentucky
Jane	Doe	769 Dominic Grove	Columbus	Ohio
Joe	Doe	566 Brakus Inlet	South Linda	West Virginia
Kevin	Vandy	722 Emie Stream	Lincoln	Nebraska
Joshua	Rolluffs	32188 Larkin Turnpike	Omaha	Nebraska

Rows per page 10 1-5 of 5

Figura 16: Tabla tanstack

Como alternativa, se implementaron tablas funcionales partiendo de unas plantillas básicas de Mantine con soporte para operaciones CRUD mínimas funcionales hasta este punto del desarrollo, y datos de prueba incrustados en el código.



Rutas

Id Ruta	Nombre	Descripción	Estado	Acciones
6	RUTA 40	BOGOTÁ	ACTIVO	
8	Test APP	La descripcion y el estado están trucadas al crear	ACTIVO	

1

Figura 17: Tabla rutas

Una **nota adicional** es que se decidió replicar el diseño que tenían estas tablas de TanStack Table, puesto que estas alcanzaron a ser propuestas al cliente y fueron de gran agrado.

9.2. Implementación avanzada de tablas y optimización de componentes

9.2.1. Desarrollo de la tabla con datos de clientes:

Se añadió una tabla para gestionar clientes; con esta se incluyeron mejoras en el módulo de las tablas:

- **Paginación dinámica:** Implementada manualmente para manejar los grandes conjuntos de datos que demandaba un negocio de recolección de residuos.
- **Filtrado y ordenamiento:** Los filtros se habilitaron mediante un componente de entrada en cada columna, con procesamiento en tiempo real.

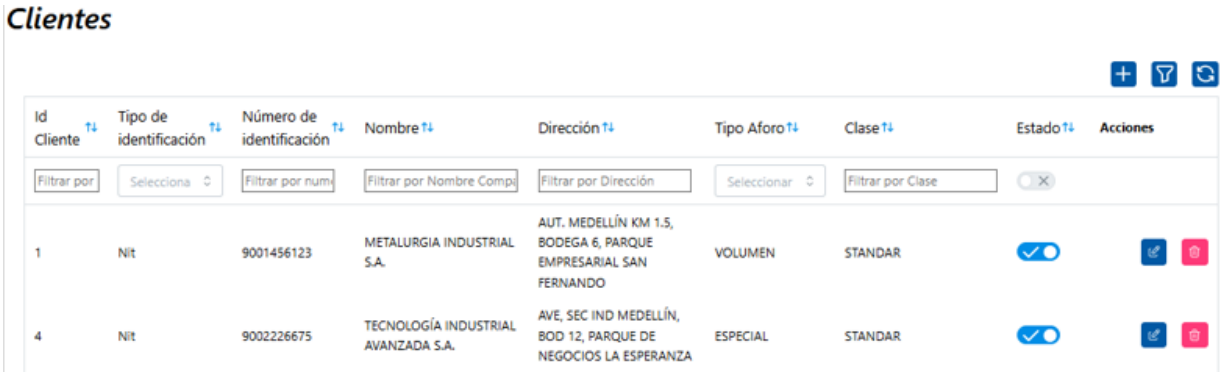


Figura 18: Tabla clientes

- **Activación/desactivación de columnas:** Correspondiente a uno de los requerimientos del cliente, el poder activar/desactivar el estado de un cliente.



Figura 19: Columna desactivación de dato

9.2.2. Funcionalidad drag-and-drop:

Se implementó un mecanismo de **ordenamiento manual** de rutas utilizando **hello-pangea/dnd**, un modificador aplicable a las tablas para hacerlas **"Drag and Drop"** y con esto poder arrastrar y soltar las filas en un orden diferente. Este enfoque permitió a los administradores ajustar el orden de las rutas de forma intuitiva.

9.3. Edición y creación in-row:

Para una experiencia más fluida, se desarrolló la edición de registros directamente dentro de las filas de la tabla:

- Al activar la edición, los campos de la fila se vuelven interactivos.
- Se integraron botones de "guardarz çancelar"dentro de la misma fila cuando se estaba en "modo edición".
- La creación de nuevos registros se realizó desde la cabecera de la tabla, permitiendo al usuario completar los datos directamente sin necesidad de modales emergentes.

9.3.1. Creación:

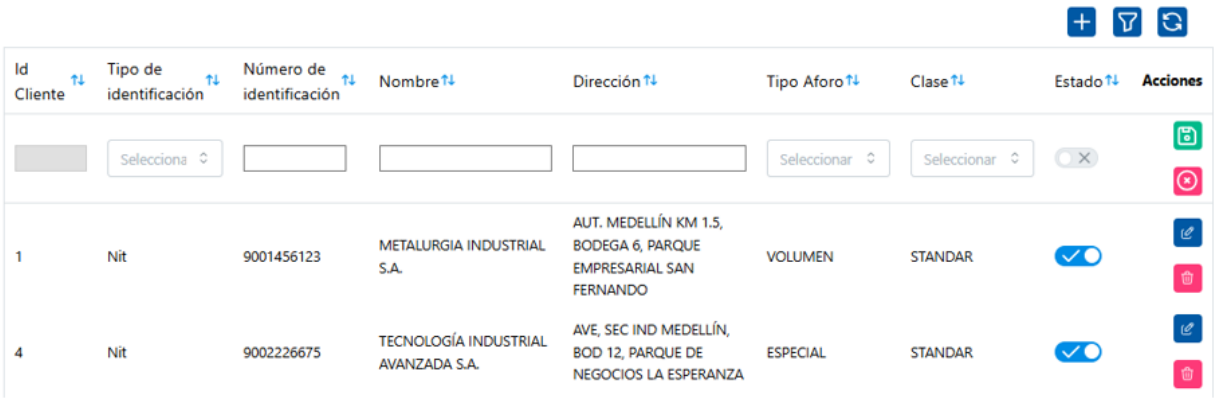


Figura 20: Desplegable de creación (in-row)

9.3.2. Edición:



Figura 21: Input de edición (in-row)

9.4. Integración con el backend y estabilidad del sistema

9.4.1. Depuración de tablas y extras:

Se corrigieron errores en la lógica de **paginación**, asegurando una transición sin problemas entre páginas y manteniendo un seguimiento preciso del estado. Además de una ligera animación al cargar nuevos datos en pantalla.

9.4.2. Conexión al backend:

Se eliminó el uso de datos estáticos en el frontend, integrando las tablas con **APIs REST** proporcionadas por el backend (desarrollado en **Spring Boot**)

Para esto se creó una carpeta "**API**"que contiene subcarpetas con su respectivo archivo de configuración hacia el backend para cada entidad significantiva, estos se nombran **Entidad + Call + Config**, por ejemplo para clientes se tiene su archivo llamado "**ClientCallConfig**", en estos se establecen los llamados HTTP necesarios.

Estas operaciones CRUD y demás personalizadas se implementaron a través de **Fetch API**, lo que permitió un manejo eficiente del estado remoto, soporte para caché y sincronización en tiempo real con el backend.

9.4.3. Manejo de errores y validación:

Se añadieron mensajes de error descriptivos para gestionar problemas como fallos en la conexión al servidor o datos inválidos enviados desde el cliente.

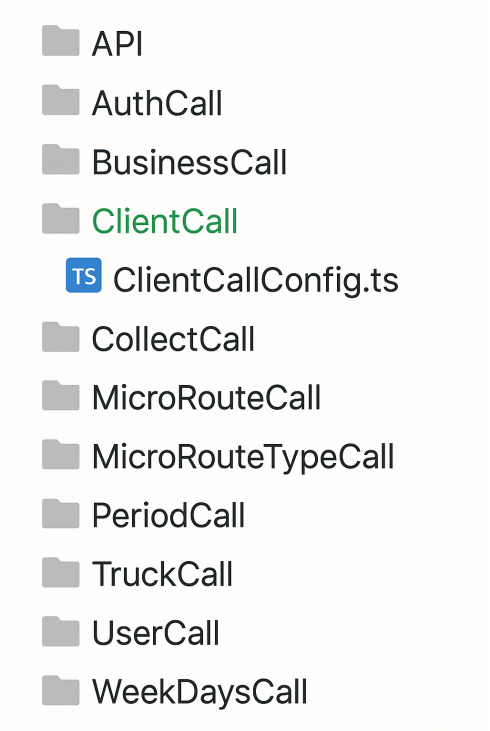


Figura 22: Estructura de archivos

Se integraron validaciones de entrada tanto en el frontend como en el backend para mantener la integridad de los datos.

Ejemplo al crear un nuevo cliente con el campo de identificación vacío:



Figura 23: Validación campo obligatorio vacío

9.5. Aplicación móvil: Android Studio & Kotlin

9.5.1. Elección de tecnologías y herramientas

El desarrollo de la aplicación móvil se realizó en **Android Studio** utilizando **Kotlin**, el lenguaje oficial de Android. Esta elección estuvo motivada por su sintaxis moderna y concisa, que facilita el desarrollo seguro y eficiente, además de ser completamente interoperable con Java.

Se integraron diversas bibliotecas de **Android Jetpack**, como **LiveData**, **ViewModel** y **Navigation**, que simplificaron la implementación del patrón **MVVC**. Este patrón permite una separación clara de responsabilidades, aportando a la mantenibilidad del código.

9.5.2. Mockups de interfaz

A partir del establecimiento de los requerimientos y las necesidades expuestas, se estructuraron detallados bosquejos de la estructura visual (front-end) de la aplicación, la idea es que sea tanto completa como intuitiva ante los operadores quienes van a estar dándole el uso diario.

Se muestran entonces los mockups respectivos de acuerdo a las historias de usuario mencionadas, correspondientes al tercer caso (CASO 3):

- **Funcionalidad:** El recolector ingresa la APP y proporciona credenciales válidas para poder estar activo en el sistema. **Anexo 2**
- **Funcionalidad:** Realizar el proceso de validación de condiciones técnicas del vehículo antes de iniciar el recorrido y registrar datos de la tripulación. **Anexo 3**
- **Funcionalidad:** Tomar los aforos con sus respectivos soportes. **Anexo 4**
- **Funcionalidad:** Iniciar el proceso de recorrido. **Anexo 5**
- **Funcionalidad:** Repostaje. **Anexo 6**
- **Funcionalidad:** Registro de Incidencias. **Anexo 7**
- **Funcionalidad:** Finalizar Ruta. **Anexo 8**
- **Funcionalidad:** Inicio de viaje al relleno sanitario. **Anexo 9**
- **Funcionalidad:** Registrar ingreso a relleno y capturar peso de disposición final. **Anexo 10**
- **Funcionalidad:** Finalización de disposición de relleno y retorno a la base. **Anexo 11**

9.5.3. Implementación del diseño de la interfaz

Las vistas se diseñaron con **Layouts XML**, proporcionando una estructura visual limpia y consistente con las directrices de diseño de **Android Material Design**, el sistema de diseño de Google que ofrece lineamientos, componentes y herramientas para crear interfaces de usuario.

Cada pantalla se optimizó para facilitar el flujo de interacción de los usuarios operativos.

- **Uso de Android Material:** Se emplearon componentes como botones, formularios y barras de navegación, adaptados a los requerimientos de la app móvil.
- **Pruebas de diseño:** Se realizaron ajustes para asegurar una correcta visualización en dispositivos específicos, considerando la resolución y tamaño de pantalla, pues la app está pensada para ser ejecutada en **dispositivos tipo tablet** con cierto margen de dimensiones.

9.5.4. Gestión de la comunicación y datos

Para la conexión con los microservicios backend (desarrollados en **Spring Boot**), se utilizó **Retrofit**, una biblioteca que simplifica las solicitudes **HTTP** en este campo. La integración incluye:

- **Consumo de APIs REST**: Configuración de endpoints para operaciones CRUD asociadas a rutas, clientes, aforos y demás funcionalidades críticas.
- **Manejo de errores:** Implementación de respuestas claras en casos de fallas en la red o datos no válidos, mejorando la experiencia del usuario.

9.5.5. Pruebas y aseguramiento de calidad

Se realizaron pruebas unitarias y de integración utilizando **JUnit** y herramientas complementarias de Android Studio, verificando la funcionalidad de cada módulo.

- Validación de datos ingresados por los usuarios.
- Simulación de escenarios con datos reales y entornos limitados de conectividad.
- Pruebas de ciclo de vida en las vistas mediante LiveData, garantizando actualizaciones en tiempo real y sincronización con el backend.

9.5.6. Optimización y rendimiento: Tareas asíncronas

Se incorporaron **coroutines** para manejar tareas asincrónicas, como la carga de rutas, la actualización de estados de los aforos y la medición de tiempos de entregas durante los recorridos.

10. Adición al trabajo movil terminado

Una vez se llegó a un punto estable del desarrollo de la app móvil, se agregó un desarrollo extra que duró alrededor de **1 mes**, correspondiente a la implementación de una base de datos local que tenía como fin hacer que la aplicación pudiera funcionar, aunque se haya perdido repentinamente su conexión a internet.

Este desarrollo **no está contemplado** en los objetivos principales del proyecto, pero se documenta de igual manera a modo de extra.

10.1. Semana 1: Análisis y propuesta inicial

10.1.1. Análisis del código existente:

Se revisó la base de código escrita en Kotlin para comprender su arquitectura y el flujo actual de datos. La aplicación estaba diseñada inicialmente para depender exclusivamente de una conexión constante a internet, interactuando con APIs REST y la base de datos remota.

10.1.2. Propuesta de base de datos local:

Tras el análisis, propuse una solución basada en la implementación de una base de datos local utilizando **Room**, la librería oficial de persistencia de datos de Android.

La funcionalidad planteada tenía como objetivo almacenar solicitudes y respuestas de las APIs en la base de datos local cuando no hubiera conexión, para sincronizar los datos una vez que la conectividad se restaurara.

10.1.3. Requerimientos funcionales:

- El sistema debía verificar constantemente la conexión a internet mediante un observable de **LiveData**.
- En caso de desconexión, las operaciones normales de la app debían continuar, almacenando, en ese caso, los datos en la base local.
- Al restablecer la conexión, se debía implementar un mecanismo de sincronización automática que asegurara la transferencia de datos locales al backend remoto sin pérdida de información.

10.2. Semana 2: Reestimación y cambio de enfoque

10.2.1. Reestimación de la complejidad técnica:

Durante los intentos iniciales de implementación, encontré que la integración de la base de datos local con una aplicación móvil ya avanzada presentaba desafíos significativos:

- **Flujo de datos interrumpido:** La introducción de una capa de persistencia local requería modificaciones sustanciales en los ViewModels y Repositorios existentes.
- **Sincronización bidireccional:** Garantizar que las operaciones realizadas offline no generan inconsistencias al sincronizar con la base remota añadía una complejidad notable al flujo de la app.

Esto llevó a una reestimación del tiempo necesario y a replantear la estrategia.

10.2.2. Segunda etapa de la implementación:

La estrategia inicial utilizaba **Room** junto con observables de **LiveData** para monitorear la conexión a internet y almacenar solicitudes y respuestas completas en tablas temporales. Sin embargo, esta implementación resultó ser invasiva, alterando el flujo normal de la aplicación y afectando el rendimiento.

Investigando técnicas más simples, decidí trabajar en una estrategia que redujera la carga computacional y los riesgos de interrupciones en la app.

10.3. Semanas 3 y 4: Implementación definitiva y refinamiento

10.3.1. Cambio en la estrategia de persistencia:

Opté por una solución simplificada, implementar una tabla de **Registro"** en la base de datos local, diseñada para almacenar únicamente un "historial" de los métodos API llamados y los datos asociados (payloads) mientras no había conexión.

En lugar de intentar ejecutar los métodos directamente offline, la app simplemente registraba:

- **Método llamado:** Sea POST, GET, PUT, DELETE, almacenando el nombre del método.
- **Ruta del endpoint:** Identificador único del recurso.
- **Payload enviado:** Datos de request.

Al restaurarse la conectividad, la aplicación procesaba los registros de manera secuencial, ejecutando las solicitudes pendientes en el orden original.

10.3.2. Flujo de sincronización:

- **Modal de espera:** La app mostraba un modal informando al usuario que los datos locales estaban siendo sincronizados con la base remota.
- **Estrategia de sincronización FIFO:** La sincronización seguía un esquema de First In, First Out (FIFO), asegurando que los llamados se ejecutaran en el mismo orden en que se almacenaron.

- **Control de errores:** Se añadieron manejadores para reintentos automáticos y notificaciones al usuario en caso de fallos persistentes.

11. Capactaciones

Para evaluar el funcionamiento y aceptación de la solución, se implementó una estrategia de validación compuesta por:

- **Pruebas funcionales**: Se realizaron pruebas internas por parte del equipo de desarrollo sobre cada módulo, validando los criterios de aceptación definidos en las historias de usuario. Estas pruebas incluyeron validación de flujos CRUD, filtros, estados de activación y comportamiento asincrónico.
- **Pruebas de campo**: Se desplegó una versión funcional del sistema en un entorno de pruebas y se realizaron ejercicios simulados con operadores reales en condiciones controladas. Durante estas sesiones se documentaron observaciones, errores y sugerencias de mejora.
- **Encuestas de satisfacción**: Se aplicaron encuestas breves al personal administrativo y operativo que participó en la prueba del sistema. Las preguntas fueron diseñadas para medir percepción de utilidad, facilidad de uso y posibles dificultades técnicas.

Las capacitaciones se debían realizar directamente con funcionarios de la empresa, ubicados en la ciudad de Bogotá, para esto, uno de los desarrolladores del equipo fue seleccionado para ir junto con uno de los ingenieros líderes del proyecto para dictar una clase presentado el programa y cómo funciona, y evidentemente realizar una prueba de campo con la mayor cantidad de funcionarios posible, esto tuvo un resultado bastante positivo y el programa entró en una fase que denominamos "Beta", fase en la que el programa estaría funcionando productivamente pero con una muy alta supervisión.

11.1. Mecanismos de recolección de feedback

El feedback fue recolectado a través de:

- Formularios estructurados en Google Forms
- Reuniones presenciales con actas de retroalimentación
- Observación directa durante pruebas de uso

11.2. Resultados de validación

Los resultados se sistematizaron y permitieron introducir mejoras iterativas en la interfaz, simplificación de flujos de navegación y refinamiento de procesos como la cotización especial. Se documentó una tasa de satisfacción general superior al 85 % entre los usuarios encuestados y se logró un tiempo de respuesta 60 % más rápido en tareas administrativas respecto al proceso manual anterior.

Nota: Se preserva la confidencialidad del contenido específico de las reuniones y formularios según lo acordado con la empresa.

12. Metodología

12.1. Metodología de desarrollo

El desarrollo del proyecto se organizó bajo los principios de la metodología ágil Scrum, esta metodología permite a equipos pequeños de desarrollo ser flexibles y adaptables a la hora de definir y aplicar una variante adecuada de Scrum centrado en las necesidades de un cliente, asegurando la entrega de una solución de calidad que cumpla con los objetivos establecidos [12].

Se garantizó que el producto evolucionara en función de las necesidades de la empresa Aguas de la Sabana.

12.2. Organización del trabajo

El proceso se estructuró en **sprints de dos semanas**, al inicio de los cuales se definían los objetivos y tareas principales, priorizadas con base en los requerimientos funcionales del sistema y las historias de usuario levantadas en la fase de análisis. Estas sesiones de planificación permitieron establecer una hoja de ruta clara para cada iteración y orientar el trabajo hacia entregables específicos.

12.3. Seguimiento y control

Durante cada sprint se realizaron **reuniones de seguimiento tipo Daily Scrum**, con una frecuencia de tres veces por semana y una duración de 15 a 20 minutos. Estas reuniones fueron espacios clave para revisar el progreso de las tareas, resolver dificultades técnicas y realizar ajustes en las prioridades si era necesario, manteniendo un flujo de trabajo constante y enfocado.

El avance de las tareas se registró en listas organizadas por etapas, lo que permitió una visualización del progreso general y el control del cumplimiento de objetivos.

12.4. Evaluación y retroalimentación

Al finalizar cada sprint, se llevaron a cabo sesiones de **Sprint Review**, en las que se presentaban los avances al director del proyecto y, en ciertos casos, al equipo empresarial. Estas presentaciones permitieron validar funcionalidades, recoger observaciones y orientar los siguientes ciclos de trabajo con base en la retroalimentación recibida.

Asimismo, se promovieron espacios de reflexión sobre el proceso, con el fin de identificar oportunidades de mejora en la organización, comunicación y ejecución de tareas.

12.5. Resultados

Como era de esperarse, la aplicación de los principios de Scrum facilitó una gestión efectiva del tiempo, una integración progresiva de funcionalidades y una comunicación fluida con los actores del proyecto. Este enfoque permitió mantener la coherencia técnica, la alineación con los requerimientos y una evolución constante del sistema.

Elemento	**Descripción**
Marco metodológico	Basado en Scrum
Duración de sprints	2 semanas
Planeación de sprint	Reuniones al inicio de cada ciclo para definir objetivos y tareas
Reuniones de seguimiento	3 veces por semana, de 15 a 20 minutos
Sprint review	Presentación de avances al cierre de cada sprint
Reflexión sobre el proceso	Revisión de aprendizajes y buenas prácticas tras cada iteración
Gestión de tareas	Basada en historias de usuario y prioridades técnicas
Interacción con el cliente	A través de reuniones presenciales y virtuales según disponibilidad

Tabla 7: Resumen de la metodología aplicada

12.6. Proceso de capacitación al personal

Con el objetivo de garantizar una adecuada adopción de la solución desarrollada, se implementó un plan de capacitación dirigido a los diferentes perfiles de usuario de la empresa Aguas de la Sabana, diferenciando entre personal administrativo y personal operativo (conductores y auxiliares de recolección).

Metodología de capacitación

Se aplicó una metodología práctica y participativa, estructurada en sesiones breves presenciales. Cada sesión incluyó:

- Explicación general del sistema (componentes, objetivos, beneficios)
- Navegación guiada por la consola web y la aplicación móvil
- Resolución de casos de uso simulados (ej.: crear cliente, registrar aforo, emitir cotización)
- Espacio de preguntas y retroalimentación

Materiales utilizados

Para acompañar el proceso se diseñaron:

- Manual de usuario para la consola web (formato digital e imprimible)
- Guía rápida ilustrada para la app móvil (con capturas de pantalla)
- Videos cortos tipo tutorial (grabados desde reuniones virtuales)

Participación

En total se realizaron entonces tres jornadas de capacitación, dos virtualizadas de las cuales se extrajo la mayor parte de documentación audiovisual y escrita, y una práctica en presencial. A esta última asistieron 2 personas del área administrativa y 12 del área operativa, aunque no todos participaron en la prueba de campo como tal. La asistencia fue registrada mediante listas físicas firmadas en posesión de la empresa cliente y registros digitales.

12.7. Métricas de adopción y mejora del proceso

Posterior al despliegue de la solución en entorno de prueba, se recolectaron indicadores clave de uso y percepción mediante encuestas y observación directa. Los principales resultados fueron:

- **Adopción inicial**: El 100 % del personal capacitado logró utilizar de forma autónoma la aplicación móvil y el sistema web en escenarios simulados.
- **Reducción de tiempos administrativos**: Las tareas de programación de rutas y generación de cotizaciones mostraron una disminución promedio del 60 % en tiempo de ejecución en comparación con el proceso manual.
- **Satisfacción del usuario**: En una encuesta de 5 ítems aplicada a 16 participantes, el 87 % calificó la solución como "útil" o "muy útil", y el 91 % consideró la interfaz "intuitiva".
- **Incidencias técnicas**: Durante las pruebas de campo se registraron 3 incidencias menores (una relacionada con sincronización de datos móviles y dos de validación en formularios), todas corregidas en la siguiente iteración.

Estas métricas fueron importantes para validar que la solución no solo era funcional, sino también aceptada y adaptable por los usuarios finales de la empresa.

13. Conclusiones

El desarrollo del sistema de información para la sistematización de la gestión administrativa y la lógica de negocio de la recolección de desechos en la empresa **Aguas de la Sabana** evidencia la importancia de la aplicación de soluciones tecnológicas como respuesta a problemáticas operativas recurrentes en entornos empresariales. A partir de una metodología de investigación mixta y un enfoque de desarrollo ágil, fue posible comprender las dinámicas internas de la organización, identificar los cuellos de botella en sus procesos y diseñar una solución integral orientada a la eficiencia y la trazabilidad.

La implementación de una arquitectura basada en microservicios, junto con una consola web administrativa y una aplicación móvil para los operadores, permitió automatizar tareas críticas, reducir errores humanos y centralizar la información operativa y administrativa. Todo ello redunda en una mejora significativa de los tiempos de respuesta, la calidad del servicio y el cumplimiento normativo, particularmente en relación con la **normatividad establecida por la CRA**.

13.1. ¿Por qué estas tecnologías?

La elección de las tecnologías utilizadas en este proyecto respondió a criterios de robustez, flexibilidad, compatibilidad con arquitecturas modernas y facilidad de integración. Se utilizaron además herramientas de código abierto y con comunidades activas, lo que asegura mantenimiento, documentación y evolución tecnológica en el tiempo.

- **React:** Elegido por su arquitectura basada en componentes reutilizables, su alta adopción en la industria y su bajo acoplamiento con el backend. Su uso permitió desarrollar una interfaz rápida, modular y de fácil mantenimiento. Se prefirió sobre alternativas como Angular por su menor complejidad inicial, y sobre Vue.js por la disponibilidad de recursos y documentación.

- **Mantine + TanStack** Estas bibliotecas permitieron construir tablas altamente dinámicas, con capacidades de paginación, edición, ordenamiento, filtros avanzados y carga eficiente de datos. Agilizaron la construcción del frontend sin sacrificar la experiencia de usuario.

- **Spring Boot:** Fue seleccionado para el backend por su capacidad de estructurar aplicaciones REST robustas, su integración con estándares de seguridad y su soporte nativo para microservicios, facilitando la división lógica del sistema. Además, es uno de las tecnologías más usadas en los proyectos de la empresa encargada del desarrollo de este proyecto.

- **Dockerización:** El uso de contenedores aseguró portabilidad y desacoplamiento del entorno, permitiendo el despliegue en distintos servidores sin necesidad de configuraciones complejas. También facilitó la colaboración y pruebas entre miembros del equipo.

A diferencia de plataformas comerciales como Epiroc, WasteIQ o GeoWaste, que requieren licenciamiento y adaptación a normativas extranjeras, la solución desarrollada está ajustada a la realidad de las empresas colombianas. Entre sus ventajas se encuentran:

- Criterio relevante sobre la evolución del sistema.
- Adaptación directa al marco regulatorio local.
- Modularidad para ajustarse a futuras necesidades.

Además, con base en las mediciones realizadas durante las pruebas y las encuestas, es posible asegurar que el impacto tecnológico generado por este proyecto hace un aporte significativo en la investigación tanto de las regulaciones sanitarias en Colombia como en la manera en la que se comportaría un sistema que gestiona entregas en puntos de una localidad; en otras palabras, brinda una base o "plantilla" sobre este tipo de lógica de negocio para proyectos futuros.

Respecto al proyecto, por parte del equipo, es prudente decir que hubo:

- Reducción del 60 % en los tiempos de respuesta en tareas administrativas, especialmente en programación de rutas y generación de cotizaciones.
- Incremento en la disponibilidad y autonomía del personal operativo gracias al modo offline de la aplicación móvil.
- Mejora en la trazabilidad de los procesos mediante registros digitales auditables por microservicio.
- Control de servicios no programados. La exigencia de registrar digitalmente todos los aforos ejecutados, incluso los no agendados en una ruta, fortaleció la trazabilidad del servicio y redujo pérdidas asociadas a la informalidad operativa. Esto permitió consolidar un flujo interno más transparente y confiable, sin limitar la flexibilidad en campo.

Todo el trabajo desarrollado permitió evidenciar el valor de los enfoques modulares, reutilizables y escalables en el diseño de sistemas de información, los cuales aseguran una continuidad tecnológica adaptable a los cambios internos y externos de la empresa. La apropiación de la solución por parte del personal, facilitada mediante capacitaciones y documentación, constituye un factor importante para su sostenibilidad y evolución a futuro.

13.2. Contribución académica

En suma, este proyecto constituye una contribución significativa al estudio de la aplicación de tecnologías modernas en contextos reales del sector público. A través del diseño, implementación y validación de un sistema completo, se generan insumos útiles para la academia y futuras intervenciones tecnológicas.

La documentación del proceso de transformación digital en una empresa prestadora de servicios públicos brinda un caso práctico valioso. El desarrollo permite entender cómo integrar la lógica operativa con patrones de diseño y arquitecturas escalables a entornos relacionados a recolecciones y/o entregas a domicilios, necesitados de un sistema de rastreo de elementos y mapeo de rutas.

13.3. Trabajos futuros...

Con el sistema actualmente implementado y en operación, se proyecta una fase de mejora continua que permitirá no solo consolidar las funcionalidades existentes, sino también ampliar las capacidades del sistema en función de nuevas necesidades operativas y avances tecnológicos. La arquitectura modular y escalable adoptada facilita la incorporación progresiva de nuevos módulos sin comprometer la estabilidad del sistema, lo cual representa una oportunidad estratégica para continuar su evolución.

Entre las líneas de trabajo más relevantes se encuentra la integración de tecnologías de geolocalización y rastreo GPS en los vehículos de recolección. Esta funcionalidad permitirá monitorear en tiempo real el recorrido de las rutas asignadas, optimizar la logística de recolección, responder de forma oportuna ante incidentes en el campo y generar trazabilidad completa del servicio prestado. La implementación de este componente implicará el desarrollo de un módulo adicional conectado al sistema actual, así como la definición de interfaces de visualización y análisis para el personal administrativo.

Adicionalmente, se prevé la incorporación de mecanismos de evaluación continua del desempeño operativo, como indicadores de eficiencia por ruta, consumo de combustible, tiempos de recorrido y cumplimiento de programación. Estos elementos enriquecerán la capacidad analítica del sistema y aportarán a la toma de decisiones basadas en datos.

La flexibilidad tecnológica del sistema, junto con el compromiso institucional con la innovación, permiten afirmar que esta solución se encuentra en constante desarrollo y que su impacto se fortalecerá en la medida en que se incorporen nuevas funcionalidades orientadas a la sostenibilidad, la eficiencia operativa y, por supuesto, el cumplimiento normativo.

14. Anexos

14.1. Anexo 1: Esquema completo de base de datos

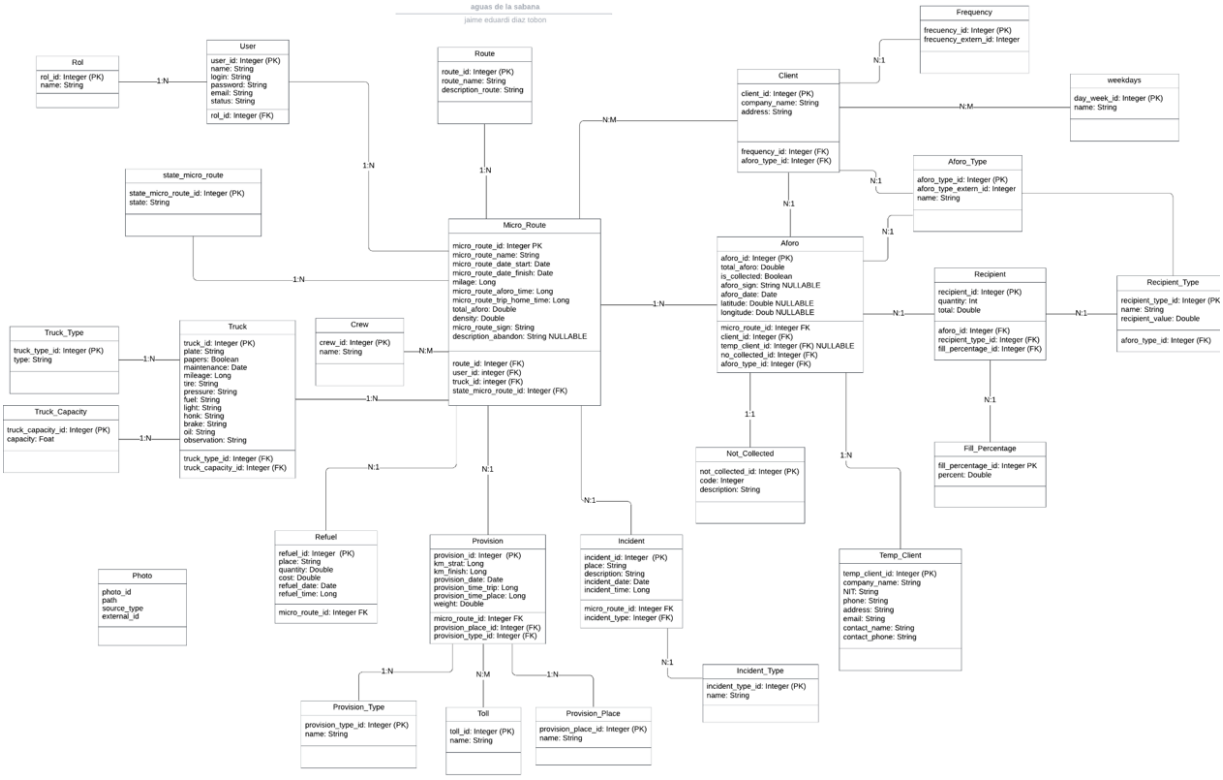


Figura 24: Esquema completo de base de datos

14.2. Anexo 2: Credenciales y primera vista



Figura 25: Login y primera vista

14.3. Anexo 3: Validación mecanica del vehiculo



Figura 26: Validación mecanica del vehiculo

14.4. Anexo 4: Toma de un aforo y reportes

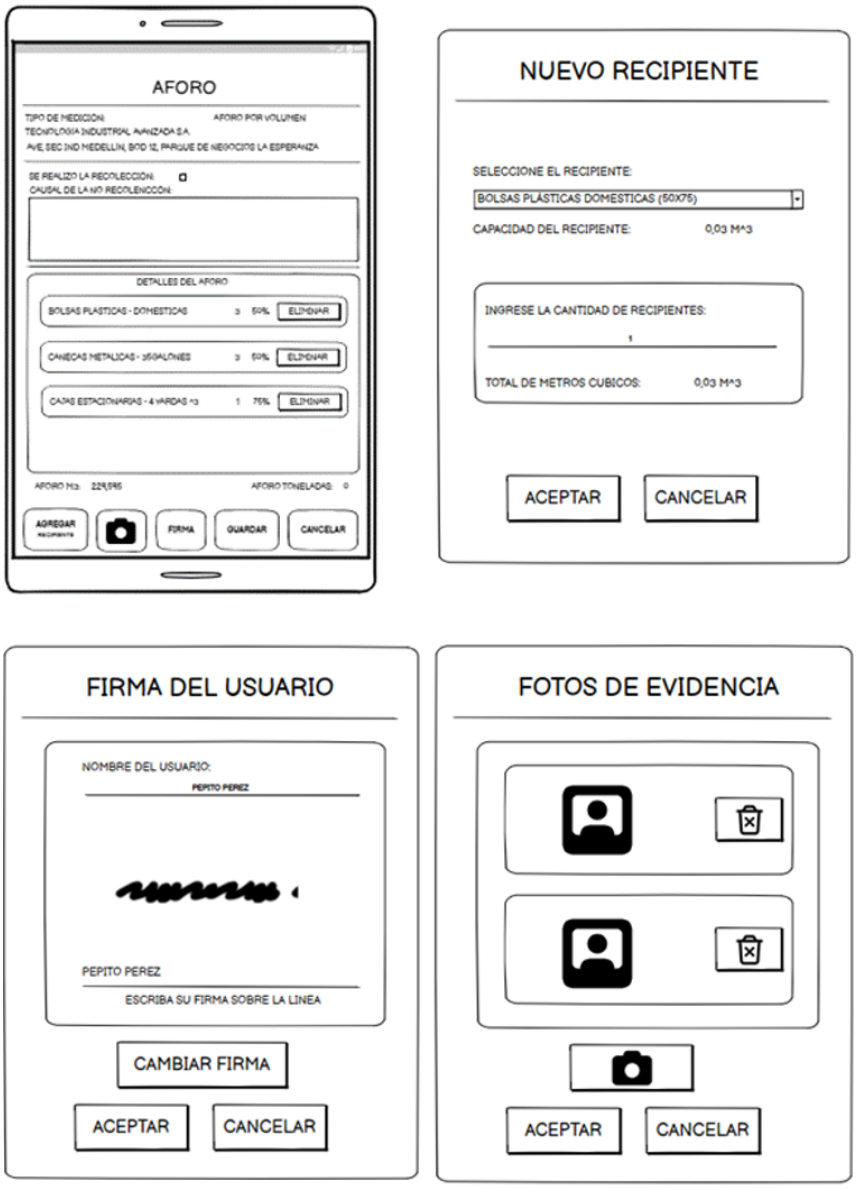


Figura 27: Validación mecanica del vehiculo

14.5. Anexo 5: Inicio de recorrido

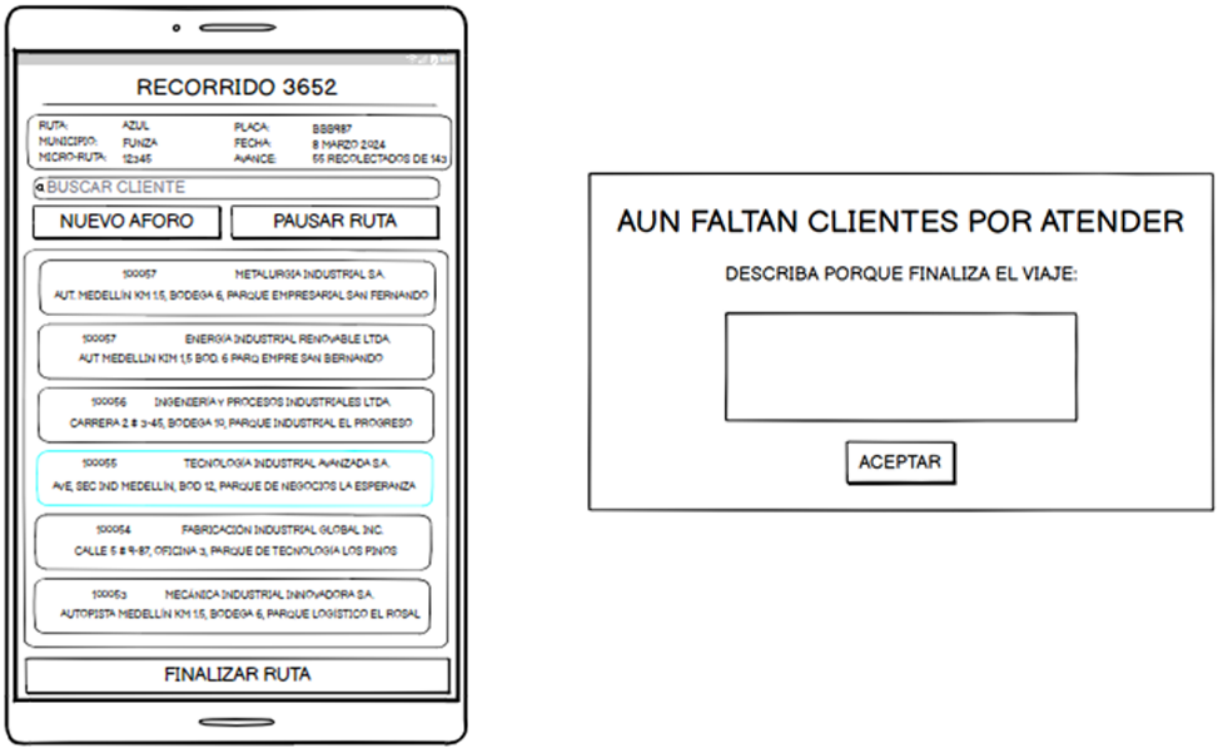


Figura 28: incio de recorrido

14.6. Anexo 6: Pausa de ruta

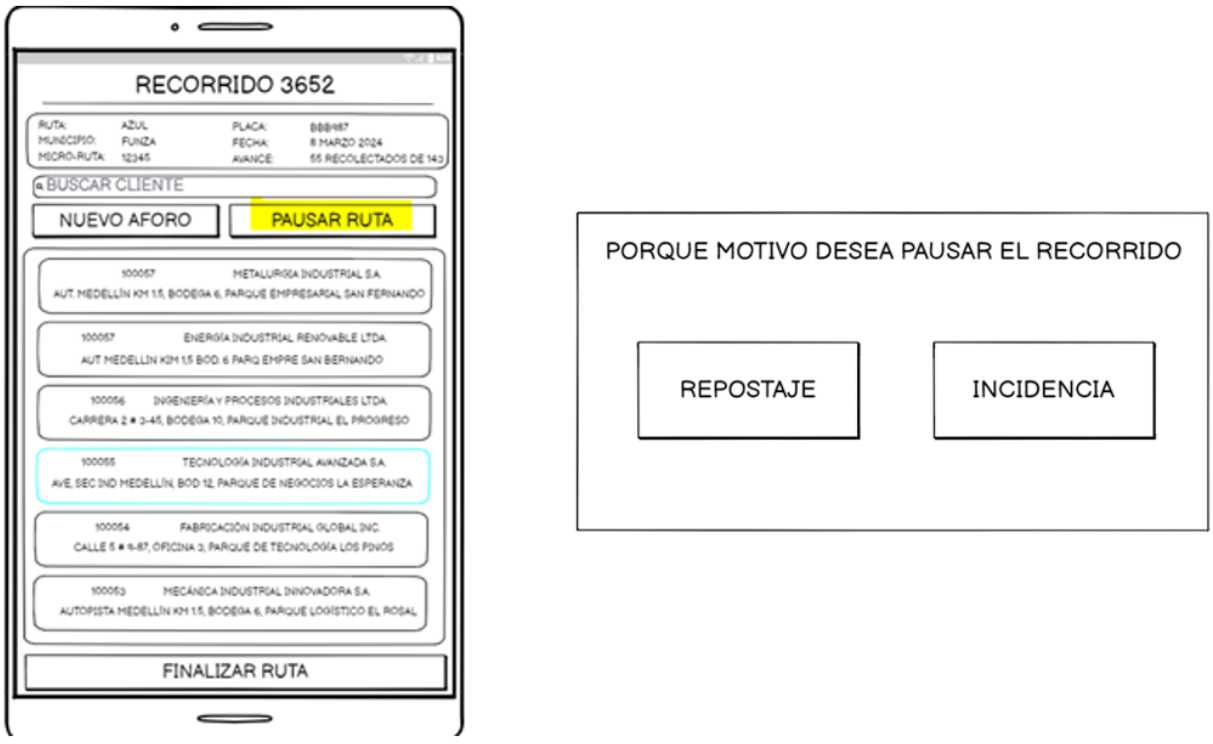


Figura 29: Pausa de ruta

14.7. Anexo 6: Repostaje



Figura 30: Repostaje

14.8. Anexo 7: Registro de incidencias



Figura 31: Registro de incidencias

14.9. Anexo 8: Finalizacion de ruta



Figura 32: Final de ruta

14.10. Anexo 9: Inicio de viaje a relleno sanitario



Figura 33: Viaje hacia relleno sanitario

14.11. Anexo 10: Registro de ingreso a relleno



Figura 34: Registro de ingreso y de peso de disposición final

14.12. Anexo 11: Finalizacion y retorno a base



Figura 35: Finalizacion y retorno a base

15. Referencias

[1] A. de la Sabana de Bogotá, "Sobre nosotros: www.asb.com.co/nosotros/," Consultado el 08/06/24.

[2] "Comisión de regulación de agua potable y saneamiento básico, www.cra.gov.co/," Consultado el 21/06/24.

[3] C. y. T. Ministerio de Vivienda, "Guía para la formulación de planes de gestión integral de residuos sólidos (pgirs): https://www.minvivienda.gov.co/sites/default/files/documentos/guia-pgirs.pdf," tech. rep., Consultado el 23/05/25.

[4] U. S. E. P. A. (EPA), "Advancing sustainable materials management: 2020 fact sheet": www.epa.gov/facts-and-figures-about-materials-waste-and-recycling," tech. rep., Consultado el 23/05/25.

[5] Laudon, "Sistemas de información gerencial: https://dspace.itsjapon.edu.ec/," 2004. Consultado el 09/06/24.

[6] S. Newman, ""building microservices". disponible en https://book.northwind.ir/bookfiles/building-microservices/building.microservices.pdf," tech. rep., Consultado el 23/05/25.

[7] V. L. D. Taibi and C. Pahl, ""architectural patterns for microservices: A systematic mapping study", in proceedings of the 8th international conference on cloud computing and services science. 2018. disponible en: www.researchgate.net/publication/323960272_architectural_patterns_for_microservices_a_systematic_mapping_study," tech. rep., Consultado el 24/05/25.

[8] Ecoembes, ""plataforma de gestión de reciclaje y logística de residuos en españa", www.ecoembes.com," tech. rep., Consultado el 26/05/25.

[9] SmartWaste, ""smartwaste environmental management software for construction", www.smartwaste.co.uk," tech. rep., Consultado el 26/05/25.

[10] O. M. de la Salud, "www.who.int/es," Consultado el 13/06/24.

[11] D. Zornosa, "Método de análisis y selección de soluciones tecnológicas de seguridad informática para el sector empresarial y las organizaciones," Master's thesis, Universidad Santo Tomás, Consultado el 13/06/24.

[12] R. L. and J. N. S, "Scrum software development process for small teams," *IEEE*, 2000. Consultado el 2/07/24.