Trabajo de Fin de Grado

Software de gestión de recogida de residuos

Recycling Planner

Juan Marrero Domínguez

La Laguna, 12 de julio de 2023

D. **Christopher Expósito Izquierdo**, con N.I.F. 78.851.649-J profesor Ayudante Doctor adscrito al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna como tutor

D. **Israel López Plata**, con N.I.F. 42.193.801-W profesor Ayudante Doctor adscrito al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

"*Software de gestión de recogida de residuos, Recycling Planner*"

ha sido realizada bajo su dirección por Don **Juan Marrero Domínguez**, con N.I.F. 51.154.160-K.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 12 de julio de 2023

Agradecimientos

Quiero expresar mis más sinceros agradecimientos a mi tutores del proyecto, Christopher Expósito Izquierdo e Israel López Plata por su inmensa dedicación, orientación y apoyo a lo largo de todo el proceso de mi TFG. Su ayuda y conocimiento han sido fundamentales para el desarrollo exitoso de este trabajo, y estoy profundamente agradecido por su paciencia y compromiso.

Las correcciones y el contacto ha sido constante durante todo el proceso de estos meses. Esta comunicación tan frecuente ha hecho que me sienta integrado y atendido en todo momento. Esto resulta en una mayor motivación e ilusión a la hora de llevar a cabo un proyecto tan importante como este.

Resumen

En el desarrollo de este Trabajo de Fin de Grado se abordan varios aspectos clave para abordar el desarrollo de un software que facilita la toma de decisiones en el ámbito de recogida de residuos. Esta herramienta tiene los medios para agilizar y optimizar tanto tiempo como recursos a los encargados del sector.

Se da especial importancia a la toma de decisiones por parte del usuario, brindando las herramientas necesarias para llevar a cabo un análisis detallado de la situación actual del entorno, permitiendo identificar requisitos y problemas, así como definir las soluciones adecuadas.

Se desarrolla una API que se encarga de gestionar las entidades principales de la lógica de negocio como almacenes, camiones, puntos de recogida y contenedores. Se detalla sobre las buenas prácticas de desarrollo de API y la utilización de una arquitectura hexagonal para modularizar y permitir que sea más escalable. Esta API es consultada mediante un frontend y opera sobre una base de datos no relacional, todo ello formando una aplicación Full Stack.

Palabras clave: Gestión de residuos, Planificación de rutas, Optimización, Aplicación Full-Stack

Abstract

In the development of this project, several key aspects are addressed to approach the development of a software that eases decision making in the field of waste collection. This tool seeks to optimize both time and resources in this sector.

Special importance has been given to decision making by the user, providing the necessary tools to carry out a detailed analysis of the current situation of the environment, allowing to identify requirements and problems, and define appropriate solutions.

An API has been developed, whose main goal is to manage the main business logic entities such as warehouses, trucks, pickup points and containers. It is detailed on good API development practices and the use of a hexagonal architecture to modularize and allow it to be more scalable. This API will be queried through a frontend and will operate on a non-relational database, all composing a Full Stack application.

Keywords: *Waste management, Route planning, Optimization, Full-Stack application*

Índice general

1. Introducción **1**
1.1. Estado de la recogida de residuos de Canarias . . . 3
1.2. Objetivos del trabajo y organización de la memoria . . . 4

2. Motivación y estado del arte **5**
2.1. Estado del arte . . . 5
2.1.1. ACTAIS Waste . . . 6
2.1.2. SeintoSOFT . . . 6
2.1.3. HSE Tools . . . 7
2.1.4. Fleetio . . . 8
2.2. Características de Recycling Planner . . . 9

3. Tecnologías y arquitectura del software **11**
3.1. Arquitectura empleada . . . 11
3.1.1. Principios de la arquitectura Hexagonal . . . 12
3.1.2. ¿Por qué la arquitectura hexagonal? . . . 12
3.1.3. Comunicación entre componentes . . . 13
3.2. Frontend . . . 14
3.2.1. Tecnologías empleadas . . . 14
3.3. Backend . . . 15
3.3.1. Objetos valor . . . 16
3.3.2. Tecnologías empleadas . . . 17
3.3.3. API REST . . . 18
3.4. Base de Datos . . . 18
3.4.1. MongoDB . . . 19
3.5. Productor de contenedores . . . 19
3.6. Tecnologías auxiliares . . . 20
3.6.1. Docker . . . 20
3.6.2. Git & Github . . . 20

4. Funcionamiento **21**
4.1. Gestión de contenedores y puntos de recogida . . . 22
4.1.1. Registrar nuevos contenedores y puntos de recogida . . . 22
4.1.2. Inspeccionar un punto de recogida o contenedor . . . 23
4.1.3. Eliminar contenedores y puntos de recogida . . . 23
4.2. Gestión de almacenes . . . 24
4.3. Gestión de camiones . . . 24
4.4. Funcionamiento del algoritmo . . . 25
4.5. Visualizador de soluciones planificadas . . . 27

4.6. Cuadro de mandos 30

5. Experimentación **32**
5.1. Operaciones CRUD 32
5.1.1. Creación de entidades 32
5.1.2. Lectura de entidades 33
5.1.3. Actualización de entidades 33
5.1.4. Eliminación de entidades 34
5.2. Cuadro de mandos 34
5.3. Lanzamiento del algoritmo 35
5.3.1. Recopilación de datos 35
5.3.2. Ejecución del algoritmo 36
5.3.3. Visualización de soluciones 37
5.4. Traducción 38
5.5. Productor de contenedores 39

6. Presupuesto **41**
6.1. Análisis y Diseño 41
6.2. Desarrollo 41
6.3. Pruebas y Calidad 42
6.4. Implementación del algoritmo y versión final 42
6.5. Redacción de memoria 42
6.6. Presupuesto final 42

7. Conclusiones y líneas de trabajo futuras **44**

8. Conclusions and further research **46**

Bibliografía **47**

Índice de figuras

1.1. Histograma de reciclaje de plástico realizado por Statista 2

2.1. Pantalla principal de ACTAIS Waste 6
2.2. Pantalla principal del software de SenitoSOFT 7
2.3. Pantalla principal de HS3 Tools 8
2.4. UI de Fleetio 9

3.1. Esquema de comunicación en la aplicación 11
3.2. Esquema de una arquitectura hexagonal 12
3.3. Esquema de las entidades principales 16

4.1. Resumen de los casos de uso de la aplicación 21
4.2. Pantalla principal para la gestión de contenedores y puntos de recogida . . 22
4.3. Formularios de creación para puntos de recogida y contenedores 23
4.4. Información del punto de recogida "Los Cancajos" 23
4.5. Pantalla principal para la gestión de almacenes 24
4.6. Pantalla principal para la gestión de camiones 25
4.7. Información de uno de los camiones 25
4.8. Formulario lanzamiento del algoritmo 26
4.9. Pantalla de visualización de soluciones 27
4.10 Valores generales de una solución 28
4.11 Valores por día de una solución 28
4.12 Valores por ruta de una solución 29
4.13 Ruta planificada 29
4.14 KPI’s y gráficos del cuadro de mandos 30
4.15 Mapa que contenedores con su estado 31

5.1. Formulario para crear un almacén 32
5.2. Almacén creado 33
5.3. Información del recién creado almacén 33
5.4. Formulario para actualizar los datos de un almacén 34
5.5. Cuadro de mandos en inglés 35
5.6. Recopilación de datos para el algoritmo 36
5.7. Estados de la ejecución del algoritmo 36
5.8. Ejecución exitosa 37
5.9. Resultado ejemplo: rutas 37
5.10 Resultado ejemplo: días 38
5.11 Cuadro de mandos en español 38
5.12 Fichero .yml para el productor 39
5.13 Productor emitiendo actualizaciones 39

5.14 Backend operando según lo que recibe del productor 40

Índice de tablas

6.1. Tabla de coste de tiempo estimado 42

Capítulo 1

Introducción

En la actualidad, el cuidado del medio ambiente es una preocupación creciente y una de las prioridades en el ámbito internacional, por tanto, también en las políticas públicas. La gestión de recogida de residuos es uno de los aspectos más importantes en este ámbito, y cada vez son más las entidades públicas que buscan optimizar la recogida de los mismos y su posterior tratamiento. Con todo ello, el reciclaje de materiales se ha convertido en una necesidad para reducir el impacto medioambiental y contribuir al desarrollo sostenible.

Actualmente en España se generan en torno a 137,8 millones de toneladas de residuos, de los cuales el 48,3 % terminaron en el vertedero; frente al 38,7 % que se recicló, el 10 % que se reutilizó y el 3 % que acabó incinerado[1], tal y como se observa en la Figura 1.1. Estos datos nos deja en un puesto respetable, estando entre los 10 países que más recicla plástico de la UE, pero sigue habiendo un gran margen de mejora[2] así como otros materiales que pueden ser reciclados en los que podemos mejorar.

Debido a la importancia de la gestión de residuos en las sociedades desarrolladas, la Unión Europea ha fijado un objetivo de reciclaje del 55 % de los residuos municipales para 2025, aumentando al 60 % en 2030 y al 65 % en 2035.[3]. Países como Alemania ya se están acercando a estos objetivos, que en 2019 consiguió que el 70 % de sus residuos fueran reciclables. Otros como Bélgica o Países Bajos también consiguieron una tasa de reciclaje del 80 % ese año.[4].

La optimización de la recogida de residuos puede aportar beneficios significativos tanto desde el punto de vista medioambiental como económico. Al reducir la cantidad de residuos recogidos de forma innecesaria, se disminuye la emisión de gases contaminantes y se ahorra en combustible y tiempo. Esto se ve reflejado en estudios como el que se hizo en una pequeña ciudad de la India [1]. Este estudio demostró que una estrategia efectiva para optimizar la recogida de residuos es reducir la distancia de viaje entre las áreas de eliminación y las ubicaciones de los contenedores de basura. Al hacer esto, se puede lograr una recolección más eficiente de los residuos, lo que puede generar ahorros

[1] https://www.elconfidencial.com/medioambiente/ciudad/2021-08-25/las-cifras-del-reciclaje-en-espana
3241930

[2] https://www.envaproblog.com/post/espa%C3%B1a-en-el-top-5-de-pa%C3%
ADses-europeos-que-recicla-m%C3%A1s-pl%C3%A1stico

[3] https://www.miteco.gob.es/es/ceneam/carpeta-informativa-del-ceneam/novedades/
objetivos-ue-economia-circular.aspx

[4] https://deutsche-recycling.es/blog/tasa-de-reciclaje-en-alemania-comparacion-con-otros-paises-eu

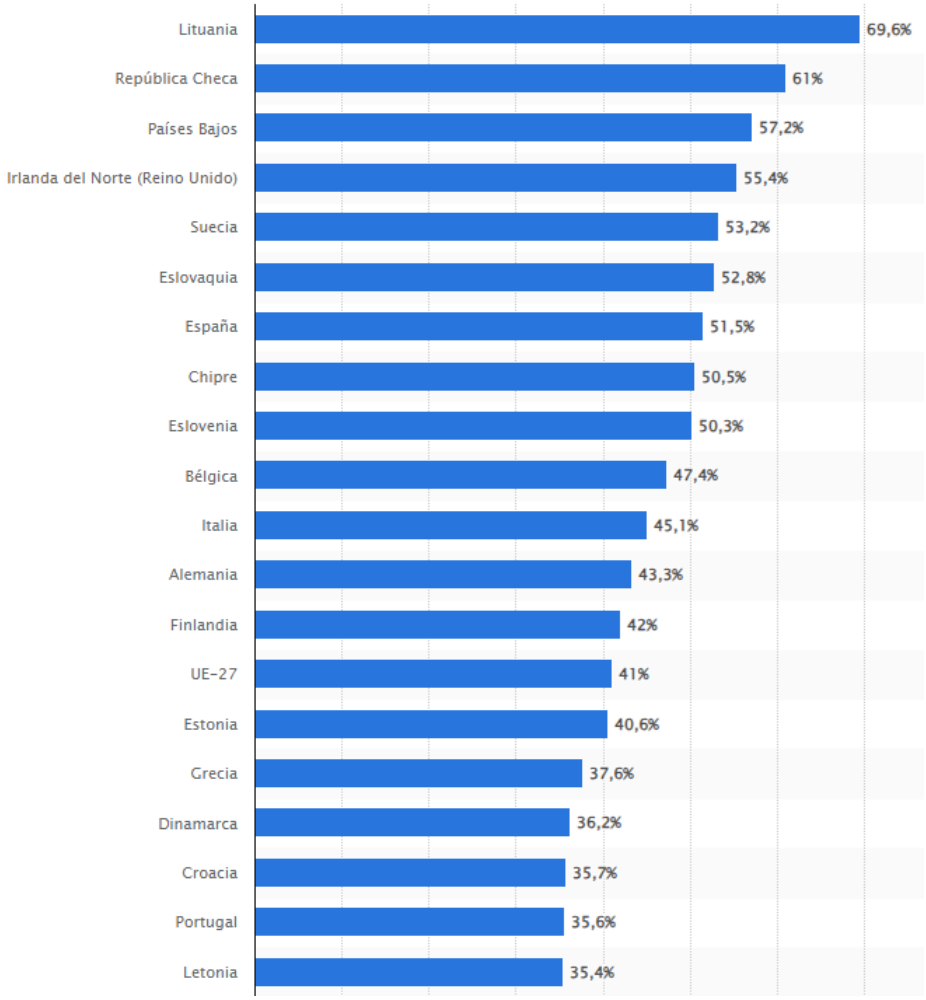


Figura 1.1: Histograma de reciclaje de plástico realizado por Statista

significativos en términos de combustible y tiempo.

En este contexto, la aplicación Recycling Planner se presenta como una herramienta fundamental para avanzar en la gestión de los residuos de forma sostenible y, sobre todo, eficiente. Cada proceso optimizado ayuda a un mejor uso de los recursos disponibles, permitiendo una mejor recogida de residuos empleando un menor número de recursos. De igual manera, surge la necesidad de contar con herramientas que permitan gestionar de manera eficiente la recogida de los residuos. La aplicación Recycling Planner opta por una solución innovadora que pretende mejorar la gestión de la recogida de los residuos, ofreciendo una herramienta para optimizar los procesos de recogida de basura gracias al tratamiento de los datos de los que dispone.

Este TFG tiene como objetivo el desarrollo, justificación y explicación de la aplicación full-stack Recycling Planner. Esta aplicación gestiona y almacena los datos necesarios para la correcta recogida de residuos. Con estos, se pretende optimizar los procesos de recogida de basura gracias a los datos con los que se va a operar. La aplicación implementada tiene como principal objetivo optimizar la ruta de recogida de basura de los camiones encargados de esta tarea. Para ello, se utilizan variables como la distancia, el porcentaje de llenado de los cubos y la capacidad de carga de los camiones. Gracias a esta información, es posible tomar decisiones sobre qué contenedor es crucial que

sea recogido y cuáles pueden esperar, ya que algunos se llenan más lentamente que otros, dependiendo de su ubicación. De esta manera, se pretende reducir el tiempo y el combustible utilizado en la recogida de residuos y, a su vez, disminuir las emisiones de gases contaminantes.

El uso de esta aplicación se orienta a un Sistema de Ayuda a la Decisión (DSS en inglés), lo que permite a los gestores tomar decisiones de manera más eficiente y basadas en datos reales. Para profundizar en el tema de los Sistemas de Ayuda a la Decisión, el estudio ”A Critical Review of DSS Foundational Articles” [2] destaca la relevancia tanto para la investigación actual como para futuros estudios, los DSS. Además, proporciona una referencia para categorizar la investigación en DSS. Con la información proporcionada por la aplicación se pueden planificar las rutas de recogida de forma más efectiva y, a su vez, optimizar los recursos disponibles para esta tarea.

La plataforma brinda una interfaz muy clara en la que los usuarios ven claramente el estado actual de la situación gracias a gráficos y representaciones de los datos. Por otro lado, cuenta con un mapa donde se muestra la ruta óptima que los camiones han de tomar, y de esta manera recoger sólo los cubos indispensables para gestionar efectivamente tanto combustible como tiempo.

1.1. Estado de la recogida de residuos de Canarias

Actualmente Canarias padece de una situación descuidada en cuanto a la recogida de residuos. En Canarias, cada camión gasta alrededor de 1000€ [5] en combustible para realizar las recogidas. Por los estudios y argumentos mencionados previamente, y dado el coste diario que implican las rutas, es necesario estudiar si los viajes de los camiones son demasiados, para así valorar la posibilidad de mejora y optimización en las rutas seguidas por los camiones con el fin de llevar a cabo una reducción en los costes.

Uno de los casos que más se pretende evitar es aquel en los que un camión recoge un contenedor que no está ni al 40 % de su capacidad total. Dicho camión podría haberse ahorrado ese recorrido extra. Además, estas zonas suelen ser las más alejadas de las zonas más pobladas, suponiendo que los viajes menos necesarios son los más costosos en tiempo y combustible.

La necesidad que se plantea con esta aplicación es la capacidad de monitorizar en tiempo real el estado de los contenedores y los camiones encargados de su recogida. Gracias a tratar los datos de manera inteligente, se espera poder tomar mejores decisiones a la hora de organizar los recorridos que cada camión debe hacer. Por ejemplo, si observamos que un pueblo alejado en la isla de La Palma solo llena 3 veces el 100 % de la capacidad de sus contenedores a la semana, se puede empezar a plantear cuestiones de hacer 4 viajes a la semana, en lugar de uno diario, con la consecuente optimización de las rutas de recogida.

[5] https://www.eldia.es/2014-02-10/PALMA/1-Cada-camion-basura-consume-euros-mes-combustible.htm

1.2. Objetivos del trabajo y organización de la memoria

El objetivo principal de este Trabajo de Fin de Grado es documentar y presentar el desarrollo de una aplicación de gestión y optimización de la recogida de residuos. Se ha creado un sistema que facilite la gestión eficiente de los residuos, permitiendo el seguimiento y control de su recolección, clasificación, transporte y muestras de datos y rutas optimizados.

El resto de los capítulos están organizados de la siguiente manera:

- *Capítulo 2. Motivación y estado del arte.* Indica cómo se encuentra la gestión de residuos en Canarias, qué aplicaciones existen sobre dicha temática y una justificación más extendida sobre la necesidad de la aplicación.
- *Capítulo 3. Tecnologías y arquitectura del software.* Se comenta la arquitectura de software sobre la que se construye la aplicación, así como su stack tecnológico correspondiente explicado de manera técnica.
- *Capítulo 4. Funcionamiento.* Muestra cómo funciona la aplicación. Contiene casos de uso, *workflows* de la plataforma y funcionalidades.
- *Capítulo 5. Experimentación.* Se realiza una experimentación con la que se comprueba que el funcionamiento de la aplicación es el esperado.
- *Capítulo 6. Presupuesto.* Expone el presupuesto total del proyecto.
- *Capítulos 7 y 8. Conclusiones y líneas de trabajo futuras.* Se aportan una serie de conclusiones y visiones de trabajo futuras con respecto a como avanzar el proyecto y seguir completándolo. El capítulo 8 es en completo inglés.

Capítulo 2

Motivación y estado del arte

2.1. Estado del arte

El tratamiento de residuos derivados de los procesos de fabricación, construcción, demolición e industriales en general tienen una normativa cada vez más estricta. La sociedad y los organismos públicos exigen una gestión medioambiental adecuada que cumpla con los requisitos establecidos. Una de las normativas más importantes es La Legislación de la UE sobre la gestión de residuos[1]. Entre los puntos clave se pueden destacar:

- Las autoridades nacionales competentes deben establecer planes de gestión de residuos y programas de prevención de residuos.
- La gestión de los residuos debe realizarse sin crear riesgos para el agua, el aire, el suelo, las plantas o los animales, sin provocar incomodidades por el ruido o los olores y sin atentar contra los paisajes ni contra los lugares de especial interés.

También la ley de Residuos y Suelos está fuertemente comprometida a la reducción notable de los suelos contaminados en España [2] en los años venideros, como fechas clave 2025 y 2030. Nuevamente, destacan los siguientes puntos clave:

- Las autoridades competentes deben adoptar las medidas necesarias para asegurar que la gestión de los residuos se realice sin poner en peligro la salud humana y sin dañar al medio ambiente.
- De acuerdo con el principio «quien contamina paga», los costes relativos a la gestión de los residuos, incluidos los costes correspondientes a la infraestructura necesaria y a su funcionamiento, así como los costes relativos a los impactos medioambientales y en particular los de las emisiones de gases de efecto invernadero, tendrán que ser sufragados por el productor inicial de residuos, por el poseedor actual o por el anterior poseedor de residuos
- Establece una jerarquía para actuar según este orden de prioridad:
    - Prevención.
    - Reutilización.

[1] https://eur-lex.europa.eu/ES/legal-content/summary/eu-waste-management-law.html
[2] https://www.boe.es/buscar/act.php?id=BOE-A-2022-5809

- Reciclado.
- Eliminación del residuo.

En el ámbito de la gestión de residuos, existen diversas aplicaciones que han abordado esta temática mediante diferentes aproximaciones. A continuación se exponen las 3 plataformas software más populares que cumplen con estos requerimientos.

2.1.1. ACTAIS Waste

Software [3] desarrollado por EcoComputer [4] para la gestión de puntos limpios, ecoparques, deixalleries, garbigunes y puntos verdes. Consiste en una aplicación, con integracion en bases de datos existentes que afirma ser modular y escalable. ACTAIS Waste es una referencia a nivel nacional, con más de 170 instalaciones digitalizadas.En concreto, en la Comunidad Valenciana el sistema ACTAIS Waste se utiliza ya en la digitalización de las redes de ecoparques de 5 Consorcios (COR-V5, CRiV-V4, Terra-A2, CREA-A3, Baix Vinalopó-A5) y 2 Ayuntamientos (Elche y Castellón)

Entre sus características más destacables se encuentran una monitorización a tiempo real de los datos, cumplimiento de las normativas vigentes y una función de trazabilidad de entradas y salidas. En la Figura 2.1 se muestra un ejemplo de la interfaz de usuario de ACTAIS Waste.

Figura 2.1: Pantalla principal de ACTAIS Waste

2.1.2. SeintoSOFT

SeintoSOFT [5]es una empresa que cuenta con software de propósito general y orientado a la gestión y toma de decisiones empresariales de varios nichos. No obstante dentro de uno de sus módulos más específicos se encuentra uno para la gestión de residuos, como el que se muestra en la Figura 2.2.

[3]https://www.actaiswaste.com
[4]https://www.ecocomputer.com/index.php/es/
[5]https://seintosoft.com/modulos/software-gestion-residuos-programa-erp

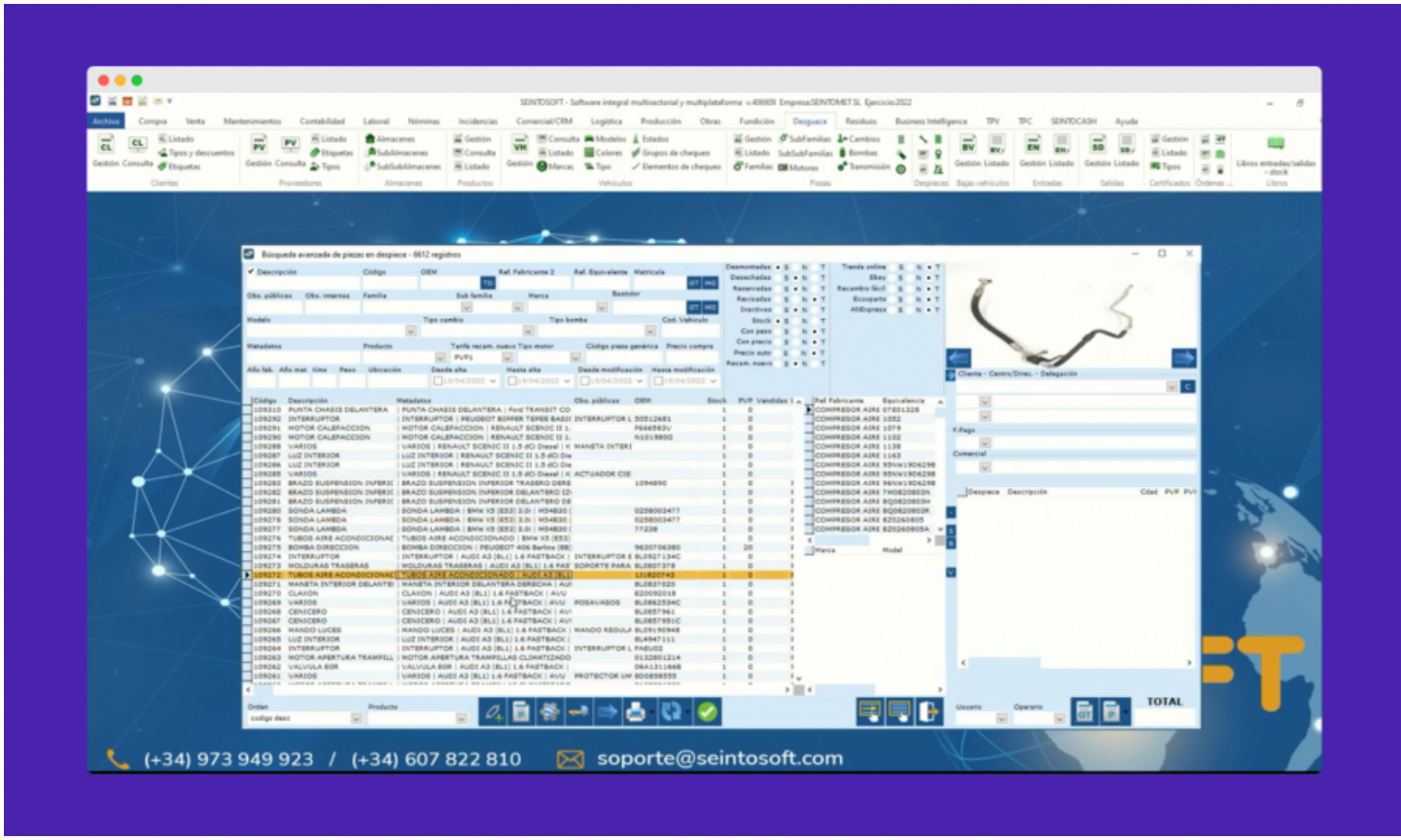


Figura 2.2: Pantalla principal del software de SenitoSOFT

Este software proporciona una solución integral para abordar todos los aspectos relacionados con la gestión eficiente de los residuos. Desde la planificación y seguimiento, la generación de informes, hasta funcionalidades robustas para optimizar los procesos y garantizar un manejo adecuado de los residuos. Una de las características destacadas de SeintoSOFT es su enfoque en la automatización de procesos. A través de la implementación de flujos de trabajo automatizados, el software agiliza las tareas y minimiza los errores manuales, lo que resulta en una mayor eficiencia operativa. La interfaz intuitiva del software de SeintoSOFT facilita su uso y aprendizaje, lo que permite a los usuarios aprovechar al máximo todas las funcionalidades disponibles.

2.1.3. HSE Tools

La empresa Grupo ESG Innova[6] ofrece un software [7] de gestión inteligente del flujo de los residuos. Este software permite a las empresas llevar a cabo un seguimiento eficiente y sistemático de sus actividades relacionadas con los residuos y evaluar su impacto ambiental. El software de gestión de residuos de HSE Tool brinda una plataforma integral para la planificación, seguimiento y control de los procesos de gestión de residuos. Permite gestionar de manera efectiva la generación, recolección, transporte, tratamiento y disposición final de los residuos. Una de las características destacadas de HSE Tool es su enfoque en la generación de informes y análisis de datos. El software proporciona herramientas para recopilar y analizar información relevante sobre los residuos generados. Esto permite a las empresas tomar decisiones informadas y establecer estrategias para

[6]https://www.linkedin.com/company/software-hse/
[7]https://hse.software/gestion-de-residuos/

mejorar la gestión de residuos y reducir su impacto ambiental. En la Figura 2.3 se muestra la interfaz de usuario de la aplicación H3S Tools.

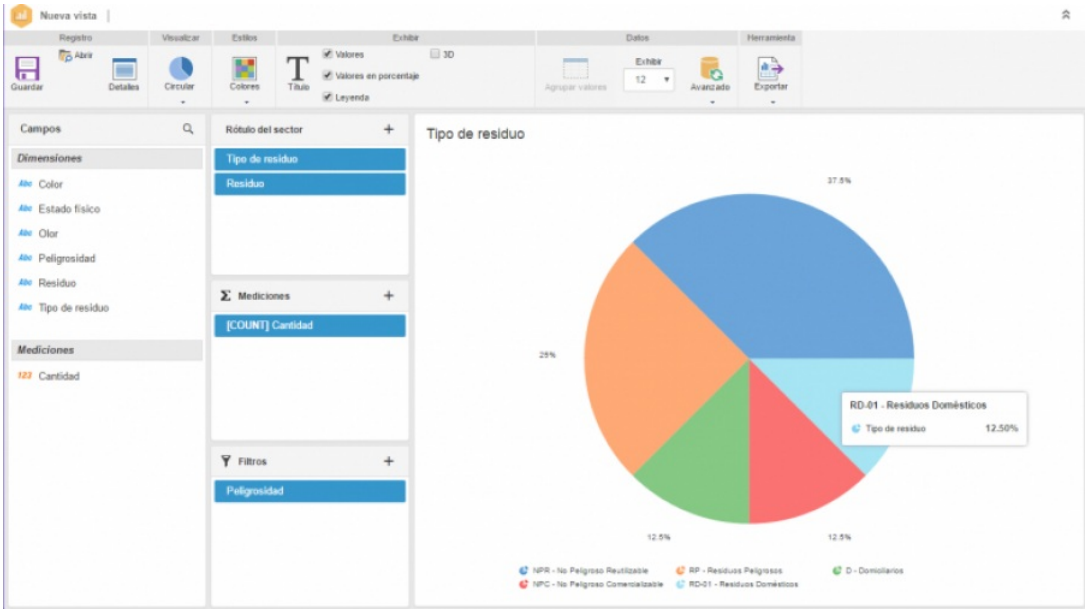


Figura 2.3: Pantalla principal de HS3 Tools

2.1.4. Fleetio

Fleetio [8] es una aplicación de una empresa estadounidense de gestión de flotas diseñada para ayudar a empresas a mantener sus vehículos en buen estado y optimizar su rendimiento. Ofrece una amplia variedad de características y funcionalidades para ayudar a los administradores de flotas a mantenerse organizados y tomar decisiones informadas. Su interfaz se puede observar en la Figura 2.4.

[8] https://www.fleetio.com/

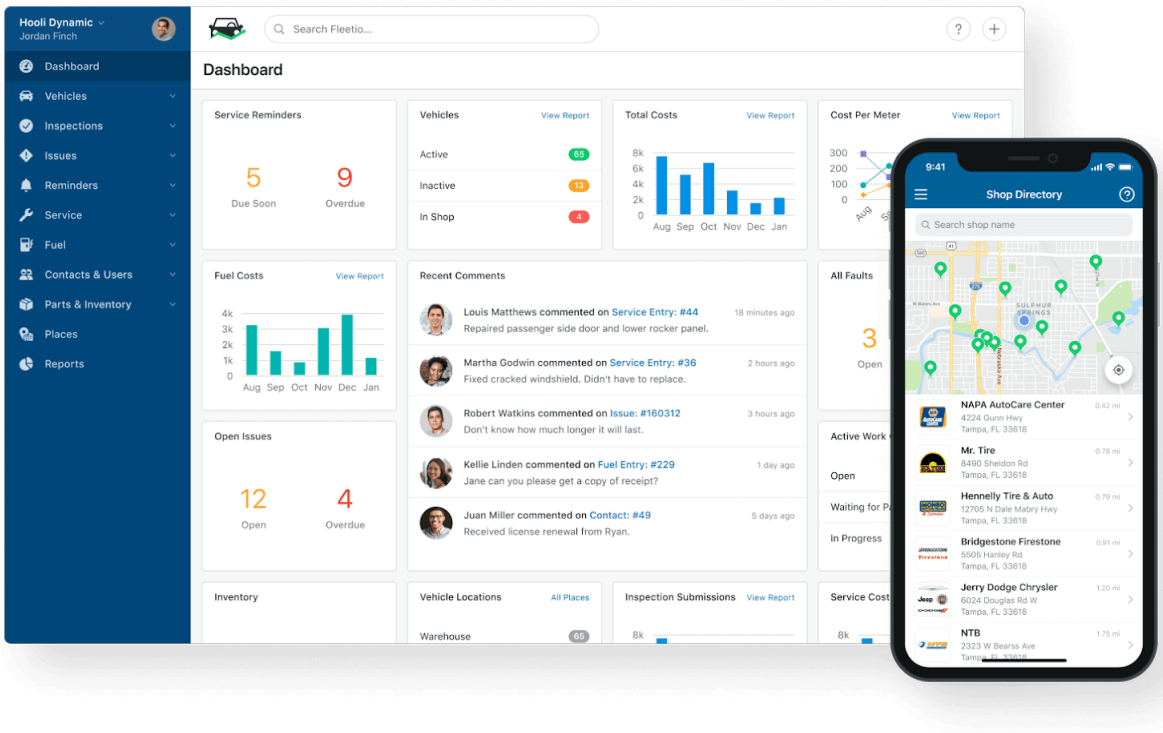


Figura 2.4: UI de Fleetio

Una de las características principales de Fleetio es su sistema de seguimiento de vehículos. La aplicación utiliza GPS para mostrar la ubicación de los vehículos en tiempo real y proporcionar datos sobre su rendimiento. También se puede hacer seguimiento del historial de mantenimiento de cada vehículo, y controlar dichos mantenimientos mediante un sistema de alertas.

Otra funcionalidad importante de Fleetio es su sistema de gestión de combustible. La aplicación te permite registrar el consumo de combustible de cada vehículo y hacer seguimiento del costo para detectar cualquier problema.

2.2. Características de Recycling Planner

El software propuesto representa una innovación en la gestión de residuos gracias a una serie de funcionalidades novedosas que no se encuentran actualmente en el mercado.

Una de estas innovaciones es la implementación de mapas que permiten la monitorización en tiempo real de los datos relacionados con la gestión de residuos, proporcionando una herramienta valiosa para la toma de decisiones.

Además, integra algoritmos de optimización de rutas que permiten la mejora en la gestión de residuos, lo que conlleva a una mayor eficiencia en el proceso. Así mismo, se ha diseñado una planificación a corto, medio y largo plazo para la gestión de residuos, lo que permite una mejor organización y un uso más efectivo de los recursos. Esto supone una ayuda real para los trabajadores encargados de la recogida de residuos, lo que optimiza y facilita su labor en el proceso.

En líneas generales, la plataforma es una solución integral que aborda los principales desafíos de la gestión de residuos, ofreciendo una serie de herramientas novedosas que contribuyen a una gestión más eficiente y sostenible de los residuos.

Capítulo 3

Tecnologías y arquitectura del software

En este capítulo, se muestra en diferentes secciones y apartados los módulos de la aplicación (con sus respectivas tecnologías) y la arquitectura empleada para el desarrollo de esta aplicación.

3.1. Arquitectura empleada

En el proyecto se va a emplear una arquitectura hexagonal [1] que divide la aplicación en 3 componentes principales: frontend, backend y Base de Datos. Además, hay dos componentes extra que se comunican con el backend exclusivamente. Estos componentes son el Algoritmo, y un productor de contenedores para hacer simulaciones. Estos componentes se intercomunican mediante *adapters* y se mantendrán aislados entre sí, pero su adhesión da forma a la aplicación. La Figura 3.1 muestra un esquema de los componentes que intervienen en la aplicación.

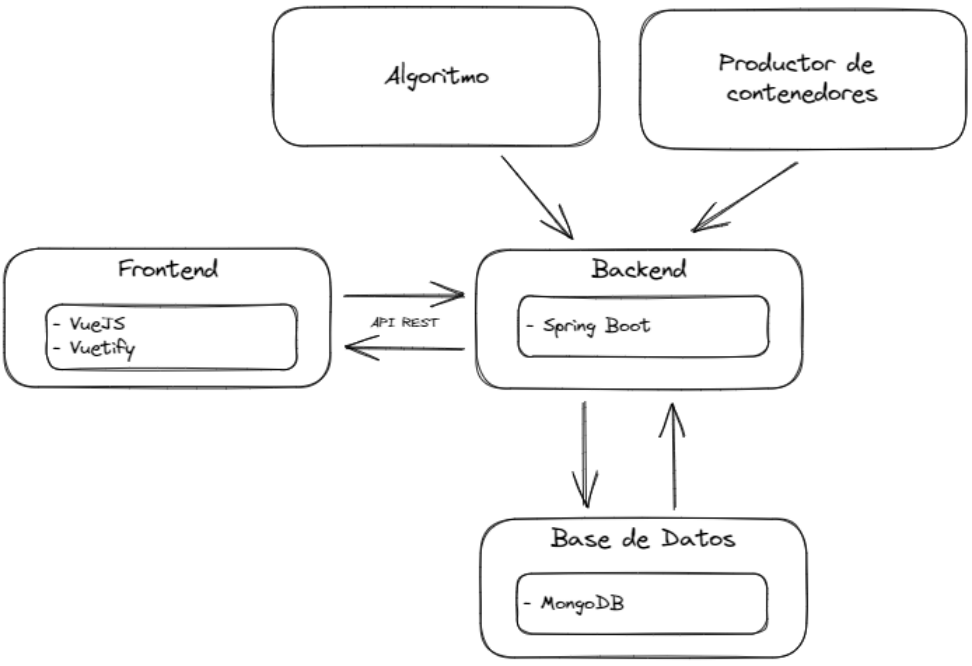


Figura 3.1: Esquema de comunicación en la aplicación

[1] https://www.baeldung.com/hexagonal-architecture-ddd-spring

3.1.1. Principios de la arquitectura Hexagonal

La arquitectura hexagonal es un patrón de diseño de software que se enfoca en separar las capas de la aplicación según su responsabilidad y dependencias, para lograr un sistema altamente cohesivo y de bajo acoplamiento. La capa central, conocida como dominio, se compone de las entidades y clases que conforman el núcleo del sistema. A medida que ascendemos en las capas, llegamos a la capa de aplicación, donde se usan las entidades del dominio para llevar a cabo los casos de uso de la aplicación. La capa más externa es la de infraestructura, que proporciona adaptadores para la conexión con otros servicios, como por ejemplo, el adaptador para el uso de Mongo DB como repositorio para la persistencia de datos. Es importante destacar que estos detalles de infraestructura no afectan al dominio de la aplicación, ya que podríamos utilizar cualquier otra base de datos sin que esto altere la estructura del dominio. En la figura 3.2 se muestra un esquema básico sobre la estructura de una arquitectura hexagonal.

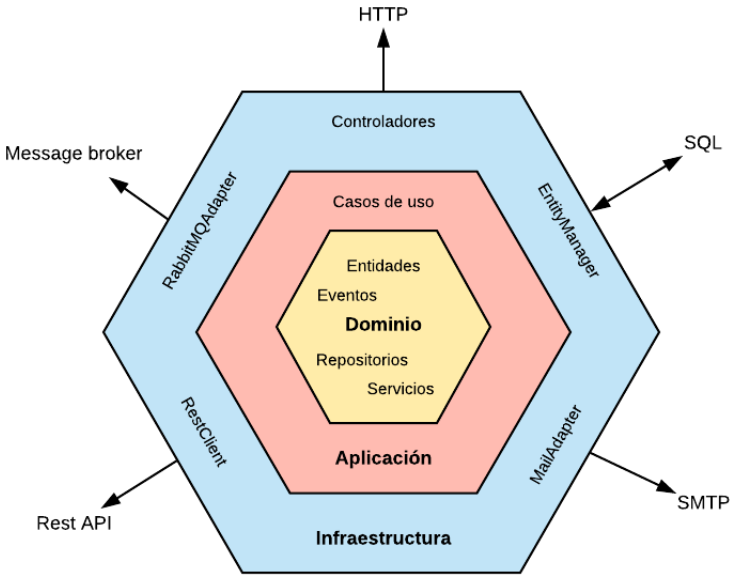


Figura 3.2: Esquema de una arquitectura hexagonal

3.1.2. ¿Por qué la arquitectura hexagonal?

La arquitectura hexagonal aísla toda la lógica de negocio de lo que llamamos *infraestructura*. Esto permite desarrollar un sistema mucho más fácil de modificar y mantener a largo plazo. Es muy probable que las capas externas cambien, a diferencia de las internas.

Cuenta con la flexibilidad que permite al hacer que los componentes se comuniquen entre sí a través de puertos y adaptadores. Por lo tanto, los cambios en un componente no afectan necesariamente a los otros componentes. Esto permite una mayor flexibilidad y evita que los cambios en una parte del sistema afecten a todo el sistema.

Se separan las responsabilidades en diferentes partes diferenciadas. Esto facilita la comprensión de la aplicación y como se puede ampliar.

Por otro lado, permite un desarrollo más eficiente gracias a la reutilización de componentes. Los componentes pueden ser reutilizados en diferentes partes de la aplicación o incluso en diferentes aplicaciones, lo que reduce el costo y el tiempo de desarrollo.

3.1.3. Comunicación entre componentes

Para seguir los principios de la arquitectura hexagonal, hay ciertas reglas que han de ser cumplidas en la manera en la que las diferentes partes de la aplicación se comunican.

El dominio central, como se ha hablado ya, contiene las reglas y entidades esenciales para la solución del problema. Este dominio se comunica con el repositorio. Guarda toda nuestra lógica interna del sistema, también llamada lógica de negocio. Este se comunica con el repositorio y no conoce ninguno de los niveles superiores.

Los repositorios son colecciones en memoria de entidades o agregados. Destacar que los repositorios no son el medio de acceso a la base de datos. Se usa para persistir entidades o agregados, recuperar las entidades conocidas y especificar casos de uso. Solo conoce el dominio y se comunica con los servicios

Los servicios son los encargados de orquestar las llamadas y encapsular toda la lógica que usan nuestras entidades. Son los intermediarios con el *adapter*.

Este *adapter* contiene un controlador que contiene la lógica para que la infraestructura externa a la capa de aplicación opere en nuestro sistema. En la aplicación se emplean varios controladores, uno por cada módulo que tenga que comunicarse con el backend.

El controlador de MongoDB, se encarga de interactuar con la base de datos MongoDB para realizar las operaciones de persistencia necesarias. Este controlador está específicamente diseñado para cada entidad del sistema y se encarga de gestionar las operaciones cruciales relacionadas con la persistencia de datos en la base de datos. Desde la creación y actualización de registros hasta la consulta y eliminación de entidades, el Controlador de MongoDB garantiza que la información se almacene y recupere de manera eficiente y precisa.

El segundo controlador es el controlador REST, cuya función principal es orquestar las comunicaciones con la API REST del sistema. Este controlador actúa como intermediario entre el *adapter* y el usuario final, facilitando el envío y recepción de datos a través de las peticiones y respuestas HTTP. El Controlador REST se encarga de recibir las solicitudes del usuario final a través de la interfaz de la aplicación y traducirlas en llamadas a la API correspondientes. Asimismo, recibe las respuestas de la API y las presenta al usuario final a través de la interfaz de la aplicación. De esta manera, el controlador REST cumple un papel fundamental en la comunicación entre el usuario y el sistema, asegurando que las interacciones se realicen de manera fluida y eficiente.

Por último, un controlador de Web Socket que escucha las actualizaciones del productor de contenedores cuando esté en funcionamiento.

3.2. Frontend

El cometido del frontend es construir una sencilla interfaz para permitir que el usuario interactúe, utilice y recopile información de la aplicación. Mediante dicha interfaz, el usuario puede ejecutar acciones tales como registrar nuevas entidades, ejecutar el algoritmo de optimización de rutas o estudiar los informes y datos relacionados.

El frontend ha sido diseñado siguiendo una arquitectura hexagonal, lo que implica una separación clara entre la capa de presentación y las capas subyacentes de la aplicación. Esto permite una mayor modularidad y flexibilidad, facilitando el mantenimiento y la evolución del sistema. Las principales capas que tiene el frontend son:

- Capa de control del estado. Contiene las *stores* que guardan las entidades traídas desde el backend
- Capa de adaptador a HTTP, se encarga de hacer las peticiones a la API y comunicarse con la capa de control de estado para guardar lo que obtenga de dichas operaciones con el backend.
- Capa de presentación. Es la parte que el usuario ve y con la que interactúa, hechas con Vue 3 y Vuetify.

La principal ventaja de mantener estas capas separadas es que es muy fácil transicionar entre tecnologías de la capa de presentación. Se podría utilizar otra tecnología como React [2] para crear la interfaz de usuario, y el cambio sería mucho menos brusco, pues sólo se tendría que hacer los nuevos componentes y acoplarlos a las otras capas, pero la lógica interna permanece idéntica.

3.2.1. Tecnologías empleadas

Vue 3

Con propósito del desarrollo del *frontend* se empleará el famoso framework de componentes de JavaScript Vue [3], que ofrece una sintaxis clara y flexible para la creación de componentes reutilizables.

Las principales ventajas de Vue 3 y los motivos por los que se ha elegido esta tecnología son:

- Su enfoque principal es la creación de componentes reutilizables, lo que facilita la construcción de aplicaciones modulares y mantenibles.
- Sintaxis clara y flexible.
- Capacidad de enlazar datos permitiendo que los cambios en los datos del componente se reflejan automáticamente en la interfaz de usuario y viceversa.
- Amplia comunidad de desarrolladores y una extensa documentación, lo que facilita el aprendizaje y la resolución de problemas.

[2] https://es.react.dev/
[3] https://vuejs.org/

Pinia

El encargado de gestionar el estado del frontend es Pinia [4], una biblioteca de gestión de estado para Vue 3. Se enfoca en proporcionar una solución moderna y escalable en comparación a su predecesor, Vuex [5].

Pinia se basa en el concepto de stores (mencionadas antes en las capas del frontend), que son contenedores de estado reactivos. Cada store en Pinia representa una porción específica del estado de la aplicación y define sus propias acciones y mutaciones para manipular dicho estado. Puesto cada store se enfoca en una parte específica de la lógica de la aplicación, hay un store por cada entidad con la que el usuario puede operar.

Vuetify

Vuetify [6] es una librería de Vue que brinda una amplia gama de componentes visuales ya hechos y listos para su uso. Esto ahorra tiempo de desarrollo, ya que estos componentes funcionales son plantillas que hay que adaptar a las necesidades del desarrollo.

Vuetify sigue una filosofía de diseño Material Design [7], que es un conjunto de pautas de diseño desarrolladas por Google. Esto no es exclusivo de Vue 3, hay otras tecnologías como React que tienen su propia versión [8], lo que resulta en una apariencia coherente con muchos estándares de otras aplicaciones.

3.3. Backend

El backend es el encargado de recoger, ejecutar y responder las peticiones que el usuario haga a través de la interfaz. Se encarga de gestionar todas las funcionalidades principales de la aplicación, proporcionando una capa de lógica de negocio y servicios que se comunican con la base de datos y exponen una API REST para que los clientes interactúen con el sistema.

Las entidades fundamentales que se encuentran en el backend son las mostradas en la Figura 3.3:

- Camiones: Son a quienes se les asigna las rutas planificadas, están relacionados a un almacén.
- Almacenes: Son los puntos de inicio y final de las rutas a planificar. Contienen camiones guardados y asociados a él.
- Puntos de recogida: Son los puntos geográficos por los que los camiones pasan para realizar la recogida. Un punto de recogida tiene uno o varios contenedores asociados a él.

[4] https://pinia.vuejs.org/
[5] https://vuex.vuejs.org/
[6] https://vuetifyjs.com/en/
[7] https://m3.material.io/
[8] https://mui.com/

- Contenedores: Pertenecen obligatoriamente a un punto de recogida. Cada uno tiene un porcentaje de llenado el cual será un indicador muy importante en el algoritmo que planifica las rutas.
- Soluciones: Son un registro de las planificaciones resultado del algoritmo. Contienen varios indicadores útiles para el análisis de la misma, así como un recorrido a realizar y los camiones asignados a él.
- Algoritmo: Ejecuta la acción para obtener una planificación según los indicadores indicados por el usuario. Esta planificación se guarda como la entidad Solución.

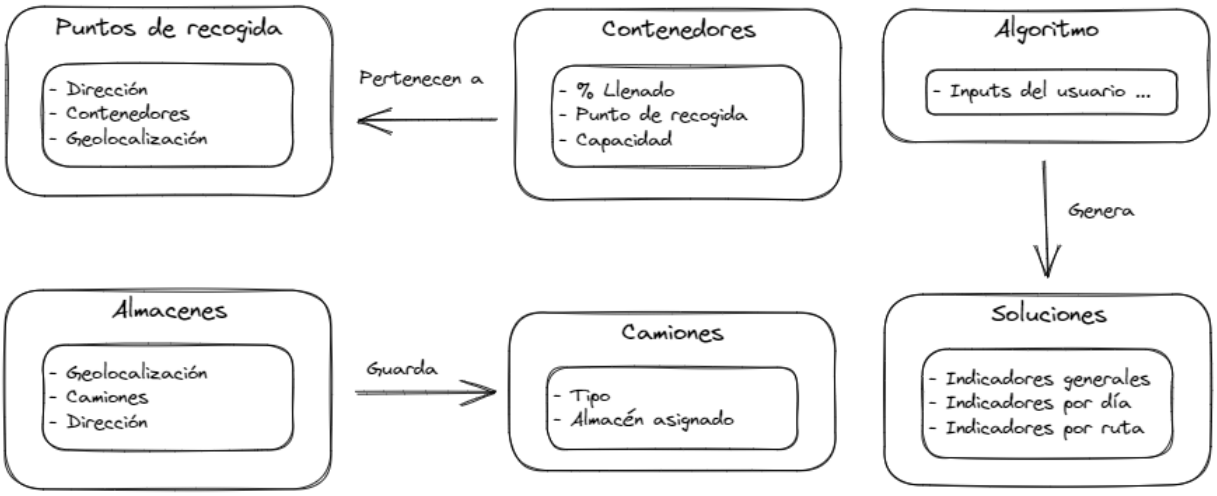


Figura 3.3: Esquema de las entidades principales

3.3.1. Objetos valor

En el desarrollo, se han creado objetos valor como una parte fundamental de la arquitectura del sistema. Al emplear objetos valor, se logra una mayor claridad en la representación de ciertos elementos del sistema, así como un código más mantenible y comprensible.

Los objetos valor[9] son elementos utilizados en programación para representar y encapsular un valor o una entidad que carece de identidad propia y se define únicamente por sus propiedades o atributos. A diferencia de las entidades, los objetos valor no tienen una identidad única y no se distinguen por su identificador único. Los objetos valor se caracterizan por ser inmutables, lo que significa que una vez creados, sus propiedades no pueden ser modificadas.

Estos objetos suelen utilizarse para representar conceptos abstractos o conceptos que no tienen una identidad propia en el dominio del problema. En este desarrollado, se han creado los siguientes objetos valor:

- Address. Guarda una dirección. La utilizan los almacenes y los puntos de recogida. Dentro de la lógica de la clase Address se comprueba que tenga una mínima y una máxima longitud y que esté bien definida.

[9] https://leanmind.es/es/blog/primitive-obsesion-value-objects/

- Geolocation. Representa unas coordenadas geográficas. Es usado por los almacenes y puntos de recogida. Guarda una latitud y una longitud, comprobando que está dentro de los valores límite que pueden tener.
- Percentaje. Es una clase que representa un porcentaje, comprueba que su valor sea entre 0 y 100 y que está correctamente definido. Es usado por la entidad Contenedor, para representar el llenado del mismo.
- Capacity. Esta clase guarda un valor para representar la capacidad total de los contenedores. Principalmente, se asegura que la capacidad no tenga un valor negativo, además de que no cambie en la vida del contenedor.
- TypeRecycling. Es un objeto tipo *enum* el cual solo permite asignar tipo "Plástico.º "Papel.ª los contenedores y camiones

Los objetos valor son útiles porque encapsulan la lógica y el comportamiento relacionados con un valor específico, lo que facilita el mantenimiento del código y control de errores. Se pueden utilizar sin preocuparse por cambios inesperados en su estado.

3.3.2. Tecnologías empleadas

Spring

El encargado del desarrollo del *backend* es Spring [10], un marco de trabajo integral y modular para el desarrollo de aplicaciones empresariales en Java que proporciona una amplia gama de características y ventajas, como la modularidad, la integración con otras tecnologías, la inyección de dependencias, la gestión de transacciones, la seguridad y el testing.

Ofrece muchas facilidades para gestionar las rutas, la seguridad y muchos otros aspectos del desarrollo. Es perfecto para desarrollos de arquitectura hexagonal gracias a su modularidad, que encaja perfectamente con la flexibilidad y su filosofía de capas.

Dentro de Spring existe Spring Boot [11]. Ha sido utilizado para simplificar la configuración y el despliegue del backend. Esta herramienta permite crear aplicaciones de Spring autocontenidas con una configuración mínima, lo que agiliza el proceso de desarrollo y facilita la gestión de dependencias.

Los motivos por los que Spring es empleado por una grandísima comunidad son:

- Spring proporciona una amplia integración con otras tecnologías y marcos de trabajo, como bases de datos, sistemas de mensajería o servicios web.
- Spring proporciona herramientas y utilidades para facilitar las pruebas unitarias e integradas.
- Spring Boot proporciona una configuración automática inteligente, detecta las dependencias y ajusta la configuración en función de las bibliotecas y componentes que estés utilizando, lo que te permite ahorrar tiempo y esfuerzo.

[10] https://spring.io/
[11] https://spring.io/projects/spring-boot

- Las funcionalidad de Spring que facilita la inyección de dependencias resulta ideal construir aplicaciones robustas y escalables.

3.3.3. API REST

La comunicación de la interfaz de usuario con el resto de módulos de la aplicación será responsabilidad de la API diseñada y programada en Spring. La API se comportará de la siguiente manera según los verbos empleados:

- Creación de entidades (POST). En concreto registra camiones, contenedores y almacenes de residuos. Este método guarda en la base de datos estas entidades para ser empleadas en el algoritmo.
- Actualización de entidades (PUT). Permite reescribir los datos internos de las entidades (coordenadas, capacidad, etc) sin la necesidad de hacer un borrado de la entidad.
- Eliminación de entidades (DELETE). Mediante este método, las entidades son eliminadas del sistema, y si es el caso, también las relaciones con otras entidades. Por ejemplo, cada almacén contiene una lista de camiones asignados. En caso de que uno de esos camiones sea eliminado, también desaparecerá de la lista de este almacén.
- Lectura de entindades (GET). Recupera en forma de objeto las entidades que sean solicitadas, permite manipular y trabajar en el frontend con la información recogida.

3.4. Base de Datos

Una base de datos no relacional es mucho más recomendable para el sistema que se necesita en esta aplicación. Uno de los principales motivos es la gran escalabilidad horizontal que ofrecen, lo que significa que pueden manejar grandes volúmenes de datos y un alto rendimiento en entornos distribuidos. El número de puntos de recogida y contenedores puede aumentar significativamente con el tiempo. Si utilizamos una base de datos relacional tradicional, es posible que lleguemos a un punto en el que la capacidad de la base de datos se vea limitada y el rendimiento se degrade a medida que se agregan más datos. Esto puede ser problemático cuando necesitamos manejar un gran volumen de información en tiempo real y asegurarnos de que el sistema pueda escalar eficientemente para satisfacer las demandas crecientes.

Otro motivo es que las bases de datos NoSQL son flexibles en términos de esquema de datos. No requieren una estructura fija y predefinida como las bases de datos relacionales. Esto es beneficioso en el desarrollo de aplicaciones donde los datos pueden tener una estructura variable o evolucionar con el tiempo.

Su colección está compuesta por las entidades principales comentadas anteriormente en el apartado de backend.

3.4.1. MongoDB

MongoDB es una base de datos NoSQL (o no relacional) de código abierto que utiliza documentos en formato JSON para almacenar información. Es escalable y flexible, lo que la hace ideal para manejar grandes volúmenes de datos y sistemas distribuidos. MongoDB utiliza un modelo de datos orientado a documentos que permite almacenar información relacionada en un mismo documento y realizar consultas de manera más eficiente. Es una herramienta poderosa y versátil para manejar grandes cantidades de datos y aplicaciones en tiempo real, que es exactamente lo que se necesita.

MongoDB es altamente escalable y puede manejar grandes volúmenes de datos y cargas de trabajo distribuidas. Permite la distribución de datos a través de múltiples servidores o clústeres, lo que facilita el escalado horizontal según sea necesario. Es una opción popular para aplicaciones en tiempo real que requieren una respuesta rápida a eventos y actualizaciones constantes gracias modelo de datos y su capacidad para manejar grandes volúmenes de datos rápidamente.

3.5. Productor de contenedores

Este módulo de la aplicación es una representación de la realidad simulando el llenado y generación de los contenedores en puntos de recogida. Está programado con Spring al igual que el backend. El productor recibe una serie de parámetros antes de su inicialización a través de un fichero *.yml*, el cual contiene los siguientes campos:

- numberOfStartingContainers. Registra el número de contenedores que el programa genera de inicio.
- numberOfStartingPickupPoints. Registra el número de puntos de recogida que el programa genera de inicio.
- perdiod. Indica cada cuantos segundos en tiempo real se actualiza la información de los puntos de recogida.
- timeFactor. Representa cuantos segundo pasan en la simulación en cada segundo de la vida real. Por ejemplo, poniendo un periodo de 1 segundo y un factor de 1000 segundos significa que un segundo en la "vida real"son 1000 en la simulación.
- startingDate. Fecha de inicio de la simulación.
- endingDate. Fecha de finalización de la simulación.
- updateChance. Es la probabilidad que cada contenedor tiene de llenarse.

Con estos datos presentes, el programa hará una representación ficticia de lo que sería la realidad de un número determinado de días de llenado en los contenedores. La simulación finaliza dependiendo de las fechas y los factores elegidos. Los contenedores tienen una posibilidad de llenarse cada periodo que indiquemos (de nuevo, si el periodo vale 2, es cada dos segundos de ejecución real), y en caso de ser llenados, envía un evento a un puerto para ser captado por quien esté escuchando.

La intención es representar lo que sería tener contenedores y sensores en la vida real que monitoricen el porcentaje de llenado de los contenedores. Ante la imposibilidad de ello, la solución es crear una simulación.

El backend cuenta con un adaptador de WebSocket que escucha al puerto en el que el módulo envíe la información de los contenedores que han sido actualizados. El backend comprueba en cada uno de los mensajes si el contenedor se encuentra en la base de datos, en caso afirmativo, actualiza el porcentaje acorde a cual sea el que indica la simulación. En caso de no encontrar una instancia de ese contenedor en la base de datos, lo crea.

3.6. Tecnologías auxiliares

Pese a no formar parte directamente de la aplicación desarrollada, su uso es de gran utilidad para mejorar el proceso de desarrollo, facilitar la colaboración y garantizar la reproducibilidad del entorno de trabajo. Exploremos brevemente estas tecnologías.

3.6.1. Docker

Docker es una plataforma software que permite la creación, el despliegue y la ejecución de aplicaciones en contenedores. Estos contenedores permiten que una aplicación se ejecute de manera aislada del resto del sistema, lo que mejora la seguridad y la eficiencia. Es ampliamente utilizado en el desarrollo de software.

Es empleado para testar y desplegar la aplicación, y así no preocuparnos de dependencias o configuraciones de los diferentes sistemas.

3.6.2. Git & Github

Como sistema de control de versiones se usa Git y Github. Git es un sistema de control de versiones que permite a los desarrolladores trabajar de manera colaborativa en un proyecto y llevar un control e historial del código escrito. Github es es plataforma web que utiliza Git para alojar repositorios de código.

Git y Github son herramientas fundamentales para la gestión del código fuente y la colaboración entre desarrolladores. Permite la integración con otras herramientas de desarrollo, como sistemas de integración continua y despliegue automático, lo que facilita el proceso de desarrollo y entrega de software. Por ejemplo, se puede realizar el despliegue de los contenedores e imágenes Docker de forma automática con cada push, mediante el uso de GitHub Actions.

Capítulo 4

Funcionamiento

En este capítulo se exploran los casos de uso de la aplicación desarrollada. Los casos de uso son representaciones detalladas de las interacciones entre los usuarios y el sistema, describiendo cómo se utilizan las funcionalidades de la aplicación para lograr las soluciones que la plataforma elabora.

El objetivo principal de esta sección es analizar y describir los diferentes escenarios en los que la aplicación puede ser utilizada, brindando una visión clara de cómo los usuarios se benefician de ella en situaciones específicas. Los casos de uso proporcionan un enfoque sistemático para comprender cómo se utilizan las funcionalidades de la aplicación y cómo estas se adaptan a las necesidades de los usuarios.

A lo largo de este apartado, se presenta varios casos de uso que abarcan diferentes situaciones y acciones. Cada caso de uso se describe detalladamente, especificando los actores involucrados, las acciones realizadas y los resultados esperados. En la figura 4.1 se muestra un esquema general de los casos de uso de la aplicación.

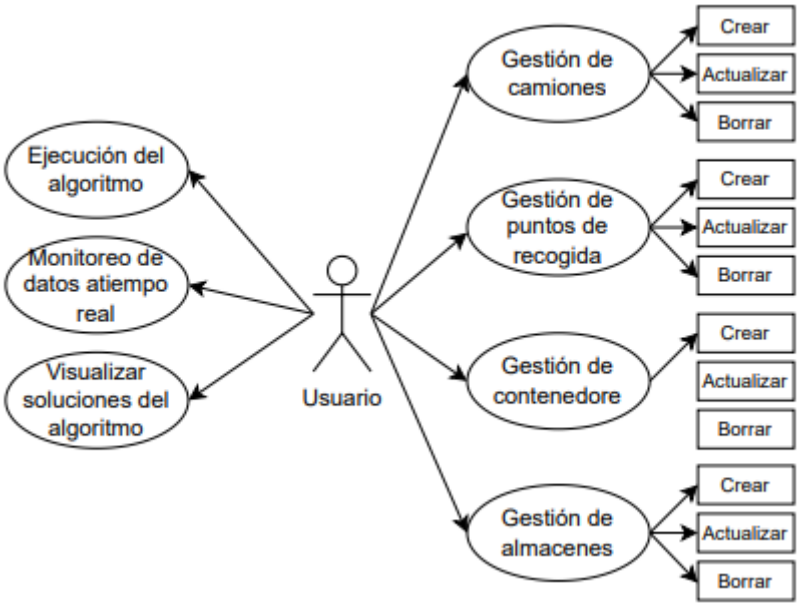


Figura 4.1: Resumen de los casos de uso de la aplicación

Mediante el análisis de los casos de uso, se pretende demostrar cómo la aplicación

desarrollada puede mejorar la eficiencia, la productividad y la experiencia de los usuarios, cumpliendo con los objetivos planteados en este TFG. También se proporciona una comprensión más profunda de la aplicación y se presenta ejemplos prácticos de cómo puede ser aprovechada en diferentes escenarios reales. Esto permite evaluar su efectividad, utilidad y potencial impacto positivo en la gestión de las recogidas de residuos.

4.1. Gestión de contenedores y puntos de recogida

La aplicación permite gestionar toda la información tanto de los contenedores como de los puntos de recogida. Estos últimos registran una geolocalización física real, y a su vez guarda todos los contenedores que le pertenecen. De esta manera vemos que estas entidades comparten una relación directa. Los contenedores por otro lado, registran su tipo (plástico o papel) y el punto de recogida al que pertenecen. No puede haber un contenedor sin asociar a un punto de recogida, y estos deben tener al menos un contenedor a su cargo para que el punto de recogida tenga sentido registrarlo. Ante este planteamiento, tiene sentido que ambas entidades sean gestionadas en la misma sección de la aplicación.

Como se puede observar en la pantalla principal para la gestión de estas entidades (figura 4.2) se pueden realizar las siguientes operaciones:

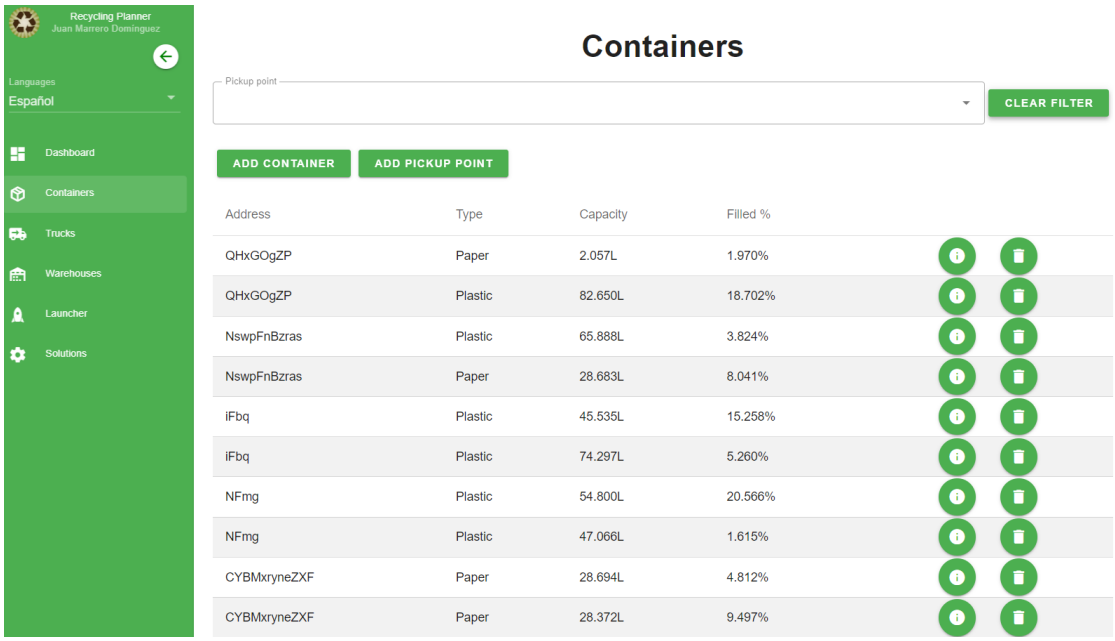


Figura 4.2: Pantalla principal para la gestión de contenedores y puntos de recogida

4.1.1. Registrar nuevos contenedores y puntos de recogida

Desde aquí, los administradores encargados de mantener la información del sistema al día crean los puntos de recogida y los contenedores correspondientes. La pantalla de gestión dispone de los botones pertinentes para crear una u otra entidad. Al pulsarlos se accede el formulario correspondiente. La figura 4.3 muestra ambos.

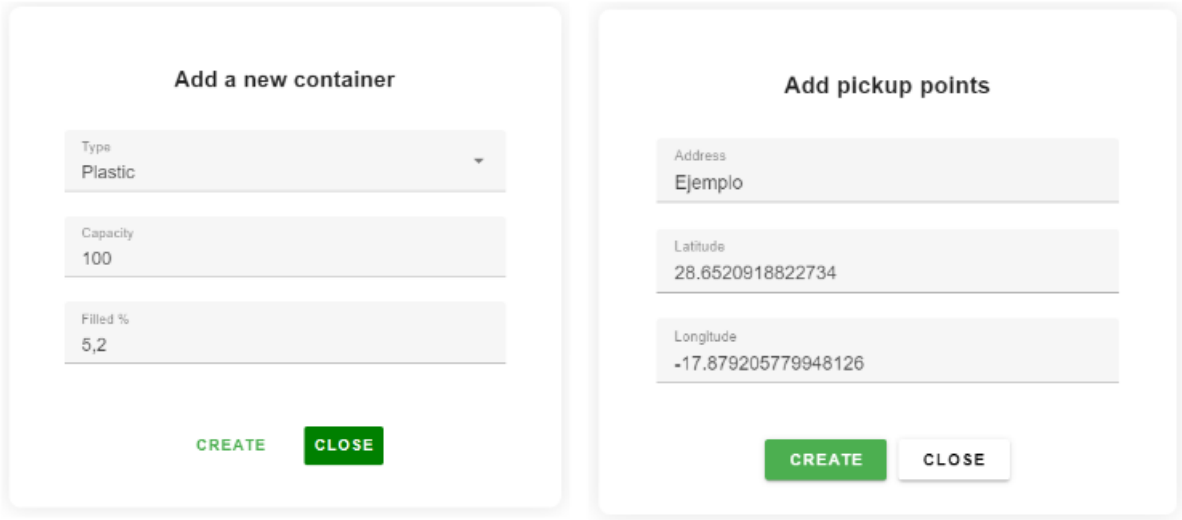


Figura 4.3: Formularios de creación para puntos de recogida y contenedores

4.1.2. Inspeccionar un punto de recogida o contenedor

Estas entidades tienen su propio botón que despliega información más precisa sobre el objeto en cuestión. En la figura 4.4 se muestra como el punto de recogida de ”Los Cancajos” es desglosado y contiene toda la información, además de un mapa con el que el usuario puede interactuar.

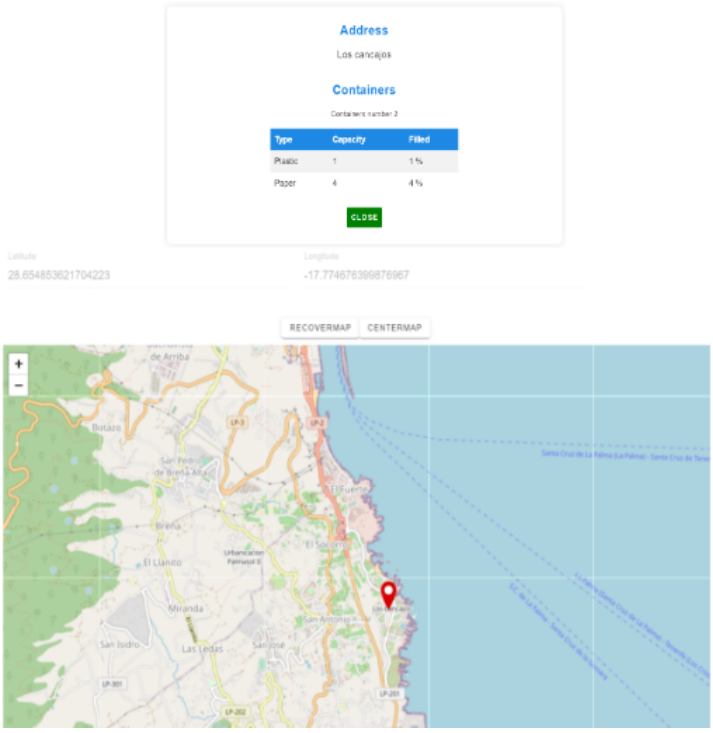


Figura 4.4: Información del punto de recogida "Los Cancajos"

4.1.3. Eliminar contenedores y puntos de recogida

Por último, los usuarios pueden eliminar las entidades que deseen mediante el botón de eliminado correspondiente brindado en el panel de control 4.2. Si borras un punto de

recogida, todos sus contenedores asociados se borran también de la base de datos.

4.2. Gestión de almacenes

Los almacenes sirven, desde un punto de vista lógico, para establecer el principio y el final de las rutas. Tienen registrados una dirección y unas coordenadas, por lo que también son visualizados en el mapa. A su vez, guardan una lista de camiones que están registrados en dicho almacén. Esto es útil para el algoritmo de enrutado (explicado en la sección 4.4). La interfaz reflejada en la figura 4.5 brinda un claro resumen de la situación de los almacenes registrados.

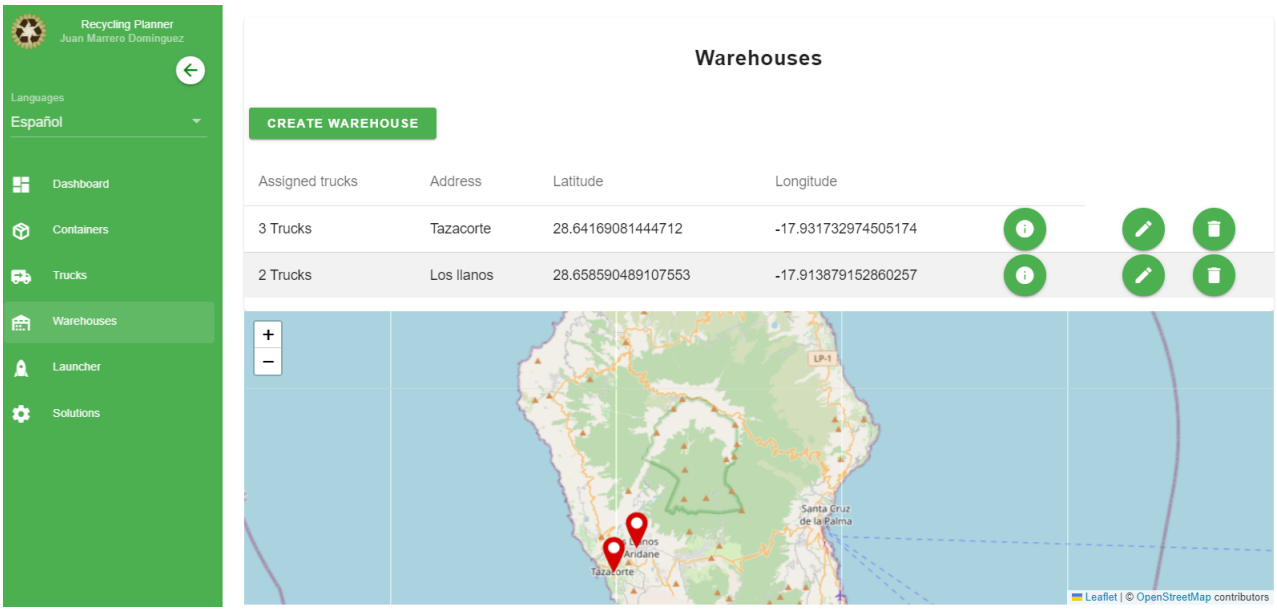


Figura 4.5: Pantalla principal para la gestión de almacenes

De igual manera que en anteriores entidades, desde aquí el administrador es capaz de consultar información concreta de uno de los almacenes, borrarlo y en este caso actualizarlo también.

4.3. Gestión de camiones

Los camiones son las entidades que a nivel lógico nos indican cuántas rutas puede generar el algoritmo, puesto que a más camiones disponibles, la ruta es repartida de diferente manera. Los camiones deben estar ligados a un almacén que exista. De ninguna manera pueden estar sin registrarse en el almacén que les corresponda. Tienen su propia pantalla de gestión 4.6. Un camión frecuenta varios almacenes y no está casado con un solo almacén, a diferencia que un contenedor con su punto de recogida correspondiente, pues no es tan normal que cambie de ubicación. Es por esto que su gestión está separada de la de los almacenes.

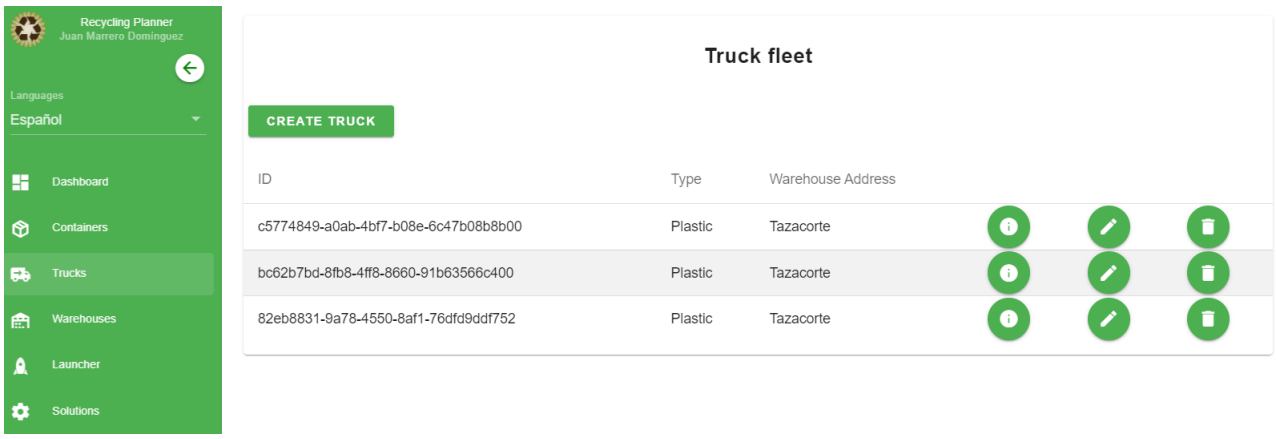


Figura 4.6: Pantalla principal para la gestión de camiones

Los camiones guardan como atributos el tipo de residuo que tratan y la dirección del almacén en el que se encuentran en ese momento. Los tipos de residuos que pueden tratar son dos: papel o plástico. Tienen las operaciones básicas que tienen el resto de entidades.

Los camiones se crean en un sencillo formulario que permite registrar la información necesaria. A su vez pueden ser actualizados o borrados según el usuario gestor quiera. También cuenta con una funcionalidad, mostrada en la figura 4.7, para desglosar más profundamente la información concreta de solo uno de los camiones.

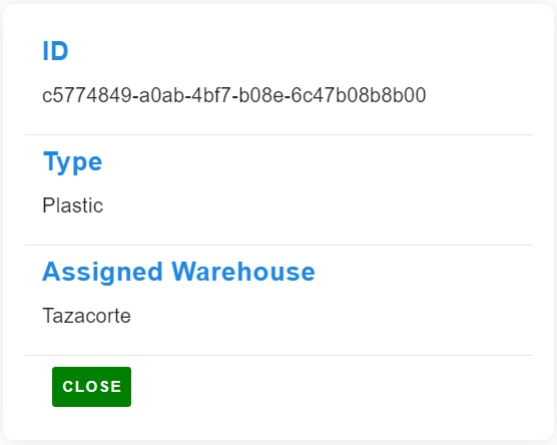


Figura 4.7: Información de uno de los camiones

4.4. Funcionamiento del algoritmo

El algoritmo es una parte fundamental de la aplicación encargada de devolver la solución que ha generado a partir de su ejecución. Este planifica las rutas a partir del input que el usuario introduzca. Hay un formulario que rellenar, mostrado en la figura 4.8, que manda los datos necesarios para la ejecución del algoritmo al backend. La entrada del algoritmo es la siguiente:

- Número de días en los que planearás la ruta.

- Tiempo máximo que puede durar un recorrido por cada camión.
- Una constante que representa el tiempo estimado en el que se tarda en recoger cada contenedor.
- Tipo de ruta, esto es para recoger Plástico o Papel.
- Almacén de salida.
- Almacén de llegada.
- Lista de camiones usados para la planificación.
- Puntos de recogida existentes.

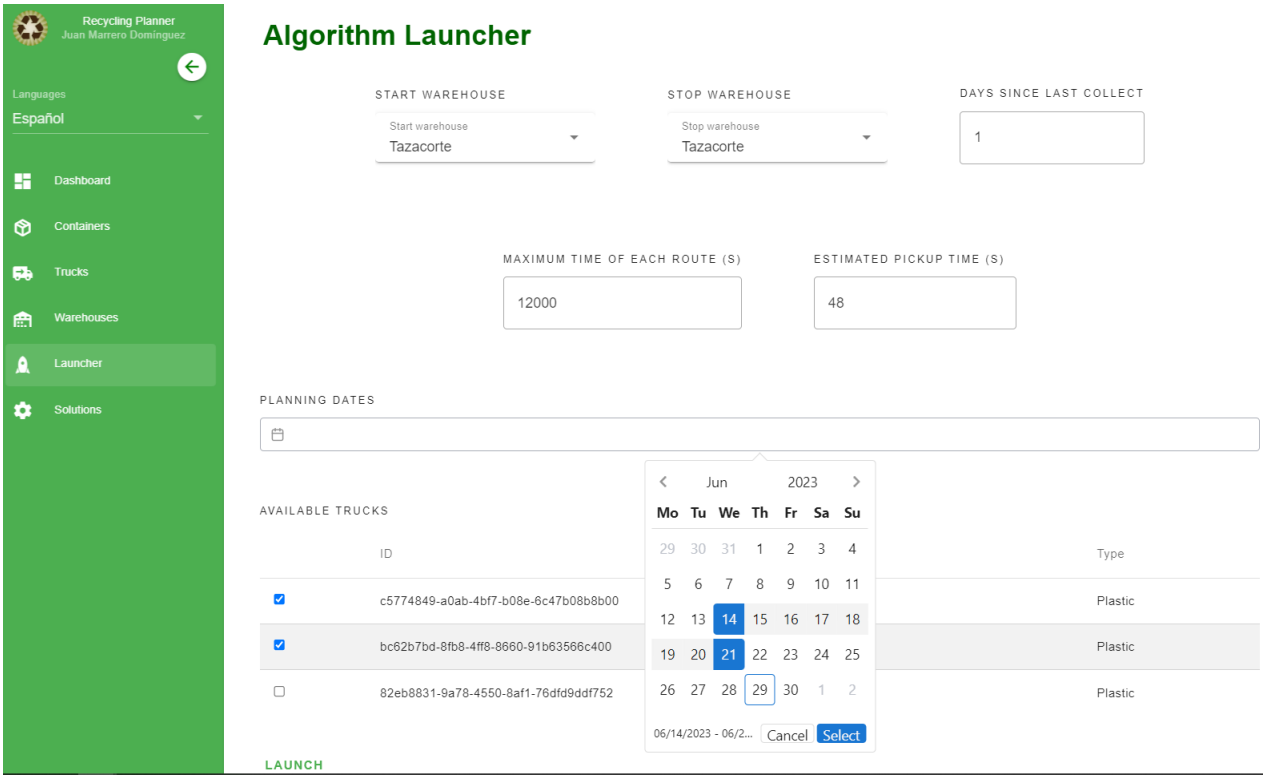


Figura 4.8: Formulario lanzamiento del algoritmo

Los datos requeridos en el formulario incluyen información como la ubicación de los puntos de recogida, la capacidad de los camiones, los tiempos estimados de recogida, las restricciones específicas y cualquier otra información relevante para la planificación de las rutas. Se generan rutas dependiendo del número de camiones y de días disponibles. En concreto, se genera una ruta por camión, y cada camión hace dos rutas al día.

El usuario completa el formulario proporcionando los datos requeridos, estos datos son enviados al backend de la aplicación para su procesamiento por el módulo del algoritmo de planificación de rutas. El algoritmo utiliza los datos proporcionados en el formulario como entrada y genera las mejores rutas posibles. Esto implica determinar la mejor secuencia posible de puntos de recogida, asignar los camiones de manera eficiente, minimizar los tiempos de recorrido y maximizar la eficiencia en la recolección de residuos. Una vez que el algoritmo ha generado las rutas de recogida, la aplicación presenta al usuario

los resultados obtenidos mediante la visualización de las rutas en un mapa interactivo, la estimación de los tiempos de recorrido, la asignación de camiones a cada ruta, entre otros detalles relevantes.

4.5. Visualizador de soluciones planificadas

Las soluciones resultantes del planificador de rutas son guardadas en la base de datos como entidades independientes. Contienen una serie de indicadores cruciales para el análisis y comparación de las soluciones. Todas las soluciones se encuentran en un panel común 4.9. Este panel cuenta con la posibilidad de eliminar o inspeccionar a fondo cada solución, según quiera el administrador.

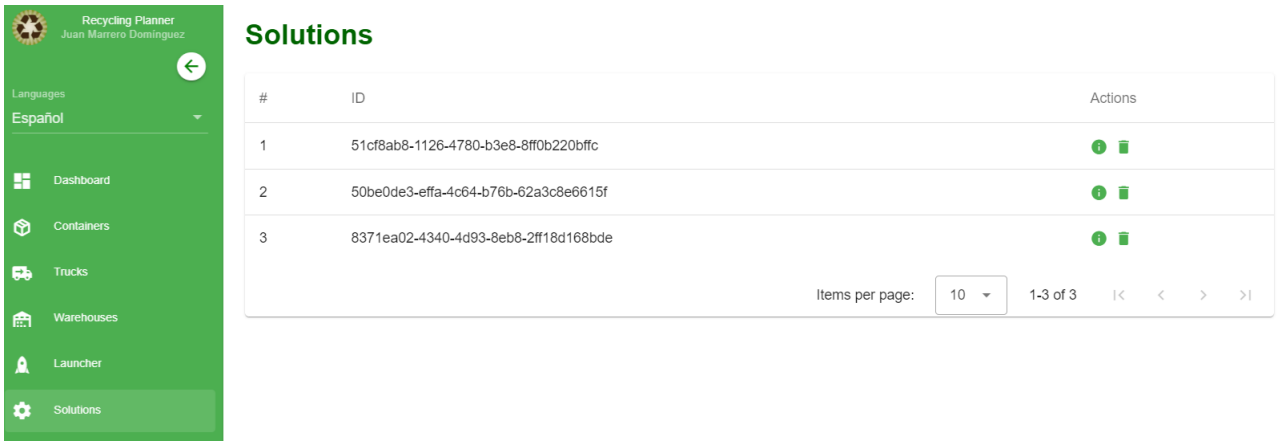


Figura 4.9: Pantalla de visualización de soluciones

Al inspeccionar una solución, se desglosan los indicadores generados. Estos son de tres tipos:

- Indicadores generales: Estos indicadores recogen información de todas las rutas y días de la planificación 4.10. Muestra, entre otros, el tiempo medio de cada día que ha sido empleado en recorridos, media de contenedores recogidos por día, combustible consumido, etc.

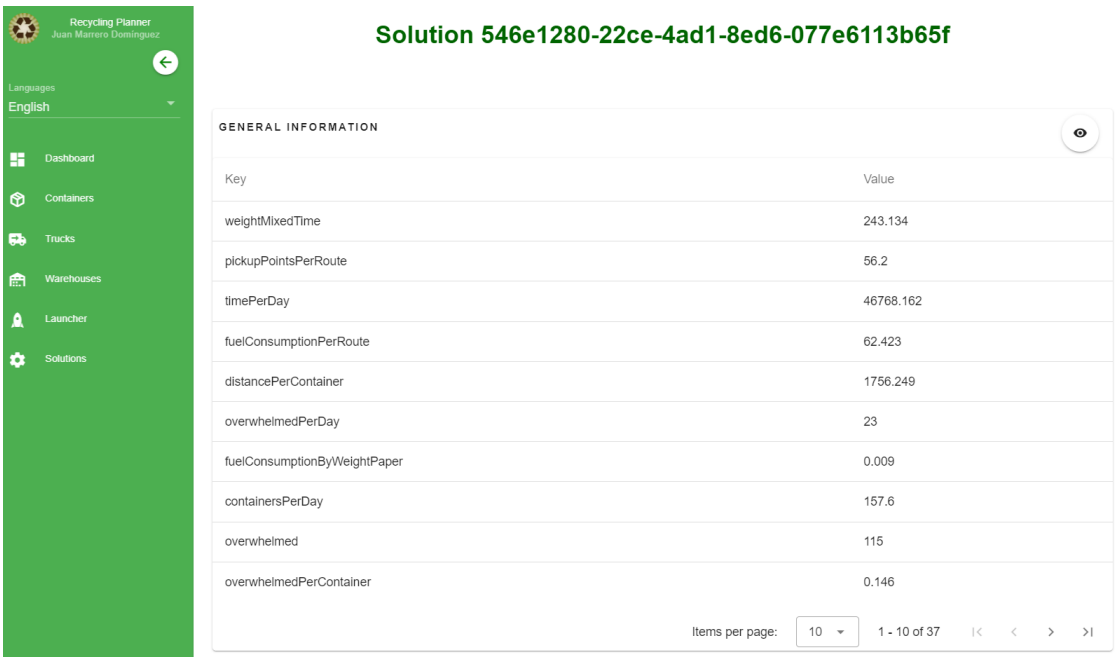


Figura 4.10: Valores generales de una solución

- Indicadores por día: Desglosa la información en una tabla dividida por días 4.11. Por ejemplo, si la planificación ha durado 13 días, cada día tiene sus propios valores. Algunos valores son distancia media recorrida ese día por cada camión, tiempo medio de cada ruta, etc.

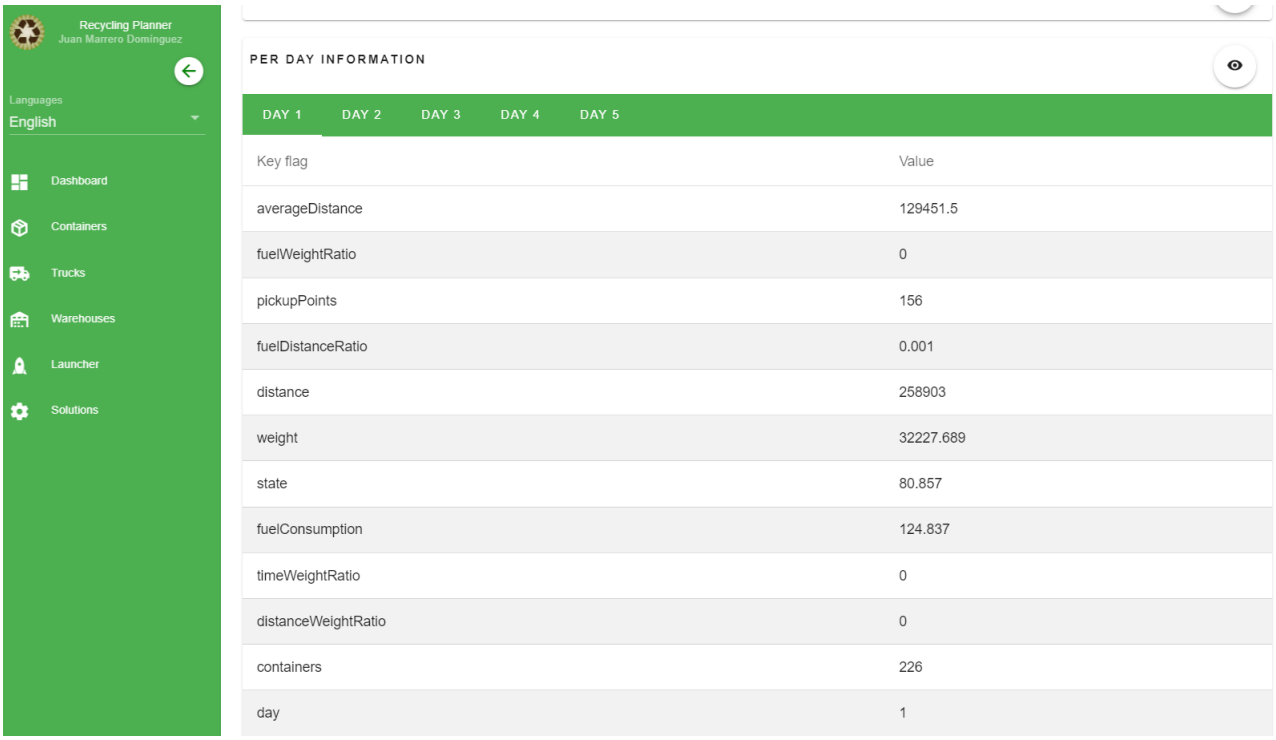


Figura 4.11: Valores por día de una solución

- Indicadores por ruta: Desglosa la información en una tabla dividida por rutas 4.12. Guarda indicadores como contenedores recogidos cada ruta, promedio de combustible gastado por ruta, etc.

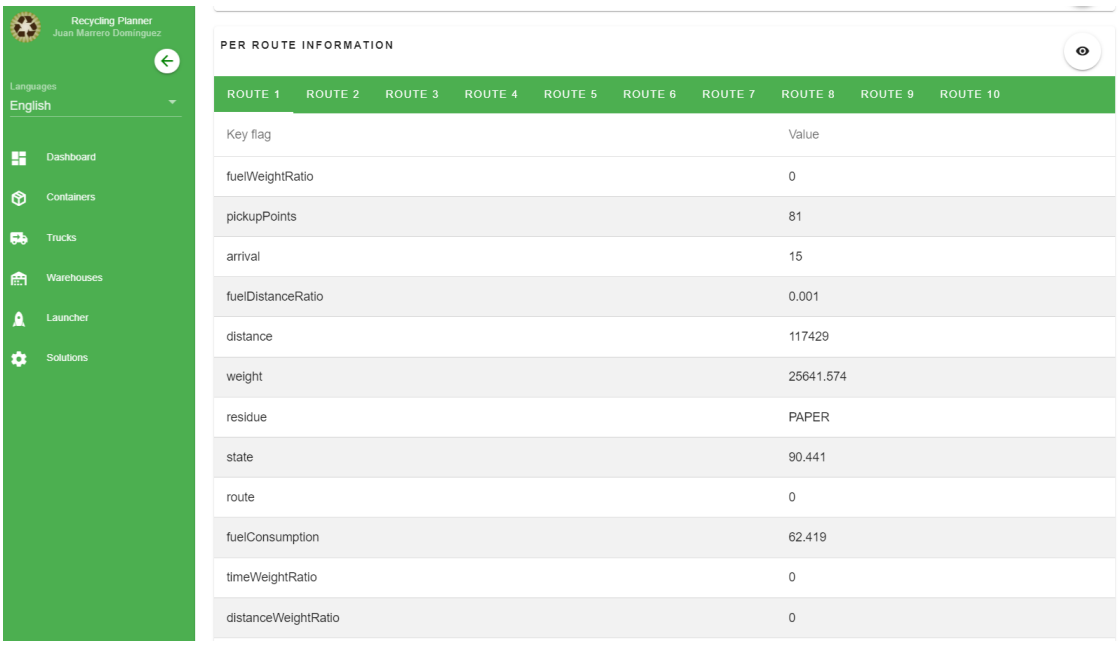


Figura 4.12: Valores por ruta de una solución

Lo más interesante de inspeccionar la solución es que cada ruta tiene un mapa en el que se ve reflejado el camino a seguir en cada una de ellas. En un sencillo ejemplo generado en el apartado 5.3.2 la ruta se ve plasmada como se ve en la figura 4.13. De momento, y se aclara de nuevo en el capítulo de líneas futuras de trabajo, los caminos se ven rectos, sin respetar las carreteras en la realidad. De momento sirve para hacer a la idea de como queda representada cada ruta. De igual manera, la ruta está representada en orden en una tabla. Al hacer click en cada punto de la tabla, el mapa se centra en esa coordenada.

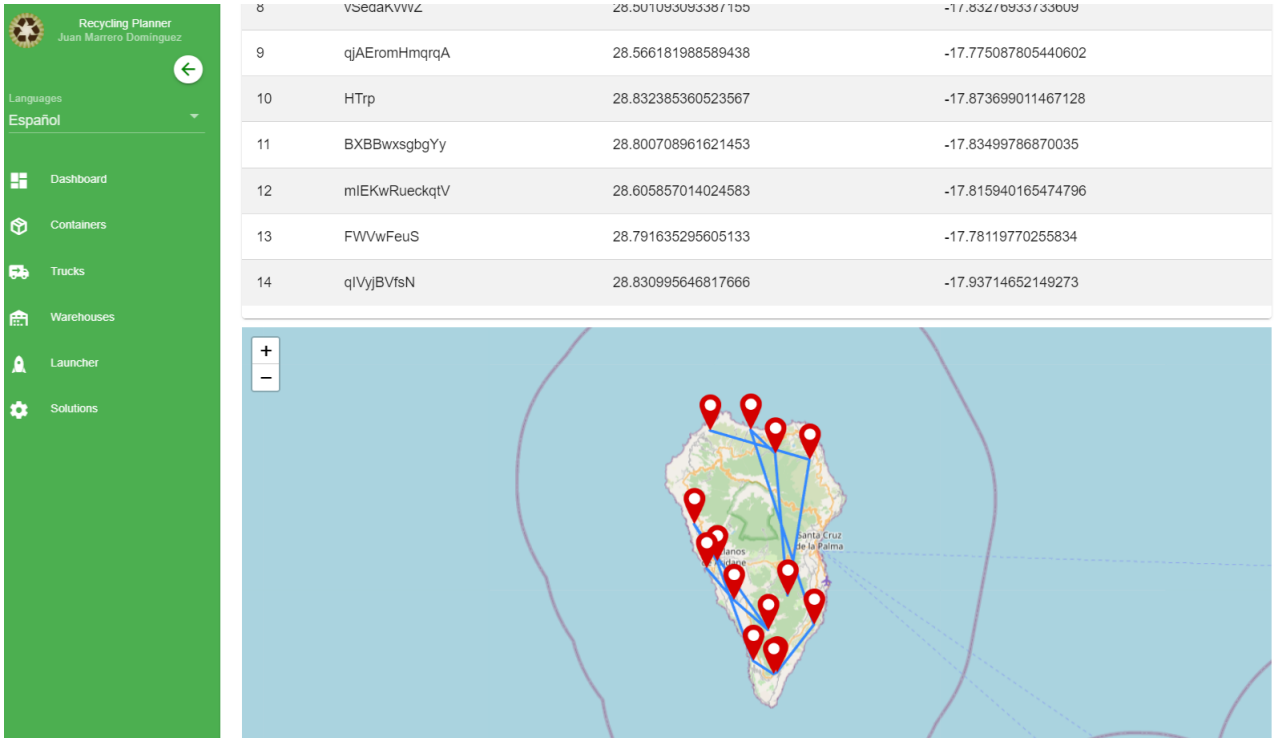


Figura 4.13: Ruta planificada

4.6. Cuadro de mandos

La plataforma provee un cuadro de mandos que desempeña un papel crucial en el seguimiento y control de las entidades registradas en el sistema. Este cuadro de mandos ofrece una visión global y actualizada del estado real de las diferentes entidades, permitiendo al gestor comprender rápidamente la situación y tomar decisiones informadas en cada momento. Es una herramienta esencial para el usuario, ya que proporciona una visión integral del estado actual de las entidades registradas y ofrece orientación sobre las decisiones que se deben tomar. Esta funcionalidad garantiza una gestión eficiente y efectiva de las operaciones de recogida de residuos, permitiendo al usuario estar al tanto de la situación en tiempo real y tomar acciones para optimizar los recursos y mejorar la calidad del servicio.

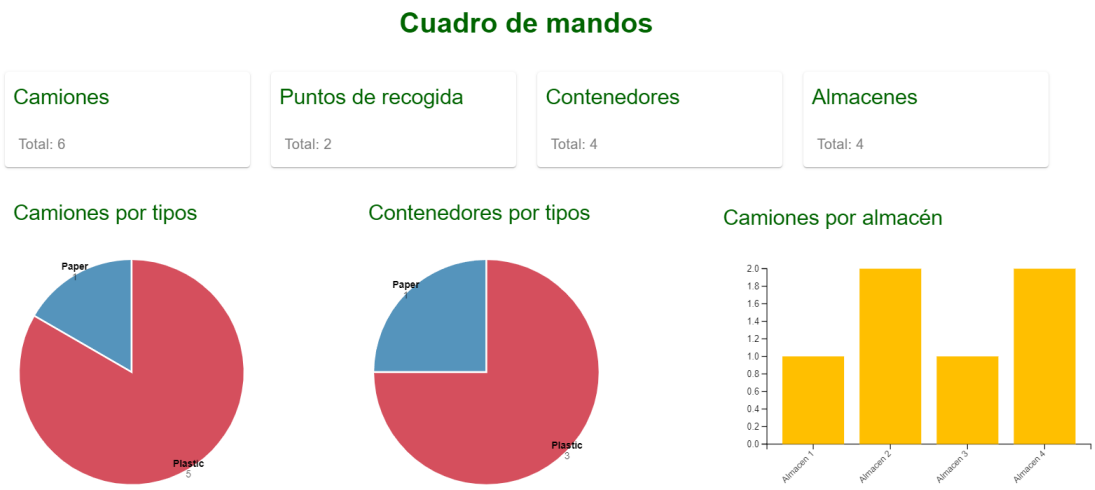


Figura 4.14: KPI’s y gráficos del cuadro de mandos

Cuenta con una sección de gráficas y KPI’s mostradas en la figura 4.14. Contiene una distribución de todas las entidades almacenadas en el sistema. Por otro lado cuenta con una tabla y un mapa que expone la información de todos los puntos de recogida y su porcentaje actualizado cada segundo a tiempo real. En el mapa, se podrá visualizar el estado de llenado de los contenedores. Según esté más o menos lleno, el color de su indicador del mapa será diferente:

- 0 % - 20 %: El indicador es verde
- 21 % - 50 %: El indicador es amarillo
- 51 % - 85 %: El indicador es naranja
- 86 % - 100 %: El indicador es rojo

Este sistema permite, con un vistazo, entender que zonas de la isla requieren más urgentemente una recogida, pues están más cerca de su llenado completo. En la figura 4.15 se presenta un ejemplo de todos los casos.

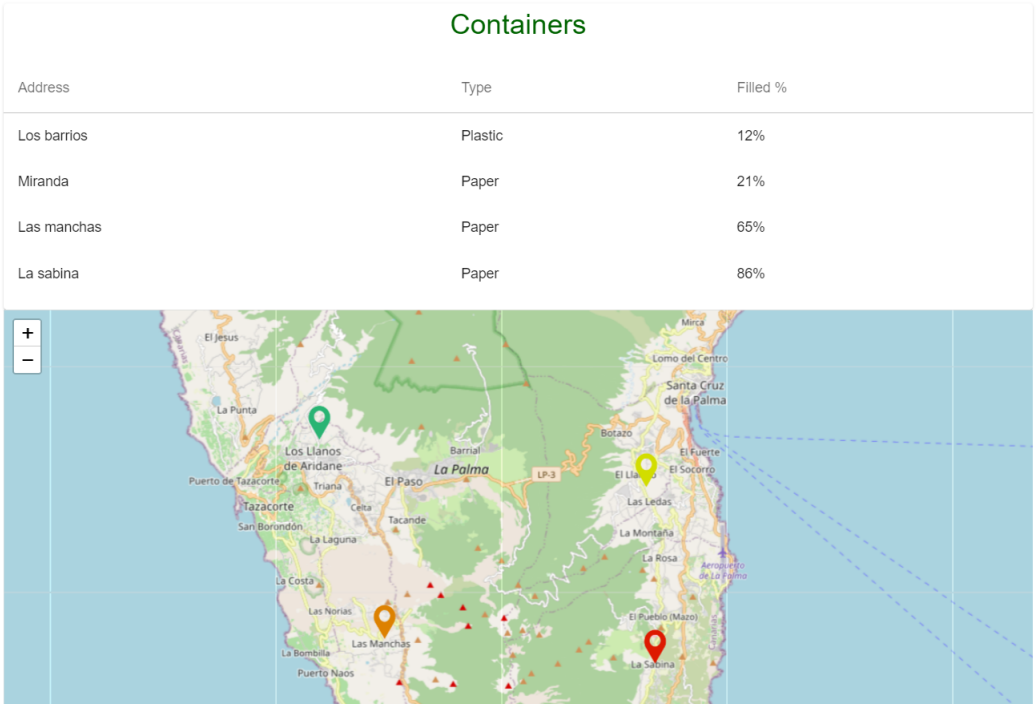


Figura 4.15: Mapa que contenedores con su estado

Capítulo 5

Experimentación

En este capítulo hace una demostración de las funcionalidades implementadas. Se abarca su funcionamiento y planteamiento de escenarios reales

5.1. Operaciones CRUD

Las operaciones CRUD son un aspecto fundamental para el buen funcionamiento de la aplicación. En esta sección se muestra el funcionamiento de cada una de ellas. Para este experimento se opera con la entidad almacén, el funcionamiento es similar para el resto de entidades.

5.1.1. Creación de entidades

Vamos a comenzar creando un almacén, tras rellenar el formulario de la siguiente figura 5.1, la nueva entidad almacén es registrada tal y como se ve en la figura 5.2.

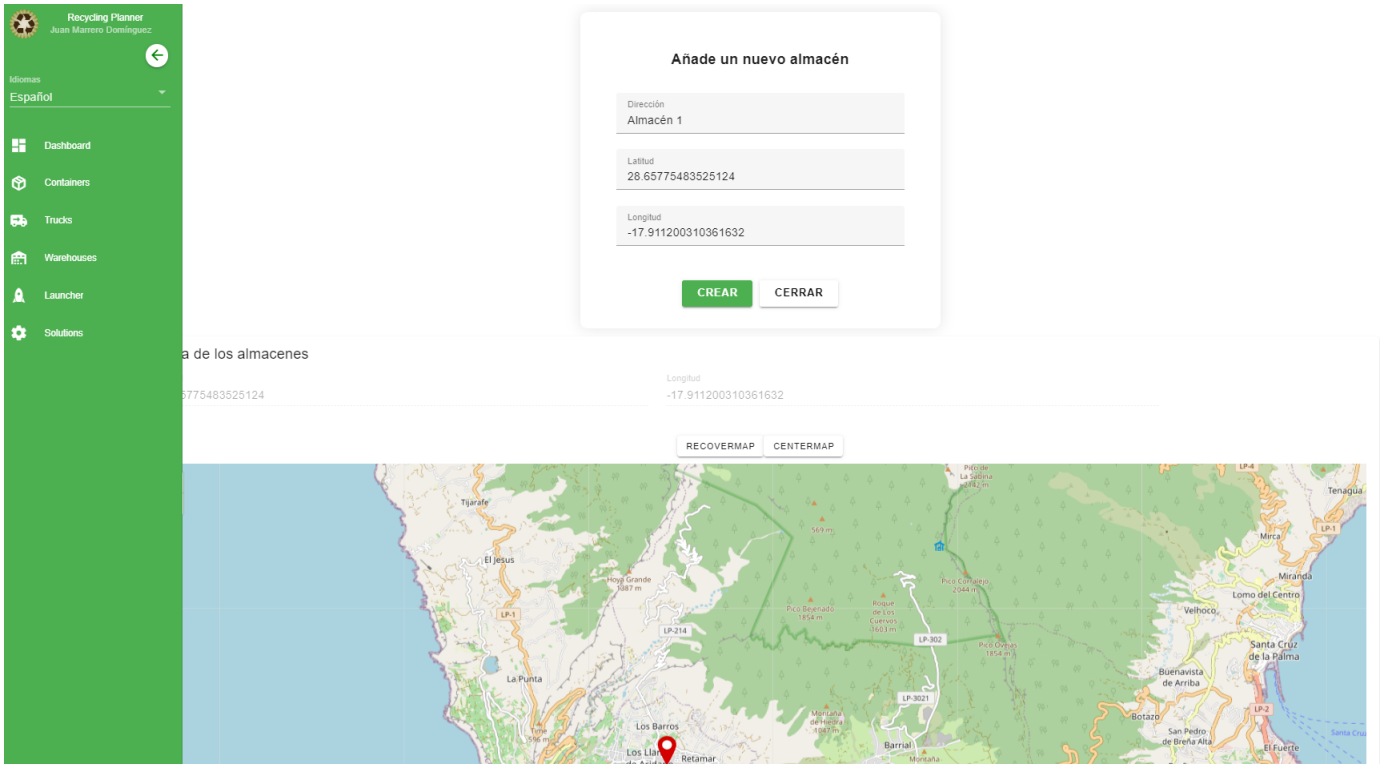


Figura 5.1: Formulario para crear un almacén

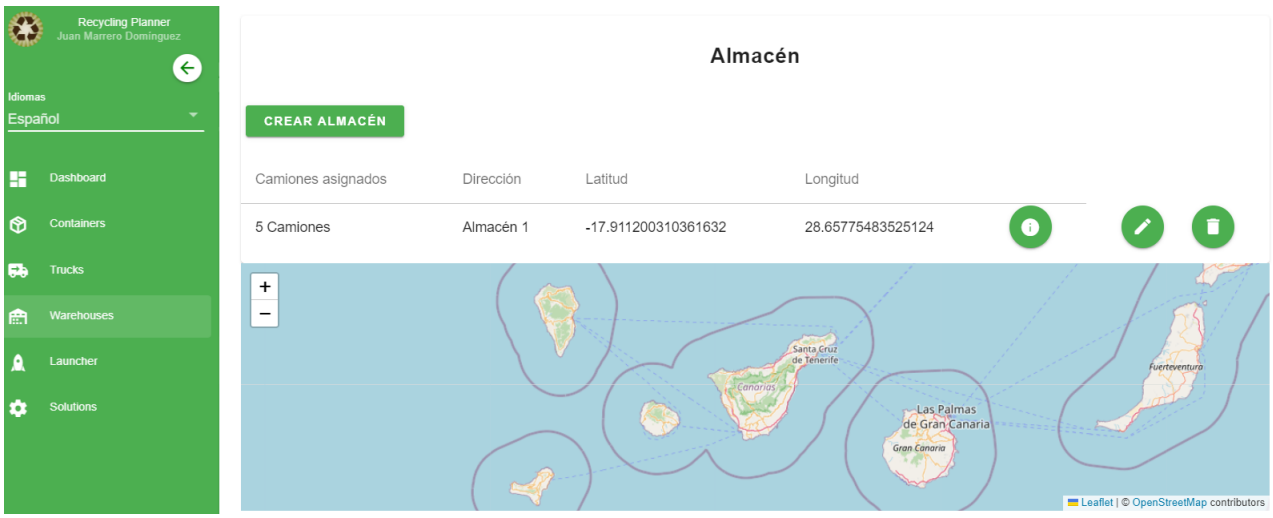


Figura 5.2: Almacén creado

5.1.2. Lectura de entidades

Ahora, se puede consultar en detalle la entidad gracias a la funcionalidad que permite inspeccionar cada entidad. En la figura 5.3 se muestra toda la información correspondiente al recién creado almacén.

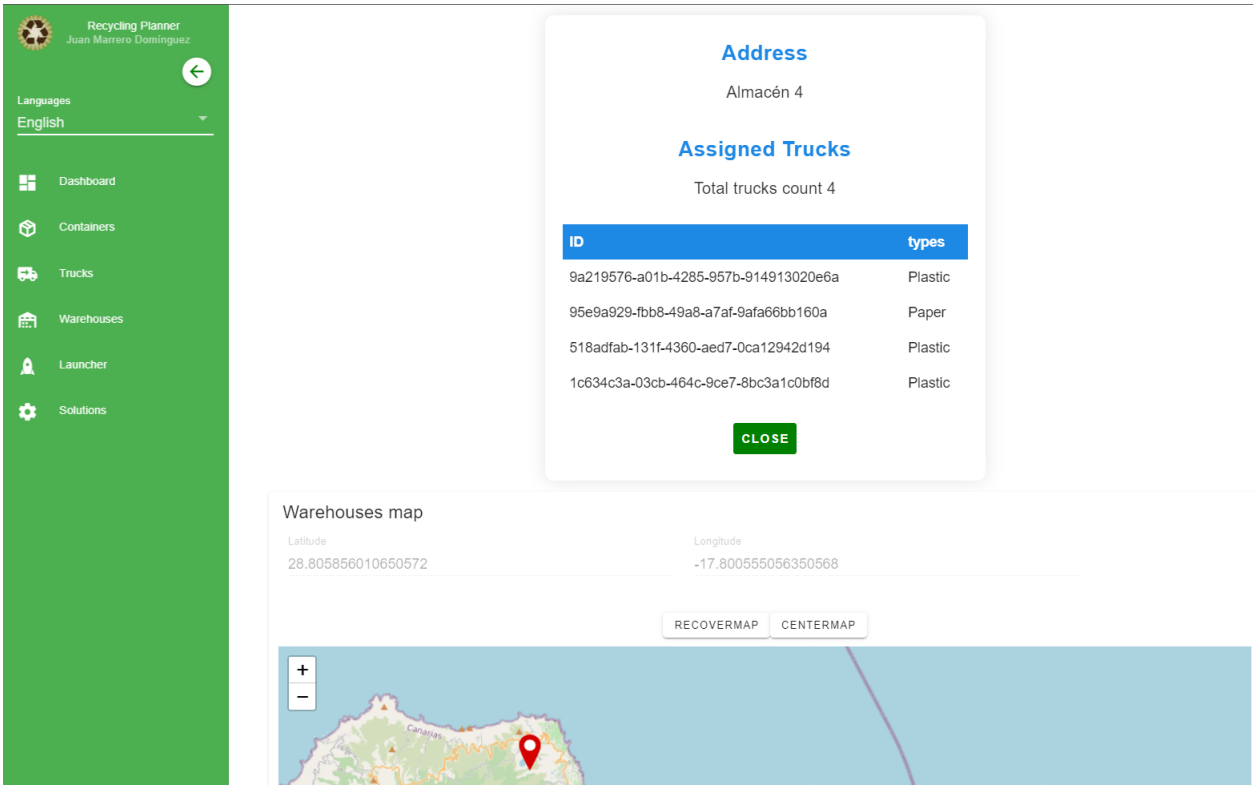


Figura 5.3: Información del recién creado almacén

5.1.3. Actualización de entidades

En caso de que se deba modificar algún que otro atributo de una entidad, se accede la funcionalidad *update* de cada entidad, en la que se rellena el formulario para efectuar

los cambios. En la figura 5.4 se muestra el formulario. En este caso se le ha cambiado la dirección del almacén.

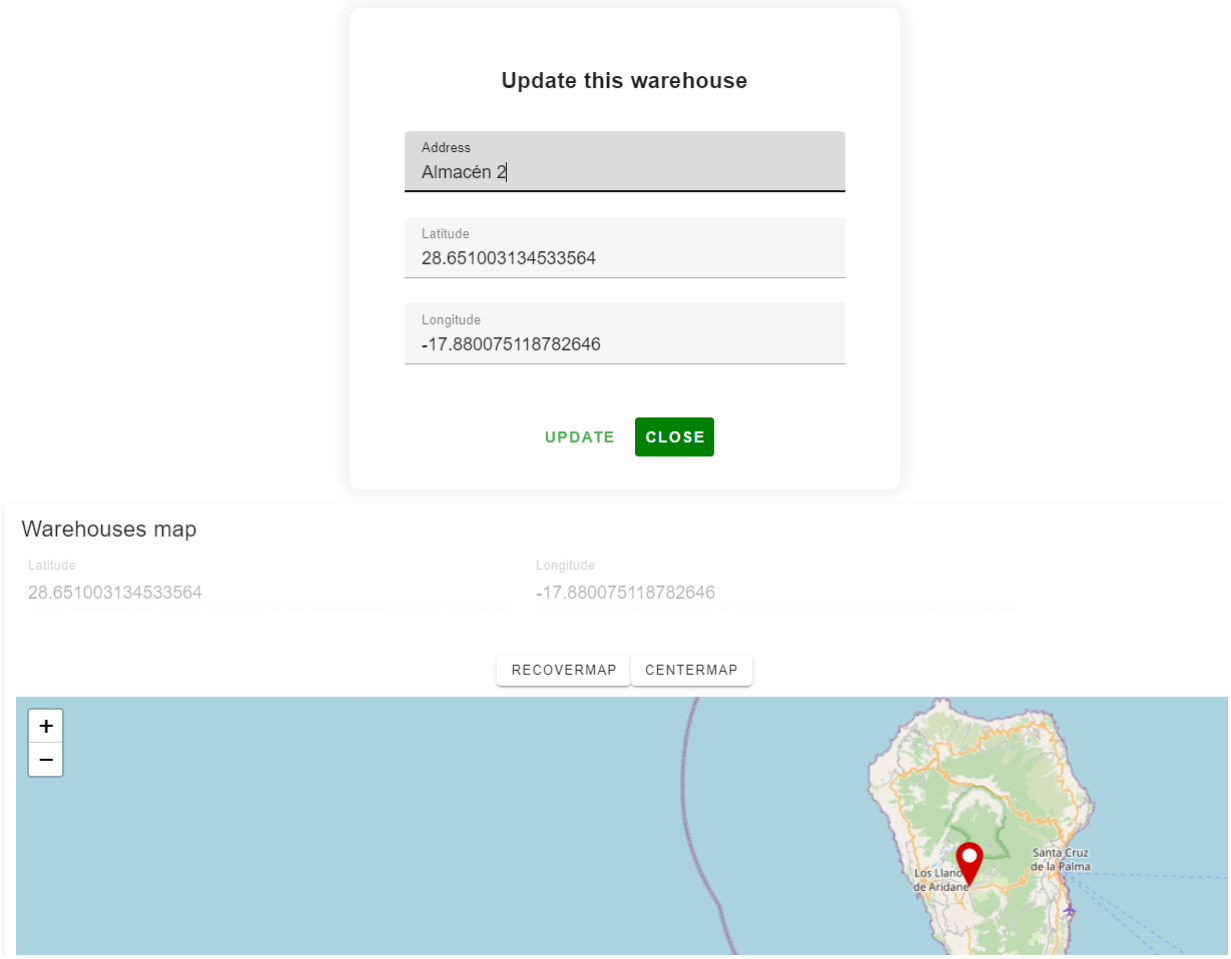


Figura 5.4: Formulario para actualizar los datos de un almacén

5.1.4. Eliminación de entidades

Por último, para eliminar basta con hacer click en el botón cuyo icono es una papelera para disparar dicha funcionalidad y eliminar la entidad de la base de datos.

5.2. Cuadro de mandos

El panel de control presenta de manera organizada y clara toda la información relevante sobre el estado actual del sistema. En esta sección, los usuarios pueden encontrar un resumen de los datos con los que deben trabajar y supervisar. Para la demostración, se han usado 9 camiones, 2 puntos de recogida con 4 contenedores, y otros 4 almacenes como datos de ejemplo, para comprobar que son visualizados correctamente.

Una parte importante del panel de control son los KPI's, que muestran el número de entidades actualmente registradas en la aplicación. Además, se presentan gráficas que representan la distribución de camiones y contenedores según el tipo de material de reciclaje al que pertenecen, como papel o plástico. También se muestra la distribución de los camiones en los diferentes almacenes o plantas de reciclaje.

Justo debajo de las gráficas se encuentra una tabla que muestra los contenedores registrados en el sistema. Esta tabla incluye información sobre el porcentaje de llenado de cada contenedor. Es importante destacar que esta información se actualiza en tiempo real, sin necesidad de recargar la página, con una frecuencia de actualización de sesenta segundos.

En las figura 5.5 se ilustra visualmente cómo es esta sección del panel de control. Esta imagen proporciona una representación visual de la interfaz y muestran la disposición y apariencia de los elementos descritos anteriormente. Esta imagen concretamente contiene los datos de 9 camiones, 2 puntos de recogida con 4 contenedores, y otros 4 almacenes.

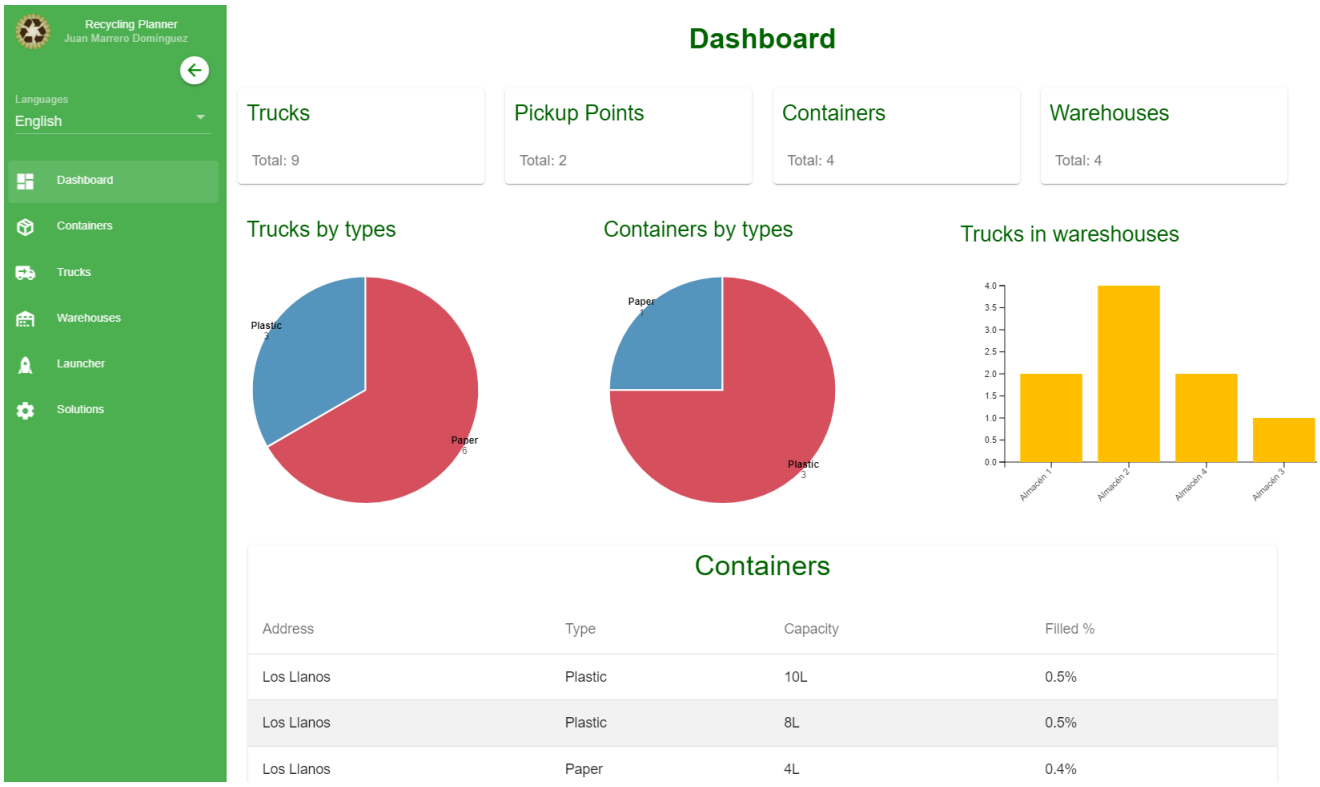


Figura 5.5: Cuadro de mandos en inglés

5.3. Lanzamiento del algoritmo

Esta sección de la aplicación permite a los usuarios generar rutas y soluciones óptimas para planear las rutas de recogida de residuos. Podemos dividir el proceso de general un plan en 3 fases principalmente. Para explicarlas, se va a simular un caso real en el que se emplearán 2 almacenes, cada uno con 3 camiones, en la planificación se tendrán en cuenta 14 puntos de recogida, cada uno con sus respectivos contenedores, de 2 a 5 por cada uno de ellos. En este ejemplo, se planifican 4 días, y solo 2 de los 3 camiones disponibles del almacén de salida.

5.3.1. Recopilación de datos

En este punto de la ejecución, el usuario indica al programa los datos con los que quiere ejecutar el algoritmo. Como se especificó antes, el usuario debe almacenar el

número de días en los que planea la ruta, el tiempo máximo que puede durar un recorrido por cada camión, el tiempo medio estimado en el que se tarda en recoger cada contenedor, el tipo de ruta, los almacenes tanto de salida como de llegada,los camiones a usar y por último, los puntos de recogida existentes. Todo esto se recoge en la pantalla mostrada en la figura 5.6.

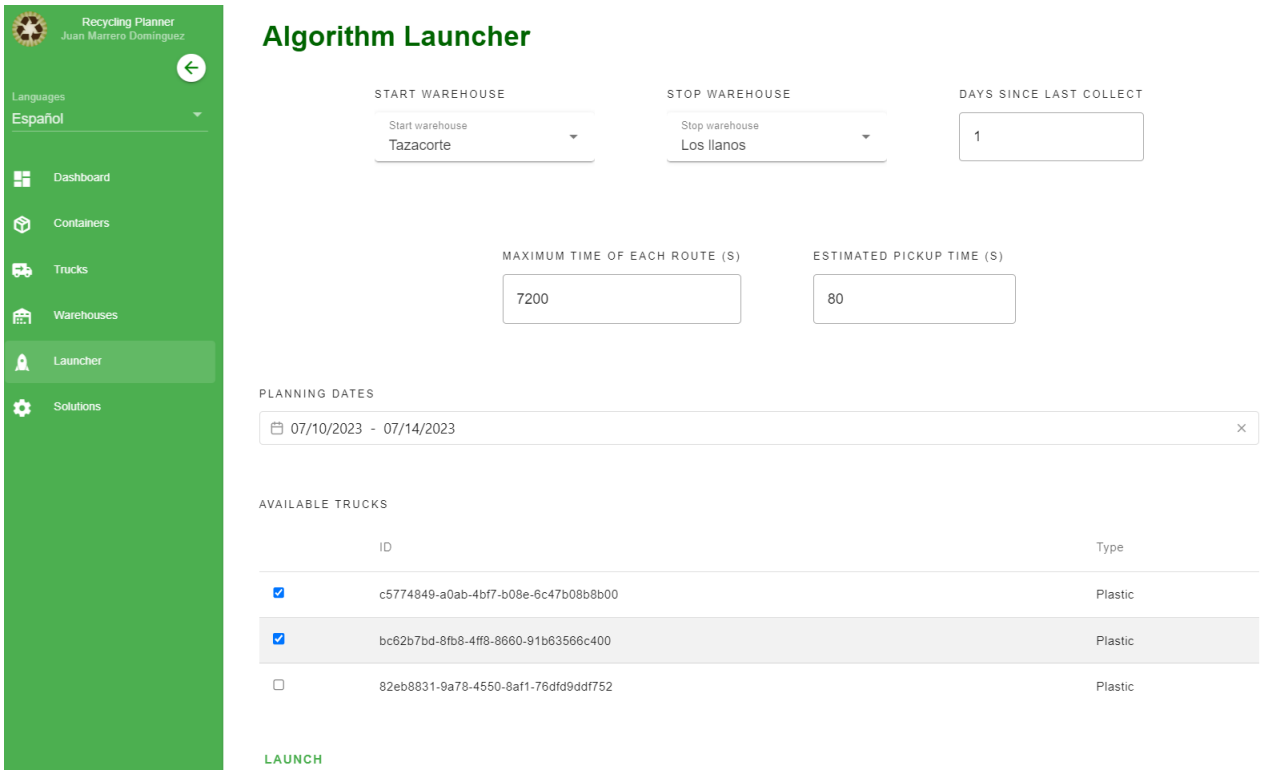


Figura 5.6: Recopilación de datos para el algoritmo

5.3.2. Ejecución del algoritmo

Tras esta recopilación de parámetros, al pulsar el botón se procede a la ejecución del algoritmo. Una vez finalice, el cliente recibe feedback en formas de alertas dependiendo de la respuesta de la aplicación. En la figura 5.7 aparece los 3 estados en los que se puede encontrar la petición de la ejecución del algoritmo.

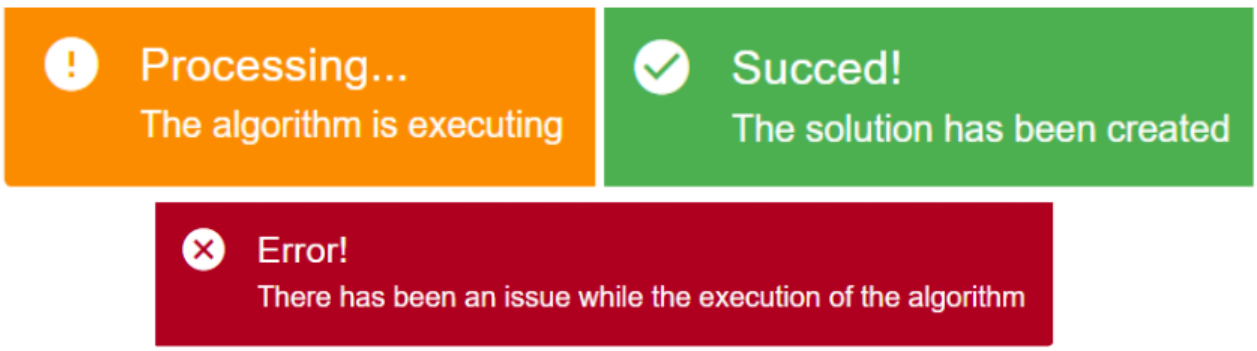


Figura 5.7: Estados de la ejecución del algoritmo

Cuando el algoritmo haya sido ejecutado exitosamente y se muestre un mensaje similar

al de la figura 5.8, se guarda una entidad solución que contiene todos los datos y resultados fruto de los cálculos del algoritmo, incluyendo las rutas para cada día y camión.

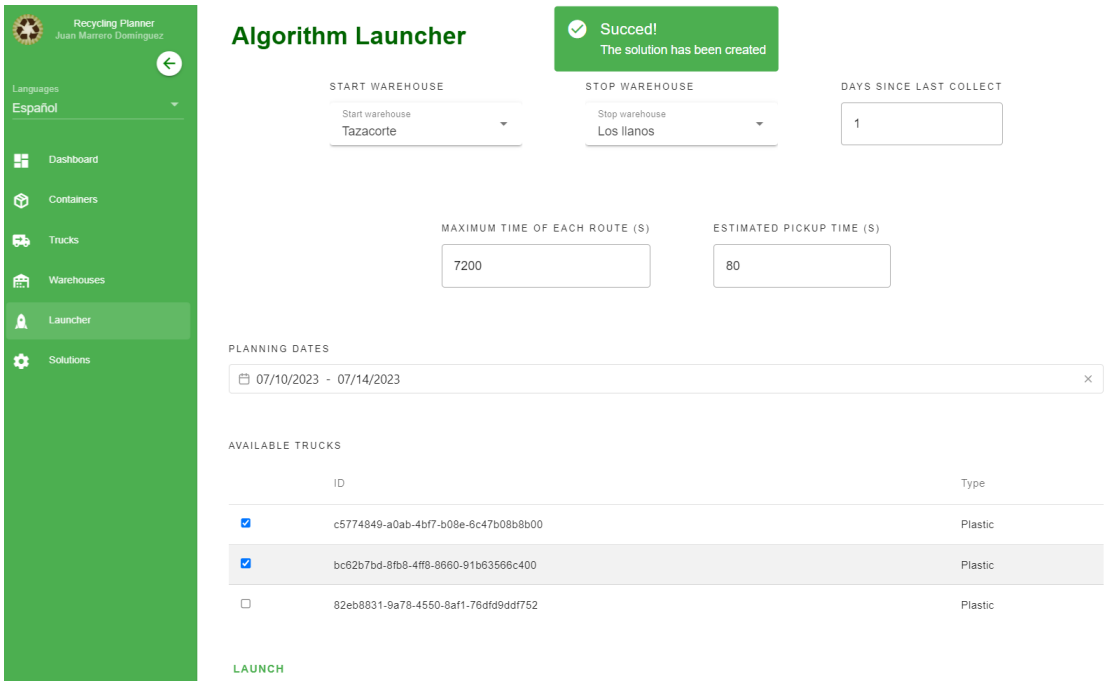


Figura 5.8: Ejecución exitosa

5.3.3. Visualización de soluciones

Ahora el último paso es acceder a la página que lista las soluciones mediante el menú lateral y encontrar la solución recién generada. Si se hace click en el botón de información (explicado en la sección 4.5) se accede a los datos en detalle.

En la sección de indicadores por ruta vemos que hay 8 (reflejado en la figura 5.9), 2 por cada día y para cada camión, y en la de indicadores por día encontramos 4 apartados, uno para cada uno de los mismos(reflejado en la figura 5.10).

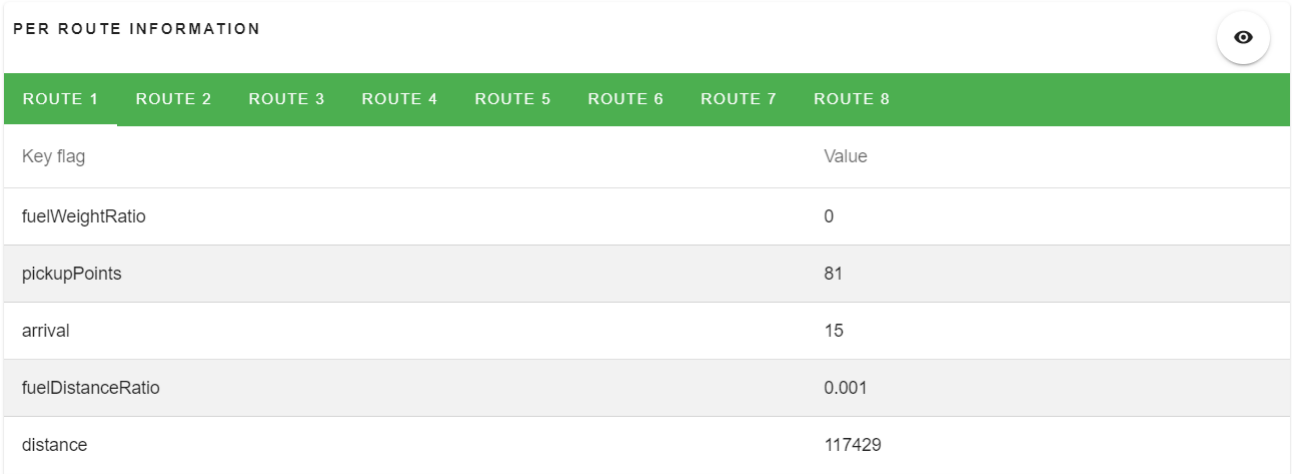
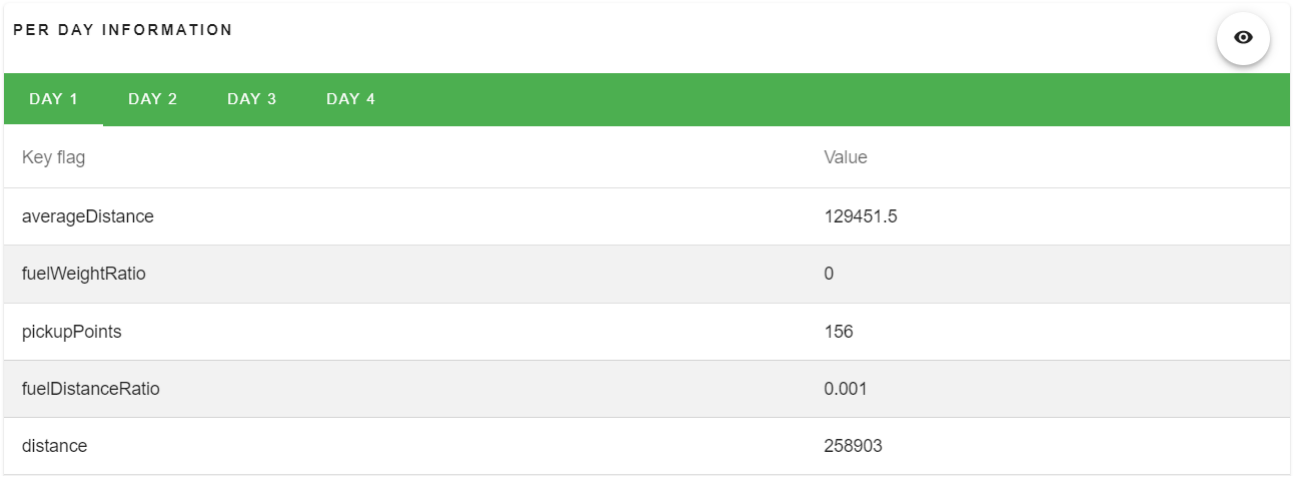


Figura 5.9: Resultado ejemplo: rutas


PER DAY INFORMATION

DAY 1 DAY 2 DAY 3 DAY 4

Key flag	Value
averageDistance	129451.5
fuelWeightRatio	0
pickupPoints	156
fuelDistanceRatio	0.001
distance	258903

Figura 5.10: Resultado ejemplo: días

Además, cada ruta tiene su propio mapa mostrando la ruta en el mapa, la ruta de este ejemplo es la misma que se muestra en el capítulo 4.13. Con todos estos parámetros, el usuario gestor puede planificar y dirigir los recursos necesarios para realizar el trabajo.

5.4. Traducción

Esta funcionalidad permite cambiar el idioma de la aplicación, ya sea al español o al inglés. Usando como ejemplo el cuadro de mandos, pues contiene más información que ninguna otra sección, en la figura 5.5 se observa que está en completo inglés. En contraparte, la figura 5.11 lo muestra traducido al idioma español tras haber cambiado el idioma empleado. Esto incluye etiquetas, botones y cualquier otro elemento de la interfaz que presente texto visible al usuario.

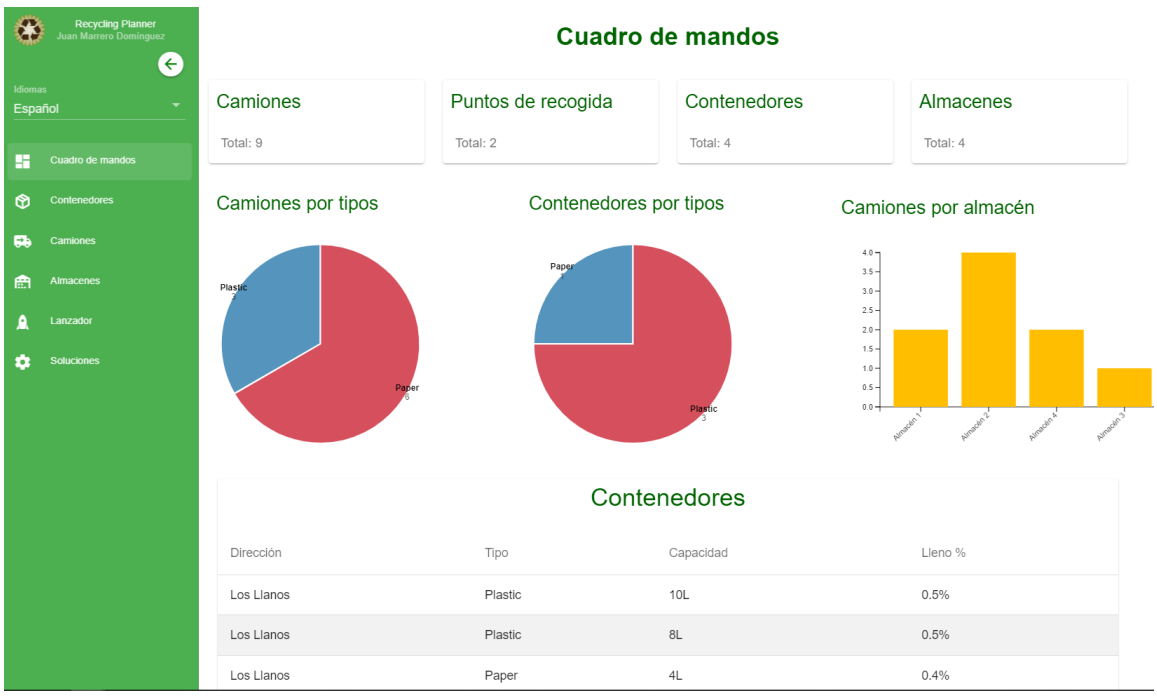


Figura 5.11: Cuadro de mandos en español

5.5. Productor de contenedores

Para realizar una simulación de la creación de los contenedores, se tiene que rellenar un fichero *.yml* como el de la figura 5.12. Como se explica en el capítulo 3.5, son unas variables que dictaminan el comportamiento y resultado de la simulación.

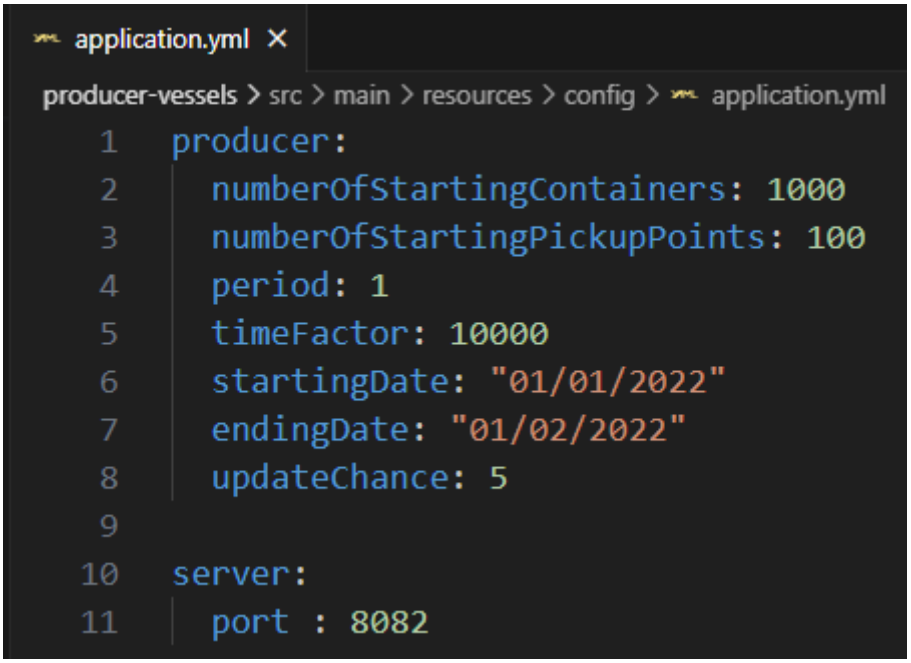


Figura 5.12: Fichero .yml para el productor

Al ejecutar el productor comienza a enviar mensajes, cuando el backend se conecte a escuchar, tramita todos los mensajes que reciba según el comportamiento descrito anteriormente.

En las figuras 5.13 5.14 se comprueba tanto la emisión de los datos por parte del productor como la recepción y las operaciones realizadas por el backend.

Figura 5.13: Productor emitiendo actualizaciones

Figura 5.14: Backend operando según lo que recibe del productor

Capítulo 6

Presupuesto

A lo largo de este capítulo, se presenta un presupuesto estimado para el desarrollo de la aplicación Recycling Planner, proporcionando detalles y consideraciones relevantes que ayudarán a comprender mejor los costos. A medida que avancemos en el desglose del presupuesto, exploraremos las distintas etapas del desarrollo de la aplicación, los recursos necesarios, los costos asociados y los aspectos clave a considerar para garantizar el éxito del proyecto. Antes que nada, vamos a comentar las diferentes fases del desarrollo.

6.1. Análisis y Diseño

Esta fase inicial comprende de las reuniones iniciales en las que se han definido los objetivos, principales retos y requisitos de la aplicación y alcance del proyecto. En esta primera etapa del desarrollo, se decide la arquitectura software, el stack tecnológico y una primera idea de como es la interfaz de usuario.

6.2. Desarrollo

El desarrollo de una aplicación software es una etapa crucial y extensa en el proceso de creación. El desarrollo se puede dividir en varios sub-apartados en las cuales engloba todo lo que ha conllevado el trabajo de creación:

- Configuración de entorno de desarrollo y pruebas.
- Implementación de la lógica de negocio
- Integración de APIs y servicios externos.
- Desarrollo de la interfaz de usuario (Frontend).
- Desarrollo del back-end y la base de datos.

La fase de desarrollo también implica la gestión y seguimiento del progreso del proyecto, con reuniones habituales para asegurarse de que se va por buen camino, así como la resolución de problemas y desafíos técnicos que puedan surgir en el camino.

6.3. Pruebas y Calidad

Esta etapa va prácticamente de la mano con el desarrollo. La etapa de revisión y aseguramiento de la calidad del código es una parte integral del desarrollo de la aplicación. Durante esta etapa, se realizan pausas en el proceso de desarrollo para examinar y evaluar el código escrito, garantizando su calidad y robustez. Se lleva a cabo una exhaustiva prueba de todas las implementaciones agregadas para asegurar su integración correcta con el resto del diseño y evitar errores inesperados en otras áreas del sistema. A medida que el proyecto se escala, se presta especial atención al acoplamiento para evitar impactos negativos en el código. Esta etapa se enfoca en garantizar que el código sea coherente, modular y siga las mejores prácticas de desarrollo, contribuyendo así a la estabilidad y mantenibilidad del software.

6.4. Implementación del algoritmo y versión final

La integración del algoritmo es uno de los aspectos fundamentales de este proyecto. En esta etapa, se acopla el algoritmo de optimización de rutas a la plataforma que ya está desarrollada. Se realizan pruebas exhaustivas para validar la precisión y efectividad del algoritmo en la generación de rutas óptimas para la recolección de reciclaje. Además, se realiza una integración cuidadosa con el resto de la aplicación, asegurando que el algoritmo funcione correctamente en la visualización correcta de la solución. Una vez completada esta etapa, se obtiene la versión final de la aplicación, lista para su implementación y uso en el mundo real.

6.5. Redacción de memoria

A lo largo de este documento, se han presentado las diferentes características del proyecto, desde el análisis y diseño inicial hasta la integración del algoritmo y la versión final de la aplicación. La información recopilada y presentada en esta memoria sirve como guía para comprender mejor el alcance, los recursos necesarios y los costos involucrados en el desarrollo de la aplicación. Por lo tanto, redactarla tiene una importancia grande, lo suficiente para que tenga su propio apartado en el presupuesto.

6.6. Presupuesto final

Actividad	**Horas Estimadas**	**Coste estimado**
Análisis y Diseño	10 horas	150€
Desarrollo	180 horas	2700€
Pruebas y Calidad	60 horas	900€
Integración del algoritmo y versión final	30 horas	450€
Redacción de memoria del proyecto	20 horas	300€
Total	300 horas	4500€

Tabla 6.1: Tabla de coste de tiempo estimado

Según la tabla de coste de tiempo estimado proporcionada 6.1, el trabajo se divide en diferentes actividades, como el análisis y diseño, desarrollo, pruebas y calidad, y la implementación del algoritmo y versión final. Utilizando una tarifa por hora y sumando el tiempo total aproximado de 300 horas, se puede calcular un precio cercano a la realidad. Poniendo el precio a 15€ la hora de un solo programador (la media de un programador en España), el coste asciende hasta los 4500€ solamente en el salario del desarrollador. Sin embargo, es importante recordar que este cálculo es solo una referencia, hay que considerar otros factores para obtener un precio más preciso y adecuado a tu situación y mercado. Es importante tener en cuenta que este precio estimado no incluye otros gastos asociados, como licencias de software o gastos de infraestructura.

Capítulo 7

Conclusiones y líneas de trabajo futuras

Durante el desarrollo de este proyecto se ha logrado implementar con éxito una aplicación de reciclaje y, mediante un exhaustivo análisis y diseño, se ha creado una solución eficiente que optimiza la planificación de las rutas de recogida, contribuyendo así a una gestión más efectiva de los recursos y a la reducción de costos operativos.

A lo largo de la implementación y las pruebas realizadas, se ha demostrado que la aplicación de gestión de residuos, basada en un Sistema de Ayuda a la Decisión, ofrece importantes beneficios en el proceso de la recogida de residuos. Mediante el análisis de variables como el estado en tiempo real de los contenedores y su nivel de llenado o la ubicación de los puntos de recogida, la aplicación permite al usuario gestor de tomar decisiones acertadas que optimicen la labor de los servicios de limpieza. Esta optimización se traduce en la reducción de los tiempos de recorrido y en una mejor distribución de los recursos disponibles, lo que contribuye a agilizar y maximizar los procesos que se deben llevar a cabo.

El Sistema de Ayuda a la Decisión brinda una herramienta poderosa para la toma de decisiones estratégicas en la gestión de residuos, permitiendo una planificación más eficiente y una asignación óptima de los recursos, en beneficio del medio ambiente y de la comunidad en general.

Mirando hacia el futuro y de cara a ampliar esta línea de trabajo, existen diversas oportunidades para seguir mejorando la aplicación. En primera instancia, la generación de rutas en línea recta da un margen de mejora amplio. A futuro se podría implementar una integración con Google Maps u otro sistema de geolocalización que permita generar los caminos respetando la realidad. Se puede hablar de igual manera de la incorporación de funcionalidades adicionales, como la integración con sistemas de geolocalización en tiempo real, lo que permitiría un seguimiento y control más preciso de las rutas y camiones en tiempo real. Así mismo, se podría implementar algoritmos de optimización más avanzados y complejos, que tengan en cuenta variables adicionales como el tráfico o las restricciones de acceso a ciertas áreas. Es posible explorar la posibilidad de ampliar el alcance de la aplicación, considerando otros tipos de materiales reciclables o la gestión de diferentes flotas de vehículos. Esto permitiría adaptar la solución a diferentes contextos y necesidades específicas.

Pero sin duda lo más atractivo sería el desarrollo de una versión móvil enfocada en los trabajadores encargados de la recolección de basura. Esta versión móvil debería contar con características específicas que optimicen su usabilidad en el campo. Sería deseable que la aplicación móvil permita la visualización de las rutas asignadas, la actualización en tiempo real del estado de los contenedores y la capacidad de informar sobre situaciones excepcionales, como contenedores dañados o llenos. Además, sería más fácil implementar la integración con herramientas de navegación y geolocalización comentadas previamente. Serían aspectos clave para facilitar el trabajo diario de los empleados y mejorar la eficiencia en la recolección de residuos.

Capítulo 8

Conclusions and further research

During the development of this project, a recycling application has been successfully implemented and, through an exhaustive analysis and design, an efficient solution has been created that optimizes the planning of collection routes, contributing to effective management of resources and the reduction of operating costs.

Throughout the implementation and testing, it has been demonstrated that the Recycling Planner application, based on a Decision Support System (DSS), offers significant benefits in the waste collection process. Through the analysis of variables such as the real-time status of the containers and their filling level or the location of the collection points, the application allows the user-manager to make the right decisions to optimize the work of the cleaning services. This optimization translates into a reduction in travel times and a better distribution of available resources, which contributes to speed up and maximize the processes to be carried out.

The Decision Support System provides a powerful tool for strategic decision making in waste management, allowing for more efficient planning and optimal allocation of resources for the benefit of the environment and the community at large.

Looking ahead to the future and extending this line of work, there are several opportunities to further improve the application. In the first instance, the generation of straight-line routes provides broad room for improvement. In the future, an integration with Google Maps or another geolocation system could be implemented to generate routes in agreement with reality. It is also possible to incorporate additional functionalities, such as integration with real-time geolocation systems, which would allow a more precise monitoring and control of routes and trucks in real time. Also, more advanced and complex optimization algorithms could be implemented, taking into account additional variables such as traffic or access restrictions to certain areas. It is also possible to explore the possibility of expanding the scope of the application, considering other types of recyclable materials or the management of different types of vehicles. This would allow the solution to be adapted to different contexts and specific needs.

But undoubtedly the most attractive feature would be the development of a mobile version focused on workers in charge of garbage collection. This mobile version should have specific features that optimize its usability in the field. It would be desirable for the mobile application to allow visualization of assigned routes, real-time updating of

container status and the ability to report exceptional situations, such as damaged items or full containers that are overflowing. In addition, it would be easier to implement integration with navigation and geolocation tools previously discussed. These would be key aspects to make easier the daily work of employees and improve efficiency in waste collection.

Bibliografía

[1] Jaydeep Lella, Venkata Ravibabu Mandla, and Xuan Zhu. Solid waste collection/transport optimization and vegetation land cover estimation using geographic information system (gis): A case study of a proposed smart-city. *Sustainable Cities and Society*, 35:336–349, 2017.

[2] Gloria Philips-Wren, Mary Daly, Daniel Power, and Frederic Adam. A critical review of decision support systems foundational articles. *AIS Electronic library*, 2017.

[3] R. Pell T. L. Saaty, P. Rogers. Portfolio selection through hierarchies. *Journal of Portafolio Management*, 6(3):16–21, 1988.