ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Smart Waste Management System: Optimización de rutas de recolección de residuos mediante IoT

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Telemática

Presentado por:

Javier Isaac Dillon Jiménez

Danny Rogelio Medina Moncayo

GUAYAQUIL - ECUADOR

Año: 2023

DEDICATORIA

JAVIER DILLON JIMÉNEZ

A Dios, fuente de sabiduría y guía constante en este camino académico. A mi amada familia, cuyo amor y apoyo incondicional han sido mi mayor fortaleza.

DANNY MEDINA MONCAYO

Dedico el presente proyecto a mis padres, Leonor Piedad Moncayo Alvarado y Roberto Guillermo Medina Vera, quiénes han sido el soporte más grande que he tenido a lo largo de este recorrido y creyeron en mí desde el primer momento y siempre me mostraron su apoyo incondicional sin importar cuán difícil se haya tornado la situación.

AGRADECIMIENTOS

JAVIER DILLON JIMÉNEZ

Agradezco a Dios por su guía constante y a mi familia por ser mi roca. A mi novia, por ser mi inspiración diaria. A mis tíos y amigos, gracias por su apoyo y enseñanzas. Este logro es el resultado del respaldo y la colaboración de cada persona que ha sido parte de este viaje.

DANNY MEDINA MONCAYO

Agradezco a Dios, mis padres, familiares, profesores y amigos, especialmente a Gabriela, mi amiga desde hace muchos años, por su apoyo constante y siempre estar presente. A mis amigos de carrera, Nathaly, Andrea y Nicolás, han hecho de mi último año en ESPOL uno de los mejores, gracias por su incondicional apoyo. Por último, me gustaría agradecerme a mí mismo por no rendirme y ser persistente en conseguir esta meta y poder decir, finalmente, “¡Lo lograste!”

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; JAVIER DILLON JIMÉNEZ y DANNY MEDINA MONCAYO damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”

JAVIER DILLON JIMÉNEZ

DANNY MEDINA MONCAYO

EVALUADORES

Firmado electrónicamente por:
MARIA JOSE RAMIREZ PRADO

Mgtr. María José Ramírez

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
WASHINGTON ADRIAN VELASQUEZ VARGAS

Ph.D. Washington Velásquez Vargas

PROFESOR TUTOR

RESUMEN

La aplicación de las tecnologías emergentes, en la generación desbocada de residuos, se centra en el manejo y control de los desechos sólidos, siendo estos los que constituyen en gran medida a la contaminación del ecosistema. Además, el poco o nulo tratamiento de estos desechos, ocasionado por la manera de incorrecta de realizar su respectiva recolección, ha contribuido significativamente al incremento y amontonamiento desmedido de los residuos en zonas urbanas.

El siguiente trabajo implementa una aplicación Web que permite generar una ruta de recolección de residuos acorde a la posición de los diferentes puntos de recolección donde se deba llevar a cabo esta actividad. En torno a la optimización del tiempo de recolección de los residuos, se propone la utilización de rutas generadas mediante el uso del algoritmo cuántico QPSO-LR, el cual genera la ruta más optima entre todos aquellos puntos que requieran ser vaciados.

Este trabajo está dividido en 5 capítulos. El capítulo 1 abarca el origen de la problemática, justificación, objetivos, alcance, limitaciones y el Estado del Arte. Capítulo 2 presenta la Metodología con la arquitectura, componentes y materiales. Capítulo 3 describe la solución, que consistió en implementar el aplicativo usando tecnologías Open Source así como los servicios de Google Maps Platform y AWS, en conjunto con el algoritmo QPSO-LR, para determinar la ruta más optimizada. Capítulo 4 analiza los resultados obtenidos, relacionados al escenario propuesto para evaluar la efectividad de la aplicación. Finalmente, el capítulo 5 detalla las conclusiones, recomendaciones y trabajo futuros alrededor de la propuesta.

Palabras Clave: Desechos Sólidos, Ruta de recolección, QPSO-LR, Google Mapas Platform, AWS

ABSTRACT

The application of emerging technologies, in the rampant generation of waste, focuses on the management and control of solid waste, which largely contributes to ecosystem pollution. Furthermore, the inadequate treatment of these wastes, caused by incorrect collection methods, has significantly contributed to the increase and disproportionate accumulation of waste in urban areas.

The following work implements a web application that allows for the generation of a waste collection route based on the positions of different collection points where this activity should take place. Regarding the optimization of waste collection time, the use of routes generated by the Quantum Particle Swarm Optimization algorithm (QPSO-LR) is proposed, which generates the most optimal route among all points that need to be emptied.

This work is divided into 5 chapters. Chapter 1 covers the origin of the problem, justification, objectives, scope, limitations, and the State of the Art. Chapter 2 presents the Methodology with the architecture, components, and materials. Chapter 3 describes the solution, which involved implementing the application using Open-Source technologies as well as Google Maps Platform and AWS services, in conjunction with the QPSO-LR algorithm, to determine the most optimized route. Chapter 4 analyzes the results obtained, related to the proposed scenario to evaluate the effectiveness of the application. Finally, Chapter 5 details the conclusions, recommendations, and future work surrounding the proposal.

Keywords: Solid Waste, Collection Route, QPSO-LR, Google Mapas Platform, AWS

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
IoT	Internet of Things
PWA	Progressive Web Apps
RSU	Residuos Sólidos Urbanos
CVRP	Capacitated Vehicle Routing Problem
API	Application Programming Interface
cGA	Compact Genetic Algorithm
ACO	Ant colony optimization
QPSO	Quantum Particle Swarm Optimization
QPSO-LR	Quantum Particle Swarm Optimization with Lorentzian Function
QPSO-RM	Quantum Particle Swarm Optimization with Rosen-Morse Function
QPSO-CS	Quantum Particle Swarm Optimization with Coulomb's Square Function
WSN	Wireless Sensor Network
LOS	Line of Sight
PSO	Particle swarm optimization
AWS	Amazon Web Services
FIEC	Facultad de Ingeniería en Electricidad y Computación
FICT	Facultad de Ingeniería en Ciencias de la Tierra
FIMCM	Facultad de Ingeniería Marítima y Ciencias del Mar
FSCH	Facultad de Ciencias Sociales y Humanísticas
CELEX	Centro de Lenguas Extranjeras
FADCOM	Facultad de Arte, Diseño y Comunicación Audiovisual
FCNM	Facultad de Ciencias Naturales y Matemáticas
FCV	Facultad de Ciencias de la Vida
FIMCP	Facultad de Ingeniería en Mecánica y Ciencias de la Producción

SIMBOLOGÍA

bps	Bits por segundo
kbps	Kilobits por segundo
ms	Milisegundos
s	Segundos

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CODIGOS DE PROGRAMA

CAPÍTULO 1

1. INTRODUCCIÓN

El manejo adecuado de los residuos es crucial para conseguir un ambiente más sustentable y sostenible para la vida humana, animal y vegetal que habita en el planeta (Carrera and Mendez, 2020). Actualmente, los índices medio ambientales muestran un crecimiento exponencial de residuos orgánicos debido a la sobrepoblación global presente en el planeta. El poco o nulo manejo de los residuos afectan de manera directa al ecosistema, puesto que estos producen dióxido de carbono (CO2) al descomponerse, ocasionando contaminación medio ambiental e incluso afectando la calidad de vida de los seres vivos (García Chumacero and Guerrero Olaya, 2023). Este problema se hace presente en mayor medida en las zonas urbanas, donde se generan miles de toneladas de basura diariamente, las cuales no suelen tener un tratamiento adecuado para minimizar el índice de contaminación presente en los residuos y el impacto que estos producen (Dedios Carrasco, 2023).

De esta manera, surge el presente proyecto integrador que nace de la necesidad de mejorar la forma de recolectar y tratar los residuos de manera responsable con el planeta. El proyecto presenta una solución programable que consiga integrar tecnologías IoT con técnicas de optimización de rutas basados en algoritmos cuánticos para obtener el recorrido más eficiente en la recolección de residuos de los diferentes puntos del campus Gustavo Galindo en ESPOL (Zambrano and Cobeña, 2020).

Se plantea la implementación de un sistema de recolección de residuos que muestre la ruta más optima en tiempo real acorde al estado de cada uno de los contenedores (Betanzo-Quezada et al., 2016). El sistema se encuentra equipado con un hardware básico que sirve para simular el comportamiento real de un contenedor de basura, donde se busca recopilar datos, que también serán simulados y ayudarán al sistema a obtener la mejor ruta. Además, constará de un componente de software donde se muestra cuál

es la ruta más eficiente que se debe tomar para la recolección de residuos.

A continuación se describe la problemática existente en torno al manejo de los desechos sólidos y la manera poco eficiente en que estos son recolectados (Calle-Loyola and Solís-Muñoz, 2021). También, se presenta una justificación del por qué este problema debe ser resuelto usando modelos y algoritmos de optimización cuánticos. Se presenta los objetivos con los cuales se prevé dar solución al problema. El alcance y limitación de la solución propuesta y el estado del arte donde se mencionan las diversas formas en que han dado solución a este problema usando diferentes tecnologías a nivel de Software y Hardware.

1.1 Descripción del problema

La gestión ineficiente de los residuos sólidos ha emergido como un desafío global, con consecuencias significativas para el medio ambiente y la sostenibilidad. Este problema se manifiesta de diversas formas alrededor del mundo, afectando tanto a entornos urbanos como rurales. Uno de los resultados más visibles del manejo deficiente de residuos sólidos es la acumulación de desechos, la cual puede generar un ambiente poco agradable y desgastado, afectando la calidad de vida de los seres vivos. Esto no solo altera el entorno social, sino que también contribuye a la contaminación del ecosistema (Bermello Giler, 2021). Además, los desafíos en la gestión de residuos sólidos son variados y complejos, la mayoría de ellos se centran en el impacto negativo que generan al momento de ser recolectados. La falta de infraestructuras adecuadas, sistemas de recolección eficientes y conciencia pública sobre la importancia del manejo adecuado de los desechos contribuye a esta problemática global(Guamán Pacheco and López López, 2023).

Preservar el ecosistema y promover prácticas de desarrollo sostenible son los principales objetivos para la conservación del medio ambiente(de Romero et al., 2020). Sin embargo, la contaminación de desechos no solo es generada de manera personal o en pocas cantidades, las grandes empresas, organizaciones e instituciones generan una cantidad significativa y diversa de residuos, que van desde desechos sólidos en descomposición hasta residuos que pueden ser reciclados, y requieren estrategias eficientes de recolección para minimizar su impacto negativo (Prudencio Cruz, 2021).

Actualmente, la gestión de residuos se ha convertido en uno de los más grandes desafíos ambientales por enfrentar, ya que existen procesos de recolección que son dañinos al momento de realizarlos. Esta falta de eficiencia no solo aumenta los costos operativos sino el tiempo de trabajo y salud de las personas encargadas de realizar esta actividad (Mayorga et al., 2021). Por lo general, los procesos de recolección se basan en horarios predefinidos y no tiene en cuenta el nivel real de llenado de los contenedores. Esto conduce a una gestión ineficiente de los recursos, ya que se pueden realizar recolecciones innecesarias o, en contraste, se pueden pasar por alto contenedores llenos(Choez-Segovia et al., 2021). También, la falta de datos en tiempo real sobre la capacidad de los contenedores dificulta la toma de decisiones informadas para mejorar la gestión de residuos. En este contexto, se considera necesario crear rutas de recolección optimizadas que permitan un manejo más eficiente de los desechos (Cardenas Cabrera and Cuadra Arevalo, 2022). Este proyecto propone el uso de tecnologías IoT para el desarrollo de una aplicación web que consiga generar una ruta de recolección entre aquellos contenedores que necesiten ser vaciados.

1.2 Justificación del problema

La implementación de un sistema inteligente de gestión de residuos representa un paso significativo hacia la resolución de los desafíos actuales en la administración de estos, ya que se conseguiría una mejora significativa en la eficiencia del personal encargado y una reducción notable en los tiempos de recolección asociados con la gestión de residuos. Esta eficiencia se traduciría en un uso más sostenible de los recursos, alineándose con los principios de responsabilidad ambiental (Valdivia Castillo and Garcia Chirinos, 2022). Además, uno de los elementos clave de este sistema es la utilización de sensores de llenado en los contenedores de residuos. Estos sensores permiten recopilar datos en tiempo real sobre el nivel de llenado de cada contenedor (Montañez Gomez, 2021). Esto representa un cambio fundamental en comparación con los métodos tradicionales basados en horarios predefinidos, ya que ahora la recolección puede programarse de manera más precisa, atendiendo a la capacidad real de los contenedores en lugar de seguir un cronograma estático (Contreras Silva, 2021).

Por otro lado, La reducción de la frecuencia de recolección no solo disminuye la huella

de carbono asociada con la movilidad de los vehículos de recolección, sino que también contribuye a un ambiente más limpio y agradable. Para conseguirlo, en este proyecto, se realiza la implementación de algoritmos cuánticos para la optimización de rutas, los cuáles desempeñan un papel crucial dentro del sistema (Sung and Hsiao, 2022). Al analizar los datos recopilados por los sensores, estos algoritmos pueden generar rutas de recolección más eficientes. Esto no solo implica una reducción en el tiempo dedicado a la recolección, sino también una disminución significativa de los costos operativos asociados con la gestión de residuos (Cáceres Sánchez, 2022). Los recursos, como combustible y horas de trabajo, se utilizan de manera más eficiente, contribuyendo así a la sostenibilidad financiera y operativa. Además, al reducir la frecuencia de recolección innecesaria y garantizar que los contenedores se vacíen cuando sea necesario, se contribuye a la reducción general de residuos y se fomenta un entorno más limpio y ordenado (Machado and Saldaña, 2022).

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar un aplicativo PWA que utilice algoritmos de optimización basados en datos en tiempo real para la localización de la mejor ruta de recolección de desechos dentro del campus Gustavo Galindo de ESPOL.

1.3.2 Objetivos específicos

- Diseñar el hardware de sensores IoT instalados en los contenedores para monitorear el estado en el campus de la ESPOL, aplicando los principios de diseño de sistemas de sensores.
- Emplear un algoritmo de optimización basado en el movimiento de partículas cuánticas utilizando geocalización y monitoreo en tiempo real para el descubrimiento de una ruta optima de recolección de residuos.
- Integrar los servicios de AWS y Google Maps dentro del aplicativo para conseguir mostrar el mapa, los puntos de recolección que deberán ser vaciados y generar la ruta más optima entre ellos.

1.4 Alcance y Limitaciones

1.4.1 Alcance

El proyecto tiene un alcance local, es decir, será implementado dentro del campus de la ESPOL. Se contabilizará la cantidad de contenedores de basura que se encuentran ubicados en las diferentes facultades para, posteriormente, marcar rutas estratégicas que faciliten la recolección y disminuyan el tiempo que se emplea en esta actividad. El sistema será portable, es decir, se podrá acceder a éste desde un dispositivo móvil para llevar el control y monitoreo del estado de los contenedores y recibir notificaciones acerca de las rutas.

1.4.2 Limitaciones

Las limitaciones que se han encontrado en la implementación del sistema se centran principalmente en la elección de los contenedores para generar la ruta. Se tomaron en consideración aquellos puntos que se encontraron fuera de las aulas de clases y oficinas, es decir aquellos puntos donde la concentración de personas es mayor en diferentes horas del día y no existe un control ni manejo adecuado de los objetos que se desechan

Encontrar un herramienta de mapeo que sirva para marcar rutas de manera más precisa ha significado un desafío, ya que, actualmente, no existe ningún registro de las ubicaciones de los contenedores en el campus, por lo que hay que identificar cada uno de ellos y obtener sus latitudes y longitudes para marcar sus posiciones exactas y usarlas en el código para generar la ruta más optima.

La elección del Framework también fue un desafío, esto se debe a que montar un aplicativo móvil con todos los requerimentos que se solicitan tales como: Segmentación de contenedores por facultad, descubrimiento de la mejor ruta y notificaciones de monitoreo y control de los contenedores, es complicado debido al tiempo de implementación que se tiene para realizar el sistema. Por lo que, se va a implementar un aplicativo web que, posteriormente, mediante el uso de una PWA, se podrá lanzar la aplicación en cualquier sistema operativo móvil, debido a que son aplicativos genéricos.

1.5 Estado del Arte

La problemática de los residuos sólidos urbanos (RSU) constituye un desafío crucial en la actualidad, manifestando un crecimiento exponencial que desencadena consecuencias sociales, económicas y ambientales significativas. El aumento constante en la generación de desechos, combinado con prácticas ineficientes de manejo, ha llevado a la saturación de vertederos, la contaminación del entorno y la pérdida de recursos valiosos. Este escenario refleja la urgente necesidad de replantear y optimizar los sistemas de recolección y gestión de RSU. Con el transcurrir de los años y el avance de tecnologías más sofisticadas y eficientes, se ha logrado implementar dispositivos para conseguir un mayor monitoreo de estos desechos (Jiménez Ordoñez and Jiménez Ordoñez, 2020).

Los ciudadanos son el principal factor por el cual los desechos poseen una gestión poco asertiva sobre su tratamiento (Gómez and Mozo, 2021). La importancia del manejo adecuado de los residuos sólidos implica la complicidad de muchas entidades gubernamentales, industria, el comercio y la sociedad en general, así como la responsabilidad de las autoridades municipales en la elaboración de un marco general de información o diagnóstico con el cual se reconozcan y definan los problemas relacionados desde la generación hasta la disposición final de los residuos sólidos.

En la cuidad de Ibarra se ha implementado un sistema para automatizar y sistematizar la recolección de residuos sólidos urbanos (RSU) en el sector “El Olivo”, utilizando sensores ultrasónicos y la tecnología SigFox para IoT (Obando Suárez, 2022). El sistema detecta el nivel de almacenamiento en tiempo real de los contenedores de basura, genera rutas de recolección para el personal encargado y proporciona información relevante para los residentes del sector. También, el sistema propuesto utiliza sensores remotos para detectar el nivel de llenado de los contenedores, transmitiendo los datos a un microcontrolador que los procesa y clasifica en tres niveles: vacío, medio y lleno. Estos datos se envían de manera inalámbrica a la red de transmisión SigFox, la cual envía un mensaje cada hora del día con la información del nivel de cada contenedor a la nube de SigFox. Esta información sirve de base para la toma de decisiones en el algoritmo de aprendizaje de la ruta óptima, que considera la ubicación predefinida en el sistema junto con el nivel de almacenamiento obtenido en tiempo real de cada contenedor. Además, todos los datos que se obtienen por medio de los sensores son enviados a una aplicación

web y móvil que muestran el estado de los contenedores, su posición, la ruta óptima y otros datos relevantes. El artículo menciona que el sistema también permite el análisis de patrones de comportamiento de las personas con sus desperdicios, lo que es información importante para la optimización de la logística y recolección de residuos, así como para mejorar los hábitos de la población en el consumo y desperdicio de alimentos y otros materiales de uso diario.

Sin embargo, el desafío que presenta conocer, con la mayor exactitud posible, el nivel de llenado de cada uno de los contenedores sigue siendo una de las preocupaciones más grandes al momento de llevar un control o monitoreo de los sensores. A nivel nacional se han implementado sistemas similares, pero que van dirigidos a distintos sectores usando distintas tecnologías.

En (Muñoz Santiana, 2021) se presenta un sistema de optimización de las rutas de recolección de residuos sólidos en el cantón Latacunga, a cargo de la Empresa Pública de Aseo y Gestión Ambiental (EPAGAL). El proyecto se enmarca en la problemática global de la recolección y transporte de residuos sólidos, especialmente en América Latina y el Caribe, donde se evidencia una preocupación por los altos costos asociados a este proceso. El autor indica que acorde a estudios del Banco Mundial y el Banco Interamericano de Desarrollo, los costos de recolección y transporte de residuos sólidos varían entre $22 y $197 USD por tonelada recolectada en ciudades de América Latina. El objetivo principal es minimizar los costos de operación de cada ruta, mejorar la capacidad de los vehículos, agilizar el trabajo, reducir las horas extras y disminuir el impacto ambiental. Para lograr esto, se propone la implementación de la metodología de ruteo de vehículos con capacidad limitada (CVRP), donde se recopila información detallada sobre las rutas de recolección de residuos sólidos, los vehículos utilizados, los puntos de acopio, y se analizan los datos obtenidos para identificar oportunidades de optimización. Para la implementación del sistema, el autor propone un modelo matemático para la optimización de las rutas, que será desarrollado con la ayuda de software especializado como ILOG CPEX Optimization Studio. Este modelo permitirá diseñar una distribución óptima de rutas que minimice los costos y mejore la cobertura del servicio de recolección y transporte de residuos sólidos en el cantón Latacunga. Además, se destaca la importancia de cumplir con normativas ambientales y de calidad en el manejo y disposición final de desechos sólidos.

En instituciones universitarias también han replicado este tipo de sistemas. (Guerrero Garcia, Limones Obando, et al., 2018) presenta la implementación de un sistema de monitoreo y optimización de desechos sólidos en el Campus Gustavo Galindo de la ESPOL. El objetivo general del que propone el autor es desarrollar un prototipo de contenedor inteligente e implementar una plataforma web que monitoree el nivel de los desechos sólidos en los contenedores distribuidos en el Campus Gustavo Galindo. Para lograr esto, se propone el uso de tecnologías en vanguardia, como dispositivos y redes orientados a Internet de las cosas (IoT), con un enfoque en la viabilidad de proyectos IoT, el análisis exhaustivo del consumo de energía, la disponibilidad de dispositivos y la factibilidad de la carga y descarga de datos. El prototipo propuesto utiliza tecnología LPWAN con la modulación LoRa y el protocolo LoRaWAN para la comunicación inalámbrica, junto con un sensor ultrasónico HC-SR04 para medir el nivel de saturación de los contenedores. Se destaca que el dispositivo está alimentado con una batería LiPo, lo que le permite funcionar de manera autónoma sin necesidad de una fuente externa. Los datos recopilados por el prototipo son almacenados en la plataforma The Thing Network (TTN) y posteriormente procesados y presentados en una plataforma web desarrollada para el monitoreo y visualización de los datos obtenidos. Además, el autor menciona que se aborda la experimentación para determinar el rango de transmisión de LoRaWAN entre el gateway y los dispositivos finales ubicados en el campus, así como el análisis del modelo de pérdida de trayectoria para este protocolo. Además, se incluye la implementación de una sección de reportes en la plataforma web, que permite a los usuarios generar consultas a la base de datos para elaborar reportes históricos y generar archivos CSV.

La implementación de sistemas de recolección de residuos en diversos países, al margen de Ecuador, se ha convertido en una necesidad imperante para abordar los crecientes desafíos ambientales a nivel global. En respuesta a la expansión de las áreas urbanas y el aumento de la producción de residuos, numerosas naciones han buscado desarrollar y perfeccionar sus propios sistemas de gestión de residuos sólidos (Chafla Borja, 2020). Estos esfuerzos abarcan desde la introducción de tecnologías innovadoras hasta la implementación de políticas efectivas que fomenten la reducción, reutilización y reciclaje de desechos.

En (Ferrer and Alba, 2019) se presenta un sistema que aborda el desafío de gestionar

de manera óptima la recolección de residuos urbanos, que representa el 70% del costo operativo en el tratamiento de residuos. El objetivo principal del sistema es reducir el costo del servicio de recolección de residuos minimizando la distancia recorrida por un camión para recolectar los residuos de un contenedor, lo que a su vez reduce el consumo de combustible. Al mismo tiempo, se aumenta la calidad del servicio para los ciudadanos al evitar los desbordamientos molestos de los contenedores gracias a las precisas predicciones del nivel de llenado proporcionadas por el sistema. Su implementación se lleva acabo en una ciudad española, donde se consigue demostrar que el uso de este evita el viaje innecesario de los trabajadores hacía los contenedores, reduce la distancia recorrida para recolectar un contenedor en un 20% y genera rutas un 33.2% más cortas que las rutas utilizadas por la empresa, lo que permite una reducción considerable de costos totales y emisiones dañinas. Además, el sistema propuesto aborda dos grandes problemas principales que enfrenta un servicio de recolección de residuos: qué contenedores deben ser recolectados y en qué orden deben ser visitados para minimizar el costo. Para resolver la primera cuestión, se utilizan técnicas de aprendizaje automático para estimar el nivel de llenado de cada contenedor, generando predicciones a partir de datos históricos para decidir cuándo un contenedor debe ser recolectado y así poder interactuar con un sistema de Internet de las cosas (IoT) para obtener el llenado real de los contenedores equipados con sensores volumétricos. En cuanto a la generación de rutas, el sistema utiliza algoritmos para calcular soluciones óptimas, es decir, las rutas completas que los camiones deben seguir para recolectar los contenedores seleccionados.

En zonas urbanas es muy común encontrar este tipo de problemas, los desechos sólidos no tienen un manejo adecuado y ocasionan un aumento en los niveles de contaminación ambiental que van en aumento día tras día. En (López et al., 2022), se plantea la problemática de la recolección de residuos tecnológicos en zonas urbanas, destacando la importancia de implementar estrategias de reciclaje y fortalecer una economía circular para abordar el creciente problema de los desechos tecnológicos. Se menciona que la recolección y transporte de residuos representa entre el 60% y el 80% de los costos totales del Sistema de Gestión de Residuos, lo que lo convierte en un factor crítico en el gasto fiscal de los sistemas de gestión de residuos. Para abordar este problema, los autores proponen la aplicación de dos metaheurísticas poblacionales:

el algoritmo genético celular (cGA) y el algoritmo de optimización basado en Colonia de Hormigas (ACO) mediante la creación de un sistema de contenedor inteligente que informe el nivel de llenado de los contenedores de residuos, con el fin de evitar recorridos innecesarios y optimizar la recolección para monitorear y controlar el nivel de llenado de los contenedores a través de internet, permitiendo la generación de rutas de recolección únicamente con aquellos contenedores que requieran ser vaciados. Estos algoritmos se utilizan para generar recorridos óptimos para la recolección de residuos tecnológicos en las ciudades de Caleta Olivia y Comodoro Rivadavia, Argentina. El algoritmo ACO mostró la obtención de las mejores rutas, minimizando los kilómetros recorridos en comparación con las obtenidas por cGA.

(Hernández Navarro, 2019) propone abordar esta problemática usando otro tipo de tecnologías de Software como lo son los algoritmos genéticos. El sistema propuesto supervisa y regula el grado de ocupación de los contenedores mediante herramientas de Hardware IoT, posibilitando la creación de rutas de recolección exclusivamente para aquellos contenedores que necesiten ser vaciados. La incorporación de algoritmos genéticos multiobjetivo simplificó la ejecución del sistema propuesto, ya que su operación específica, diseñada para abordar problemas con el objetivo de minimizar tanto la distancia como la elevación del terreno en una ruta de recolección de residuos sólidos urbanos (RSU), se adaptó de manera eficiente tomando en cuenta restricciones clave en el proceso. También, se menciona que el costo de la recolección constituye entre el 70% y el 85% del costo total del manejo de los RSU, y se detallan factores que determinan dicho costo, como los vehículos recolectores, el personal de limpia, la frecuencia de recolección, las distancias recorridas y el tiempo invertido en el servicio.

(Ricatti et al., 2017) propone la implementación de un sistema de recolección que se enfoque en validar la factibilidad y necesidad de la utilización de sensores y rutas dinámicas para mejorar el proceso de recolección. Se establece que la solución propuesta no contempla la utilización de la plataforma y sensores para residuos domiciliarios, ya que la recolección de estos sigue un proceso completamente diferente. Además, se especifica que los sensores no funcionarán en condiciones extremas, como temperaturas bajo cero, demasiado elevadas, inundaciones y otras situaciones climáticas fuera de lo normal. El sistema va dirigido a mayores escalas, su implementación se centra en la ciudad de Dublin, Irlanda. Se menciona que el problema principal en la recolección de residuos

de los cestos de basura en la vía pública es la falta de eficiencia debido a la escasez de información en el proceso. Por lo tanto, el enfoque del proyecto es implementar un prototipo de sensores de capacidad conectados que permitirán optimizar las rutas de recolección y la planificación de instalación de nuevos basureros. El autor destaca la importancia del análisis de la información, proporcionado por las autoridades locales de la ciudad, para tomar decisiones tanto de diseño como de negocio, y se menciona que la validación inicial de si la información a recolectar sirve se realizará en forma manual.

En el constante esfuerzo por mejorar la eficiencia de la gestión de residuos sólidos urbanos (RSU), la integración de enfoques avanzados, como el uso de algoritmos genéticos, ha emergido nuevas tecnologías que permitan conseguir una mayor precisión al momento de implementar sistemas más robustos y eficaces (Chaveinte García et al., 2023). La complejidad inherente a la determinación de rutas óptimas para la recolección de residuos exige métodos que no solo aborden múltiples variables, como distancia y tiempo de recolección, sino que también consideren la variabilidad dinámica de los datos en tiempo real. En este contexto, los algoritmos cuánticos, basados en los principios de la mecánica cuántica, ofrecen un enfoque innovador para abordar problemas complejos de optimización, como la determinación de rutas eficientes. Este enfoque promete superar las limitaciones de los métodos clásicos al explorar múltiples soluciones simultáneamente y encontrar resultados más rápidos y precisos (Kumar et al., 2020).

En (Alvarez-Alvarado et al., 2021), los autores proponen tres nuevos algoritmos de optimización inspirados en la mecánica cuántica y el comportamiento de enjambres. Estos algoritmos se centran en la exploración y explotación de espacios de búsqueda, con el objetivo de encontrar soluciones óptimas para problemas de optimización. Los autores describen la importancia de la exploración, que se refiere a la capacidad de examinar ampliamente las áreas prometedoras del espacio de búsqueda, así como la explotación, que se relaciona con la capacidad de aprovechar las soluciones prometedoras encontradas. También, mencionan que estos tres algoritmos se basan el algoritmo QPSO, el cual es una técnica avanzada de optimización heurística que utiliza el concepto de movimiento cuántico de partículas para alcanzar la solución óptima. El proceso implica la identificación de estas posiciones para guiar el movimiento de las partículas hacia soluciones óptimas. Se proponen tres nuevos algoritmos de optimización de enjambre con comportamiento cuántico basados en los campos

potenciales de Lorentz (QPSO-LR), Rosen-Morse (QPSO-RM) y raíz cuadrada tipo Coulomb (QPSO-CS). QPSO-LR, QPSO-RM y QPSO-CS se inspiran en modelos de interacción donante-aceptador entre partículas, vibraciones moleculares y confinamiento de electrones en grafeno, respectivamente. Como resultado de su investigación, los autores lograron determinar que entre las tres técnicas de optimización propuestas, no hay una que demuestre el mejor rendimiento en todos los escenarios de prueba realizados (exploración, explotación y tiempo de simulación). Sin embargo, QPSO-LR es el más equilibrado, lo que lo convierte en un buscador global potente para diferentes aplicaciones.

Tomando en consideración lo antes expuesto, esto da a entender que los algoritmos cuánticos proponen una mejora considerable en la optimización de rutas, en comparación con los algoritmos genéticos.

En (Velasquez et al., 2023) se muestra como se aplica el algoritmo QPSO-LR en una red de sensores inalámbricos (WSN), la cual busca minimizar el número de nodos sensores dentro de la red. Los autores mencionan que el enfoque propuesto incorpora un modelo de propagación que localiza los nodos sensores, calcula la distancia de separación aproximada entre cada uno, verifica la línea de visión (LOS) y evita intrusiones considerables en la primera zona de Fresnel. El estudio que se realiza en el artículo se basa en la preocupación global por el aumento de incendios forestales, y busca proporcionar una solución a los problemas de las redes de sensores inalámbricos en áreas boscosas, considerando la topografía del terreno y las pérdidas entre los nodos. Se comparan los algoritmos de optimización cuántica con el algoritmo tradicional de optimización de enjambre de partículas (PSO), y se demuestra que los algoritmos cuánticos tienen una robustez superior en comparación con el PSO tradicional. También, se menciona que se presentan tres escenarios de estudio con diferentes modelos de propagación y se evalúan con cada algoritmo de optimización utilizando un número variable de partículas. Los resultados muestran que todos los algoritmos convergen a un resultado único en términos de la ubicación de los nodos, pero presentan diferencias en el tiempo de ejecución. Por lo que, esto da a entender que, que los algoritmos cuánticos tienen un tiempo de ejecución más corto y convergen a valores óptimos con menos partículas o iteraciones en comparación con el algoritmo PSO tradicional. Además, los autores mencionan, que el modelo de propagación influye en la distancia entre los nodos

y, por lo tanto, en el número de nodos necesarios.

CAPÍTULO 2

2. METODOLOGÍA

En el presente capitulo se da a conocer la metodología empleada en la realización e implementación del proyecto, así como también los diferentes componentes, tecnologías y materiales que actúan en este para conseguir evaluar y, posteriormente, cumplir los objetivos planteados.

La metodología escogida para el proyecto es la de Cadena Crítica, ya que permite ordenar tareas pendientes acorde a la prioridad de cada una y optimizar los recursos disponibles dentro del proyecto. A diferencia de otros métodos de gestión de proyectos implementados con tecnologías IoT, este es el más indicado, considerando los alcances y limitaciones identificadas, ya que permite gestionar el proyecto de la manera más eficiente posible. Además, al identificar los tipos de recursos usados, el alcance local y el tiempo de implementación estimado se define la ruta crítica a seguir, la cual sirve para definir las tareas que se deben cumplir para finalizar el proyecto con éxito.

Para este proyecto, se propone la implementación de un sistema de recolección de residuos diseñado y codificado en React usando algoritmos cuánticos para determinar la ruta más optima, al igual que un hardware básico que permite emular el estado de un contenedor de basura para identificar si este se encuentra vacío o lleno e identificar si este puede ser considerado en la ruta de recolección. En este capítulo se explica cómo se aplica la metodología escogida dentro del contexto del proyecto, las tecnologías a nivel de Hardware y Software, la arquitectura propuesta para el sistema y el manejo de los datos. Además, se describen los criterios de inclusión y exclusión que se consideran para cada uno de los contenedores, las métricas de evaluación y rendimiento empleadas para conocer el estado del sistema y definir las métricas de satisfacción del usuario o beneficiario final que usará el sistema implementado.

2.1 ¿Por qué es necesario crear una ruta de recolección de residuos?

Una ruta de recolección, se define como el camino escogido para recoger los residuos o desechos que se encuentran dentro de los diferentes contenedores que se encuentran ubicados a lo largo del campus Gustavo Galindo, ESPOL. Sin embargo, para conseguir crear esta ruta, se deben considerar qué aspectos ambientales permiten identificar las necesidades y requerimientos para la implementación del sistema.

Un aspecto crucial es la reducción de residuos que llegan a los contenedores. La creación de una ruta de recolección eficiente no solo evita la acumulación descontrolada de desechos en el campus, sino que también contribuye a la disminución de la carga que se envía a los vertederos. Fomentar prácticas de reciclaje y reutilización impulsa una gestión responsable de los recursos naturales y disminuye la dependencia de nuevos materiales, promoviendo así una visión más holística de la sostenibilidad.

Finalmente, la eficiencia operativa del campus se ve directamente beneficiada por la creación de una ruta de recolección bien organizada. La ubicación estratégica de contenedores, al igual que la implementación de un proceso eficiente de recolección facilitan la operación diaria, reduciendo los tiempos y costos asociados con la gestión de residuos. La disposición adecuada de los desechos contribuye a un entorno más limpio y ordenado, mejorando la calidad de vida en el campus y proporcionando un ambiente propicio para el aprendizaje.

2.2 Arquitectura

La implementación del sistema sigue un esquema basado en el uso de microservicios en la nube donde se aloja el aplicativo. Además, el esquema muestra en resumen el detalle de los diferentes componentes a nivel de Software que intervienen para conseguir un aplicativo funcional y eficiente.

La figura 2.1 muestra la arquitectura planteada del sistema, donde se observa el despliegue de la aplicación, sus componentes y el aplicativo móvil obtenido mediante el uso de una aplicación web progresiva (PWA). La arquitectura se divide en tres grandes

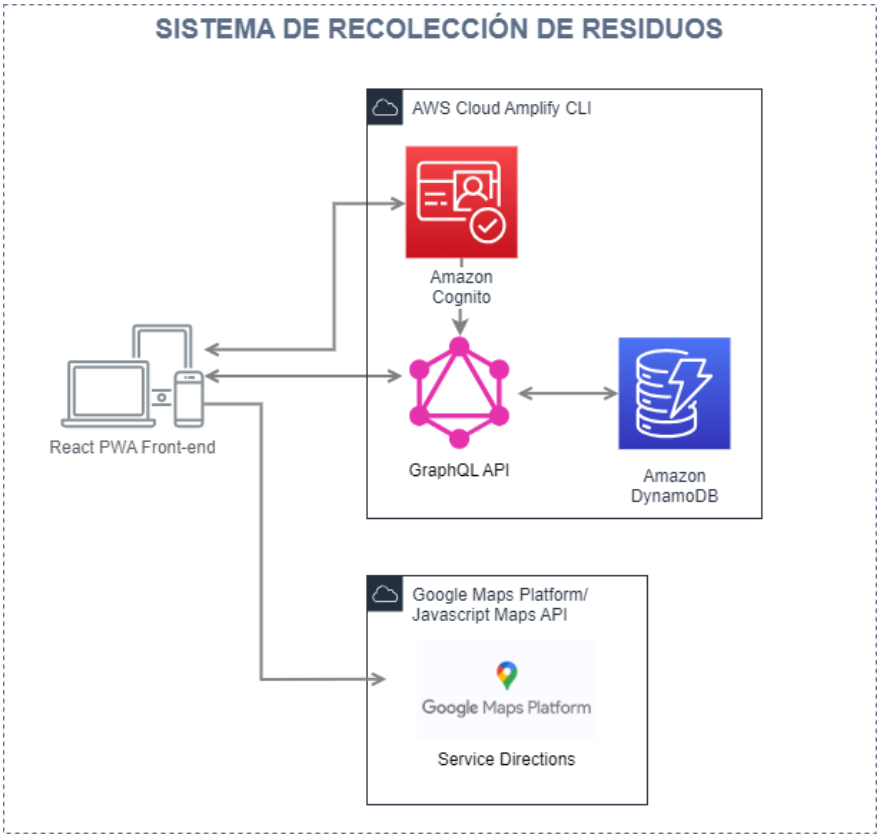

Figura 2.1: Arquitectura propuesta del Sistema de recolección de residuos

secciones:

- Identificación y acceso al sistema
- Uso del algoritmo QPSO-LR para optimizar la ruta
- Uso de librerías y herramientas Open Source

Cada una de las secciones mencionadas se encarga de cumplir una función en específico dentro del sistema donde se encuentran alojados en la nube, es decir, cada uno de los componentes mostrados forman parte de una arquitectura distribuida en la nube.

Acorde a la metodología de Cadena crítica implementada en el proyecto, nos indica que se debe identificar y ordenar las tareas según su prioridad. Por lo que, en primera instancia, se determinó qué tareas a nivel de Software y Hardware son las más importantes y cuáles deben ser realizadas de forma inmediata. La primera de ellas fue la recopilación de las ubicaciones de cada contenedor donde se registró su latitud y longitud para, posteriormente, calcular la distancia entre un punto y otro. Esta información es

usada en la aplicación, ya que se conoce la ubicación exacta donde se encuentran los contenedores y se puede definir una posible ruta.

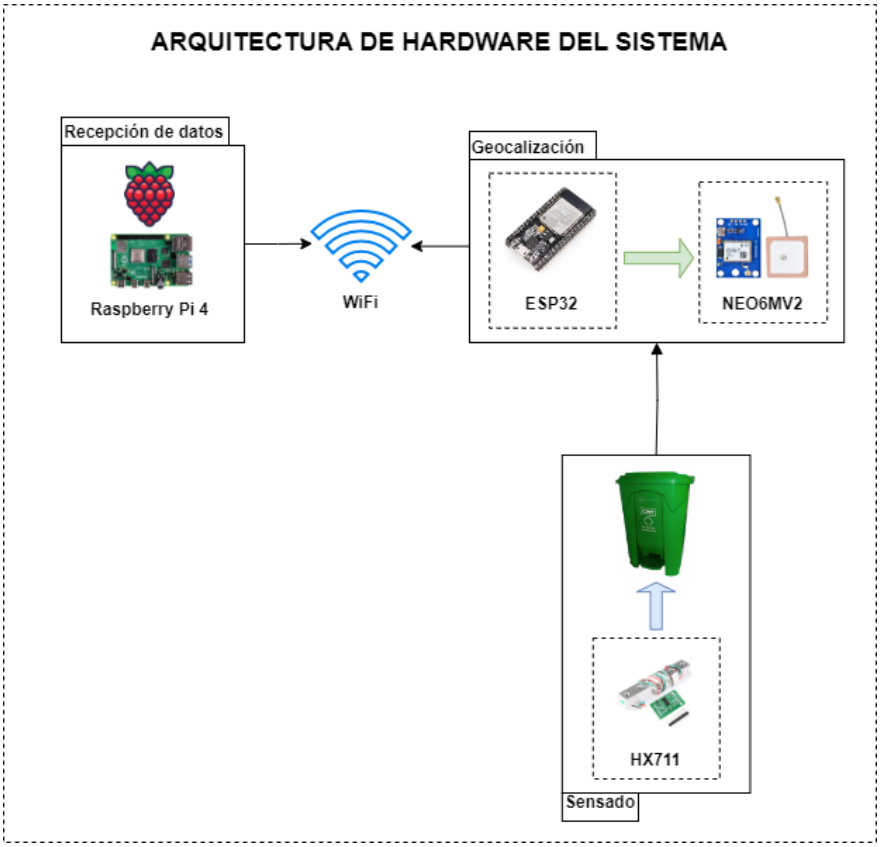

Figura 2.2: Arquitectura propuesta a nivel de Hardware

Por otro lado, una de las siguientes tareas es identificar que herramientas y/o tecnologías intervienen en el sistema. A nivel de Software se plantea el uso de microservicios en la nube de AWS para implementar y alojar el sistema, así también el uso de herramientas como Google Maps Platform para montar el mapa donde se colocaran las ubicaciones de cada contenedor y Maps JavaScript API para crear una ruta dentro del mapa. Además, como se muestra en la figura 2.2, a nivel de Hardware se plantea el uso de un microcontrolador ESP32 en conjunto con un módulo GPS (NEO6MV2) el cual proporciona la ubicación del contenedor y un sensor de peso en conjunto con un módulo HX711 que permite amplificar la lectura del sensor y medir la fuerza (peso) del contenedor. Con cada uno de los componentes de Software y Hardware identificados, se comienza con la codificación del sistema de recolección en el Framework React, donde, posteriormente, se emplea el uso del algoritmo QPSO-LR para conseguir la ruta de recolección más optimizada posible.

La Cadena Crítica permite optimizar la cantidad de recursos necesarios en la

implementación del proyecto, por lo que el uso de herramientas Open Source no solo reduce el margen de inversión del proyecto, sino que también ofrece la posibilidad de modificar el código fuente y ser usado acorde los requerimientos del sistema, además de ofrecer portabilidad.

2.2.1 Almacenamiento de datos

Amazon DynamoDB[1] es un servicio de base de datos NoSQL totalmente administrado y escalable proporcionado por Amazon Web Services (AWS). DynamoDB se convierte en la fuente central de almacenamiento para datos clave, como la ubicación de contenedores y el historial de rutas optimizadas (Elhemali et al., 2022).

Con DynamoDB, se pueden diseñar tablas que almacenen datos relacionados con usuarios, roles y preferencias individuales. Los índices y capacidades de consulta de DynamoDB facilitan la recuperación eficiente de información específica, lo que es crucial para proporcionar a cada usuario la perspectiva y funcionalidades necesarias.

La asignación de roles y permisos garantiza que cada usuario acceda solo a las tablas y datos pertinentes a su función. Esto no solo mejora la seguridad, sino que también simplifica la gestión y mantenimiento del sistema. Además, DynamoDB permite la consulta eficiente de datos, lo que facilita la obtención de información específica para cada rol (Talha et al., 2020).

2.2.2 Procesamiento de datos

2.2.2.1 Google Maps API

Google Maps API[2] es un conjunto de herramientas y servicios proporcionados por Google para permitir a los desarrolladores integrar capacidades de mapas en sus aplicaciones y sitios web. Esta API brinda acceso a diversos servicios relacionados con mapas, ubicaciones y geocodificación, permitiendo a los desarrolladores incorporar mapas interactivos, mostrar información sobre ubicaciones específicas, trazar rutas, y mucho más (Nurdin et al., 2021).

[1] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
[2] https://developers.google.com/maps/apis-by-platform?hl=es-419

Permite integrar mapas interactivos en aplicaciones y sitios web, con opciones para personalizar el aspecto y el contenido de los mapas. Facilita la obtención de direcciones entre dos puntos y la representación visual de rutas en el mapa, con opciones para modos de transporte (a pie, en automóvil, en bicicleta, etc.).

Google Maps API, al ser altamente personalizable, también se adapta a las necesidades específicas de diferentes roles de usuario. Los administradores pueden gestionar la configuración de la aplicación y personalizar la presentación de las rutas en el mapa según los requisitos específicos del sistema.

2.2.2.2 Maps JavaScript API

Maps JavaScript API [3] es una interfaz de programación de aplicaciones (API) proporcionada por Google Maps Platform. Esta API permite a los desarrolladores incorporar mapas interactivos basados en JavaScript en sus sitios web. Al utilizar la Maps JavaScript API, los desarrolladores pueden personalizar la presentación de mapas, integrar funciones de búsqueda de lugares, mostrar rutas y proporcionar diversas interacciones con mapas geoespaciales (Wu, 2020).

La implementación de Maps JavaScript API en conjunto con el algoritmo QPSO-LR garantiza que las rutas sean dinámicamente actualizadas y optimizadas en función de factores en tiempo real y la disponibilidad de rutas.

2.2.3 Autorización de usuarios

Amazon Cognito[4] es un servicio de Amazon Web Services (AWS) que proporciona servicios de autenticación y autorización para aplicaciones web y móviles. Además, permite una autenticación segura y una administración eficiente de roles, facilitando la creación de perfiles personalizados para diferentes usuarios, como conserjes, supervisores y administradores.

Amazon Cognito define roles y permisos específicos para cada tipo de usuario. Los conserjes pueden tener acceso a datos relacionados con sus rutas y preferencias de recolección, mientras que los supervisores pueden visualizar métricas de rendimiento

[3] https://developers.google.com/maps/documentation/javascript/overview?hl=es-419
[4] https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

y administrar equipos. La capacidad de controlar el acceso a recursos y funciones específicas garantiza una gestión segura y eficaz de la aplicación.

Además, la sincronización de datos entre dispositivos proporcionada por Amazon Cognito asegura que la información relevante para cada usuario se mantenga consistente en todas las plataformas y dispositivos utilizados en el sistema de recolección de residuos.

2.2.4 Criterios de inclusión y exclusión

Los criterios de inclusión son aquellas características especificas que sirven para determinar qué elementos deben ser considerados para la implementación del sistema. Estos criterios pueden ser considerados en diferentes ámbitos, en este caso, se plantean criterios con base a la implementación del Software del sistema.

Los criterios de inclusión para el aplicativo web se centran en la eficiencia y la funcionalidad. El algoritmo QPSO-LR se utiliza para optimizar las rutas de recolección, por lo que el aplicativo debe ser capaz de integrar este algoritmo de manera efectiva y proporcionar resultados precisos en tiempo real. Debe ser capaz de gestionar grandes datos de geocalización y garantizar una alta velocidad de procesamiento para mantener la eficiencia operativa. La usabilidad y la accesibilidad son otros criterios clave de inclusión. El aplicativo debe ser fácil de usar tanto para los conserjes y, en general, para los usuarios que usen el aplicativo. Además, debe ser accesible desde diferentes dispositivos y navegadores, ya que se planea su posterior conversión en una PWA para dispositivos móviles. La capacidad de respuesta y la adaptabilidad son esenciales para garantizar una experiencia de usuario óptima en diferentes plataformas.

La seguridad de los datos es un criterio de inclusión crucial. Dado que el aplicativo manejará información sensible, como datos de ubicación y programación de rutas, debe contar con las suficientes medidas de seguridad para proteger la privacidad y la integridad de los datos.

Por otro lado, los criterios de exclusión se refieren a las características o funcionalidades que no serán parte del aplicativo web. Esto puede incluir características que no sean relevantes para la recolección de residuos. Para este caso en particular, el proyecto se centra en la optimización de rutas, por lo que datos relacionados al tiempo de inicio y fin de la recolección no son considerados para el sistema, ya que no influye de manera directa en la determinación de la ruta más optima. Además, a nivel de hardware,

se excluye un estado intermedio para definir si un contenedor se encuentra vacío o lleno. Solo se necesita saber si el contenedor requiere ser vaciado o no, por lo que un estado intermedio no un dato significativo para el sistema.

2.2.5 Métricas de evaluación y rendimiento del sistema

El aplicativo tendrá como finalidad un uso recurrente, por lo que es importante definir métricas que permitan evaluar el estado y rendimiento del sistema, con el objetivo de identificar los puntos críticos en donde el aplicativo puede ser vulnerable y mejorar o solventar estos inconvenientes. Las métricas cumplen tres funciones, las cuales son: Informar cuando proporcionan datos sobre ciertos parámetros del sistema y proveer algún diagnóstico si cierto parámetro se encuentra fuera de los límites de lo normal, causando que cierta parte del sistema presente problemas de rendimiento llevando a limitaciones que son llamadas cuellos de botella. Además, sirven para llevar una evaluación de impacto de las modificaciones realizadas en un sistema o proceso.

En la arquitectura propuesta en la figura 2.1 se muestra cómo las tres secciones convergen en el aplicativo final, por lo que los inconvenientes que pueden surgir entre la interacción del usuario final con el aplicativo son los siguientes:

- Rendimiento: Mide la capacidad y la eficiencia con la que una entidad puede realizar tareas o llevar a cabo operaciones en un lapso específico. El rendimiento se mide en bits por segunfo (BPS).

- Latencia: Mide el tiempo que tarda una señal, dato o comando en viajar desde su punto de origen hasta su destino final a través del sistema. Su unidad de medida es milisegundos (ms).

- Tiempo de servicio: Cuantifica el tiempo que el sistema requiere para satisfacer una solicitud. Su unidad de medida es segundos (s).

- Tiempo de carga de la aplicación: Tiempo que tarda una página en cargarse completamente. Su unidad de medida. Su unidad de medida es segundos (s).

- Número de Bloqueos: Indica cuántas veces la aplicación ha experimentado un bloqueo durante un período de tiempo específico.

2.2.6 Métricas de satisfacción del usuario

Es importante determinar el impacto que la aplicación tendrá en las personas directamente relacionadas al momento de realizar la recolección de residuos, en este caso serán los conserjes o también llamados Auxiliares de Servicios de la institución. Esto se realiza con el fin de determinar si la aplicación realmente cumple con las necesidades y requerimientos que los usuarios deben solventar. Por ende, se va a realizar una encuesta dirigida a ellos donde se evalué de forma cuantitativa su nivel de conformidad con el sistema de recolección actual. El objetivo principal de esta encuesta es conocer cuáles son los puntos de dolor que se relacionan al momento de realizar la recolección de residuos. En ESPOL hay cerca de 100 Auxiliares de Servicios[5] distribuido a lo largo de todo el campus, los cuales serán considerados nuestra población, se procede a aplicar la fórmula 2.1 para hallar la muestra de una población finita:

$$muestra = \frac{\frac{z^2 p(1-p)}{e^2}}{1 + \frac{z^2 p(1-p)}{e^2 N}} \tag{2.1}$$

Donde:

- *N*: Tamaño de la población.
- *e*: El margen de error (%)
- *p*: Probabilidad a favor (%)
- *z*: Puntuación z; que es la cantidad de desviaciones estándar que las proporciones se alejan de la media.

Acorde a la tabla 2.2, y considerando la población de 100 Auxiliares de Servicio, se decidió seleccionar un margen de error del 10 % con un nivel de confianza del 80 %. Aplicando la fórmula, se obtiene una muestra de aproximadamente 29 Auxiliares de Servicios. Finalmente, la cantidad de Auxiliares que tienen en común con la variable que se busca medir, indicado por la letra p, en estos casos se coloca 0,5. De forma general, esta métrica es simple pero muy útil para medir la satisfacción de los consumidores respecto a un servicio o producto. Las preguntas estarán orientadas a respuestas bajo

[5]https://www.espol.edu.ec/sites/default/files/transparencia

Tabla 2.1: Nivel de confianza relacionada a la puntuación z

Nivel de confianza deseado	Puntuación z
80	1,28
85	1,44
90	1,65
95	1,96
99	2,58

escalas numéricas, que buscan indagar la opinión sincera de los Auxiliares de Servicios al integrar un nuevo sistema de recolección. Para obtener el resultado de esta métrica, suele

Tabla 2.2: Grados de calificación definidos para la métrica de satisfacción al usuario

Calificación	Significado
1	Totalmente en desacuerdo
2	En desacuerdo
3	Neutral
4	De acuerdo
5	Totalmente de acuerdo

calcularse el promedio que se convertirá en la satisfacción media de los Auxiliares de servicio al usar el nuevo sistema. Sin embargo, el resultado también puede apoyarse en un porcentaje que determinará la satisfacción neta de los Auxiliares utilizando la fórmula 2.2 a continuación:

$$SN = \frac{DTA}{TD}100 \tag{2.2}$$

Donde:

- *SN*: Satisfacción neta.

- *DTA*: Cantidad total de Auxiliares de servicio que calificaron con 4 y 5
- *TD*: Total de Auxiliares de servicio participantes

Se realizará un análisis individual en promedio y satisfacción neta por pregunta, apoyado por gráficos que ilustren las tendencias. Si bien es importante que las repuestas sean precisas y representen cómo se sienten los Auxiliares al usar el nuevo sistema, el punto crítico de esta métrica es comprender las respuestas obtenidas, al ser la retroalimentación recibida importante para las posibles futuras mejoras del sistema.

CAPÍTULO 3

3. DISEÑO Y FUNCIONALIDAD DEL SISTEMA

En el presente capítulo se muestra el escenario de prueba plantaedo donde se implementarán los dispositivos IoT en el campus de ESPOL, el proceso de diseño de la aplicación mediante el uso de diferentes diagramas, tales como: Diagramas de Entidad - Relación, Clases UML, flujo e interacción con el usuario, tomando en consideración los actores involucrados en el escenario planteado en el capítulo anterior.

En principio se muestra el diseño de red para el escenario de prueba y cómo se interconectan los dispositivos entre sí, seguido del diagrama Entidad – Relación del sistema donde se puede observar cómo los componentes de Software se encuentran relacionados. En el diagrama de clases UML se podrá observar cómo las diferentes entidades identificadas en el proyecto interactúan entre sí permitiendo la comunicación a nivel de Back-End del sistema. El diagrama de flujo permitirá conocer el proceso de interacción intuitivo que existe entre los actores y la aplicación, esto abarca desde el ingreso del usuario al aplicativo hasta el momento en que se genera una nueva ruta. Finalmente, se muestra el diagrama de interacción de usuario que permite explicar de manera más detalladas las interacciones que existen entre el Front-End y el Back-End, y cómo estas convergen en el aplicativo.

Además, se explica el funcionamiento de la aplicación en conjunto con el algoritmo cuántico QPSO-LR y cómo se consigue generar la ruta más óptima entre los diferentes puntos de recolección identificados.

3.1 Escenario de prueba

El sistema tiene como escenario principal los puntos de recolección que se encuentran dentro de la ESPOL, ambas arquitecturas propuestas trabajarán en conjunto para determinar el estado de cada uno de los puntos de recolección.

En la figura 2.2 se muestra la arquitectura para un solo nodo recolector el cual se encarga de sensar los contenedores y a su vez enviar estos datos, en conjunto con la ubicación, a la Raspberry Pi 4 mediante el protocolo WiFi para poder alojarlos en un servicio en la nube. Sin embargo, al aplicar el sistema a nivel institucional se requiere que esta arquitectura se replique reiteradas veces en los distintos puntos de recolección y asegurarse que cada uno de ellos tenga salida a Internet para enviar los datos al microprocesador.

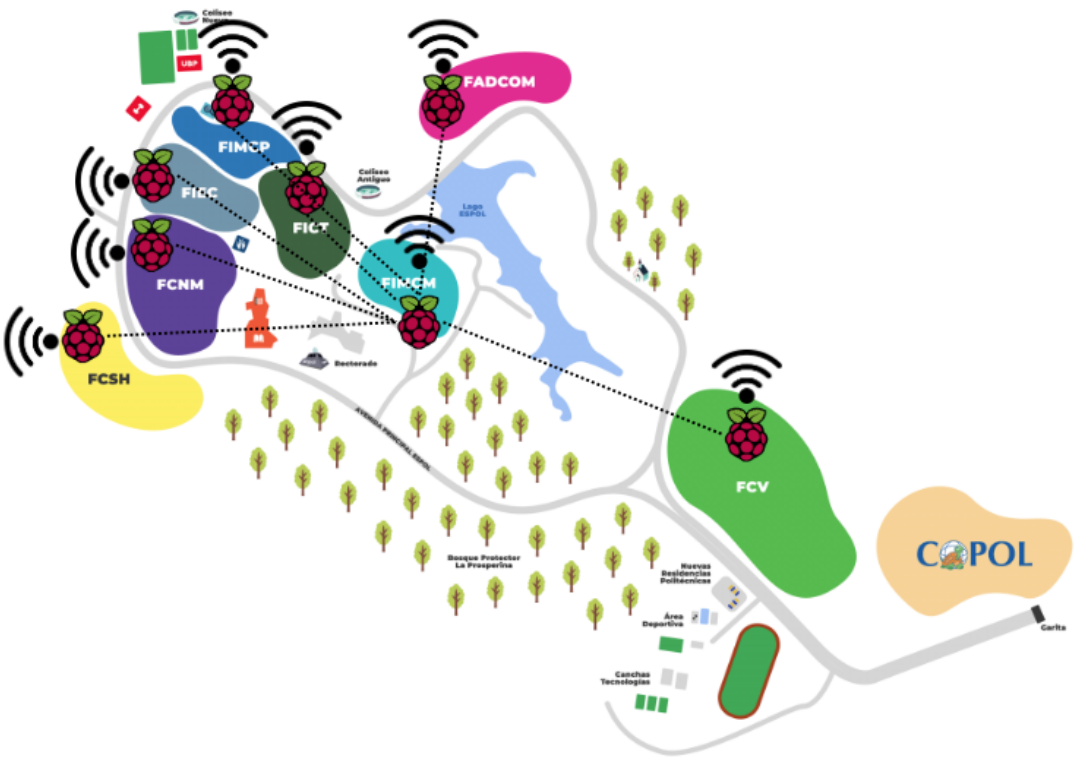

Figura 3.1: Red de malla

Para esto se plantea el uso de una red de tipo malla[1], la cual se define como una red inalámbrica en la que cada dispositivo de la red, como los nodos recolectores, están interconectado entre sí formando una malla. La implementación de este tipo de red proporciona redundancia en las rutas de comunicación, por lo que si un nodo llega a

[1] https://support.eero.com/hc/en-us/articles/207646676-What-s-a-mesh-network

fallar o a desconectarse, la red puede encontrar automáticamente una ruta alternativa para mantener la conectividad y enviar los datos sensados.

En la figura 3.1 se muestra una aproximación de cómo será la implementación de una red de malla para asegurar una conectividad entre todos los dispositivos Raspberry Pi 4 instalados en cada área en específico donde se encuentren sus respectivos puntos. Todos estos nodos estarán conectados al nodo central o máster que se encuentra en la FIMCM, el cual fue escogido debido a su cercanías con los demás dispositivos Raspberry Pi 4.

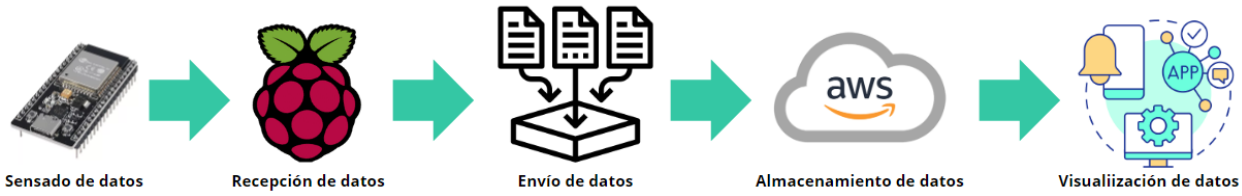

Figura 3.2: Proceso de los datos

Con la implementación de ambas arquitecturas se tiene en claro cómo funciona el proceso de los datos, desde su recolección hasta su visualización. En la figura 3.2 se aprecia el comienzo, desarrollo y final del proceso de los datos, estos son recolectados gracias a los nodos recolectores enviándolos a la Raspberry Pi 4 para ser receptados y enviados a los elementos de la arquitectura de Software, específicamente al servicio de AWS, donde se almacenan todos los datos del sistema, y se envían a la aplicación para poder ser visualizados y que el usuario pueda observar el estado del contenedor.

3.2 Diseño del sistema

3.2.1 Diagrama Entidad – Relación

Un diagrama entidad-relación es una representación visual de las entidades que interactúan en un sistema y cómo están relacionadas entre sí. Las entidades que se presentan en el diagrama son las mismas que se encuentran en el diagrama de clases UML. En este caso, los atributos descritos en cada una de las entidades del diagrama no solo están formados por las características esenciales de cada una de estas, sino también por atributos de otras tablas que se relacionan entre sí y forman parte de los atributos de la entidad. Estos atributos poseen un identificador “FK” (Foreign key), es decir clave foránea, lo que indica que la tabla se relaciona con otra por medio de ese atributo foráneo. En el caso de la entidad UsuarioAreas, esta nace de la relación que existe entre las tablas Usuario y Area la cual no solo posee claves foráneas, sino que estas a su vez son las “PK” (Primary Key), es decir clave primaria, de la entidad.

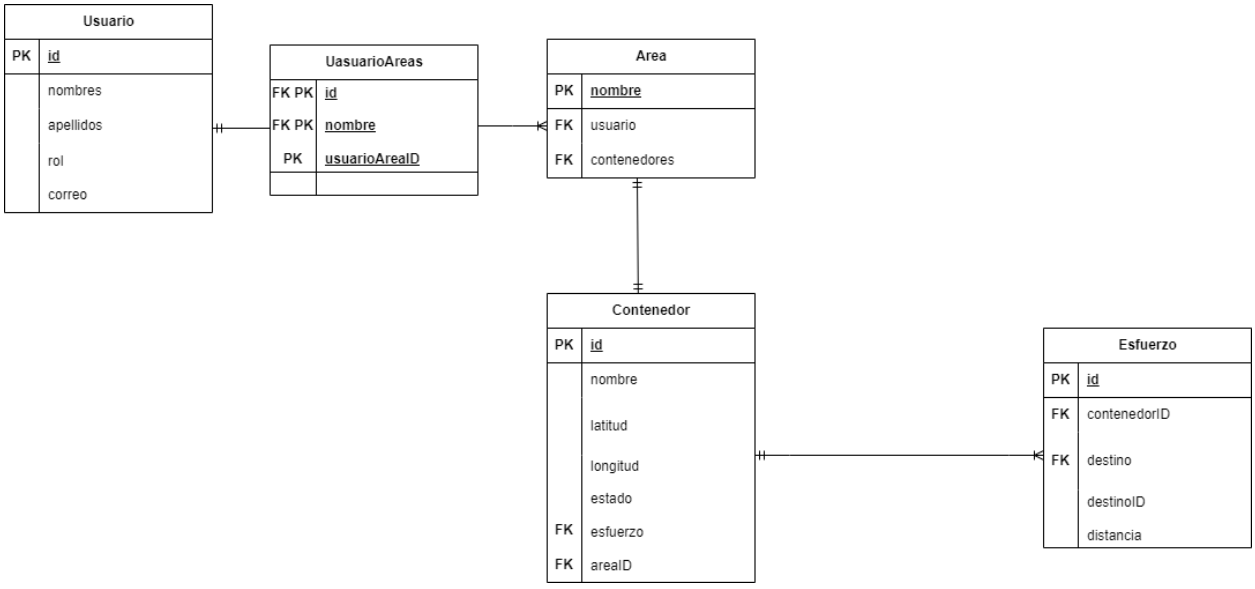

Figura 3.3: Diagrama Entidad - Relación del sistema

3.2.2 Diagrama de Clases UML

La aplicación interactúa de manera dinámica acorde al escenario planteado donde el usuario puede acceder a la aplicación desde el navegador Web de su preferencia y registrase en este para acceder y generar rutas. Con base a esto, se plantea el siguiente

diagrama UML para describir la interacción de las diferentes clases y los métodos que posee cada una de estas. La clase Usuario es la encargada de alojar los diferentes

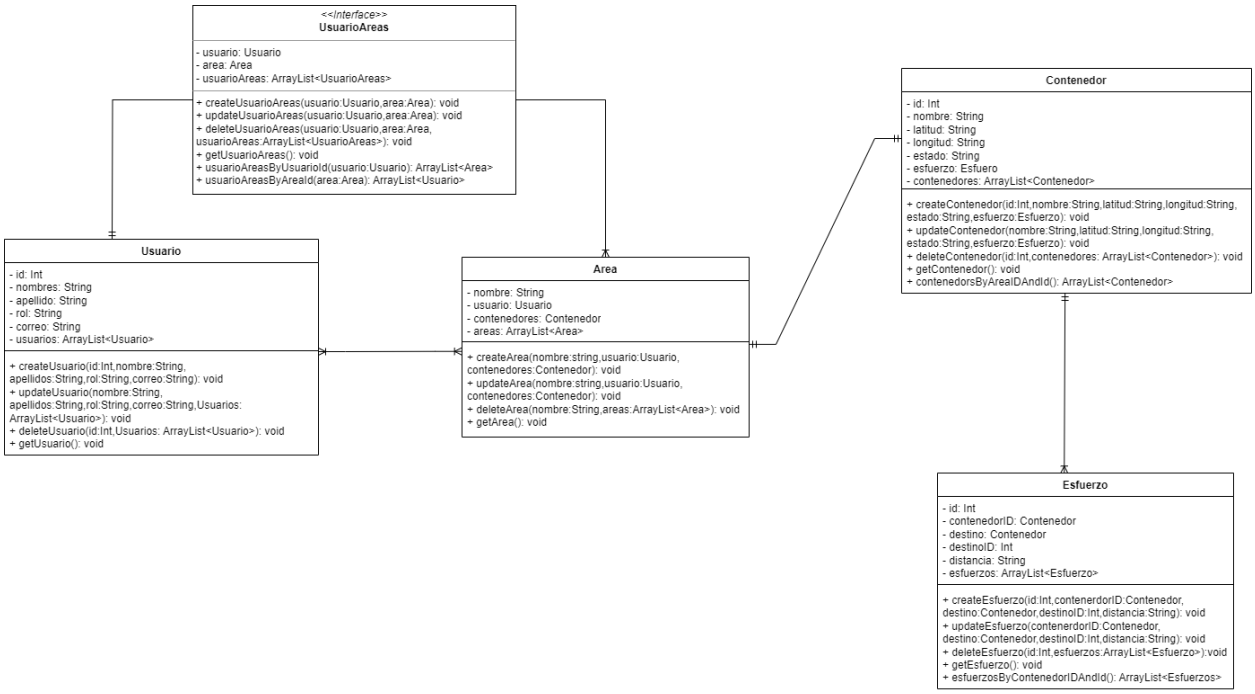

Figura 3.4: Diagrama de Clases UML del sistema

atributos que se solicitan al momento de registrarse en el aplicativo. Esta clase posee una relación con la Clase Area que a su vez se relacionan con la Interfaz UsuarioAreas. La clase Usuario posee métodos asociados a la obtención de información de cada uno de los usuarios, así como también un atributo de tipo ArrayList donde se almacenará todos los usuarios registrados.

En la clase Area se alojan los diferentes lugares del campus donde se hayan encontrado contenedores, en esta ocasión las áreas se definen por cada una de las facultades donde se almacenan su información básica, la cual está asociada a los atributos de la clase. Esta clase interactúa con las clases Usuario y Contenedores, así como también con la interfaz UsuarioAreas. Al igual que todas las clases dentro del diagrama, también posee métodos de obtención de información que permitirá asociar a cada uno de los contenedores con un área respectiva.

La Interfaz UsuarioAreas posee atributos de tipo Usuario y Area, esto con el fin de acceder a los demás atributos de estas clases y usarlos en los métodos propios de la interfaz. El método usuarioAreasByUsuarioId() es el encargado de obtener las áreas asignadas por cada uno de los usuarios con el rol de Recolector dentro del sistema. El

método usuarioAreasByAreaId() es el encargado de obtener los usuarios asignados con el rol de Recolector dentro del sistema por cada una de las áreas. Es importante mencionar que todo esto se realiza con base a la relación que existe entre las tablas Usuario y Area, la cual es una relación uno a muchos, por lo que su interpretación resultaría en que cada usuario Recolector está asociado a un área y que esta área puede tener asociado varios usuarios Recolectores.

La clase Contenedor es la encargada de almacenar la información de cada uno de los diferentes contenedores asociados en cada una de las diferentes áreas del campus. Entre sus atributos se encuentra uno de tipo Esfuerzo y uno de tipo Area, el cual mediante el método ContenedorsByAreaIDAndId(), se pueda determinar la cantidad de contenedores que existen por cada una de las áreas y con el atributo esfuerzo asociar a cada contenedor el esfuerzo o distancia recorrida desde un determinado contenedor con respecto a otros.

La clase Esfuerzo es la encargada de almacenar la información relacionada a la distancia que existen entre los contenedores con respecto a otros. Esta clase posee entre sus atributos uno de tipo Contenedor, el cual mediante el método esfuerzosByContenedorIDAndId(), se puede determinar la cantidad de esfuerzos asociados que existen en un contenedor con respecto a los demás.

3.2.3 Diagrama de Flujo

Un diagrama de flujo es la representación visual de un proceso donde se ilustra la secuencia de pasos necesarios para llevar a cabo una tarea o resolver un problema. En esta ocasión, el diagrama de flujo se centra en la descripción del proceso de inicio de sesión hasta que el usuario genera una ruta de recolección dentro del sistema. El diagrama inicia con el ingreso de las credenciales de inicio de sesión donde se explica el proceso de lo que ocurre en caso de no poder acceder con la contraseña registrada. La pantalla de inicio de la aplicación permite la recuperación de contraseña mediante correo electrónico, una vez cambiada la contraseña se vuelven a ingresar las credenciales y si estas son correctas ingresan al sistema, caso contrario se vuelve al proceso anterior.

Una vez ingresado al sistema, el usuario puede visualizar la interfaz interactiva donde puede apreciar los puntos de recolección, ruta, información de personal y perfil. En el mapa se muestran los diferentes puntos de recolección, para acceder a la generación de ruta optimizada, se debe considerar todos los puntos de recolección con un estado

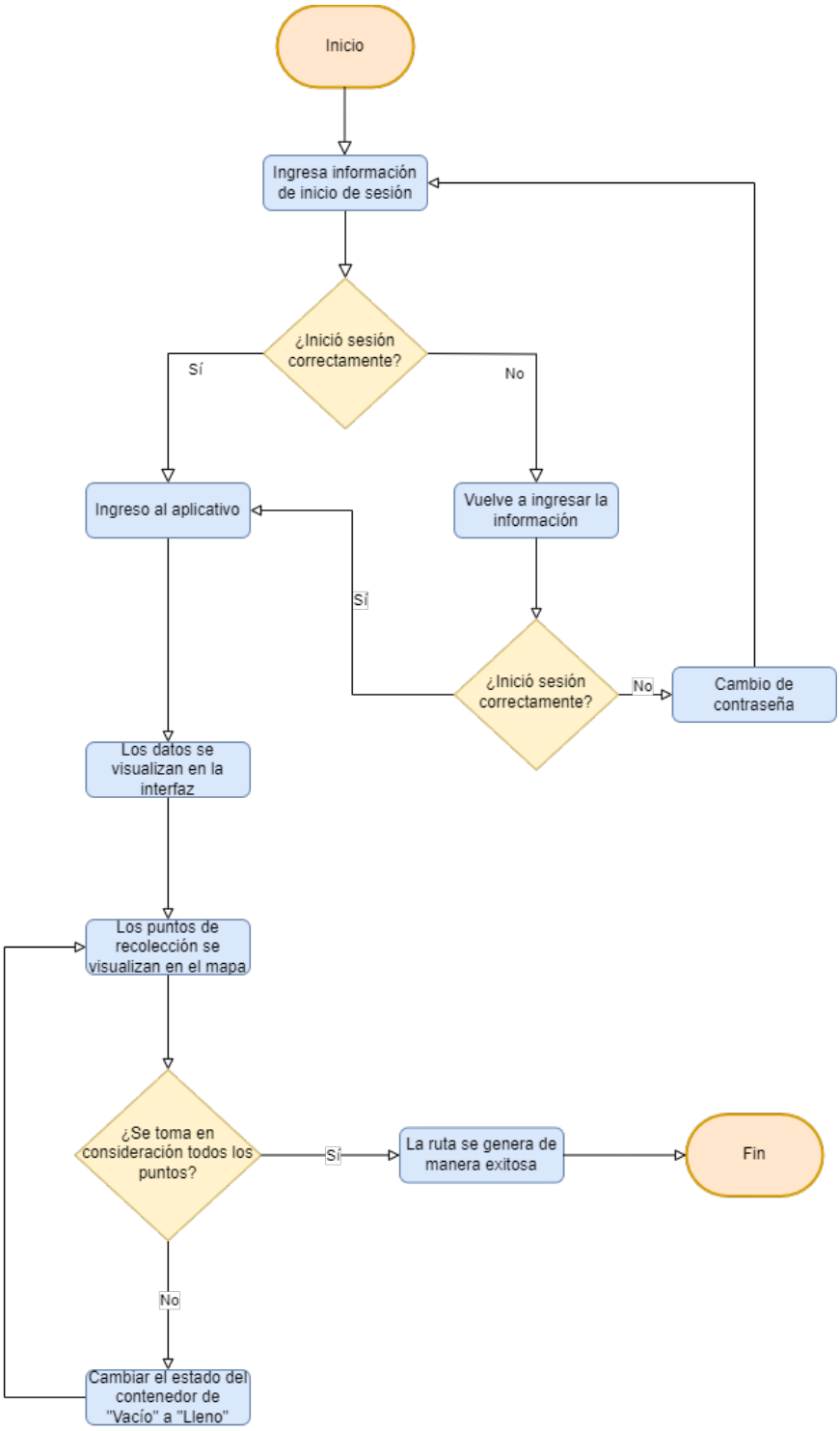

Figura 3.5: Diagrama de flujo del sistema

“Lleno”, en caso de que no se pueda generar una ruta con los puntos actualizados, el proceso indica que se debe cambiar el estado de los contenedores para poder considerar todos los puntos dentro de la ruta optimizada.

3.2.4 Diagrama de interacción

El aplicativo propuesto está compuesto por un componente de Back-End, en donde se contempla las herramientas utilizadas de AWS para alojar el sistema y las herramientas de Google Maps Platform para la geocalización. Además, está compuesto por un Front-End el cual interactúa con el Back-End mediante la interfaz.

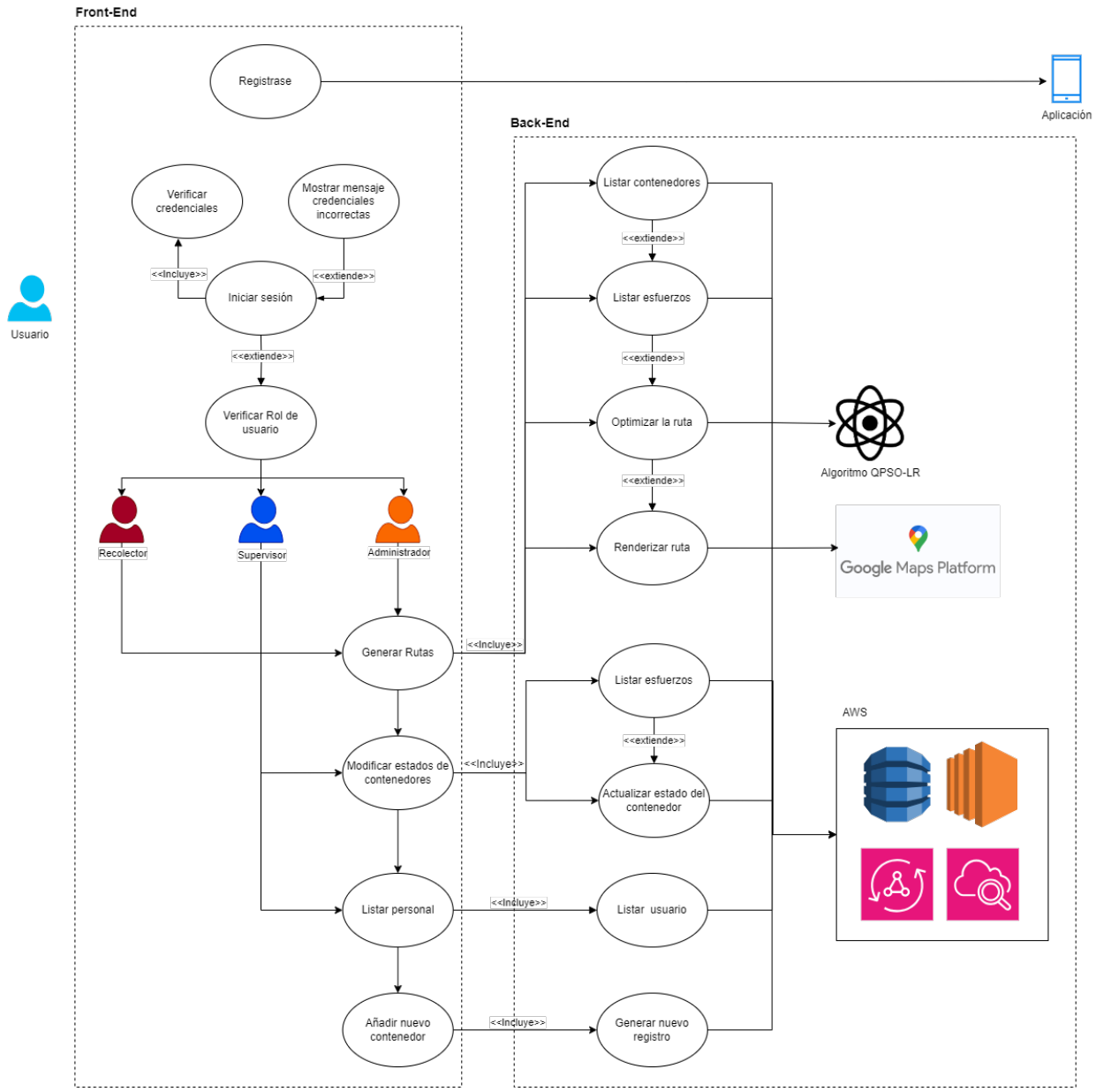

Figura 3.6: Diagrama de interacción del sistema

El sistema tiene representado con un rectángulo los componentes de Back-End y Front-End, el cual contiene los casos de uso que son los diferentes procesos o tareas que se ejecutan en él. Por otro lado, a los costados se muestra las herramientas en el lado del Back-End y los actores que intervienen en el Front-End. Como único actor que interactúa en el sistema se tiene al usuario, quien ingresa al sistema mediante el registro y al inicio de sesión. Luego de esto se realiza la verificación de los diferentes roles de usuario existente en el sistema. Hay tres diferentes roles: Administrador, Supervisor y Recolector.

Cada uno de los actores posee diferentes responsabilidades, las cuales están representadas por las elipses y contienen las tareas para cada usuario. En el caso del usuario Recolector, este solo puede generar la ruta, para el usuario Supervisor tiene asignado las tareas de modificar los estados de los contenedores y listar personal, y para el usuario Administrador tiene asignado todas las demás tareas además de añadir nuevo contenedor. Cada una de estas tareas tiene se relacionan con las herramientas del Back-End. Para la generación de rutas se incluyen los procesos donde intervienen los listas de los contenedores y los esfuerzos asociados a cada uno de estos, esto se encuentra alojado en los servicios de AWS en la nube. Para la optimización de la ruta se usa el algoritmo cuántico QPSO-LR. Finalmente, para renderizar la ruta se usaan los servicios de Google Maps Platform.

Los procesos de listar esfuerzos, actualizar estado del contenedor, listar usuario y generar nuevo registro también se alojan en AWS. Finalmente, todos estos procesos convergen en el proceso final con la presentación del aplicativo mediante el navegador web.

3.3 Funcionalidad del sistema

3.3.1 Puntos de recolección

El aplicativo es capaz de generar una ruta optimizada donde los auxiliares de servicio consigan agilizar los procesos de recolección mediante la implementación del algoritmo cuántico QPSO-LR. Para ello fue necesario realizar un levantamiento de información sobre los diversos puntos de recolección que se encuentran a lo largo del campus de la

ESPOL. Acorde al esquema planteado previamente, donde se considera a una facultad como un área dentro de la aplicación, por cada uno de los puntos identificados, se obtuvo su ubicación exacta en términos de latitud y longitud usando Google Maps.

Tabla 3.1: Detalle de los puntos de recolección del campus por área

Área	Puntos identificados
FIEC	18
FICT	4
FIMCM	4
FCSH y CELEX	11
FADCOM	3
FCV	4
FCNM	3
FIMCP	5
Varios	14
Total de puntos	66

Acorde a la tabla 3.1 se contabilizaron un total de 66 puntos de recolección en todo el campus de la ESPOL. Cabe mencionar que solo se tomaron en consideración aquellos puntos que se encontraron fuera de las aulas de clases y oficinas, es decir aquellos puntos donde la concentración de personas es mayor en diferentes horas del día y no existe un control ni manejo adecuado de los objetos que se desechan.

Dentro del aplicativo los puntos se almacenan en AWS DynamoDB en la nube, posteriormente, son llamados y cargados en el mapa e identificados por un marcador que indica la ubicación de cada de los puntos se recolección.

3.3.2 Algoritmo QPSO-LR

La abreviatura “QPSO-LR” se refiere a un algoritmo de optimización llamado “Quantum Particle Swarm Optimization with Lorentzian”. Este algoritmo es una variante de

optimización de enjambre de partículas (PSO) que incorpora elementos de la mecánica cuántica, como el enjambre cuántico, y también utiliza una función específica conocida como la “Lorentzian Function” (Función Lorentziana), la cual puede utilizarse como parte del proceso de evaluación o cálculo de la aptitud de las soluciones propuestas por las partículas en el espacio de búsqueda.

En (Velasquez et al., 2023) se muestra la implementación del algoritmo en una red de sensores inalámbricas donde se obtiene como resultado que, entre todos los demás algoritmos cuánticos implementados, QPSO-LR es aquel que posee la mayor convergencia en todos los escenarios planteados, por lo que resulta ser el mejor algoritmo en términos de optimización.

El algoritmo QPSO-LR es el encargado de optimizar la ruta entre todos los puntos de recolección que requieran ser vaciados. Para ello el algoritmo debe tener como entrada cada uno de los puntos de recolección identificados previamente, dentro del aplicativo, estos puntos de recolección tendrán dos estados “Lleno” y “Vacío”, cada vez que un punto cambie de estado a “Lleno” será considerado como parte de la ruta próxima a generar. Dentro del código se representan a estos estados como “0” para indicar que el contenedor está vacío y “1” para indicar que está lleno, esto permite identificar las posiciones de cada uno de los puntos de recolección con respecto a los demás.

Código 3.1: Esquema de estados “1” y “0”

```
const garbageCollector = contenedores.map((obj, i) => {
    if (obj?.estado == “LLENO”) return 1;
    else return 0;
});
console.log(garbageCollector);
```

Con cada una de las posiciones se crea una matriz de distancia donde se almacenará la distancia que existe entre cada uno de los puntos de recolección con respecto a todos los demás, es decir se genera una matriz de adyacencia que debe ser reducida, para ello solo se toma en consideración los puntos de recolección que posean el estado “1” y, posteriormente, volver a generar una nueva matriz de distancia solamente tomando en consideración estos puntos.

Código 3.2: Creación y reducción de la matriz de distancia

```
const distances = contenedores.map((obj) => {
  var f = contenedores.map((obj2) => {
    var d = esfuerzos
      .filter((esfuerzo, i) => {
        if (
          obj?.id == esfuerzo?.contenedorID &&
          obj2?.id == esfuerzo?.destinoID
        )
          return esfuerzo;
      })
      .map((dis) => {
        return dis?.distancia;
      });
    return (d as any)[0];
  });
  return f;
});
console.log(distances);

const reduceGarbageCollector: number[] = [];
garbageCollector?.map((o, i) => {
  if (o == 1) reduceGarbageCollector.push(i);
});
console.log(reduceGarbageCollector);

const reducedMatrixDistances: number[][] = Array.from(
  { length: reduceGarbageCollector.length },
  () => Array(reduceGarbageCollector.length).fill(0)
);

for (let i = 0; i < reduceGarbageCollector.length; i++) {
  for (let j = 0; j < reduceGarbageCollector.length; j++) {
    reducedMatrixDistances[i][j] =
      distances[reduceGarbageCollector[i]][reduceGarbageCollector[j]];
```

```
    }
  }
  console.log(reducedMatrixDistances);

  const startPosNodo: number = 1;
```

El algoritmo QPSO-LR se basa en la optimización por enjambre de partículas, dentro del aplicativo estas partículas representaran las rutas que se generen por cada uno de los puntos identificados. Cada una de las partículas tendrán información sobre su posición, el punto de llegada, la ruta, el costo de la ruta y la mejor ruta de todas. Se calcula el costo total y la ruta para un conjunto dado de posiciones de partículas. Utiliza la matriz de distancias generada anteriormente y devuelve el costo y la ruta asociada.

Dentro del el código se inicializan las diversas variables y estructuras de datos necesarias para la ejecución del algoritmo QPSO-LR tal como se muestran en (Alvarez-Alvarado et al., 2021). Se toma como población inicial de partículas la matriz de distancia con los puntos de recolección con estado “1” y se inicializa la mejor partícula global. A continuación, se entra en el bucle principal del algoritmo, que se ejecuta durante un número máximo de iteraciones.

Dentro del bucle, se actualizan las posiciones y velocidades de las partículas utilizando fórmulas específicas del algoritmo QPSO-LR. Se evalúa el costo de cada partícula y se actualizan los mejores valores individuales y globales. El bucle continúa hasta alcanzar el número máximo de iteraciones. De esta forma se obtiene la partícula con el costo y la ruta más optimizada.

3.3.3 Generación y Optimización de la ruta

Con los puntos de recolección ya identificados y el algoritmo QPSO-LR listo para ser utilizado se procede con la generación de la ruta más optima. Para conseguir que la ruta generada sea vista por el usuario se usa los servicios de Google Maps Platform con el fin de usar la API de Google Directions Service, la cual permite renderizar la ruta generada y lograr optimizarla.

Código 3.3: Módulo QPSO y servicios de Google Maps Platform importados

```
import { QPSO } from “@/utils/QPSO”;
```

```
import {
  DirectionsRenderer,
  DirectionsService,
  GoogleMap,
  Marker,
  useJsApiLoader,
} from "@react-google-maps/api";
```

Como se observa en el código 3.3 se importan los diferentes módulos para realizar la generación de la ruta. El módulo QPSO contiene la codificación del algoritmo QPSO-LR con todos los métodos necesarios para determinar la partícula o ruta más optima y los servicios de la API de Google Directions Service proporcionados por la biblioteca "@react-google-maps/api" para facilitar la integración de Google Maps en aplicaciones o entornos codificados en React.

Código 3.4: Variables declaradas

```
  const [mapcenter, setMapCenter] = useState({ lat: -2.14668, lng: -79.96444 });
  const [location , setLocation] = useState<[number, number]>();
  const router = useRouter().pathname;
  const [contenedores, setContenedors] = useState<ResultContenedoresQuery>([
    { areaContenedoresId: "", estado: "", id : "" },
  ]) ;
  const [areas, setAreas] = useState<CreateAreaInput[]>();
  const [esfuerzos, setEsfuerzos] = useState<CreateEsfuerzoInput[] | any>();
  const [selectedArea, setSelectedArea] = useState<Area>({ nombre: "TODOS" });
  const [selectedContainers, setSelectedContainers] = useState<
    CreateContenedorInput[]
  >([]) ;
  const [startedRoute, setStartedRoute] = useState(false);
  const [pathRoute, setPathRoute] = useState<ResultContenedoresQuery | any>();
  const [nextStep, setNextStep] = useState<CreateContenedorInput | any>();
  const [distanceToNextStep, setDistanceToNextStep] = useState<number>(-1);
  const [response, setResponse] = useState<google.maps.DirectionsResult | null>(
    null
```

```
);
```

El código 3.4 muestra las variables declaradas que serán usadas para la visualización del mapa y la ubicación de los puntos de recolección en este y el proceso de generación, optimización y renderización de la ruta. Se utiliza el hook “useState” para gestionar múltiples estados locales. Cada estado almacena información específica, como la ubicación central del mapa (mapcenter), la ubicación actual (location), información sobre contenedores (contenedores), información sobre áreas (areas), información sobre esfuerzos (esfuerzos), la área seleccionada (selectedArea), los contenedores seleccionados (selectedContainers) y el resto de variables que estarán asociadas al cálculo del siguiente punto de recolección dentro de la ruta y de la distancia en metros que existirá entre el punto actual y siguiente.

Código 3.5: Generación de ruta con QPSO-LR

```
function generateRoute() {
  if (selectedContainers.length != 0) {
    setPathRoute(QPSO(selectedContainers, esfuerzos));
  }
}
```

La función que se muestra en el código 3.5 es la encargada de generar la ruta mediante el uso del algoritmo QPSO-LR, es decir entrega un arreglo con los puntos de recolección ya ordenados. Los parámetros “SelectedContainers” y “esfuerzos” hacen referencia a las puntos de recolección que necesitan con estado “1” y los costos asociados a cada uno de esos puntos. La ruta generada es seteada en el arreglo “setPathRoute” donde, posteriormente, será procesada y renderizada usando los servicios de la API de Directions.

Código 3.6: Cálculo del siguiente punto y su distancia asociada

```
useEffect(() => {
  if (location) {
    if (nextStep) {
      const nextStepCoords = new google.maps.LatLng(
        Number((nextStep as CreateContenedorInput).latitud),
```

```
            Number((nextStep as CreateContenedorInput).longitud)
          );
          const currentLocation = new google.maps.LatLng(
            location [0],
            location [1]
          );
          setDistanceToNextStep(
            google.maps.geometry.spherical.computeDistanceBetween(
              nextStepCoords,
              currentLocation
            )
          );
        }
      }
    }, [location]);

    console.log“(geolocation: ”, location);
    console.log“(distance: ”, distanceToNextStep);

    useEffect(() => {
      getContenedors().then((coords) => {
        setContenedors(coords);
        setSelectedContainers(coords);
      });
      getAreas().then((areas) => setAreas(areas));
      getEsfuerzos().then((esfuerzos) => setEsfuerzos(esfuerzos));
    }, []);

    useEffect(() => {
      setSelectedContainers(
        contenedores.filter ((coord) => {
          if (selectedArea.nombre != “TODOS”)
            return coord.areaContenedoresId == selectedArea.nombre;
          else return coord;
```

```
      })
    );
  }, [selectedArea]);
```

El código 3.6 está compuesto de tres bloques donde se usa el hook “useEffect”. El primer bloque useEffect se ejecuta cada vez que hay un cambio en la ubicación (location). En este caso, si location está presente y existe un nextStep, calcula la distancia entre la ubicación actual y el siguiente paso en la ruta utilizando la API de geometría de Google Maps. La distancia se almacena en el estado distanceToNextStep. Luego, imprime la ubicación y la distancia en la consola.

El segundo bloque se ejecuta una vez al montar el componente (cuando se carga por primera vez). En este bloque, se realizan tres llamadas asincrónicas a funciones (getContenedors, getAreas, y getEsfuerzos) para obtener información sobre contenedores, áreas y esfuerzos. Los resultados se utilizan para actualizar los estados de contenedores, areas, y esfuerzos, respectivamente.

Finalmente, el últmo bloque se ejecuta cada vez que hay un cambio en la selectedArea. Este bloque filtra los contenedores según la selectedArea actual y actualiza el estado selectedContainers con los contenedores filtrados.

Código 3.7: Progreso de la ruta

```
useEffect(() => {
    if (pathRoute) {
      if (distanceToNextStep < 15 && distanceToNextStep !== -1) {
        const nextIndex = pathRoute.indexOf(nextStep) + 1;
        if (nextIndex == pathRoute.length) {
          alert“(RUTA TERMINADA”);
        } else {
          setNextStep(pathRoute[nextIndex]);
        }
      }
    }
  }, [pathRoute, distanceToNextStep]);
  console.log“(Siguiente punto: ”, nextStep);
```

El código 3.7 controla la progresión de la ruta generada, la cual está representada por pathRoute y realiza acciones específicas cuando la distancia al siguiente paso es menor que 15 unidades. Si el siguiente paso es el último en la ruta, muestra una alerta, de lo contrario, actualiza el siguiente paso.

Código 3.8: Solicitud de direcciones

```
const directionsCallback = useCallback(
  (
    result: google.maps.DirectionsResult | null,
    status: google.maps.DirectionsStatus
  ) => {
    if (result !== null) {
      if (status === "OK") {
        setResponse(result);
        setStartedRoute(true);
      } else {
        console.log( "response: ", result);
      }
    }
  },
  []
);
```

La función que se muestra en el código 3.8 se utiliza para manejar la respuesta de una solicitud de direcciones a la API de Google Maps. Si la solicitud es exitosa, actualiza el estado con el resultado y marca que la ruta ha comenzado.

Código 3.9: Renderización de la ruta

```
{router == "/" && pathRoute && (
          <DirectionsService
            options={{
              origin: {
                lat: Number(pathRoute[0].latitud),
                lng: Number(pathRoute[0].longitud),
              },
```

```
        destination: {
          lat: Number(pathRoute.at(-1).latitud),
          lng: Number(pathRoute.at(-1).longitud),
        },
        waypoints: pathRoute.slice(1, -1).map((p: any) => {
          return {
            location: { lat: Number(p.latitud), lng: Number(p.longitud) },
          };
        }),
        travelMode: google.maps.TravelMode.WALKING,
      }}
      callback={directionsCallback}
    />
  )}

  {router == "/" && response && startedRoute == true && (
    <DirectionsRenderer
      // required
      options={{
        directions: response,
        polylineOptions: { visible: false },
      }}
    />
  )}
```

El código 3.9 está compuesto de dos bloques, el primero de ellos renderiza el componente DirectionsService si se cumple que pathRoute existe. DirectionsService se utiliza para solicitar direcciones a la API de Google Maps. La configuración del servicio incluye el origen, el destino, los puntos intermedios (waypoints), y el modo de viaje.

El segundo bloque se utiliza para mostrar visualmente las direcciones en el mapa en función de la ruta generada.

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

En el presente capítulo se muestran los resultados obtenidos en la simulación del escenario de prueba del sistema. Además, de las diferentes pruebas realizadas en herramientas externas para conocer el comportamiento de la aplicación a nivel de rendimiento, carga y tiempo de ejecución. También, se realiza un análisis de satisfacción de los usuarios donde se desglosa de manera gráfica el nivel de aceptación de la aplicación, acorde a las afirmaciones encuestadas en capítulos anteriores, y el impacto que tendrá la incorporación de este nuevo sistema en las actividades de recolección cotidianas dentro del campus. Además, se realiza un análisis del costo relacionado al aplicativo a nivel de Software, Hardware y mano de obra.

4.1 Análisis de la simulación del escenario de prueba

Para realizar las pruebas de simulación necesarias del escenario planteado se usó la herramienta de desarrollo visual Node-RED[1] para demostrar cómo se asemejan el uso y acciones a realizar por los dispositivos IoT en el escenario planteado. Node-RED es muy usado en el desarrollo de sistemas de integración con dispositivos IoT lo cual lo hace una herramienta ideal para interactuar con Hardware.

En la figura 4.1, en la parte "1" de la figura, para simular la acción de enviar datos desde la ESP32 se utiliza un nodo Inject. Este nodo se configurará para inyectar un mensaje que contenga el estado de los contenedores y el ID del contenedor o las coordenadas del contenedor. Además, para fines didácticos de la simulación, se plantea un nodo Inject llamado Reset el cual deja en cero el valor del nodo Counter para simular la acción de vaciar el contenedor.

[1] https://nodered.org/about/

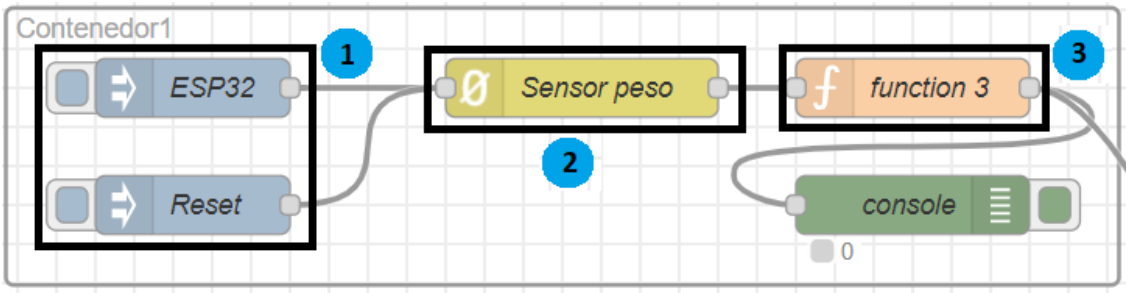

Figura 4.1: Simulación del sistema de recolección de datos

Luego, en la parte “2” de la figura 4.1, el nodo Counter es el que funcionará como el Sensor de peso que se encuentra instalado en los contenedores, tal como se muestra en la arquitectura de Hardware propuesta en los capítulos previos, para ejemplificar de la manera más real posible este proceso, este nodo será el encargado de otorgar un valor de 25 Kilos, hasta llegar a 100, cada vez que se inyecte desde la ESP32; es decir, esto se compara con el funcionamiento en conjunto que existe entre la ESP32, el sensor de peso y el módulo GPS los cuales se encargan de recolectar los datos de sensado y ubicación de cada contenedor.

Sin embargo, al igual que en un microcontrolador, debe existir una función que permita conectar con la base de DynamoDB para publicar actualizaciones del estado de los contenedores, para esto, como se muestra en la parte “3” de la figura 4.1, se usa el nodo Function donde se programa una pequeña función que recibe el conteo del nodo “Sensor peso” y, si se alcanza el límite máximo de 100 Kilos, el estado del contenedor cambia de “No Lleno” a “Lleno”.

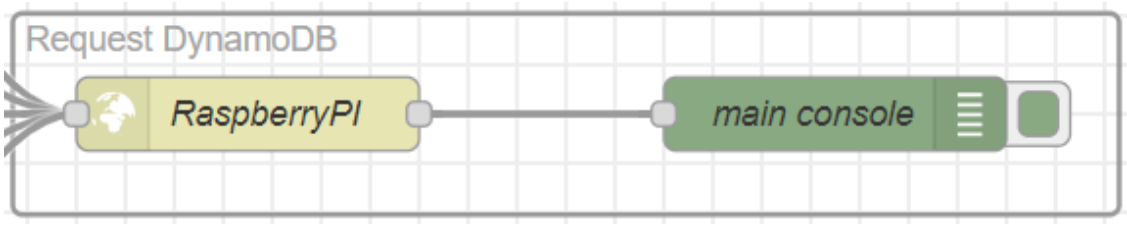

Figura 4.2: Simulación del envío de los datos a las base de DynamoDB

Lo descrito hasta ahora representa la captación de los datos, sin embargo aún no se han enviado los datos hacía la Raspberry para actualizar el estado de los contenedores en la base de DynamoDB. Para ello en la simulación se usó un HTTP Request para conectar con DynamoDB, tal como se aprecia en la figura 4.2, mediante el API KEY de la base en AWS y se consiga realizar la modificación del estado del contenedor acorde a todos los

parámetros obtenidos en los procesos anteriores.

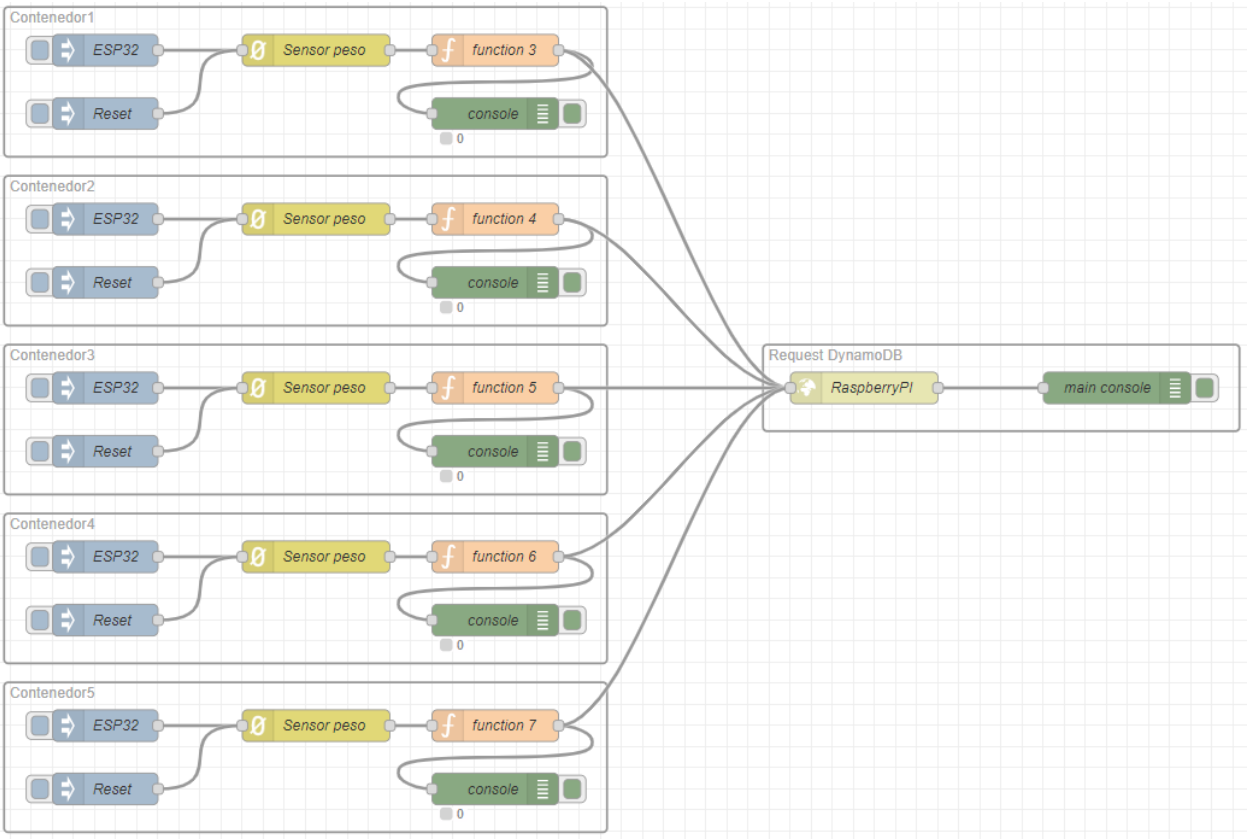

Figura 4.3: Estructura completa de la simulación del escenario de prueba

La estructura completa de la simulación, como se muestra en la figura 4.3, consta de 5 nodos recolectores y el HTTP Request que funciona como un Gateway para receptar todos los datos y, posteriormente, ser enviados. Como ejemplo de la simulación se toman los puntos de recolección de la FIEC para mostrar el funcionamiento en tiempo real de la simulación y cómo se cambia esto en la aplicación. A continuación se detalla los ID de los contenedores dentro de la base de datos DynamoDB.

Tabla 4.1: Listado de contenedores en FIEC

Número de contenedor	ID
1	539a405e-30a9-426d-8fbe-c9056bdad12e
2	bd22ae0c-b073-4d0c-97bd-198e40279408
3	66e1b4f1-fd2b-4f85-b83e-c8e71b9970aa
4	5e6712e0-d664-468c-878a-bce3d34a690f
5	2978e9cd-98cb-44c0-896c-4f1d17f372e2

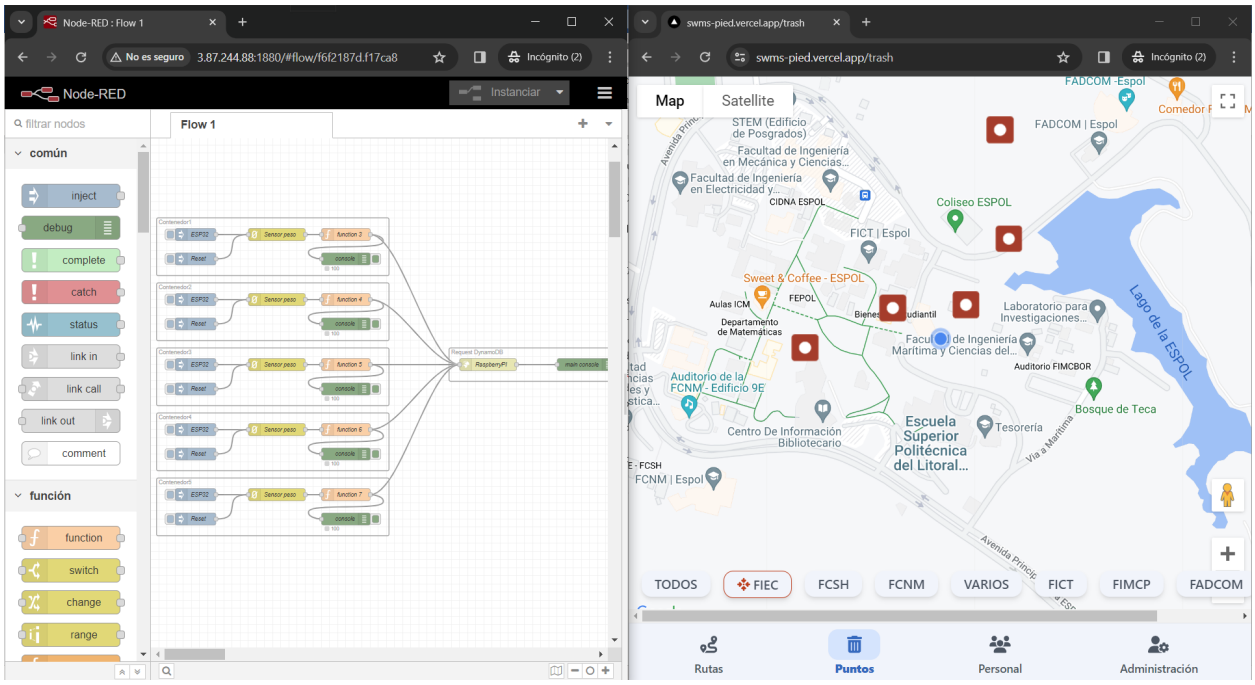

Figura 4.4: Demostración del cambio de estado de “No Lleno” a “Lleno” mediante la simulación

Como se observa en la figura 4.4 se denota el cambio de estado de todos los puntos de recolección de la FIEC, esto se consigue al inyectar un mensaje hasta que el nodo “Sensor peso” llegue a 100 Kilos, lo cual se muestra en la parte inferior de cada nodo Console de cada contenedor, y la función cambie de estado a “Lleno” y, posteriormente, estos cambios sean enviados por el HTTP Request.

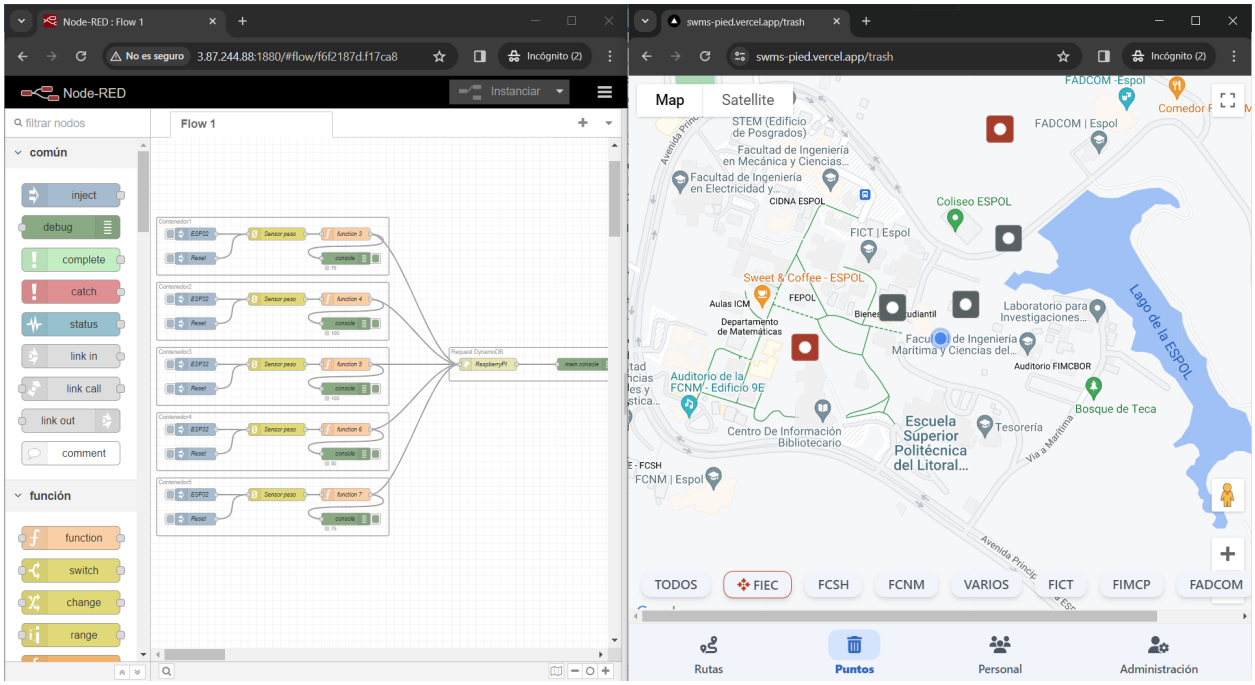

Figura 4.5: Cambio de estado de “No Lleno” a “Lleno” mediante la simulación de algunos puntos

Como se observa en la figura 4.5 se denota el cambio de estado de los puntos de recolección de la FIEC que solo posean una peso de 100 Kilos, lo cual se muestra en la parte inferior de cada nodo Console de cada uno de los contenedores.

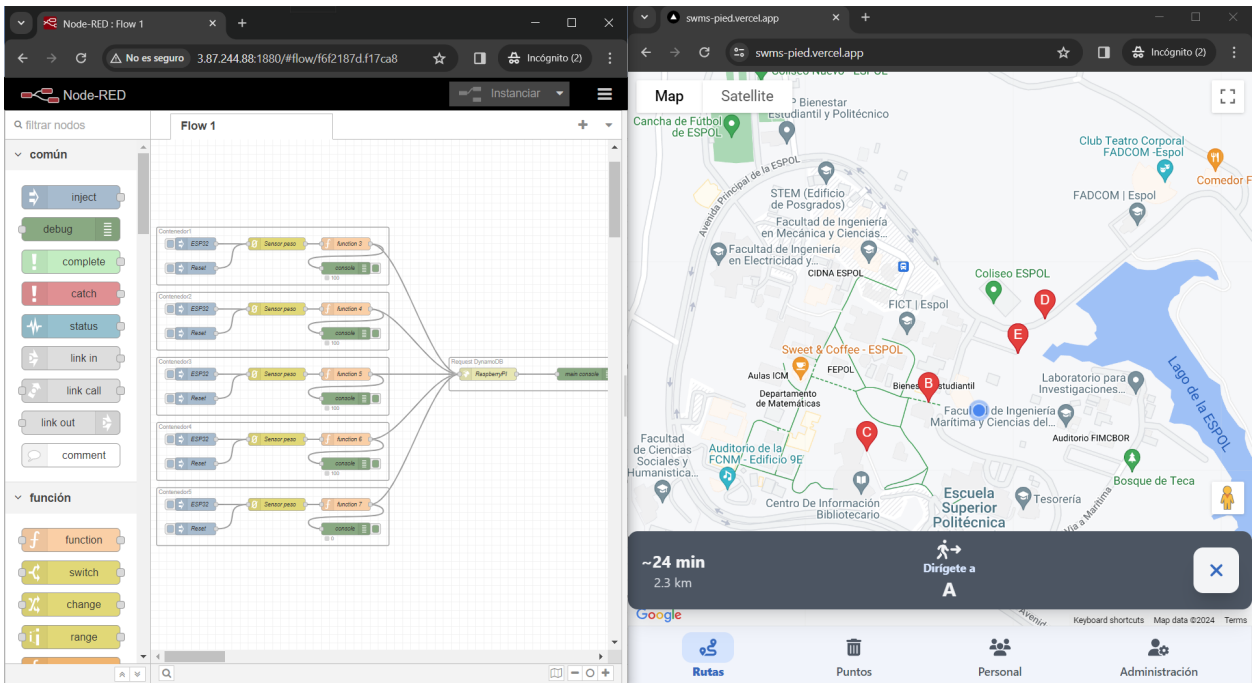

Figura 4.6: Demostración de la generación de la ruta con los puntos de la simulación

Finalmente, en la figura 4.6, se muestra la creación de la ruta con los puntos que

cumplen la condición de llegar a los 100 Kilos los cuales pertenecen a los puntos simulados en Node-RED de la FIEC. Como se aprecia, el cambio de estado ocurre en tiempo real sin afectar la generación de la ruta y respetar el orden planteado de acuerdo con las letras del alfabeto.

4.2 Análisis del sistema

Con el propósito de evaluar la carga máxima simultánea que el sistema puede manejar, se diseñaron varios escenarios para llevar a cabo pruebas de solicitudes HTTP, comenzando con 1 solicitud, luego 100, seguido de 500 y finalmente 1000, donde se puede apreciar notables cambios en la pérdida y éxito en el envío de paquetes. Se enfocó especialmente en cuatro variables distintas:

- **Latencia Media:** Indica el tiempo que el cliente tarda en enviar un paquete y recibir una respuesta.
- **Rendimiento:** Evalúa el rendimiento del sistema bajo una carga generada.
- **Datos Enviados:** Representa la cantidad de información enviada con éxito al servidor.
- **Datos Recibidos:** Son los datos recibidos por el servidor después de realizar una solicitud.

Para llevar a cabo las pruebas de estrés, se utilizó el software Apache JMeter[2] , que genera hilos para realizar envíos simultáneos de solicitudes a un destino específico.

A continuación, se detalla el proceso en los cuatro escenarios y cómo cambiaron las métricas en cada uno de ellos.

4.2.1 Escenario A: Envío de 1 solicitud HTTP

En este caso, se observa que la latencia de la solicitud es de 239 ms. La latencia, que representa el tiempo que transcurre desde que se inicia la solicitud hasta que se obtiene la primera respuesta, al ser solamente una solicitud HTTP se considera que el tiempo de

[2]https://jmeter.apache.org/

Tabla 4.2: Tabla de resultados del escenario A

Métrica	Página de inicio
Latencia(ms)	239
Rendimiento(ms)	2,1
Datos enviados(kb/s)	0,50
Datos recibidos(kb/s)	4,36

latencia es razonable. Por otro lado, el rendimiento refleja el tiempo que tarda el sistema en procesar y responder a la solicitud. Un rendimiento de 2,1 ms sugiere una respuesta rápida por parte del servidor, lo cual es fundamental para una experiencia de usuario fluida.

En cuanto a los datos enviados y recibidos, se observa que se están transmitiendo 0,50 kilobytes por segundo y se están recibiendo 4,36 kB por segundo. Estos valores indican la cantidad de información que se está moviendo entre el cliente y el servidor durante la ejecución de la solicitud. El hecho de que se envíen menos datos de los que se reciben sugiere que la mayor parte de la información está siendo proporcionada por el servidor en respuesta a la solicitud del cliente. Además, la velocidad de transmisión de datos recibidos indica una respuesta eficiente y rápida en la entrega de la información solicitada.

4.2.2 Escenario B: Envío de 100 solicitudes HTTP

Para este escenario se muestra un incremento en la latencia en contraste con el escenario anterior, esto se debe a que hay un mayor número de solicitudes, por lo que es normal que su latencia haya aumentado. Una latencia más elevada podría indicar que la aplicación está experimentando cierta carga, lo que puede afectar la capacidad de respuesta para cada solicitud individual.

El rendimiento en este escenario indica una mayor carga o complejidad en el backend de la aplicación. Hay que tener en cuenta que esta métrica, es un indicador del desempeño o esfuerzo de la red por donde se transportan las solicitudes, es decir muestra un comportamiento directamente proporcional, por lo que a mayor cantidad de solicitudes,

Tabla 4.3: Tabla de resultados del escenario B

Métrica	Página de inicio
Latencia(ms)	472
Rendimiento(ms)	68,8
Datos enviados(kb/s)	16,40
Datos recibidos(kb/s)	143,56

el rendimiento también será mayor.

En lo que respecta a los datos enviados y recibidos, se observa que se están transmitiendo 16,40 kB por segundo y se están recibiendo 143,56 kB por segundo. Estos valores más altos en comparación con el escenario de una única solicitud indican la carga adicional debida a la realización de múltiples solicitudes simultáneas.

Al ejecutar la consulta de 100 solicitudes se generó la gráfica que se observa en la figura 4.7, esta se muestra creciente en segunda mitad de su dominio. Es decir que el tiempo empleado para generar respuestas era cada vez mayor, lo cual puede deberse a la sobrecarga que se generó en el servidor. Además, conforme el tiempo sigue en aumento, se ve que va disminuyendo y nuevamente aumenta, esto es por la calidad que ofreció la red al momento de realizar la prueba del sistema.

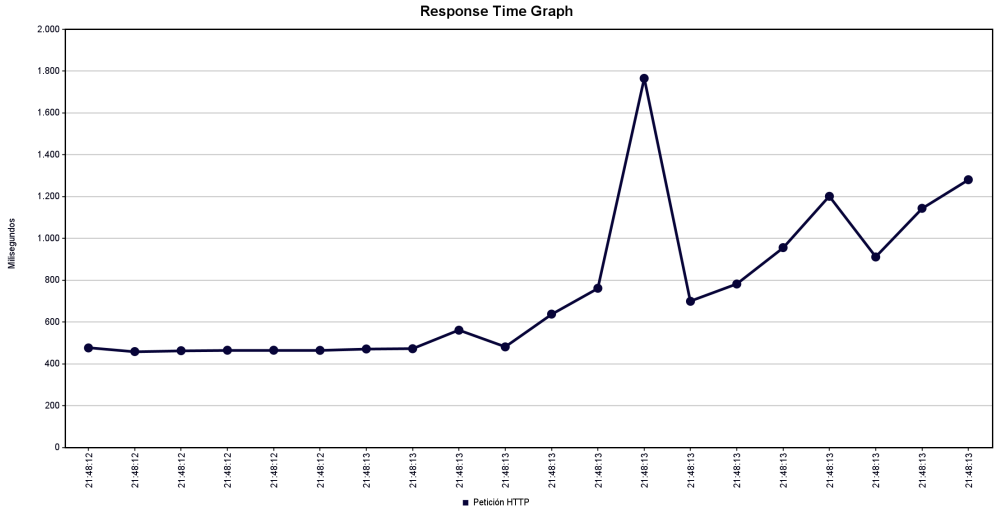

Figura 4.7: Tiempo de respuesta de la página de inicio con 100 solicitudes

4.2.3 Escenario C: Envío de 500 solicitudes HTTP

Al ejecutarse las 500 solicitudes, se puede observar que la latencia aumentó 3 veces más de lo que se tenía en el escenario de anterior, los datos recibidos y enviados conservan el mismo comportamiento que en los escenarios anteriores, lo que da a entender que el sistema posee un nivel de respuesta acorde a las consultas realizadas. En cuanto al rendimiento, aumentó un poco en comparación con los escenarios previos y la relación con la latencia es un poco menor que los otros escenarios, esto se debe a que hay un mayor número de solicitudes lo que ocasiona que el rendimiento se vea afectado. En la

Tabla 4.4: Tabla de resultados del escenario C

Métrica	Página de inicio
Latencia(ms)	1237
Rendimiento(ms)	188,1
Datos enviados(kb/s)	44,82
Datos recibidos(kb/s)	392,38

figura 4.8, se puede apreciarse la latencia de las 500 solicitudes por segundo. A diferencia de la gráfica anterior, se puede observar que la latencia posee una silueta en forma de curva donde resaltan dos picos, comienza de forma descendente y subiendo de manera constante hasta adquirir esta forma. En cuanto a los picos, se puede asumir que durante la prueba, la calidad de la red estuvo ligeramente oscilante.

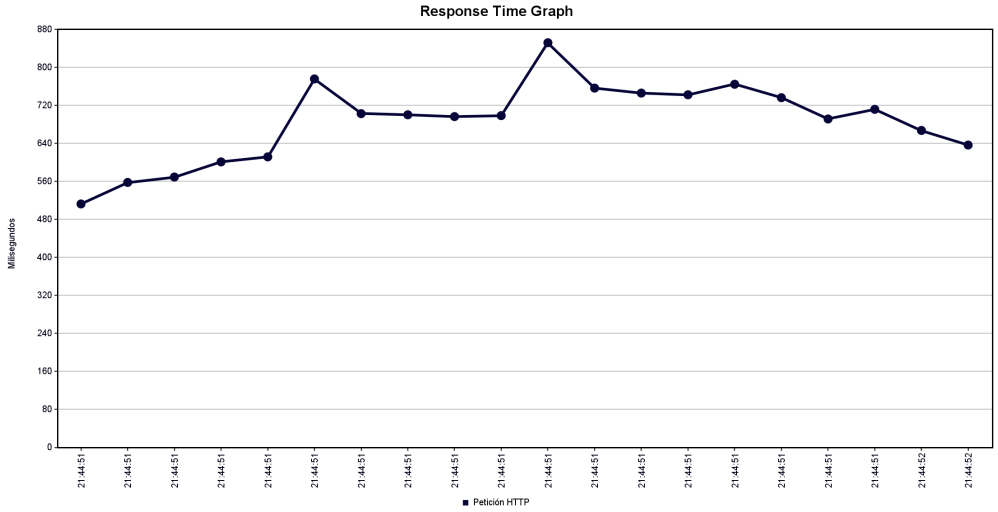

Figura 4.8: Tiempo de respuesta de la página de inicio con 500 solicitudes

4.2.4 Escenario D: Envío de 1000 solicitudes HTTP

Para este último escenario, se puede apreciar que la latencia aumenta de manera muy considerable. Este valor más alto sugiere una mayor carga de trabajo, lo que indica que el servidor está experimentando una carga significativa. Por otro lado, aunque el valor del rendimiento en comparación con la latencia es relativamente bajo, la latencia más alta podría indicar que el backend de la aplicación está enfrentando una carga sustancial, lo que podría influir en la eficiencia del procesamiento de solicitudes.

Tabla 4.5: Tabla de resultados del escenario D

Métrica	Página de inicio
Latencia(ms)	2098
Rendimiento(ms)	195
Datos enviados(kb/s)	46,46
Datos recibidos(kb/s)	406,65

Con respecto a los datos enviados y recibidos, se aprecia que estos valores son más altos en comparación con escenarios de menor carga indicando la presión adicional sobre los recursos de red y del servidor al manejar un mayor número de solicitudes.

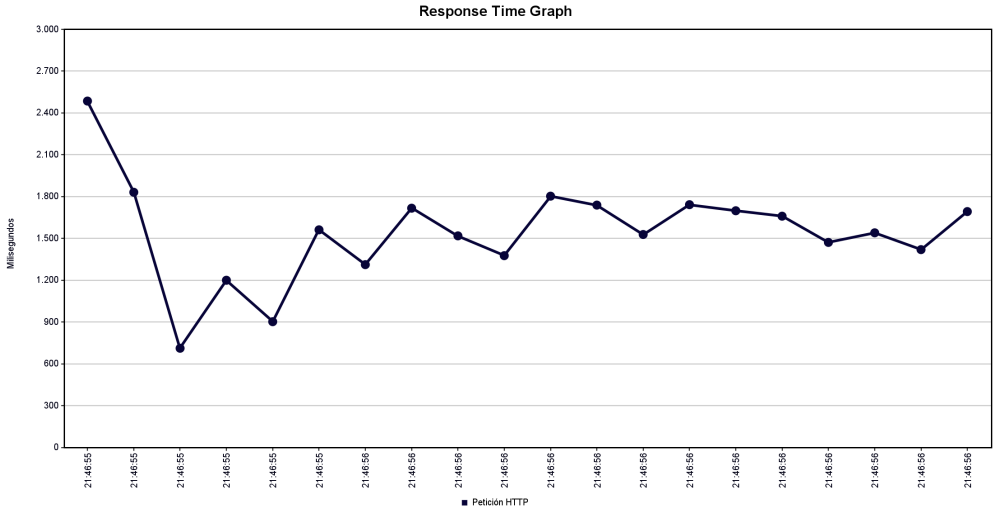

Figura 4.9: Tiempo de respuesta de la página de inicio con 1000 solicitudes

Finalmente, en la gráfica que se muestra en la figura 4.9, se puede apreciar la latencia de las 1000 solicitudes por segundo. Al igual que en el escenario anterior, se observan picos a lo largo de la gráfica, sin embargo estos picos se presentan en la mayoría del dominio de la esta. En este caso, podemos suponer que la calidad de la red osciló más a comparación con los demás escenarios.

4.3 Análisis de satisfacción del usuario

Mediante una muestra realizada a 29 auxiliares de servicio de distintas facultades, se buscó analizar la aceptación de la propuesta, presentando como exposición el sistema, junto con la aplicación que permite interactuar con este. Cabe recalcar que la encuesta está estructurada de modo de que las personas encuestadas muestren su opinión a través de una escala numérica frente a ciertas afirmaciones presentadas dentro de la encuesta, de esta forma se puede conocer las diferentes posturas para proceder a calcular la satisfacción neta descrita en el capítulo 2 utilizando la fórmula 2.2.

Conocer el impacto del nuevo sistema en comparación con el actual es crucial para la propuesta de aplicación que se ha planteado, por lo que es importante que los usuarios puedan percibir que va a existir una mejora al momento de utilizar el aplicativo. La encuesta está formada por 9 afirmaciones, todas ellas están relacionadas a lo que el usuario espera conseguir con la implementación de la propuesta, el beneficio que se puede obtener a futuro, el aporte ambiental que se logra con la implementación

del sistema en comparación con la metodología actual y el nivel adaptación que experimentarán al cambiar de un sistema a otro.

Desde un panorama general, las personas encuestadas opinan que la propuesta tendrá un gran impacto de manera positiva a la manera de recolectar los desechos y que se conseguirá reducir los niveles de contaminación ocasionados por el manejo erróneo de los desechos al momento de realizar su respectiva recolección.

Figura 4.10: El sistema de recolección de residuos es accesible para todos los usuarios (Administradores, recolectores y supervisor).

En esta afirmación se expone los roles que tendrán los diferentes usuarios de la aplicación, se muestra que los usuarios están totalmente de acuerdo con esta manera de seccionar los diferentes cargos que se han definido en la aplicación, lo cual, en opinión de algunos de los encuestados, les resulta apropiado ya que existirá un mayor control de las tareas asignadas para cada persona, evitando de esta forma una asignación desproporcionada de las actividades. Además, refleja la necesidad y el deseo de que existan personas que puedan supervisar el trabajo realizado y conseguir maximizar el tiempo de cada uno de los auxiliares de servicio, ya que habría una persona con un rol designado exclusivamente para llevar el manejo y control de los contenedores que requieren ser vaciados.

Figura 4.11: El sistema es lo suficientemente visual para los usuarios.

Esta afirmación se centra en la manera en que los usuarios perciben la aplicación al momento de verla, es decir si ellos consideran que el sistema propuesta es lo suficientemente explícito e intuitivo al momento de interactuar con la interfaz y si la distribución de los diferentes componentes es la adecuada para las actividades de recolección de desechos. Como se puede observar, existe un gran nivel de aceptación por la propuesta de Frontend y refleja que el sistema es amigable con el usuario.

Figura 4.12: El sistema es accesible y abarca todas las áreas del campus.

Uno de los objetivos esenciales para los auxiliares de servicio es que el sistema

sea accesible para cada uno de los usuarios que lo usan, además de que uno de los requerimientos por cumplir era de que cada auxiliar de cada área pueda acceder únicamente a los contenedores asignados por área, es decir a la facultad en la cual se encuentran laborando. Los encuestadores mostraron una gran aceptación, debido a que el aplicativo permite escoger el área asignada y así mostrar solamente los contenedores asociados a esa área.

Figura 4.13: El sistema implementado promueve prácticas sostenibles y amigables con el medio ambiente.

El sistema no solo busca beneficiar a los auxiliares de servicio, sino también aportar en la sostenibilidad del ecosistema del campus y alinearse a las necesidades tanto del usuario como del medio ambiente. Esta afirmación promueve el objetivo del sistema, el cual es usar diferentes herramientas tecnológicas de Software y Hardware para ofrecer una solución sostenible a la recolección de desechos dentro del campus. Los encuestados se mostraron a favor de esta afirmación mencionando que como trabajadores, es una preocupación constante el observar prácticas poco amigables y sostenibles con el sistema de recolección actual, por lo que implementar un sistema que les ofrezca una nueva manera de recolección les parece llamativa y oportuna. Además, esta afirmación cuenta con la aceptación del 96.55% de los encuestados.

Figura 4.14: El sistema proporciona una solución adecuada al problema actual sobre la manera de cómo realizar la recolección de los desechos.

Actualmente, en el campus, no existe un sistema de recolección de residuos, los auxiliares de manera intuitiva y empírica han descubierto cuáles son los puntos más cercanos para ser recolectados. Sin embargo, no hay una manera de conocer si estos contenedores deben ser necesariamente vaciados; por lo que, el aplicativo proporcionaría esta información adicional al conserje optimizando el tiempo de recolección, ya que no desperdiciarían tiempo en ir a cada uno de los contenedores, sino solo a aquellos que lo requieran.

Figura 4.15: El tiempo de recolección actual será menor al momento de implementar el nuevo sistema.

Al generar una ruta más óptima entre los contenedores que necesiten ser vaciados, el tiempo se reduce considerablemente puesto que solo se toman en cuanta a aquellos contenedores que el Supervisor indique en el mapa que están llenos.

Figura 4.16: El tiempo de adaptación del nuevo sistema les resultará más rápido a los usuarios.

El sistema es muy intuitivo, al igual que en otras aplicaciones de uso diario, se requiere un inicio de sesión con un correo electrónico y contraseña para acceder al aplicativo. La manera en que la interfaz fue desarrollada permite que el usuario se pueda adaptar de manera ágil al aplicativo, cada una de las pestañas del aplicativo indica de manera visual la acción que puede desarrollar. Al interactuar con ella, los conserjes pudieron observar el proceso de generación de la ruta y la representación de cada uno de los contenedores dentro de la aplicación dando como resultado un porcentaje de aceptación del 82.75%.

Figura 4.17: La frecuencia de recolección de residuos es suficiente para mantener limpio el campus.

Con la implementación del sistema, los conserjes consideran que sí será suficiente para mantener un mejor aseo en los diferentes puntos de recolección, ya que se conseguiría evitar la aglomeración de desechos y mantener los contenedores más limpios.

Figura 4.18: La implementación de un nuevo sistema de recolección de residuos resultará más beneficioso a largo plazo que el sistema actual.

La introducción de un nuevo sistema de recolección de residuos se perfila como una inversión a largo plazo altamente beneficiosa en comparación con el sistema actual.

Este enfoque innovador, promete mejorar significativamente la eficiencia operativa y la sostenibilidad. La optimización de rutas y la coordinación eficaz del personal de servicio, junto con la capacidad de adaptarse a las cambiantes condiciones del entorno, proporcionan ventajas notables en términos de costos, tiempos de recolección reducidos y un impacto ambiental más positivo. Esta afirmación cuenta con el 93.10% de aceptación por parte de los conserjes.

Tabla 4.6: Cálculo de Satisfacción Neta

Afirmación	Auxiliares que puntuaron con 4 y 5	Satisfacción Neta(%)
1	21	72.41%
2	21	72.41%
3	23	79.31%
4	28	96.55%
5	27	93.10%
6	23	79.31%
7	24	82.75%
8	24	82.75%
9	27	93.10%
Promedio		83.52%

La tabla 4.6 desglosa por afirmación dentro de la encuesta, el total de auxiliares de servicio que calificaron cada afirmación con 4 y 5. Con ese total, de forma individual se obtiene la satisfacción neta por pregunta; sin embargo, al ser de interés general conocer el nivel de aceptación, se realiza un promedio, obteniendo un resultado que indica una aceptación del 83.52%.

4.4 Análisis de costos

En esta sección se incluyen los aspectos a tener en cuenta como un presupuesto para la reproducción del sistema. Esto incluye los gastos relacionados con hardware, software y

mano de obra. Se realiza una estimación proyectada de cuatro meses para el desarrollo completo de la solución, determinando así el valor total de la inversión.

4.4.1 Mano de obra

Tabla 4.7: Costo de mano de obra

Programador	Rol	Costo por hora	Sueldo
Javier Dillon	Frontend	$5	$1200
Danny Medina	Backend	$5	$1200
Total			$2400

Los costos asociados a la implementación por parte de desarrolladores profesionales se encuentran asociados a la tabla salarial del Ministerio de Trabajo, por lo que estos valores corresponden al salario mensual que percibe un desarrollador Full Stack Junior, el cual posee todos los conocimientos necesarios para programar, evaluar y analizar el proceso del proyecto en cada una de sus etapas iniciales y finales, tanto en Frontend como en Backend. Además, se muestra el detalle del pago por cada hora de trabajo considerando la carga laboral de 8 horas diarias establecido en el Código del Trabajo de la constitución de la República del Ecuador.

4.4.2 Hardware

Los precios de desarrollo y producción de Hardware forman parte del costo total final del proyecto. En la tabla 4.8 se puede observar el detalle de los componentes utilizados a lo largo de la realización del proyecto, los cuales están asociados a los dispositivos electrónicos personales hasta los materiales IoT a considerar en la implementación del proyecto, los valores de estos dispositivos varían acorde a la función que cada uno de estos ocupen dentro del proyecto. Además, cabe mencionar que la cantidad de dispositivos IoT considerados en el proyecto es proporcional a la cantidad de puntos de recolección identificados dentro del campus Gustavo Galindo de ESPOL.

Tabla 4.8: Tabla de implementos de trabajo: Hardware

Dispositivo	Costo unitario	Cantidad	Subtotal
Lenovo Yoga: i7 10th Gen	$1500	2	$3000
ESP32	$12.50	6	$825
Rapsberry Pi 4	$95	3	$285
NEO6MV2	$8.50	66	$561
HX711	$9.50	66	$627
Total			$5298

4.4.3 Software

Tabla 4.9: Tabla de implementos de trabajo: Software

Componente	Costo unitario	Cantidad	Subtotal
Servicios de Google Maps: Directions	-	-	-
Consultas API Directions	$0.005	$2000	$10
Servios de AWS: DynamoDB	-	-	-
Servios de AWS: CloudWatch	-	-	-
Servios de AWS: Cognito	-	-	-
Licencia Windows Home	-	-	-
Total			$10

Los costos de producción en el desarrollo de Software están asociados a los servicios y/o componentes utilizados en el proyecto, con el propósito de otorgarle un funcionamiento óptimo y adecuado al sistema acorde a los requerimientos solicitados por el cliente. Para el desglose de componentes de Software se consideraron las licencias y servicios proporcionados por plataformas como AWS y Google Maps Platform, muchos de estos servicios no poseen costo alguno relacionado, debido a que el alcance del proyecto

permitía usar los servicios de la capa gratuita de cada una de las plataformas. Por otro lado, el único servicio que tuvo un costo asociado fue el ofrecido por la API de Maps JavaScript llamado Directions, el cual cobra céntimas de centavos por cada una de las consultas realizadas a su API. En este caso, para las pruebas e implementación del sistema se usaron 2000 consultas, por lo que el valor mencionado en la tabla 4.9 se asocia al número de consultas realizadas.

Sin embargo, si se desea escalar el proyecto a organizaciones o instituciones con mayor cantidad de puntos de recolección de basura, es altamente recomendable adquirir los servicios de pago para obtener un sistema más robusto, eficiente y que cumpla con las necesidades y requerimientos del cliente.

4.4.4 Costo final de implementación

Tabla 4.10: Costo final del proyecto

Recursos	Costos
Hardware	$5298
Software	$10
Mano de obra (Por 4 meses)	$9600
Total	$14,908

Una vez que se han establecido los costos relacionados a la mano de obra por 4 meses de trabajo, hardware y software, se calcula el costo final del proyecto integrador. La tabla 4.10 muestra este costo siendo un total de $14,908 abarcando la implementación del sistema completo y la instalación de los dispositivos IoT para el monitoreo de los contenedores. Debido a que el presente proyecto se considera un servicio de suscripción anual dirigido a organizaciones o instituciones, se considera que este precio base es adecuado para la implementación dentro del campus de la ESPOL. Sin embargo, para un funcionamiento más eficiente y apegado a las necesidades reales de la institución, se debe usar un plan de suscripción mensual de no menos de un millón de consultas a la API proporcionados por el servicio de Directions, lo cual representaría un costo adicional de $4000 dando como costo final $18,908.

CAPÍTULO 5

5. CONCLUSIONES Y LINEAS FUTURAS

En este capítulo se contemplan las conclusiones y recomendaciones obtenidas a partir del desarrollo del proyecto para poder cumplir con los objetivos planteados al inicio del trabajo. A lo largo del capítulo se explica de forma sintetizada qué fue lo que limitó ciertos resultados así como cuáles son los puntos importantes del sistema propuesto.

5.1 Conclusiones

- La aplicación desempeña un papel crucial en el manejo y control de desechos al permitir la recolección de estos en puntos estratégicos definidos como contenedores. Este enfoque no solo agiliza esta actividad, sino que también asegura que los desechos sean retirados cuando sea necesario, contribuyendo así al mantenimiento de un ambiente más limpio y sostenible.

- La aplicación, al utilizar el algoritmo cuántico QPSO-LR, logra una optimización efectiva de las rutas para los auxiliares de servicio encargados de la recolección de desechos. Este enfoque cuántico permite trazar rutas óptimas, reduciendo significativamente el tiempo necesario para realizar las actividades diarias. La integración de tecnologías en la nube, específicamente AWS, para el ingreso y control de acceso, asegura una plataforma segura y accesible, garantizando que solo personal autorizado tenga acceso al sistema, protegiendo así la privacidad y la integridad de los datos.

- La integración de tecnologías en la nube, específicamente a través de AWS, ha permitido un acceso seguro y eficiente a nuestra aplicación. El control de acceso y la gestión de datos sensibles se han llevado a cabo de manera efectiva, asegurando

que solo usuarios autorizados, con sus roles designados, puedan acceder y operar en la plataforma. Esto no solo garantiza la seguridad de la información, sino que también proporciona una base sólida para el escalado y la expansión de la aplicación a medida que crece su uso. También, la integración de la API de Google Maps Platform permite la visualización eficiente de los puntos de recolección en el mapa, brindando a los auxiliares de servicio una visión clara y detallada de las ubicaciones y rutas planificadas.

- La elección de desarrollar la aplicación como una PWA ofrece accesibilidad desde cualquier navegador web, lo que mejora la flexibilidad y la disponibilidad para los usuarios. Esto asegura que los auxiliares de servicio puedan acceder a la aplicación desde diversos dispositivos sin necesidad de instalación adicional. La implementación en navegadores web también facilita la actualización constante de la aplicación, garantizando que siempre utilice las últimas funcionalidades y mejoras sin requerir la intervención del usuario.

- El aplicativo, al asignar a cada usuario uno de los tres roles específicos (Administrador, Supervisor o Recolector), facilita una organización más efectiva del personal de servicio. Cada rol tiene funciones claras, lo que mejora la coordinación, la asignación de tareas y el seguimiento del progreso. La segmentación de usuarios por áreas asignadas en la aplicación garantiza una distribución equitativa de responsabilidades y optimiza la gestión del personal, permitiendo una cobertura eficiente de todas las áreas designadas.

- Además, el uso de esta aplicación tiene un impacto significativo en la eficiencia operativa del servicio de recolección de desechos. Desde la optimización de rutas hasta la gestión organizativa y el impacto ambiental positivo, la aplicación demuestra ser una herramienta integral que mejora la calidad y eficiencia del servicio. Este enfoque tecnológico no solo beneficia a los auxiliares de servicio, sino que también contribuye a un manejo más efectivo y sostenible de los desechos, apoyando así la salud del ecosistema y promoviendo prácticas de desarrollo sostenible.

5.2 Recomendaciones

- Para mantener la eficacia de la aplicación a largo plazo, se recomienda implementar un proceso de optimización continua de rutas. Esto implica monitorear el rendimiento del algoritmo cuántico QPSO-LR y realizar actualizaciones periódicas según las fluctuaciones en los puntos de recolección. Además, se sugiere incorporar funcionalidades predictivas que utilicen datos históricos para prever posibles cambios en las rutas y anticiparse a eventos que puedan afectar la eficiencia de la recolección. Esto asegurará que la aplicación siempre ofrezca las rutas óptimas, mejorando continuamente la experiencia para los auxiliares de servicio.

- Dada la importancia de la seguridad y la accesibilidad, se recomienda explorar y ampliar la integración con tecnologías en la nube, aprovechando las capacidades de AWS para garantizar la escalabilidad de la aplicación. Esto incluiría la implementación de servicios adicionales que mejoren la gestión de datos, el rendimiento y la redundancia del sistema. La expansión de la integración con tecnologías en la nube también permitirá una mayor flexibilidad y adaptabilidad a medida que la aplicación crezca en usuarios y requerimientos. Se debe considerar la posibilidad de implementar servicios de monitoreo y gestión automatizada para optimizar los recursos de la nube, asegurando una operación más eficiente.

- La interfaz de usuario juega un papel fundamental en la adopción y eficiencia de la aplicación. Se recomienda realizar evaluaciones continuas de la UI/UX para garantizar que sea intuitiva y fácil de usar para los auxiliares de servicio. Esto incluye la implementación de herramientas de retroalimentación directa de los usuarios para identificar áreas de mejora. Además, se sugiere incorporar funciones personalizables en la interfaz para que los usuarios puedan adaptar la aplicación según sus preferencias y necesidades específicas. Una interfaz de usuario bien diseñada no solo agilizará la adopción de la aplicación, sino que también mejorará la eficiencia y satisfacción de los usuarios.

- Para garantizar un monitoreo efectivo del desempeño y la eficacia de la aplicación, se recomienda la implementación de herramientas analíticas. Esto incluiría paneles

de control que proporcionen información en tiempo real sobre el uso de la aplicación, la eficiencia de las rutas, y la productividad del personal de servicio. La recopilación y análisis de datos permitirán identificar patrones, realizar evaluaciones de impacto y tomar decisiones informadas para mejorar la funcionalidad de la aplicación.

5.3 Líneas Futuras

- La implementación de sensores en los contenedores de desechos permitiría un monitoreo en tiempo real del nivel de llenado. Esto facilitaría una recolección más eficiente, ya que los auxiliares de servicio podrían priorizar los contenedores que necesitan atención inmediata, reduciendo así tiempos muertos y mejorando la sostenibilidad ambiental al evitar recogidas innecesarias. Por otro lado, la conexión de la aplicación con dispositivos IoT también abriría oportunidades para la automatización de ciertos procesos, mejorando aún más la eficiencia operativa y permitiendo una respuesta más rápida a las necesidades cambiantes de la recolección de desechos.
- La implementación de actualizaciones frecuentes y la exploración de técnicas predictivas permitirán anticiparse a cambios en los patrones de recolección de desechos y ajustar las rutas de manera proactiva. Además, se sugiere considerar la integración de tecnologías emergentes como Machine Learning para mejorar la capacidad predictiva de la aplicación. Esto permitiría adaptarse dinámicamente a cambios en el comportamiento de recolección y optimizar las rutas de manera más eficiente.
- Considerando la organización del personal y la segmentación por áreas asignadas, se sugiere expandir las funcionalidades de colaboración y comunicación dentro de la aplicación. La integración de herramientas de comunicación en tiempo real, como chats, facilitará la coordinación entre los roles de Administrador, Supervisor y Recolector. Además, se podría implementar un sistema de notificaciones inteligentes para informar a los usuarios sobre cambios en las rutas, eventos importantes o actualizaciones críticas.
- A medida que evolucionan las tecnologías y cambian los estándares ambientales, la aplicación debe adaptarse para seguir siendo relevante y efectiva. La

aplicación podría incorporar métricas ambientales, como la reducción de emisiones de carbono o la cantidad de desechos reciclados, para medir su impacto en la sostenibilidad. La adaptación constante a las innovaciones tecnológicas y los estándares medioambientales asegurará que la aplicación siga siendo una herramienta líder en la gestión moderna de residuos.

BIBLIOGRAFÍA

Alvarez-Alvarado, M., Alban-Chacón, F., Lamilla-Rubio, E., Rodríguez-Gallegos, C., & Velásquez, W. (2021). Three novel quantum-inspired swarm optimization algorithms using different bounded potential fields, nature scientific reports 11 (2021) 11655. *URL: https://www. nature. com/articles/s41598-021-90847-7*.

Bermello Giler, D. R. (2021). *Impacto ambiental ocasionado por desechos sólidos generados en el control de plagas y enfermedades en bananeras del cantón valencia, 2021* [Master's thesis, Quevedo-Ecuador].

Betanzo-Quezada, E., Torres-Gurrola, M. Á., Romero-Navarrete, J. A., & Obregón-Biosca, S. A. (2016). Evaluación de rutas de recolección de residuos sólidos urbanos con apoyo de dispositivos de rastreo satelital: Análisis e implicaciones. *Revista internacional de contaminación ambiental*, *32*(3), 323–337.

Cáceres Sánchez, R. E. (2022). Metodología para establecer los costos para la gestión de los residuos sólidos en la emgirs ep.

Calle-Loyola, J. E., & Solís-Muñoz, J. (2021). Estudio del manejo de desechos sólidos e impacto en la población de la troncal, ecuador. *CIENCIAMATRIA*, *7*(3), 1082–1110.

Cardenas Cabrera, C., & Cuadra Arevalo, J. A. (2022). Sistemas de información geográfica para optimizar la ruta de recolección de residuos sólidos municipales, moche, 2022.

Carrera, B., & Mendez, Y. (2020). Axiología ambiental para el desarrollo sustentable desde el aprovechamiento de los residuos sólidos. *Revista AMBIENTIS Occidentales*, *2*.

Chafla Borja, E. J. (2020). *La contaminación de los desechos sólidos y el derecho a vivir en un ambiente sano* [B.S. thesis].

Chaveinte García, M., et al. (2023). Caso de estudio de aplicación de algoritmos genéticos para la optimización de rutas marítimas.

Choez-Segovia, C. J., Parrales-Cantos, G. N., & Alvarez-Alvarez, M. J. (2021). Influencia de la recolección de desechos sólidos en la operación del relleno sanitario de jipijapa. *Dominio de las Ciencias*, *7*(2), 1417–1432.

Contreras Silva, C. A. (2021). *Procedimientos de servución para la gestión de recolección de desechos mediante el uso de las tecnologías de información y la metodología design thinking* [B.S. thesis]. Universidad del Azuay.

Dedios Carrasco, J. L. (2023). Contaminación ambiental y de la salud generado por actividades antrópicas en el mercado de talara y propuesta de medidas de solución y/o mitigación 2022.

de Romero, J. G., García, J. C., Gavidia, A., & Santana, A. G. V. (2020). Desarrollo sostenible: Desde la mirada de preservación del medio ambiente colombiano. *Revista de Ciencias Sociales*, *26*(4), 293–307.

Elhemali, M., Gallagher, N., Tang, B., Gordon, N., Huang, H., Chen, H., Idziorek, J., Wang, M., Krog, R., Zhu, Z., et al. (2022). Amazon {dynamodb}: A scalable, predictably performant, and fully managed {nosql} database service. *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 1037–1048.

Ferrer, J., & Alba, E. (2019). Bin-ct: Urban waste collection based on predicting the container fill level. *Biosystems*, *186*, 103962.

García Chumacero, R. A., & Guerrero Olaya, L. F. (2023). Gestión de residuos agrarios.

Gómez, P. L., & Mozo, H. P. B. (2021). La gestión ambiental en los gobiernos locales en américa latina. *Ciencia Latina Revista Científica Multidisciplinar*, *5*(1), 212–228.

Guamán Pacheco, L. J., & López López, K. A. (2023). Desarrollo de un sistema inteligente para la gestión de la recolección de desechos sólidos urbanos basado en comunicación inalámbrica.

Guerrero Garcia, J. J., Limones Obando, G. J., et al. (2018). *Diseño de un sistema de monitoreo y optimización de la recolección de desechos sólidos para el campus de la espol* [B.S. thesis]. Espol.

Hernández Navarro, J. L. (2019). Sistema para optimización de rutas de recolección de residuos sólidos urbanos implementando algoritmos genéticos.

Jiménez Ordoñez, H. A., & Jiménez Ordoñez, A. A. (2020). Prácticas pedagógicas agroambientales que promueven el reconocimiento de la diversidad del territorio, en el grado décimo de la institución educativa nuestra señora del rosario, municipio de san sebastián, cauca.

Kumar, N., Mahato, S. K., & Bhunia, A. K. (2020). A new qpso based hybrid algorithm for constrained optimization problems via tournamenting process. *Soft Computing*, *24*, 11365–11379.

López, M., Villagra, A., & Pandolfi, D. (2022). Optimización de rutas en la recolección de residuos tecnológicos. *Informes Científicos Técnicos-UNPA*, *14*(3).

Machado, J. T., & Saldaña, Y. M. V. (2022). Manejo de residuos sólidos para reducir la contaminación del medio ambiente: Revisión sistemática. *Ciencia Latina Revista Científica Multidisciplinar*, *6*(4), 578–601.

Mayorga, D. I. M., Córdova, H. D. F., García, S. T. D. L. C., & Villegas, J. B. S. (2021). Efectos de la contaminación ambiental producidos por los desechos sólidos. *Pro Sciences: Revista de Producción, Ciencias e Investigación*, *5*(38), 149–155.

Montañez Gomez, M. A. (2021). *Prototipo de sistema de monitoreo basado en una red inalámbrica de sensores simulada, como apoyo a la planeación de rutas de recolección de residuo sólido urbano.* [Doctoral dissertation, Facultad Ingeniería de Sistemas e Industrial].

Muñoz Santiana, C. H. (2021). *Optimización de rutas para la recolección de residuos sólidos del cantón latacunga* [B.S. thesis]. Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas …

Nurdin, N., Pettalongi, S. S., Mangasing, M., et al. (2021). Implementation of geographic information system base on google maps api to determine bidikmisi scholarship recipient distribution in central sulawesi indonesia. *Journal of Humanities and Social Sciences Studies*, *3*(12), 38–53.

Obando Suárez, J. P. (2022). *Sistema de recolección de desechos optimizado, mediante iot en el sector el olivo de la ciudad de ibarra provincia de imbabura* [B.S. thesis].

Prudencio Cruz, L. A. (2021). Modelo de sistema de gestión ambiental escolar para prevenir la contaminación en la institución educativa n° 20930 “virgen de la merced” del distrito de sayán-2019.

Ricatti, A., et al. (2017). *Optimización de recolección de residuos con iot* [B.S. thesis].

Sung, W.-T., & Hsiao, S.-J. (2022). Utilizing the improved qpso algorithm to build a wsn monitoring system. *Computers, Materials & Continua*, *70*(2).

Talha, M., Sohail, M., & Hajji, H. (2020). Analysis of research on amazon aws cloud computing seller data security. *International Journal of Research in Engineering Innovation*, *4*(3), 131–136.

Valdivia Castillo, L. O., & Garcia Chirinos, D. A. (2022). Modelo tecnológico para la optimización de rutas de recolección y transporte de residuos sólidos en lima metropolitana.

Velasquez, W., Jijon-Veliz, F., & Alvarez-Alvarado, M. S. (2023). Optimal wireless sensor networks allocation for wooded areas using quantum-behaved swarm optimization algorithms. *IEEE Access*, *11*, 14375–14384.

Wu, Q. (2020). Geemap: A python package for interactive mapping with google earth engine. *Journal of Open Source Software*, *5*(51), 2305.

Zambrano, P. A., & Cobeña, D. C. (2020). Análisis de la tasa de generación de los diferentes tipos de residuos sólidos no peligrosos que se generan en la espol y determinación de nuevos puntos de recolección de desechos no peligrosos.

APÉNDICES

A Manual de usuario de la aplicación: GreenPath Collector

La aplicación GreenPath Collector es una solución innovadora que busca ofrecer un servicio de optimización de rutas de recolección dentro de un área determinada, ya sea para una organización o institución que posean dificultades con la manera de realizar la recolección de los residuos sólidos generados y desechados de forma irresponsable. El sistema permite a los usuarios, en este caso, un auxiliar de servicio o colaborador de limpieza, generar una ruta de recolección optimizada entre los diferentes puntos de recolección de su institución. Además, el sistema ofrece una interfaz de usuario completamente amigable e intuitiva de usar para interactuar con cada una de las pestañas de la aplicación, las cuales proporcionan diferentes funcionalidades a los usuarios que le servirán de ayuda para realizar la recolección de residuos.

El presente apéndice permite conocer el funcionamiento de la aplicación desde el momento en que los usuarios acceden a ella mediante el aplicativo web de su preferencia, hasta la creación de un nuevo contenedor. También, se explica y muestra de manera gráfica cada uno de los pasos que se deben realizar para generar una ruta de recolección optimizada con aquellos contenedores que se encuentren en estado “Lleno” dentro de la aplicación y los diferentes puntos de recolección que se encuentran a lo largo del área. Además, se presentan detalles en cuanto a la animación que se utiliza en el menú de la aplicación.

Enlace de la aplicación: https://swms-pied.vercel.app/

A.1 Pantalla de Inicio

Al ingresar a la aplicación el usuario podrá visualizar una pantalla de inicio acorde a su calidad de usuario registrado o no registrado en el sistema. Para aquellos usuarios registrados se les pedirá que ingresen sus credenciales de acceso, para los usuarios no registrados, pueden optar por escoger la pestaña superior derecha “Create Account” para poder acceder al sistema.

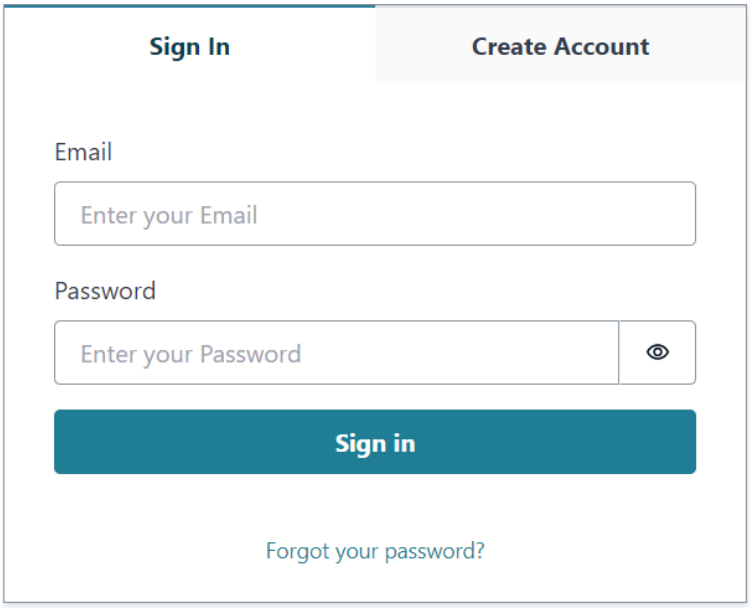

Figura 1: Pantalla inicial al acceder a la aplicación por primera vez, para usuarios registrados

En el primer caso, como se aprecia en la figura 1, la pantalla para loggearse presenta un apartado donde el usuario podrá ingresar su correo electrónico y su contraseña, si las credenciales son las correctas, el usuario podrá ingresar al aplicativo y observar el menú de servicios que ofrece la aplicación.

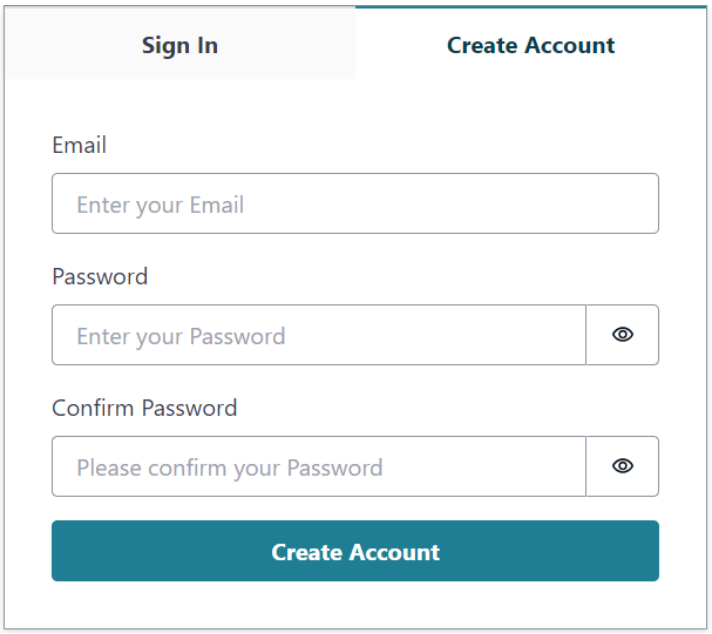

Figura 2: Pantalla inicial al acceder a la aplicación por primera vez, para usuarios no registrados

En caso de no contar con una cuenta, el usuario podrá visualizar varios campos que se le solicitarán para su registro en el sistema tal como lo muestra la figura 2.

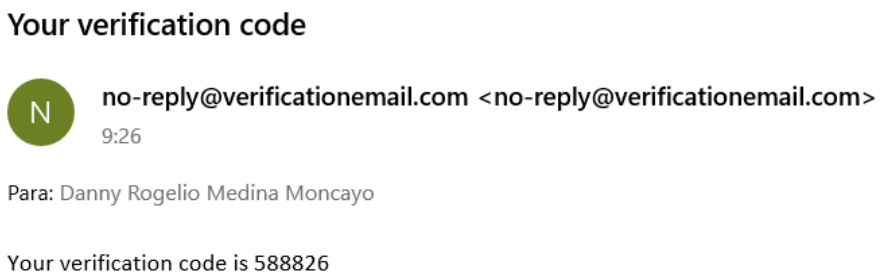

Figura 3: Código de verificación enviado al correo electrónico del usuario para su registro

Como lo muestra la figura 3, una vez registrado, se enviará un mensaje al correo electrónico ingresado en el sistema, proporcionando un código único, el cual debe ser ingresado en el aplicativo para poder validar al usuario registrado.

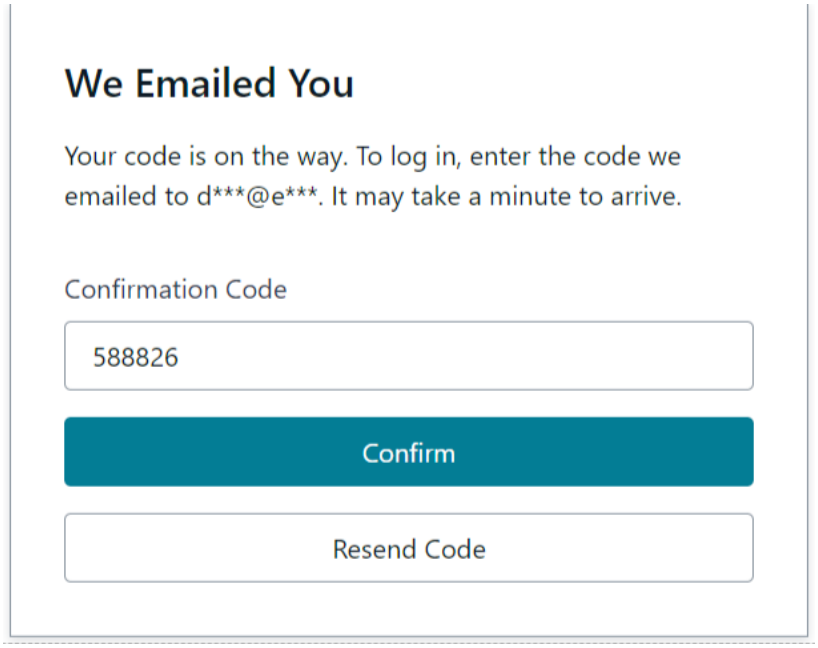

Figura 4: Ingreso del código de verificación del usuario al aplicativo

Una vez realizado estos pasos, el usuario podrá ingresar al sistema con sus credenciales creadas y estará listo para usar el aplicativo.

A.2 Pantalla Principal

Una vez validado las credenciales ingresadas e iniciar sesión, el usuario es redireccionado por el sistema a la pantalla principal de la aplicación, la cual se muestra en la figura 5. Al acceder a esta pantalla se observa una vista general del mapa, proporcionado por el servicio de geocalización usado en el sistema, con los puntos de

recolección ubicados en este. Además, se aprecian las pestañas que intervienen en la aplicación en la parte inferior de la pantalla, allí se muestran las pestañas de: Ruta, Puntos, Personal y Administración. Cada una de estas pestañas cumplen un rol diferente dentro de la aplicación; las pestañas Ruta y Administración permiten ejecutar una acción determinada, mientras que las pestañas restantes sirven para visualizar información relaciona al personal y puntos de recolección registrados en el sistema.

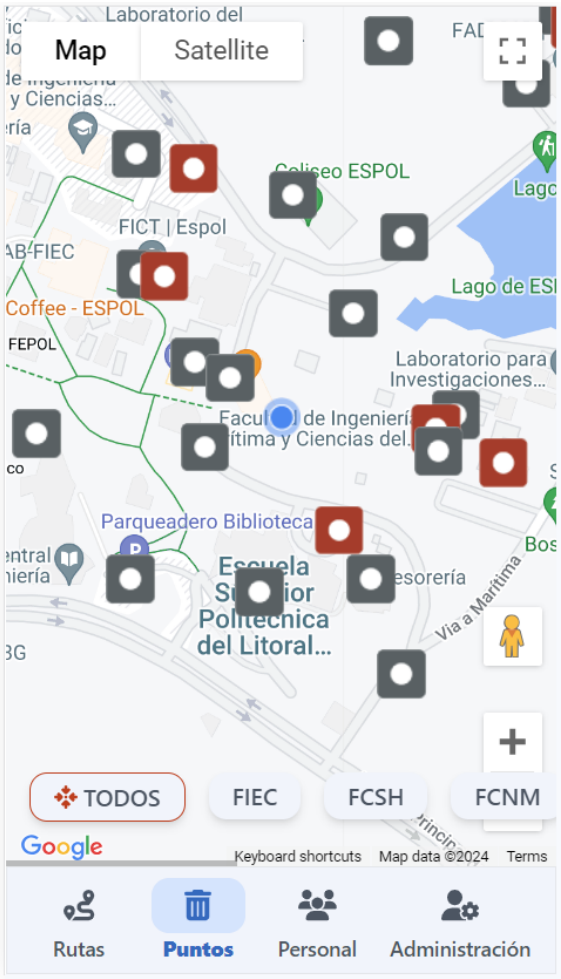

Figura 5: Pantalla principal del inicio

Además, se puede observar que existe una segmentación por áreas y facultades, las cuales se muestran en la parte inferior del mapa, siendo un total de 8 áreas segmentadas dentro del sistema. En este caso, se muestran todas las facultades de ESPOL en conjunto con dos segmentos adicionales que contemplan la totalidad de los puntos de recolección marcados en el mapa y puntos que no pertenecen a una facultad o un área determinada, sin embargo son considerados puntos de recolección.

A.3 Pantalla de la pestaña Personal

La pestaña personal muestra el detalle de los usuarios registrados en el sistema. Al elegir esta opción se muestra, de manera general, la información esencial del personal y la cantidad de contenedores asignados a cada uno de ellos tal como lo detalla la figura 6.

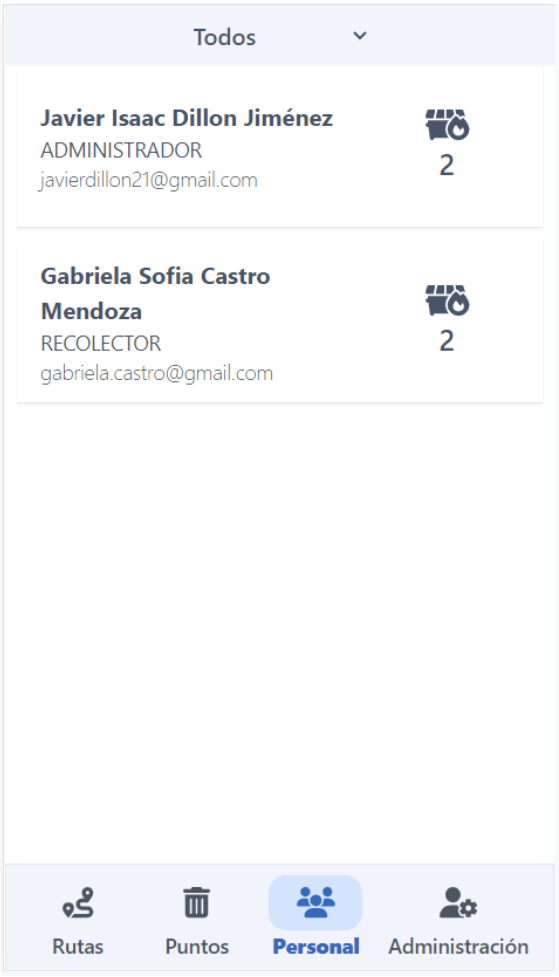

Figura 6: Pantalla principal de la pestaña Personal

El aplicativo cuenta con tres tipos de usuarios: Administrador, Supervisor y Recolector. Cada uno de ellos posee una diferente función y podrá tener acceso a diferentes acciones en el aplicativo, por ejemplo, el usuario administrador podrá designar el tipo de usuario que se registra a la aplicación y asignar áreas y puntos de recolección a otros usuarios, tal como lo muestra la figura 7.

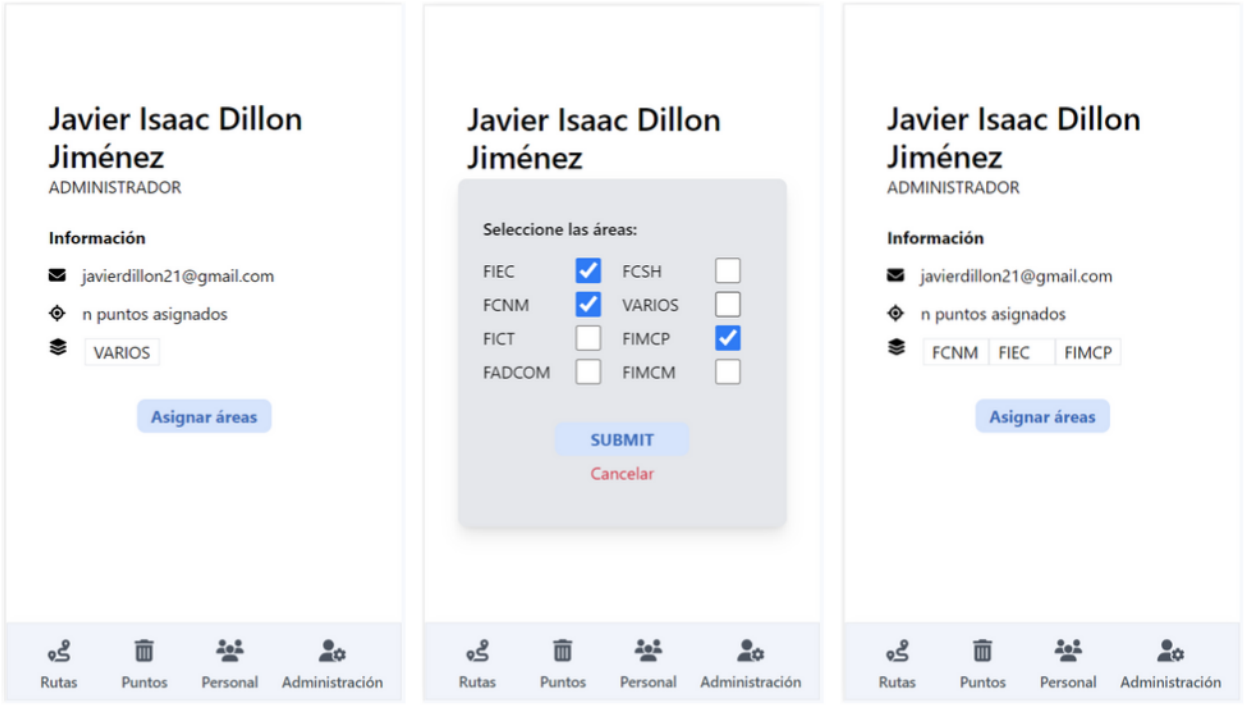

Figura 7: Asignación de áreas del personal

A.4 Pantalla de la pestaña Administración

En esta pestaña, el usuario administrador podrá agregar más contenedores a los ya existentes en el aplicativo, para ello necesita escoger una de las 8 áreas identificadas y ubicadas en la parte inferior del mapa, colocar el marcador de color verde en el punto donde desee colocar el nuevo contenedor y hacer clic en el botón “Añadir”, si todos los pasos indicados se han hecho de forma adecuada, se muestra un mensaje en la parte superior indicando que el contenedor se ha creado de manera exitosa, caso contrario aparecerá un mensaje indicando que no se puede proceder a crear el contenedor, debido a que no se ha escogido el área deseada para su creación, tal como se observa en la figura 8.

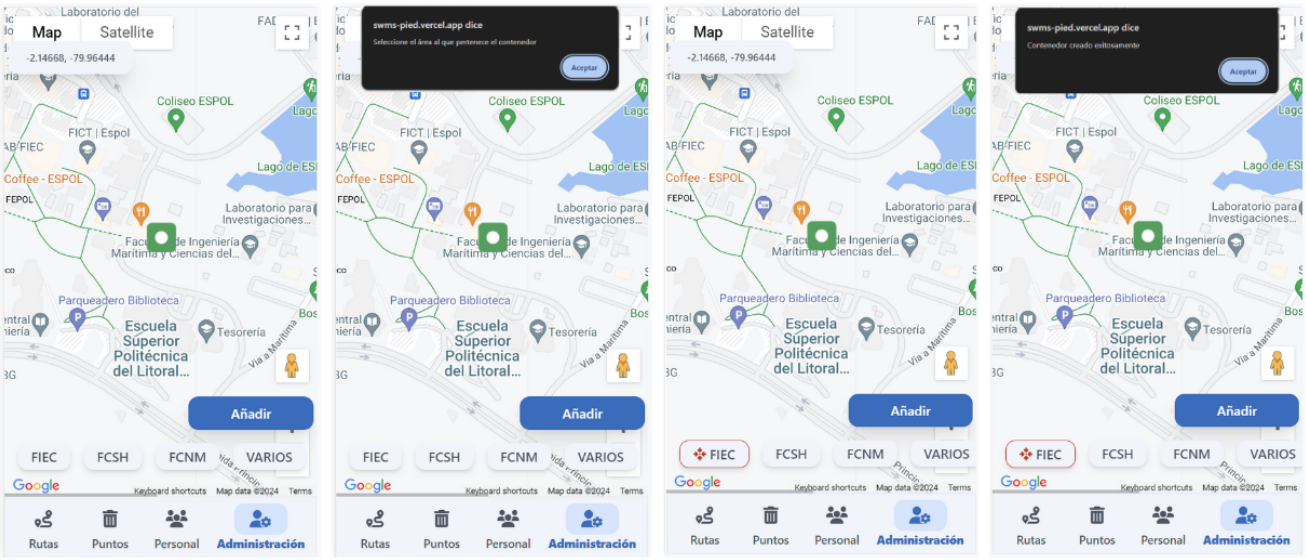

Figura 8: Creación de nuevo contenedor

A.5 Pantalla de la pestaña Puntos

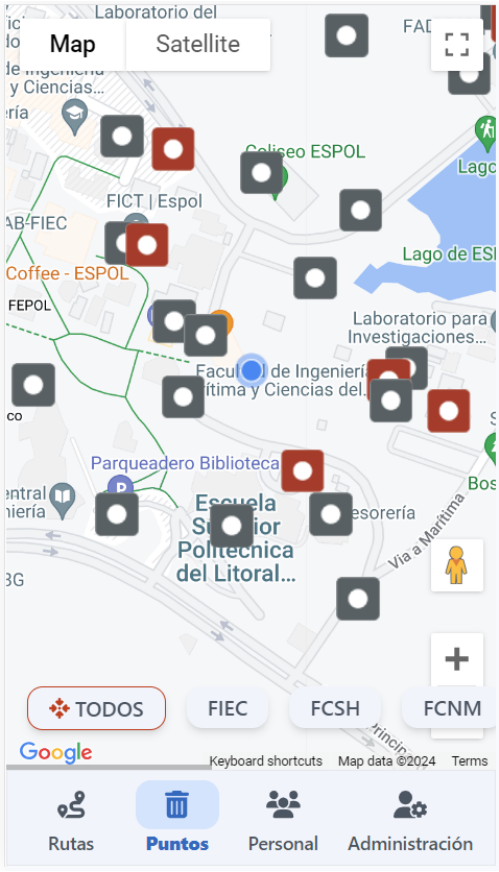

Figura 9: Pantalla principal de la pestaña Puntos

Al seleccionar la pestaña de Puntos, el usuario puede visualizar la distribución de los

diferentes puntos de recolección identificados a lo largo del campus, estos puntos de recolección estarán representados por un indicador de color rojo y gris para diferenciar a los contenedores con estado “Lleno” y “Vacío” respectivamente, como lo muestra la figura 9.

Además, el usuario tiene la posibilidad de cambiar el estado de los contenedores manteniendo aplastado el indicador en el mapa, en caso de revertir esta acción solo debe volver a mantener aplastado el indicador y este volverá a su estado anterior, tal como lo muestra la figura 10.

Figura 10: Visualización de cambio de estado de los puntos de recolección

En la figura 11 se muestra la manera en que el usuario selecciona un área diferente y puede observar los puntos asociados a esa área determinada. Además, el usuario tiene la posibilidad de ubicar el marcador de color azul, el cual representa su ubicación actual dentro el mapa, en cualquier posición cercana a los puntos de recolección para comodidad de este.

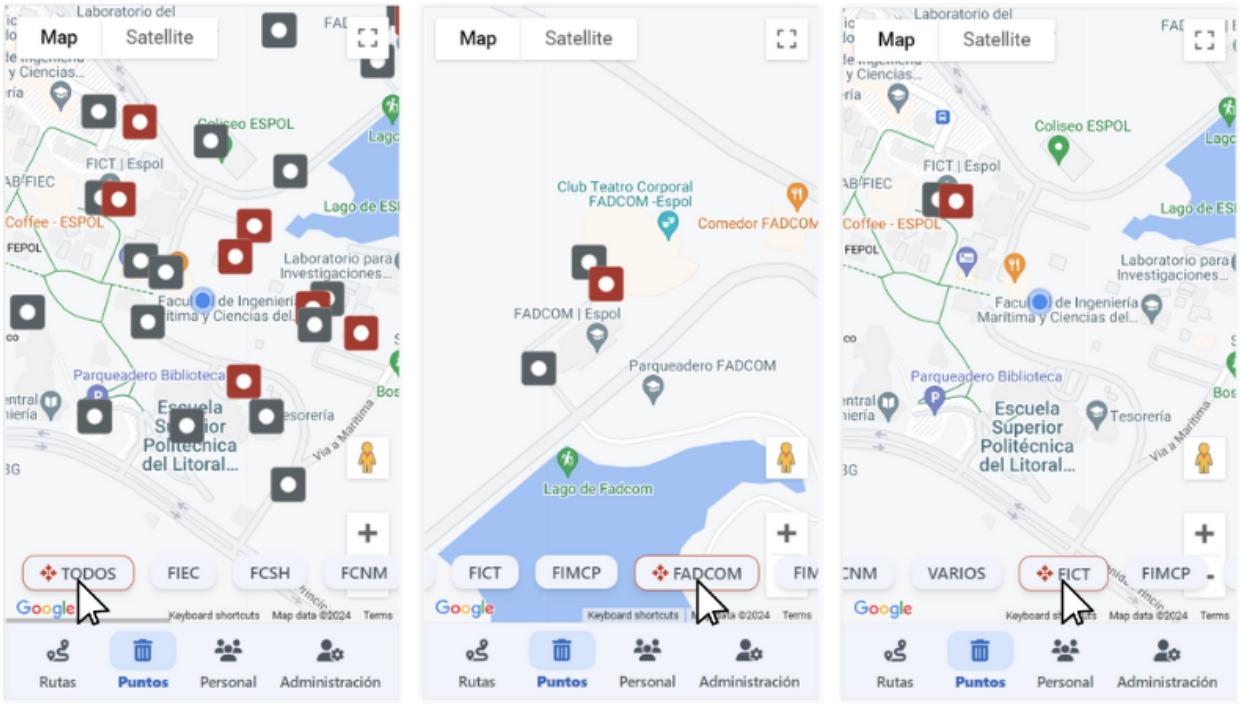

Figura 11: Visualización de los puntos por área escogida

A.6 Pantalla de la pestaña Rutas

En esta pestaña, el usuario puede generar la ruta de recolección entre todos los puntos que se encuentren con un indicador de color rojo dentro del mapa, incluso si el usuario desea agregar un nuevo contenedor y actualizar su estado a “Lleno”, tal como se lo detalla en las secciones anteriores, este nuevo contendor será incluido en la nueva ruta a generar, este se debe a que el aplicativo funciona en tiempo real. Para generar la ruta se debe hacer clic en el botón del lado inferior derecho de color azul, como se detalla en la figura 12.

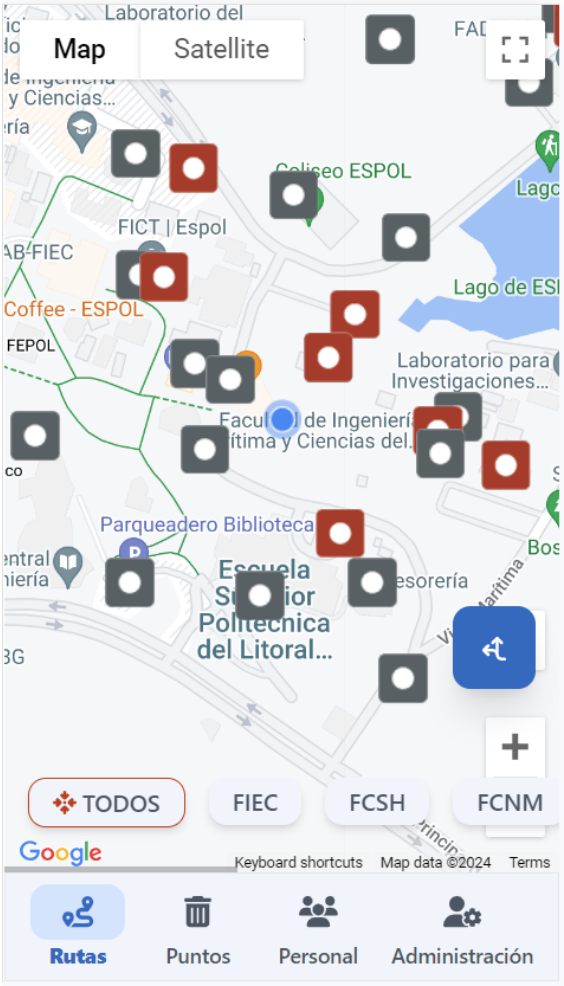

Figura 12: Pantalla principal de la pestaña Rutas

Acto seguido, se genera la ruta de recolección optimizada entre todos los contenedores. El aplicativo utiliza un sistema basado en el orden de las letras del alfabeto para mostrar de manera visual cuál contenedor se debe recolectar primero. En la figura 13 se puede apreciar que estas letras están ubicadas dentro del indicador de color rojo

ubicado en cada una de las posiciones donde se encuentran los contenedores.

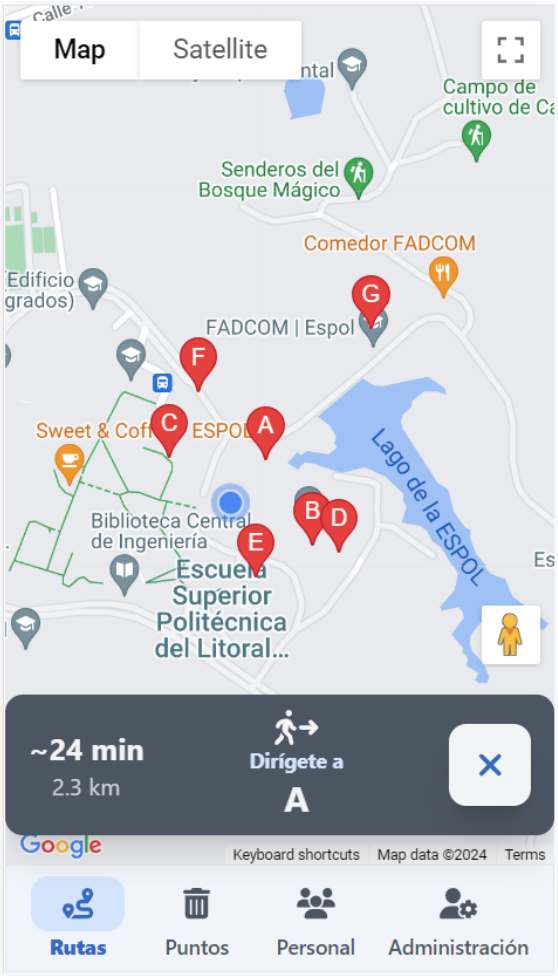

Figura 13: Generación de la ruta de recolección optimizada

A.7 Recomendaciones e Indicaciones Generales

- El aplicativo se lo puede ejecutar desde cualquier dispositivo móvil o de escritorio, ya que este se ejecuta en el navegador web, tales como: Google Chrome, Opera, Firefox, entre otros. Por lo que no es necesario tomar en cuanta qué sistema operativo posee el dispositivo en el que se vaya a ingresar al sistema.

- La ruta de recolección es generada tomando en cuenta un radio de 15 metros de proximidad entre cada punto, por lo que, en caso de crear un nuevo contenedor, se recomienda crearlo en un radio mayor a 15 metros para asegurar que el sistema lo tome en consideración al momento de generar la ruta.

- En caso de haber generado la ruta de recolección y desea volver a generarla una

vez más de manera consecutiva, se deber esperar un tiempo de 30 segundos para volver a generar la ruta. Esto se debe a que sistema está programado para no exceder el consumo del número de consultas en la API de Directions, proporcionada por Google Maps Platform. En caso de requerir aumentar el número de consultas, se debe contactar con los administradores del sistema para realizar los cambios respectivos.