Sistema de información basado en IoT para el análisis de la calidad del aire en rutas del transporte público del Área Metropolitana de Bucaramanga

Diego Andrés Ortega Gelvez y Jose Fredy Navarro Motta

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director
Gabriel Rodrigo Pedraza Ferreira
Doctorado en Ciencias de la Computación

Codirector
Duván Yahir Sanabria Echeverría
Magíster en Ingeniería de Sistemas e Informática

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingeniería en Sistemas e Informática
Bucaramanga
2026

Dedicatoria

Este trabajo de grado se dedica a nuestros padres, Jose Fredy Navarro Penaranda, Johanny Motta Ortega y Betsabe Gelvez Hernández, por su apoyo, ánimo y valores que nos han proporcionado a lo largo de nuestras vidas y que nos inspiran para seguir adelante y crecer.

Agradecimientos

Agradecemos profundamente a nuestros padres Jose Fredy Navarro Penaranda, Johanny Motta Ortega y Betsabe Gelvez Hernández, que lo han dado todo para que nosotros podamos estar aquí y que con su sabiduría y valores nos han formado en las personas que ahora somos y estamos orgullosos de ellos.

A nuestro director de tesis Gabriel Rodrigo Pedraza Ferreira y co-director Duvan Yahir Sanabria Echeverría por todo el apoyo que nos proporcionaron durante todo el proceso de creación, formalización y finalización del proyecto, así como la entrega del dispositivo y los sensores que permitieron el desarrollo del prototipo.

A nuestras familias que durante toda nuestra vida han compartido con nosotros y con las cuales hemos vivido experiencias y aprendido enseñanzas para nuestro futuro.

A nuestras parejas, en especial a Tatiana Ávila y a Chloé Catelain, por su amor, comprensión y apoyo incondicional durante el desarrollo de esta tesis. Su paciencia en los momentos difíciles y sus palabras de aliento fueron fundamentales para mantener nuestra motivación y culminar con éxito este proyecto.

A todos los profesores que con su pasión por enseñar han inspirado nuestra formación como ingenieros de sistemas.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Glosario

AMB: Área metropolitana de Bucaramanga.

Arduino: Placa electrónica programable que permite la configuración de microcontroladores de forma sencilla.

Arquitectura software: Estructura organizada de un sistema que define componentes, relaciones, y demás interacciones con el fin de cumplir con los requerimientos software.

Broker: Es un intermediario que facilita la comunicación entre dispositivos y aplicaciones mediante un protocolo de mensajería como MQTT.

Calidad del aire: Se refiere al estado de contaminación en el aire que respiramos.

CDMB: Corporación autónoma regional para la defensa de la meseta de Bucaramanga.

Contaminantes criterio: Son aquellos contaminantes en el aire que según su concentración pueden generar efectos nocivos en la salud.

Dashboard: Herramienta destinada a la visualización y organización de datos para el análisis y la toma de decisiones.

Geolocalización: Se refiere a la capacidad de asociar una medida o un dato a un determinado lugar en el espacio.

IoT: Internet de las cosas.

Microservicio: Servicio independiente que cumple con una única función y comúnmente se integra con otros servicios asegurando independencia entre los mismos.

MQTT: Mosquitto, protocolo de publicación / suscripción, máquina a máquina para el envío de datos que corre por encima de TCP/IP.

Plataforma: Conjunto de software y hardware que proporcionan el entorno necesario para ejecutar aplicaciones.

Requerimientos funcionales: Especificaciones que determinan las funciones que el sistema debe cumplir con el fin de cumplir los objetivos del usuario.

Requerimientos no funcionales: Especificaciones que definen aspectos no funcionales como el rendimiento, la disponibilidad, la usabilidad y la escalabilidad.

Sensor: Dispositivo transductor con la capacidad de obtener y transformar señales del entorno.

Sistema de información: Conjunto de componentes que recopilan, procesan y distribuyen información para apoyar la toma de decisiones y mejorar la eficiencia organizacional

Smart Campus UIS: Macroproyecto enfocado a la gestión de aplicaciones y dispositivos a través de una plataforma basada en microservicios y IoT

Transporte público: Medio de transporte sustentado por medios públicos para la movilidad en el AMB.

Resumen

Título: Sistema de información basado en IoT para el análisis de la calidad del aire en rutas del transporte público del Área Metropolitana de Bucaramanga.*

Autor: Diego Andrés Ortega Gelvez, Jose Fredy Navarro Motta**

Palabras Clave: IoT, Sistema de información, Contaminantes Criterio, Dashboard

Descripción: Los contaminantes criterio son sustancias consideradas nocivas para la salud humana y el medio ambiente. Su monitoreo permite evaluar la calidad del aire y establecer límites de exposición seguros para la población. Sin embargo, en el área metropolitana de Bucaramanga, el monitoreo de la calidad del aire depende principalmente de estaciones fijas que no siempre reflejan las condiciones reales de exposición de los ciudadanos, especialmente en entornos de movilidad como el transporte público. Esta limitación reduce la comprensión de los indicadores de calidad del aire a lo largo de las rutas, lo que dificulta la toma de decisiones informadas para la gestión ambiental y la protección de la salud pública. Por ello, este trabajo propone el diseño e implementación de un prototipo de sistema de información basado en Internet de las Cosas (IoT) para el monitoreo de la calidad del aire en rutas de transporte público. La solución se integra con Smart Campus UIS, un macroproyecto escalable basado en microservicios, que permite recopilar datos provenientes de sensores de PM2.5, CO_2, temperatura y geolocalización. A partir de estos datos, se desarrolló un dashboard interactivo que visualiza dinámicamente mapas de rutas codificados por colores, gráficos comparativos e indicadores estadísticos, facilitando la identificación condiciones específicas y zonas localizadas de alta exposición en las rutas de transporte público, ofreciendo un análisis mucho más preciso de la calidad del aire que experimentan realmente los ciudadanos durante sus desplazamientos diarios.

---

* Trabajo de Grado

** Facultad de Fisicomécanica. Escuela de Ingeniería de Sistemas e Informática. Director: Gabriel Rodrigo Pedraza Ferreira. Doctor en Ciencias de la Computación. Codirector: Duván Yahir Sanabria Echeverrí. Magíster en Ingeniería de Sistemas e Informática

Abstract

Title: IoT-based information system for analyzing air quality on public transport routes in the Bucaramanga Metropolitan Area[*]

Author: Diego Andrés Ortega Gelvez, Jose Fredy Navarro Motta[**]

Key Words: IoT, information system, Criteria pollutants, Dashboard

Description: Criteria pollutants are substances that are harmful to human health and the environment. Monitoring them is essential for assessing air quality and establishing safe exposure limits. In the Bucaramanga Metropolitan Area, monitoring depends on fixed stations that fail to capture the actual exposure conditions of citizens, particularly in mobile environments such as public transportation. These confined spaces, exposed to direct emissions and traffic congestion, present conditions where air quality differs significantly from external measurements, creating potential risks for users. This could conceal dangerous exposure levels for users. This limitation reduces the understanding of air quality indicators along urban routes, making it difficult to make informed decisions for environmental management and public health protection.

Therefore, this work proposes the design and implementation of a prototype information system based on the Internet of Things (IoT) for monitoring air quality directly on public transportation routes. The solution is integrated with Smart Campus UIS, a scalable macro-project based on a microservices architecture, which allows real-time spatial and temporal data to be seamlessly collected from portable PM2.5, CO2, temperature, and geolocation sensors. Based on this data, an interactive web dashboard is developed that facilitates the dynamic visualization of color-coded maps, comparative graphs, and statistical indicators. Ultimately, this system serves to identify specific conditions and localized zones of high exposure on public transportation routes, offering a much more accurate reflection of the air quality citizens experience during their daily commute.

---

[*] Degree Work

[**] Faculty of Physico Mechanical Engineering. School of Systems and Computer Engineering. Advisor: Gabriel Rodrigo Pedraza Ferreira. Doctor in Computer Science. Co-advisor: Duván Yahir Sanabria Echeverrí. Master of Science in Systems Engineering and Computer Science.

Introducción

La calidad del aire es un tema de creciente preocupación en muchas ciudades, donde la contaminación atmosférica representa una problemática seria para la salud pública. De acuerdo con la Organización Mundial de la Salud (OMS), los contaminantes criterio como el monóxido de carbono (CO), el dióxido de nitrógeno (NO2), y el material particulado (PM10 y PM2.5) son particularmente nocivos debido a su relación con enfermedades respiratorias y cardiovasculares. En el Área Metropolitana de Bucaramanga (AMB), la Corporación Autónoma Regional para la Defensa de la Meseta de Bucaramanga (CDMB) monitorea la calidad del aire en puntos estratégicos mediante estaciones de vigilancia. Sin embargo, estos puntos de monitoreo no cubren áreas móviles, como las rutas del transporte público, que son esenciales para evaluar la exposición de los usuarios a estos contaminantes.

Ante esta necesidad, este proyecto busca implementar un sistema de información basado en el Internet de las Cosas (IoT) para medir y geolocalizar contaminantes criterio a lo largo de las rutas de transporte público en el AMB. El uso de IoT permitirá la recolección de datos en tiempo real sobre la calidad del aire en el interior de los vehículos, generando información clave para identificar zonas de alta contaminación y mejorar las estrategias de gestión ambiental. La plataforma Smart Campus UIS apoyará este sistema mediante la integración y procesamiento de datos, facilitando el análisis de patrones de contaminación y su relación con la movilidad urbana.

1. Planteamiento y Justificación Del Problema

Según la Organización Mundial de la Salud (OMS), la contaminación del aire es el resultado de una combinación de partículas sólidas, líquidas y gases provenientes de diversas fuentes, como la quema de combustibles, los desechos industriales, las emisiones vehiculares, entre otras. Entre estos contaminantes, algunos se han clasificado como "contaminantes criterio" por ser perjudiciales para la salud, como el monóxido de carbono (CO), el dióxido de Azufre (SO2), el dióxido de nitrógeno (NO2) y el material particulado inferior a 10 micras (PM 10) y 2.5 micras (PM2.5). Para estos compuestos se han establecido niveles de concentración admisibles con el objetivo de proteger la salud de la población humana.

En el Área metropolitana de Bucaramanga (AMB), la corporación autónoma regional para la defensa de la meseta de Bucaramanga (CDMB) se encarga de monitorear la calidad del aire a través de un sistema de vigilancia conformado por 3 estaciones con la capacidad de monitorear y enviar datos hora a hora. La CDMB afirma que una fuente principal de contaminación es el incremento notable del mercado automotriz generando grandes cantidades de contaminación en el aire.

Aunque las estaciones de la CDMB recopilan información en puntos estratégicos del AMB, se hace relevante considerar y analizar la calidad del aire en espacios reducidos y expuestos a constantes fuentes de contaminación, como el interior de vehículos de transporte público, en especial los autobuses, donde otros parámetros también pueden influir en la medida de calidad del aire como la congestión vehicular y la afluencia de usuarios. Estas condiciones pueden diferir en los datos de calidad de aire tomados en el exterior, lo que podría llegar a omitir

escenarios de alta exposición a los usuarios de transporte público potencialmente peligrosos para la salud.

Smart Campus UIS, es un macroproyecto enfocado a la gestión de aplicaciones y dispositivos a través de una plataforma basada en microservicios y IoT (Internet de las Cosas). Este macroproyecto se ha ido desarrollando en la Universidad a través de la realización de diversos proyectos de grado previos, por lo cual, se cuenta con una base e infraestructura sólida para la implementación de proyectos enfocados a distintas problemáticas.

Uno de los casos de aplicación considerado dentro del Smart Campus UIS es la gestión ambiental y particularmente el tema de calidad del aire, dentro de este contexto previamente se desarrolló el trabajo de grado "Plataforma IoT para medición de la calidad del aire y seguimiento de rutas en sistemas de transporte público." El enfoque principal del trabajo fue el dispositivo para la recolección de datos en tiempo real limitado a solo el dióxido de carbono como indicador de la calidad del aire.

Considerando lo anterior, con este proyecto se propone enfocar el alcance al análisis de calidad del aire en el interior de autobuses de rutas de transporte público del AMB por medio de un sistema de información basado en Smart Campus UIS, que permita recopilar con un dispositivo información espacial y temporal de contaminantes criterio para identificar condiciones de alta exposición a través de un aplicativo web que permita a los usuarios el análisis de los datos para la toma de decisiones.

2. Objetivos

2.1 Objetivo General

Desarrollar un sistema de información basado en IoT para la medición y geolocalización de contaminantes criterio para el análisis de la calidad del aire en rutas del transporte público del Área Metropolitana de Bucaramanga y la identificación de condiciones de alta exposición.

2.2 Objetivos Específicos

Definir requerimientos del sistema de información basado en IoT para el análisis de calidad del aire basado en la medición de contaminantes criterio en rutas de transporte público del AMB.

Diseñar una arquitectura basada en la plataforma de microservicios Smart Campus UIS que permita la obtención de datos de contaminantes criterio.

Implementar el prototipo del sistema de información basado en la arquitectura propuesta.

Ejecutar pruebas del funcionamiento de los componentes del sistema de información, recolectando, analizando y visualizando datos de las rutas de transporte público en el AMB con el fin de identificar condiciones de alta exposición.

3. Marco de Referencia

Esta sección tiene como objetivo explicar y relacionar los conceptos teóricos fundamentales para la ejecución del proyecto. Se incluyen conceptos clave como la composición y calidad del aire, los contaminantes criterio, los efectos en la salud humana y la importancia de su medición. Para la implementación del proyecto, se describen conceptos como los dispositivos y sensores para la recolección y transmisión de datos, el Internet de las Cosas (IoT), la

plataforma Smart Campus UIS y tecnologías para la visualización de información a través de un dashboard.

3.1 Marco Conceptual

3.1.1 Composición del Aire

Según el Instituto de Hidrología, Meteorología y Estudios Ambientales (IDEAM, s. f.), la atmósfera se compone de una mezcla de gases y aerosoles. Estos componentes de pueden dividir en dos grupos:

***Gases Constantes*:** Son componentes que mantiene una proporción casi permanente en la atmósfera y se encuentran en mayor proporción, como el nitrógeno con un 78,08%, el oxígeno con 20,95% y el argón con 0,93%, entre otros componentes.

Gases Variables: Son componentes que sufren cambios considerables en su proporción. En este grupo se encuentra el dióxido de carbono, con concentraciones relativamente altas del 0,035%, pero con concentraciones variables en el tiempo y espacio.

3.1.2 Contaminación del Aire

El IDEAM (IDEAM, s. f.-b) se refiere a la contaminación en el aire como la presencia de pequeñas partículas o productos secundarios gaseosos que implican el riesgo, daño o molestia a la vida humana, animales y plantas que se exponen a dichos compuestos.

Las principales fuentes de contaminación provienen de múltiples fuentes, incluidas las emisiones causadas por el ser humano y sucesos naturales, por ejemplo, el uso de combustibles fósiles, actividades agrícolas o incluso por la liberación de partículas por volcanes activos.

3.1.3 Calidad del Aire

Según el Programa de las Naciones Unidas para el Medio Ambiente (PMUNA) para determinar la calidad del aire es clave llevar a cabo la medición de los contaminantes atmosféricos.

La medida aceptada internacionalmente es conocida cómo el Índice de Calidad del Aire (ICA), a mayor densidad de contaminantes, mayor será este índice, que se maneja a partir de una escala con un rango de 0 a 500, entre 50 o menos se considera seguro y por encima de 100 se considera poco saludable.

3.1.4 Contaminantes Criterio

Los contaminantes criterio hacen referencia a aquellos contaminantes en el aire que según su concentración pueden generar efectos nocivos en la salud, estos incluyen principalmente el material particulado, el dióxido de Azufre, Ozono, el dióxido de nitrógeno y monóxido de carbono.

Material Particulado PM2.5 Y PM10. El material particulado se refiere a las partículas inhalables compuestas de sulfato, nitratos, amoniaco, cloruro de sodio, carbono, polvo mineral o agua. El PM puede ser de distintos tamaños y en general es categorizado por su diámetro aerodinámico, siendo el inferior a 2.5 micrómetros e inferior a 10 micrómetros los más relevantes para la salud.

Dióxido de Azufre So2. El dióxido de azufre SO2 es un gas incoloro que se disuelve en agua. Se produce principalmente en procesos de combustión de combustibles fósiles como industrias o generación de energía. La exposición a este contaminante puede resultar en casos de asma.

Dióxido de Nitrógeno No2. El dióxido de nitrógeno NO2 es un gas de color marrón rojizo, es soluble en agua y un fuerte oxidante. Proviene de fuentes ambientales como la combustión de combustible a alta temperatura como los procesos usados en la calefacción, transporte, industrial, calentadores, estufas y hornos de gas. La exposición al NO2 puede irritar las vías respiratorias y es un contaminante estrechamente relacionado con el asma.

Ozono O3. El Ozono a nivel del suelo es el componente principal del Smog. Está formado a partir de reacciones fotoquímicas con contaminantes como el monóxido de carbono y óxidos de nitrógeno emitidos por vehículos e industrias.

La exposición a altos niveles de ozono puede causar problemas respiratorios, ataques de asma, reducir la función pulmonar y llevar a enfermedades respiratorias.

Monóxido de Carbono CO. El monóxido de carbono es un gas incoloro e inoloro el cual se produce a partir de la combustión incompleta de combustibles de base carbónica como la madera, el petróleo, el carbón, el gas natural y el queroseno. La principal fuente de CO en el ambiente es por motores de vehículos de transporte.

El CO se puede difundir a través de tejidos pulmonares y entrar al torrente sanguíneo, lo que dificulta que las células del cuerpo consigan oxígeno. La exposición prolongada al monóxido de carbono puede causar dificultades para respirar, agotamiento, mareos y otros síntomas. La exposición a altos niveles de CO se considera mortal para el ser humano.

A continuación, en la tabla 1, se presenta la información de la resolución 2254 de 2017 del Ministerio del Ambiente referente a los niveles máximos permisibles de los contaminantes criterios en el aire.

Tabla 1

Niveles máximos permisibles de contaminantes criterio en el aire

Contaminante	Nivel máximo Permisible (µg/m^3)	Tiempo de exposición
PM_{10}	50	Anual
	100	24 horas
$PM_{2.5}$	25	Anual
	50	24 horas
SO_2	50	24 horas
	100	1 hora
NO_2	60	Anual
	200	1 hora
O_3	100	8 horas
CO	5000	8 horas
	35000	1 hora

Nota. Adaptado del Ministerio del Ambiente (2017). Resolución 2254 de 2017.

Tabla 2

Clasificación de indicador de calidad del aire basado en concentración de material particulado

AQI Category and Index Value	Previous AQI Category Breakpoints	Updated AQI Category Breakpoints	What changed?
Good (0 – 50)	0.0 to 12.0	0.0 to 9.0	EPA updated the breakpoint between Good and Moderate to reflect the updated annual standard of 9 micrograms per cubic meter
Moderate (51 – 100)	12.1 to 35.4	9.1 to 35.4	
Unhealthy for Sensitive Groups (101 – 150)	35.5 to 55.4	35.5 to 55.4	No change, because EPA retained the 24-hour fine PM standard of 35 micrograms per cubic meter.
Unhealthy (151 – 200)	55.5 to 150.4	55.5 to 125.4	EPA updated the breakpoints at the upper end of the unhealthy, very unhealthy, and hazardous categories based on scientific evidence about particle pollution and health. The Agency also combined two sets of breakpoints for the Hazardous category into one.
Very Unhealthy (201 – 300)	150.5 to 250.4	125.5 to 225.4	
Hazardous (301+)	250.5 to 350.4 and 350.5 to 500	225.5+	

Nota. Adaptado del U.S. Environmental Protection Agency. (2024). Final updates to the Air Quality Index (AQI) for particulate matter

3.1.5 Sensores

Los sensores se utilizan en un amplio espectro de transducción y transformación de señales, con diferencias en la complejidad técnica. El uso de sensores puede ir desde la medición de temperatura hasta la detección de especies de bacterias mediante sistemas ópticos sofisticados. La proliferación de aplicaciones basadas en sensores está creciendo en una variedad de objetos de detección como el agua, bacterias, movimiento y aire. Así como en cualquier tipo de tecnología, los sensores tienen sus ventajas y desventajas. El rendimiento y calidad de los datos de un sensor puede depender del método de transducción, el entorno de implementación o los componentes del sistema. (McGrath & Scanaill, 2013).

Sensores de Bajo Costo. En el mercado actual han surgido diversos sensores de bajo costo que facilitan realizar monitoreo a condiciones ambientales como la calidad del aire, registrando datos con una frecuencia de segundos permitiendo conocer cambios en el ambiente en el transcurso del día. Los sensores de bajo costo tienen la capacidad de detectar y medir contaminantes gaseosos y particulados en el aire, con la opción de poder monitorear el aire en interiores y exteriores (Buitrago Mesa & Rodriguez Rodriguez, 2022, p.22). Según Buitrago Mesa & Rodriguez Rodriguez (2022, p.23) el término de "bajo costo" en los sensores hace referencia a los dispositivos de valor inferior a diez millones de pesos. Aunque el término puede variar según el autor o el propósito.

3.1.6 Internet de las Cosas IoT

El internet de las cosas hace referencia a una tecnología basada en la conexión de objetos comunes del día a día con la internet, que agregan o procesan información del entorno real, así como comunicarse entre sí para prestar servicios de valor agregado a los usuarios finales. Capaces de reconocer cambios o eventos y reaccionar de manera autónoma y apropiada. Su

finalidad es brindar una infraestructura que permita superar la barrera entre los objetos en el mundo físico y su representación en los sistemas de información. (Moíses, 2020, p. 21-22).

Ahora bien, este proyecto involucra la interacción con el mundo físico y la toma de datos con los sensores previamente indicados, es de gran importancia diseñar un sistema que sea adecuado a esta tecnología. Además, es de alta importancia considerar el tamaño del dispositivo, el protocolo de conexión con sensores, la conectividad y las capacidades tecnológicas que el sistema requiera dentro del diseño.

3.1.7 Arduino

Es una empresa de tecnología avanzada que se encarga de diseñar dispositivos electrónicos y software de fácil acceso que permiten interactuar con el entorno físico. Dentro de sus ofertas, se incluyen diversos microcontroladores diseñados para IoT con capacidades WIFI y Bluetooth conocidos cómo SoC (System On a Chip).

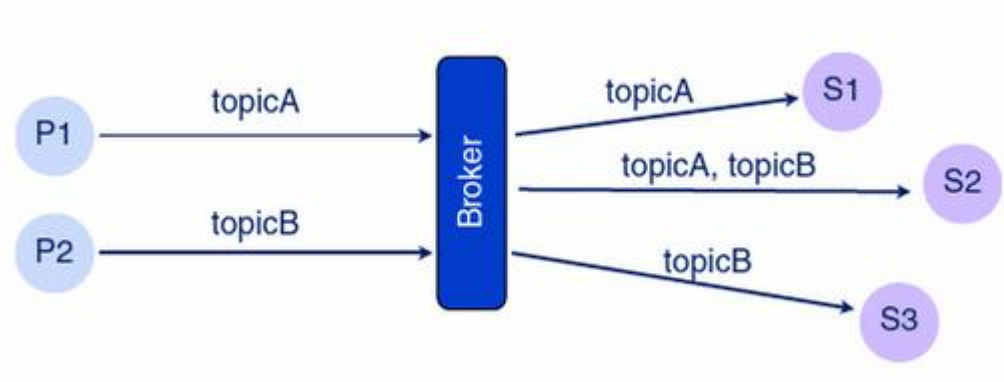
El chip que se elija es la base del proyecto para permitir la conexión y compatibilidad con los sensores que van a tomar los datos para el desarrollo del sistema IoT.

3.1.8 Modelo de Comunicación Publicación-Suscripción

Es uno de los modelos de comunicación más utilizado para sistemas IoT, en estas dos entidades existen, los publicadores que envían mensajes a un tópico del servidor y los suscriptores que se suscriben al tópico para recibir una copia de todos los mensajes que se publican con ese tópico, permitiendo que un mensaje pueda ser consumido por múltiples consumidores. Para esta comunicación se usan unos intermediarios conocidos cómo brokers que se encargan de despachar los mensajes de los publicadores a las entidades que están suscritas. En la figura 1, se puede apreciar un diagrama del funcionamiento de este modelo. (Cirani et al, 2019, p. 99).

Figura 1

Modelo de comunicación publicación-suscripción



Nota. Adaptado de Cirani, S., Ferrari, G., Picone, M., & Veltri, L. (2019). Figura 3.12, p. 99.

3.1.9. MQTT

Una de las implementaciones del modelo anterior más populares para el internet de las cosas, es MQTT o Mosquitto, un protocolo Machine-to-Machine (M2M) ligero que corre por encima del Protocolo de control de transmisión/Protocolo de Internet (TCP/IP). Consume menos recursos que el protocolo HTTP 1.1 y, por lo tanto, es bastante útil para enviar y recibir datos aproximadamente en tiempo real y con la menor huella posible. (Gastón, 2017, p. 8-9)

3.1.10 Smart Campus UIS

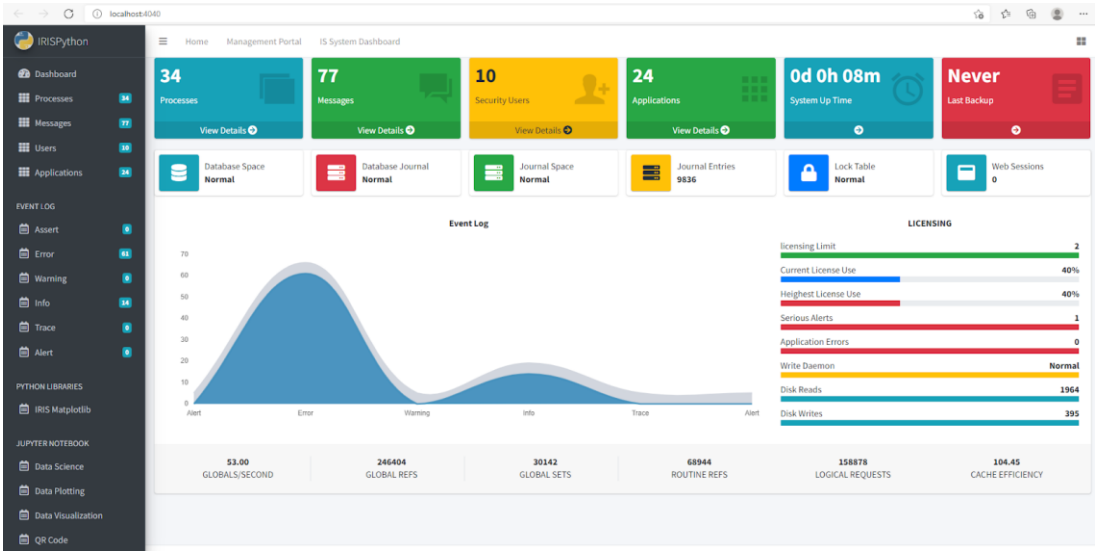
En el ámbito de los campus inteligentes, gran parte de las investigaciones se enfocan en resolver problemas específicos como el monitoreo ambiental o el control de parqueaderos, sin ofrecer una solución integral que permita la interoperabilidad y escalabilidad de las distintas tecnologías IoT. Frente a este vacío, Jiménez, Cárcamo y Pedraza (2020) proponen una plataforma software extensible basada en microservicios, Smart Campus UIS, con propiedades de autonomía e integración, compuesta de sensores, actuadores, gateways y aplicaciones en una arquitectura capaz de soportar múltiples contextos dentro del campus.

3.1.11 Dashboard

Un dashboard es una herramienta en forma de panel de control para visualizar datos reales de manera clara, organizada e interactiva en tiempo real permitiendo con un solo vistazo ver la información más relevante y apoyando la toma de decisiones. Existen diferentes maneras de desarrollar un dashboard y su conexión con los microservicios, las más populares y mejores documentadas son las que se desarrollan en Python, incluyendo los frameworks como Flask y Streamlit, en la figura 2 se muestra un ejemplo de un dashboard desarrollado con el primero.

Figura 2

Diseño de dashboard desarrollado con Python



Nota. Adaptado de InterSystems (2022) Building IRIS Responsive dashboard with Python Flask Web Framework.

3.2 Antecedentes del Tema

A través del macroproyecto Smart Campus UIS, se han desarrollado proyectos basados en la medición de condiciones ambientales como la calidad del aire. En un proyecto previo desarrollado por Sebastián y Pablo (2023), titulado “Plataforma IoT para medición de la calidad del aire y seguimiento de rutas en sistemas de transporte público”, se realizó la medición

a través de la instalación del dispositivo en un vehículo particular de parámetros como la temperatura, ubicación y medición del CO_2.

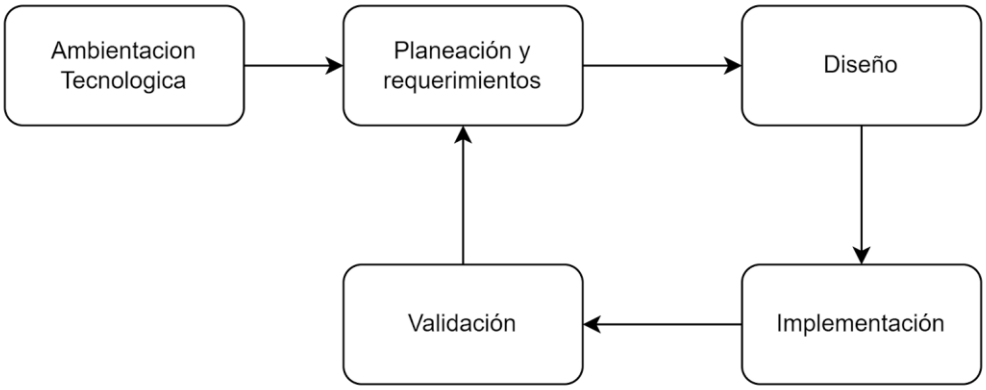
El trabajo anterior se enfocó principalmente en la implementación del dispositivo, el cual se limitó por cuestiones económicas únicamente a medir el dióxido de carbono a través de la instalación en un vehículo particular. Por otro lado, la visualización y análisis de los datos se limitó a mostrar los datos en un mapa de calor con todos los datos recopilados. Por tanto, el proyecto actual busca darle un mayor valor a la información recopilada de contaminantes adicionales a través de un procesamiento y consumo de datos más robusto.

4. Metodología

Para la ejecución del proyecto se planea usar una metodología de investigación basada en cinco fases, en el siguiente diagrama, se ilustra el flujo general y posteriormente se explica cada fase y las actividades que se realizarán en cada una.

Figura 3

Diagrama de metodología por fases del proyecto



Nota. Diagrama denotando el orden de las fases de la metodología que se va a llevar a cabo para la ejecución del proyecto.

4.1 Fase de Ambientación Tecnológica

Es la fase inicial, en la que se realiza una exploración preliminar a las diferentes tecnologías que se van a emplear en el proyecto.

Actividades:

1. Explorar la conectividad y funcionamiento del dispositivo IoT y los sensores
2. Realizar una ambientación con los microservicios de Smart Campus UIS

4.2 Fase de Planificación y Requerimientos

Es la fase en la que se realiza la planificación de la ejecución, los recursos y la información necesaria ya poseyendo conocimientos previos de las tecnologías para garantizar el éxito del proyecto.

Actividades:

1. Definir requerimientos funcionales y no funcionales
2. Definir actores
3. Definir diagramas UML

4.3 Fase de Diseño

Consiste en la especificación de la arquitectura, hardware e interfaces. En esta fase se tienen en cuenta dos pasos, el diseño de alto nivel y el diseño detallado. El diseño de alto nivel es un entregable que contiene aspectos generales del funcionamiento del sistema y el diseño detallado se enfoca en aspectos y especificaciones particulares de funcionamiento.

Actividades:

1. Definir arquitectura
2. Definición de componentes
3. Maquetación UX

4.3 Fase de Implementación

Consiste en el proceso de desarrollo del producto mediante los requerimientos y la arquitectura especificada en la fase de diseño.

Actividades:

1. Despliegue de la plataforma Smart Campus UIS
2. Montaje del prototipo del dispositivo de captura de datos
3. Desarrollo del dashboard y procesamiento de datos
4. Procesamiento de datos e Integrar componentes

4.5 Fase de Validación

Es la fase en donde se somete el sistema o producto desarrollado a una serie de pruebas que verifiquen la existencia de errores o comportamientos no esperados en la funcionalidad del proyecto, validando que lo desarrollado sea correcto y que los requerimientos se hayan cumplido correctamente, o en caso de que los requerimientos no sean apropiados o necesiten de modificaciones, permitir la retroalimentación y el cambio de estos regresando a la segunda fase del proyecto.

Actividades:

4. Realizar pruebas funcionales
5. Realizar pruebas de captura de datos
6. Definir rutas de transporte y recolectar datos
7. Elaborar una conclusión a partir del análisis de la información recolectada
8. Elaborar el informe final

5. Desarrollo

5.1 Ambientación Tecnológica

5.1.1 Smart Campus UIS - Core

Es el repositorio que contiene los componentes de la plataforma Smart Campus UIS, cada microservicio se encuentra previamente configurado a través de contenedores de Docker para facilitar su despliegue y configuración. Se encuentra publicado en un repositorio público de GitHub a través del siguiente enlace:

https://github.com/UIS-IoT-Smart-Campus/smart_campus_core_images.

5.1.2 Lenguaje de Programación

Para el desarrollo del sistema de información se eligió Python debido a sus ventajas en términos de productividad y simplicidad. Van Rossum (1997) señala que los programas en Python suelen ser entre 3 y 10 veces más cortos que los escritos en lenguajes como Java o C++, lo que acelera el proceso de desarrollo, aunque implique cierta pérdida en la velocidad de ejecución. Además, su sintaxis clara y legible facilita tanto el aprendizaje como el mantenimiento del código. A estas características se suma una comunidad activa y la amplia integración con múltiples sistemas y frameworks, factores que potencian un desarrollo ágil y eficiente.

5.1.3 Dispositivo IoT

El dispositivo IoT constituye el componente esencial para la captura de datos ambientales dentro del sistema. A partir de desarrollos anteriores ya se contaba con el microcontrolador ESP8266 y diversos sensores básicos, sin embargo, no se habían integrado sensores específicos para la medición de contaminantes criterio. Debido a consideraciones de presupuesto y facilidad de instalación, se optó por incluir la medición de un contaminante de alto impacto en la salud, el

material particulado. Por lo tanto, se incorpora el sensor SDS011, reconocido por su capacidad de medir concentraciones de PM2.5.

5.2 Planificación y Requerimientos

Cómo parte esencial del desarrollo del proyecto es importante hacer una clara definición de los requerimientos para tener claro el alcance y las funcionalidades esperadas. A continuación, se da una explicación de los requerimientos funcionales y no funcionales, así como de los diferentes actores y diagramas UML que permiten visualizar y racionalizar las diferentes funciones del sistema.

5.2.1 Definición de Actores

Los actores son agentes que interactúan con el sistema para realizar un caso de uso. En este caso se identifican:

Usuario: Persona que accede al sistema para consultar, filtrar y descargar información procesada por el sistema.

Desarrollador: Responsable de la implementación, mantenimiento y evolución del sistema, garantizando su correcto funcionamiento.

5.2.2 Requerimientos Funcionales

A continuación, se describe la lista de requerimientos funcionales del sistema de información para permitir el análisis de la calidad del aire en rutas de transporte público

Tabla 3

Lista de requerimientos funcionales.

ID	Nombre	Descripción
RF01	Consulta de datos	El sistema debe permitir la consulta de información de dispositivos IoT con los datos de PM2.5, CO_2, Temperatura, Nombre de la ruta, Longitud y Latitud.

RF02	Visualización de puntos georreferenciados	El sistema debe mostrar en un mapa interactivo los puntos de ubicación correspondientes a cada medición registrada por los dispositivos IoT.
RF03	Generación de trayectorias	El sistema debe trazar rutas en el mapa conectando los puntos georreferenciados de un mismo dispositivo, siguiendo el orden temporal de las mediciones.
RF04	Representación mediante escalas de color	El sistema debe aplicar colores a los tramos de las rutas de acuerdo con los valores de los parámetros ambientales (ej. PM2.5), basándose en los umbrales definidos en los índices de calidad del aire.
RF05	Representación de parámetros adicionales	El sistema debe permitir seleccionar las capas de contaminantes y/o parámetros (Ej. temperatura) representados en el mapa de forma distintiva y reconocible.
RF05	Leyenda del mapa	El sistema debe mostrar una leyenda interactiva sobre el mapa que permite alternar entre tipos de contaminantes y/o parámetros.
RF05	Consulta en tiempo real e histórico	El sistema debe permitir alternar la visualización en el mapa entre datos en tiempo real y datos históricos almacenados en la base de datos.
RF06	Construcción de gráficos	El sistema debe mostrar al usuario una serie de gráficos con información promedio de contaminación por ruta y diario.
RF07	Filtrado y selección	El sistema debe permitir al usuario mediante controles interactivos poder filtrar la información representada por parámetros como rutas, fechas, horas y tipo de categoría de calidad del aire.
RF08	Apartado analítico global	El sistema debe mostrar al usuario indicadores relevantes como periodos de mayor concentración, rutas con mayor y menor concentración de contaminantes, categoría de calidad del aire más común.
RF09	Puntos críticos geo-referenciados	El sistema debe mostrar los puntos de CO_2 y PM2.5, en mapas distintos con mayor concentración.

RF10	Estadísticas diarias	El sistema debe mostrar un apartado interactivo donde el usuario puede seleccionar días de registros con valores representativos como máximos o mínimos de contaminación y temperatura.
RF11	Datos tabulares	El sistema debe mostrar al usuario los datos en forma tabular para su inspección con las capacidades de filtro de rutas y fechas.
RF12	Descarga	El sistema debe permitir al usuario la descarga de los valores filtrados en la tabla.
RF13	Configuración de ruta a través del dispositivo IoT	El usuario debe poder configurar el nombre de la ruta desde el dispositivo para su respectiva clasificación y monitoreo.
RF14	Configuración del mensaje de dispositivo IoT	El payload del mensaje que envía el dispositivo debe seguir una estructura JSON con los nombres de las métricas y valores obtenidos de los sensores.

Nota. Cada requerimiento funcional es único y tiene una descripción de su función.

5.2.3 Requerimientos No Funcionales

Además de las funcionalidades específicas que debe ofrecer el sistema, es fundamental establecer las condiciones de calidad, rendimiento y restricciones bajo las cuales estas funcionalidades operarán. A continuación, se listan los requerimientos no funcionales del sistema de información:

Tabla 4

Lista de requerimientos no funcionales

ID	Nombre	Descripción
RNF01	Diseño Responsive	El sistema debe contar con un diseño de interfaz responsive. La disposición de los elementos, menús y componentes gráficos debe ajustarse de manera automática, manteniendo la coherencia visual y la facilidad de uso en móvil y escritorio.
RNF02	Actualización de datos en tiempo real	El sistema debe garantizar la actualización de las fuentes de datos en intervalos no superiores a 10 segundos.

RNF03	Manejo de errores en captura de datos	El sistema debe implementar mecanismos de validación que permitan filtrar automáticamente los registros de sensores que lleguen incompletos, corruptos o con valores fuera del rango esperado. A su vez, cuando no sea posible la consulta al servicio de datos el sistema debe notificar a través de una alerta en la interfaz sobre los fallos en la conexión.
RNF04	Recarga activa de datos	El sistema debe implementar un mecanismo de recarga activa que actualice únicamente los componentes necesarios sin recargar toda la aplicación para optimizar el rendimiento, disminuir el consumo de recursos y mejorar la experiencia del usuario.
RNF05	Navegación y experiencia UX	El sistema debe contar con una interfaz intuitiva que facilite la navegación entre las diferentes secciones, manteniendo una organización clara de menús, botones y flujos de interacción.
RNF06	Mantenibilidad y escalabilidad	El sistema debe estar diseñado bajo criterios de mantenibilidad que facilite la corrección de errores y escalabilidad que permita aumentar las capacidades del sistema.

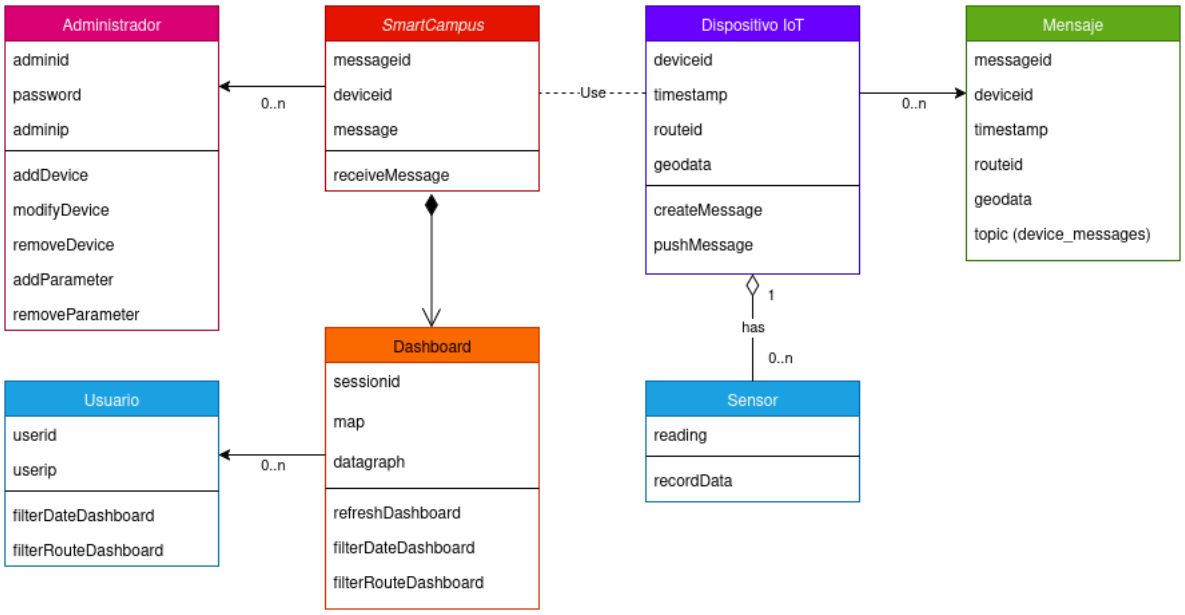
Nota. Cada requerimiento no funcional es único y tiene una descripción de su función.

5.2.4 Modelo de Objetos del Dominio

Tiene como fin representar los conceptos claves del dominio del problema, así como de identificar las relaciones entre las entidades del sistema y sus atributos, en la figura 4 se muestra el diagrama representando los objetos del dominio.

Figura 4

Diagrama de modelo de objetos del dominio



Nota. Diagrama UML denotando los objetos del dominio del sistema.

Basado en el diagrama, se da una explicación a detalle de cada objeto y sus propiedades:

Administrador: Hace parte del ecosistema de los microservicios y puede añadir, eliminar y modificar los distintos dispositivos IoT, así como configurar los parámetros y atributos de cada uno, en este caso cada medición de cada sensor, por ejemplo, Material Particulado o Dióxido de Carbono.

Usuario: Todo individuo que haga uso del dashboard podrá interactuar con el mapa, y podrá modificar filtros de rutas, de los atributos y por fecha.

Smart Campus UIS: Los microservicios son integrales al proyecto, ya que realizan todos los procesos requeridos para recibir información de cada dispositivo IoT, hace el consumo y guarda cada mensaje asociado a cada dispositivo dentro de la base de datos.

Dashboard: El centro de información y donde se pueden ver las concentraciones de contaminantes identificados por ruta y fecha en un mapa interactivo, así como diversas gráficas

para presentar y determinar nuevas conclusiones, tiene como funciones realizar un refresh de la información automáticamente cada cinco minutos, y recibir las solicitudes de filtrado del user.

Dispositivo IoT: Se encarga de recibir la información de los sensores y de empaquetar las mediciones en un mensaje con información crítica para organizar por fecha y ruta, y permitir enviarlo a los microservicios para su consumo.

Sensor: Tiene una simple función y es capturar el dato de la concentración de cada contaminante definido y medirlos cada cierto intervalo de segundos.

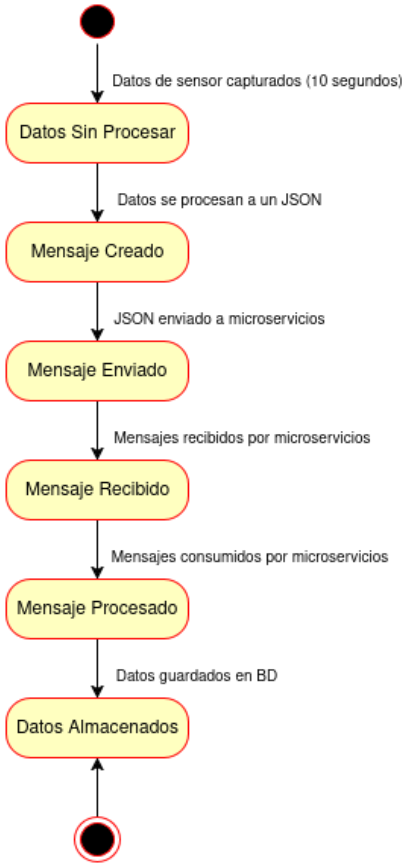
Mensaje: Es creado por el dispositivo IoT y está compuesto por la medición de concentración de contaminantes de cada sensor, el identificador del dispositivo IoT, del mensaje y de la ruta, así como una marca de tiempo y la ubicación en coordenadas geográficas.

5.2.5 Diagrama de Estados

Se utilizan para optimizar cualquier proceso de desarrollo donde sea útil visualizar los estados del objeto y las condiciones para que se produzca la transición de un estado a otro, en la figura 5 se puede apreciar el diagrama para la comunicación del dispositivo con los microservicios.

Figura 5

Diagrama de estados de la comunicación de mensajes



Nota. Diagrama demostrando los estados de la comunicación de mensajes y los procesos que se realizan para cada cambio de estado.

A continuación, se da una explicación a detalle de cada estado:

Datos Sin Procesar: Es el primer estado del envío de un mensaje, cuando los sensores capturan los datos y se tiene la información sin procesar.

Mensaje Creado: Después de obtener la información de los sensores, el dispositivo IoT compila un mensaje JSON con el tópico device_messages, los atributos o mediciones de cada sensor, un identificador de mensaje y de la ruta del transporte, así como la fecha de creación.

Mensaje Enviado: En este estado, el mensaje JSON se envía a los microservicios en un intervalo de 10 segundos.

Mensaje Recibido: Si el mensaje fue recibido correctamente por los microservicios entra a este estado.

Mensaje Procesado: Después de que el mensaje es recibido y los microservicios consumen el mensaje, entra a este estado.

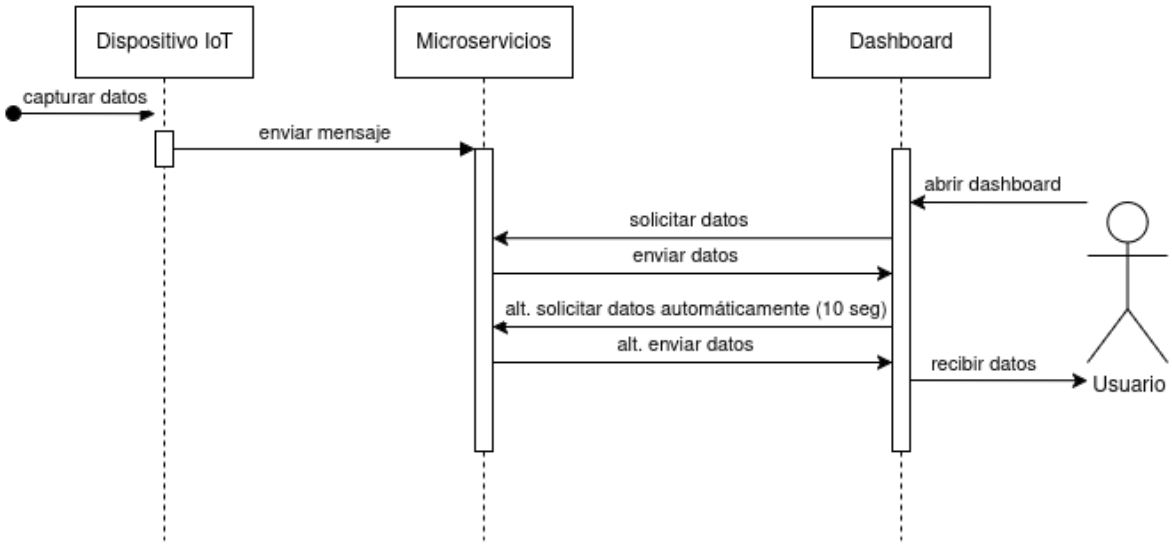
Datos Almacenados: Es el último estado, después de que el mensaje es procesado, se agregan los datos de los mensajes a la base de datos de los microservicios para su uso en el dashboard.

5.2.6 Diagrama de Secuencia

Se usan para visualizar los procesos y objetos que coexisten simultáneamente y la comunicación que ocurre entre ellos para ejecutar una función antes de que la línea de vida termine, en la siguiente figura 6 se puede apreciar el diagrama de secuencia para todo el sistema.

Figura 6

Diagrama de secuencia del sistema



Nota. Diagrama denotando la comunicación entre los distintos elementos del sistema y sus procesos en una línea de vida.

Comunicación entre el Dispositivo IoT, Microservicios, Dashboard y el Usuario. El proceso inicia desde el dispositivo IoT con la captura de los datos de los sensores, después de la creación del bulto de mensajes se envía a los microservicios, que después de recibir el paquete de mensajes envía una confirmación devuelta al dispositivo IoT.

Ya con los datos procesados y almacenados en la base de datos de los microservicios, un usuario puede entrar al dashboard, generando una solicitud a los microservicios para recibir datos y enviarlos de vuelta al usuario para popular el dashboard, También, un *refresh* cada treinta segundos que es enviado automáticamente para actualizar los datos del dashboard.

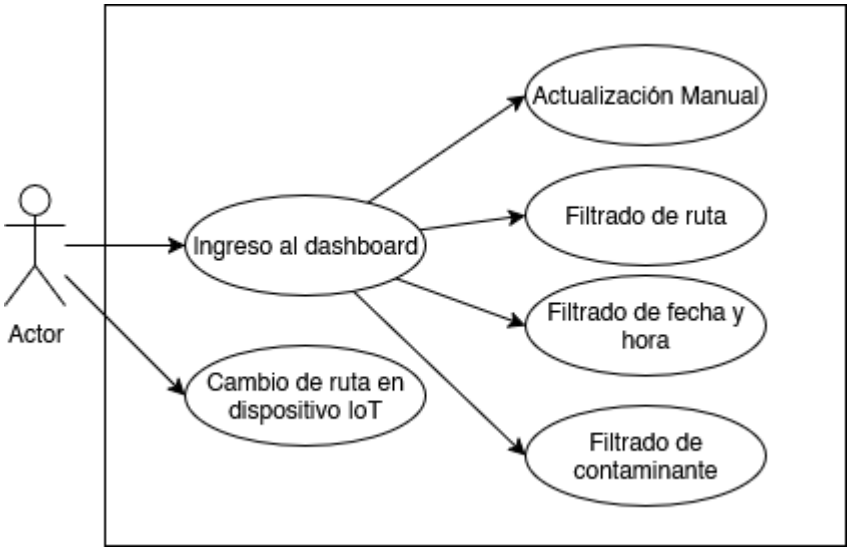
Procesos Únicos del Dispositivo IoT. Por su cuenta, el dispositivo IoT, después de enviar un bulto de mensajes y de recibir la confirmación por parte de los microservicios, elimina el anterior bulto de mensajes de su almacenamiento para asegurar que el dispositivo no tenga pérdidas de rendimiento a largo plazo.

5.2.7 Diagrama de Casos de Uso

Se usa para describir una secuencia de interacciones entre el sistema, en este caso el dashboard y un actor externo, el usuario en este ejemplo, que resultan en un resultado de valor para el actor cómo se puede ver en la figura 7.

Figura 7

Diagrama de Casos de Uso



Nota. Diagrama denotando las interacciones con el sistema por parte de un actor externo, en este caso el usuario.

A continuación, por cada interacción se proporciona a detalle una descripción, las precondiciones y postcondiciones de cada una para derivar cómo el sistema debería responder a cada solicitud por parte del usuario:

Ingreso al Dashboard

Descripción: Este caso de uso empieza cuando el usuario ingresa a la página del dashboard. El usuario ingresa desde el navegador de uso al sitio web del dashboard. El servidor responde y envía la información al usuario para su presentación en el navegador.

Precondiciones: El usuario debe tener un navegador compatible y una conexión a internet.

Poscondiciones: El usuario obtiene la información del servidor y tiene disponible para su uso y visualización el dashboard para la visualización de contaminantes criterio en Bucaramanga.

Actualización Manual

Descripción. Implica una actualización manual del dashboard por parte del usuario. El usuario presiona un botón dedicado para actualizar los datos del dashboard, incluyendo el mapa interactivo y las gráficas complementarias.

Precondiciones: Dependiendo de los filtros seleccionados, al hacer la actualización se quedarán guardados.

Poscondiciones: El servidor proporciona datos actualizados al usuario y se actualizan el mapa interactivo y las gráficas apropiadamente.

Filtrado de Ruta

Descripción: Este caso de uso implica al usuario modificando los filtros de cada ruta, hay tres maneras con las que el usuario puede interactuar, presionando los botones de seleccionar todos, deseleccionar todos o seleccionando con un cuadro de verificación a la izquierda de cada ruta de manera específica.

Precondiciones: La conexión con el sitio web del dashboard es estable y se encuentra conectado al momento de interactuar con los filtros.

Poscondiciones: Después de interactuar con el filtro, el mapa se actualiza inmediatamente y se esconde o reaparece el filtro apropiado.

Filtrado de Fecha

Descripción: Este caso de uso, de manera similar al filtrado de ruta implica al usuario modificando en este caso el filtro de fecha, que por defecto tiene un rango desde la fecha actual hasta los siete días anteriores, y en dos cuadros que contienen la fecha de inicio y la fecha de finalización.

Precondiciones: La conexión con el sitio web del dashboard es estable y se encuentra conectado al momento de interactuar con los filtros.

Poscondiciones: Después de interactuar con el filtro, el mapa se actualiza inmediatamente y esconde o reaparece el filtro apropiado.

Filtrado de Contaminante

Descripción: Al igual que el filtrado de rutas, implica al usuario interactuando con los tres botones de seleccionar de manera similar, pero en vez de rutas, se presentan los diferentes contaminantes criterio.

Precondiciones: La conexión con el sitio web del dashboard es estable y se encuentra conectado al momento de interactuar con los filtros.

Poscondiciones: Después de interactuar con el filtro, el mapa se actualiza inmediatamente y esconde o reaparece el filtro del contaminante apropiado.

Cambio de Ruta en Dispositivo IoT

Descripción: Mediante el acceso a una conexión al dispositivo IoT, permitir cambiar el campo de “location” para asignar la ruta a monitorear.

Precondiciones: Conexión establecida entre el punto de acceso y el dispositivo IoT.

Poscondiciones: Después de guardar la ubicación, el dashboard debe mostrar en los filtros la nueva ruta registrada.

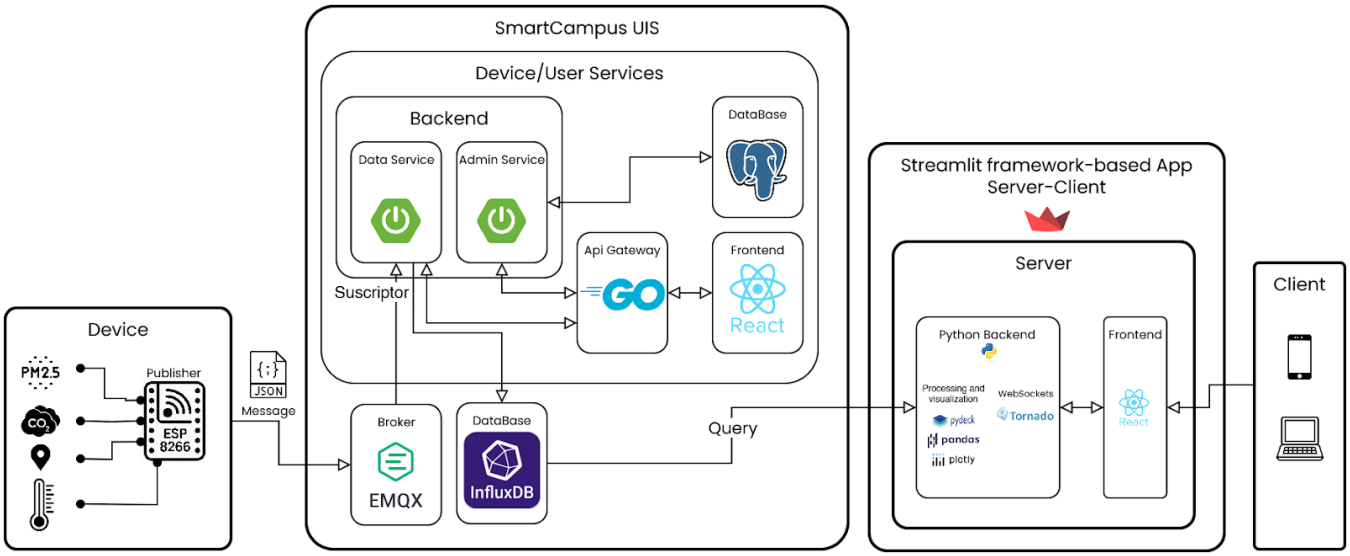
5.3 Diseño

5.3.1 Definición de Arquitectura

El siguiente diagrama en la figura 8 representa la arquitectura propuesta para la implementación del sistema de información:

Figura 8

Arquitectura propuesta para la implementación del sistema de información



Nota. Representación gráfica de la arquitectura propuesta

La arquitectura propuesta se integra con la plataforma Smart Campus UIS a través de los microservicios expuestos, estructurados en diferentes capas funcionales:

Capa de Dispositivos. Está conformada por sensores (ej. PM2.5, CO_2, temperatura) conectados a un microcontrolador ESP8266, que actúa como publicador (publisher). Este dispositivo envía vía Wi-Fi la información en un payload con formato JSON, el cual incluye métricas asociadas a la información espaciotemporal.

Capa de Comunicación y Almacenamiento (Broker y Bases de Datos). El broker EMQX recibe los mensajes provenientes de los dispositivos y los distribuye a los suscriptores correspondientes, en este caso el servicio de datos.

La información recolectada se almacena en la base de datos InfluxDB, dentro de un bucket predefinido. Esta base de datos está especializada en series temporales, lo que facilita el análisis de las variables ambientales registradas de forma continua.

Capa de Servicios (Device/User Services). Compuesta por microservicios desarrollados en Spring Boot (Data Service, Admin Service), los cuales gestionan la lógica de negocio y el acceso a la información.

Un API Gateway desarrollado en Go centraliza las solicitudes y enruta el tráfico hacia los servicios correspondientes.

La información se integra con una base de datos relacional (PostgreSQL) y un frontend en React, que provee interfaces web para la administración de registros de dispositivos y aplicaciones.

Capa de Análisis y Visualización (Streamlit App). Se implementa un framework basado en Streamlit, el cual opera bajo el modelo cliente-servidor.

En el servidor, un backend en Python procesa los datos mediante librerías como Pandas, Plotly y PyDeck, gestionando la comunicación a través de Tornado y websockets. La configuración de puertos y conexiones al cliente se delega directamente al framework.

En el cliente, la información procesada se presenta de manera interactiva en navegadores web o dispositivos móviles, permitiendo la consulta tanto de datos históricos como en tiempo real.

Interacción Cliente-Servidor. El usuario final accede desde un cliente (PC o dispositivo móvil) al frontend de Streamlit, el cual consulta la información almacenada en InfluxDB y la presenta mediante visualizaciones dinámicas e interactivas.

5.3.2 Definición de Componentes

A continuación, se describen los componentes que hacen parte de la categoría de Smart Campus UIS:

Procesamiento y Administración

Gateway (GO). Centraliza las solicitudes de los clientes y las redirige a los microservicios internos adecuados (por ejemplo, consultas a InfluxDB, MongoDB, o endpoints de autenticación).

Data Service (SpringBoot). Microservicio dedicado al manejo de registros, actúa como suscriptor para recibir los mensajes del broker con un topic específico.

Admin Service (SpringBoot). Microservicio para la administración de usuarios y datos relacionados.

Almacenamiento

InfluxDB. Base de datos principal de series temporales donde se almacenan las métricas IoT.

Postgres. Base de datos PostgreSQL para almacenamiento relacional. Para información complementaria a las métricas de IoT (usuarios, proyectos, configuraciones, metadatos).

Captura de Datos

Broker (EMQX). Broker MQTT que recibe los datos de los sensores distribuidos y gestiona la comunicación IoT mediante el protocolo MQTT (topics, QoS, suscripciones).

Data-Service. Agente de recolección de métricas y logs, se conecta a las fuentes de datos y envía la información a InfluxDB. Permite definir inputs (MQTT, HTTP, etc.) y outputs (InfluxDB, archivos).

Visualización

Frontend (React). Interfaz gráfica para gestionar la infraestructura de dispositivos, modelos y aplicaciones.

Para el sistema de información y procesamiento de datos, se describe el siguiente componente:

Componente para la Implementación del Sistema de Información. La elección de la tecnología a usar se basó en la facilidad de integración, la frecuencia de actualización y respaldo. En primer lugar, Python ofrece una integración completa con casi cualquier componente, lo cual facilita el consumo de datos de InfluxDB.

Respecto al framework, se realizó una investigación de librerías modernas que facilitaran el desarrollo del sistema de información, considerando que los requerimientos se enfocan principalmente en el tratamiento y análisis de información espaciotemporal. Como resultado de este análisis, se eligió **Streamlit**, un framework de código abierto desarrollado en Python y orientado a la construcción de aplicaciones para el análisis de datos. Su funcionamiento se basa en una arquitectura cliente-servidor, donde el servidor es gestionado mediante un script en Python que se comunica a través de websockets utilizando el framework Tornado. Por su parte, en el cliente, el renderizado de los componentes se realiza mediante React.

5.3.3 Maquetación UX

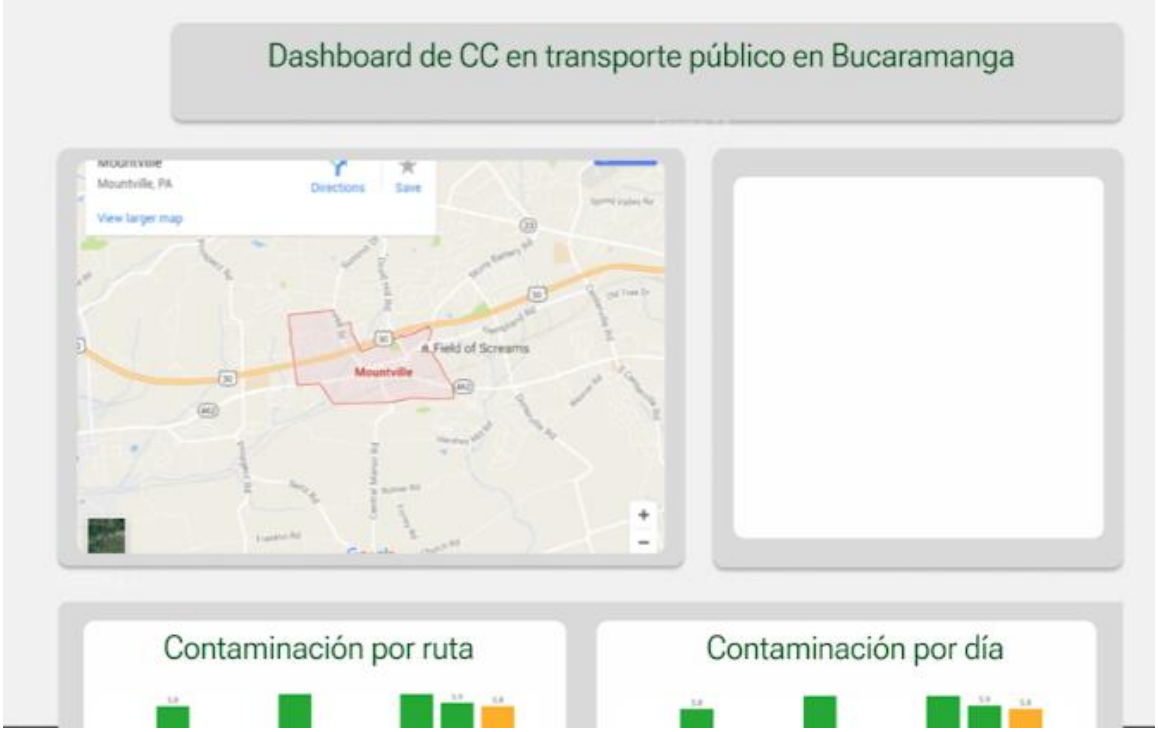
La maquetación del UX (User Experience) permite visualizar y ajustar el orden de los elementos antes del desarrollo para que el usuario tenga una mejor experiencia y se de valor agregado al software, en este caso el dashboard, así mismo basados en principios modernos se tiene en cuenta el diseño responsivo, que permitirá que el dashboard se pueda visualizar y usar

correctamente en cualquier dispositivo, teniendo en cuenta diferentes navegadores y resoluciones de pantallas cómo móviles y tablets.

A continuación, se presentan tres propuestas para el dashboard, para pantallas de computadoras, tablets y para móviles, realizadas en Figma.

Figura 9

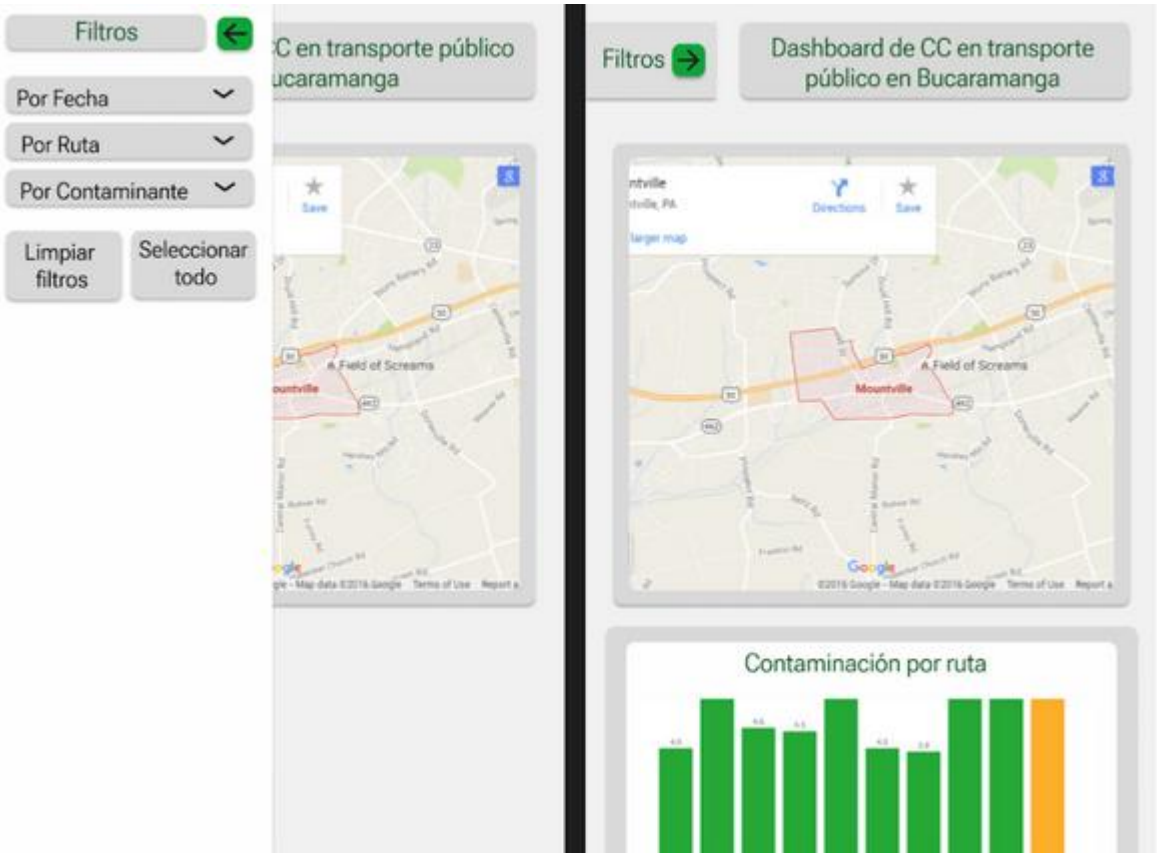
Captura del prototipo de interfaz del dashboard para tabletas en modo horizontal



Nota. Captura de Figma mostrando el prototipo de la interfaz para su uso en dispositivos como tablets en modo horizontal.

Figura 10

Captura del prototipo de interfaz del dashboard para tabletas en modo vertical



Nota. La interfaz de la izquierda muestra el menú que se despliega desde la parte izquierda de la pantalla para interactuar con los filtros del mapa y respectivamente la interfaz en la derecha muestra el menú minimizado.

Figura 11

Captura de Figma del prototipo de interfaz del dashboard para computadoras

Nota. Esta interfaz incluye el menú de filtros activado.

Con estas propuestas de maquetación se llega a un mejor entendimiento de cómo los filtros funcionan según los casos de uso, cómo se puede aprovechar de la mejor manera el espacio disponible en pantalla y la compatibilidad con todo tipo de dispositivos así cómo evitar la pérdida de información, el objetivo es que, sin importar el usuario, pueda de manera intuitiva interactuar con todas las funciones del dashboard.

5.4 Implementación

5.4.1 Despliegue de Smart Campus UIS

Para el despliegue de la plataforma *Smart Campus UIS*, el repositorio de GitHub se clonó directamente en el almacenamiento local. Posteriormente, utilizando Docker Desktop, se gestionó la orquestación de contenedores y el despliegue de las imágenes mediante el comando docker compose.

A través del archivo docker-compose.yml del repositorio, se encuentra ya establecida la asignación de puertos y configuración de los microservicios necesarios para la implementación del sistema de información.

Figura 12

Microservicios de Smart Campus UIS en Docker Desktop

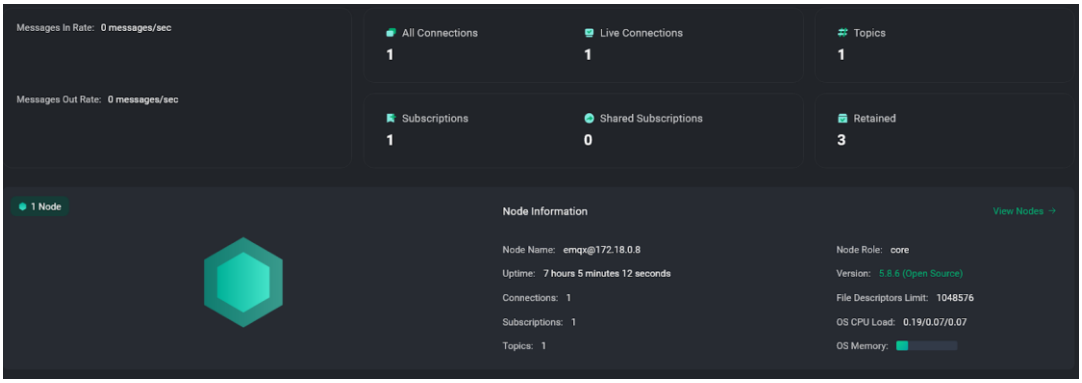
Nota. Microservicios de Smart Campus UIS desplegados y monitoreados a través de docker-desktop, incluyendo el puerto con el que se puede acceder a cada servicio.

Al desplegar los servicios en el .env del proyecto se encuentran las configuraciones de credenciales y direcciones necesarias para el correcto funcionamiento y configuración de componente de Smart Campus UIS.

El servicio que incluye el Broker expone un frontend que permite monitorear mensajes entrantes y salientes, así como las conexiones establecidas y el topic de los mensajes.

Figura 13

Frontend de EMQX

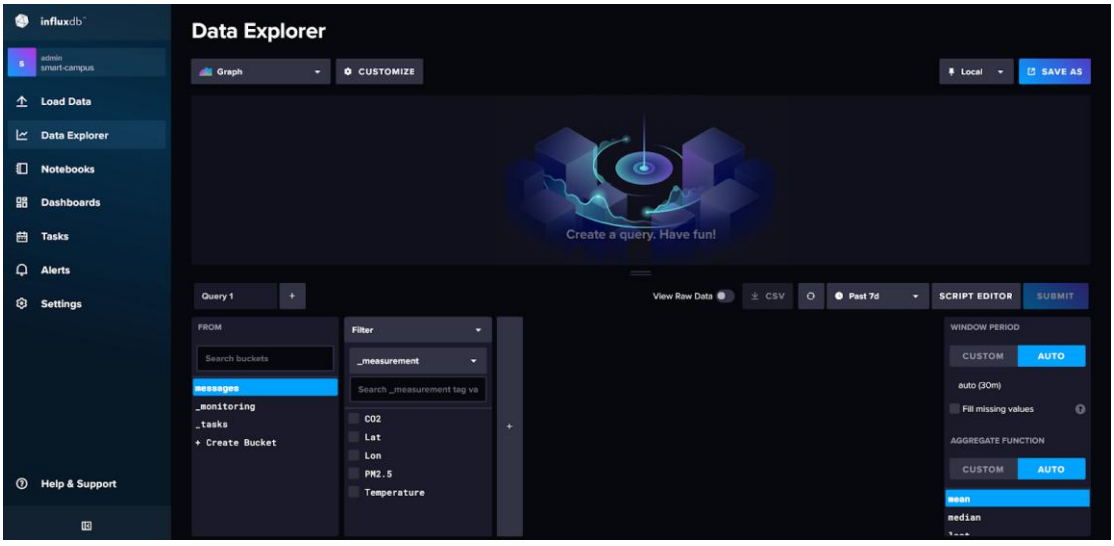


Nota. Se pueden visualizar diferentes aspectos, incluyendo conexiones e información sobre el nodo.

InfluxDB también expone un frontend que permite configurar y consultar información enviada por el broker.

Figura 14

Frontend de InfluxDB

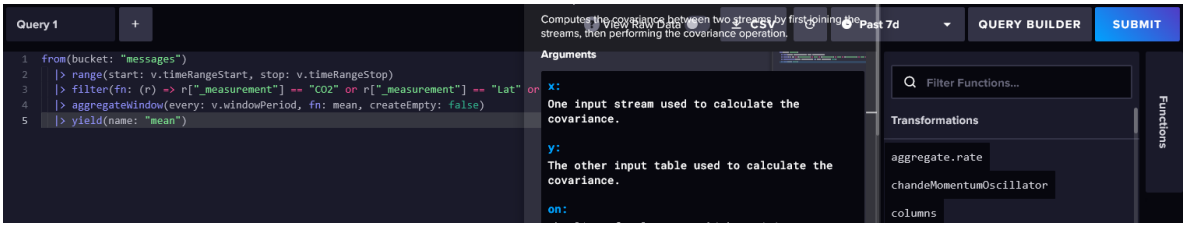


Nota. Frontend expuesto por el servicio de InfluxDB, mostrando la sección de Data Explorer donde se pueden visualizar los datos almacenados.

InfluxDB se enfoca en datos de series de datos temporales, es necesario generar un token de autenticación para utilizar datos mediante consultas redactadas en Flux, un lenguaje que InfluxDB emplea con una leve similitud a SQL.

Figura 15

Editor de consultas de InfluxDB



Nota. A diferencia de una query común, estas consultas se crean basándose en el lenguaje Flux.

InfluxDB permite realizar pruebas y visualizar los datos registrados, lo cual resulta útil para validar la información antes de su consumo por parte de la aplicación. Asimismo, ofrece un mecanismo de autenticación basado en tokens, que facilita la integración de la base de datos con otros servicios que requieran acceder a ella. Por este motivo, se generó un token específico que fue almacenado en la configuración del dashboard para hacer la conexión.

Smart Campus UIS a su vez, integra un front-end propio conectado a los servicios de administración y datos que permite registrar información adicional como modelos, aplicaciones y dispositivos.

Figura 16

Frontend de Smart Campus UIS

Nota. Desde este frontend se pueden administrar y monitorear los distintos dispositivos y modelos que se comunicarán con el sistema, así como el estado de los servicios.

5.4.2 Integración del Dashboard con Smart Campus UIS

Para realizar la integración con Smart Campus UIS, basado en la arquitectura propuesta, la conexión se implementó a través de la configuración de un cliente que se conecta a la base de datos al desplegar el sistema de información.

Figura 17

Fragmento de código para la conexión con InfluxDB

```
def _new_client() -> InfluxDBClient:
    """
    Crea el cliente
    """
    return influxdb_client.InfluxDBClient(
        url=INFLUX_URL,
        token=INFLUX_TOKEN,
        org=INFLUX_ORG,
        timeout=30_000,
        enable_gzip=True
    )
```

Nota. Función en el archivo de conexión para crear el cliente de InfluxDB.

Para su correcto funcionamiento se requieren parámetros especificados en las variables de entorno del repositorio de Smart Campus UIS y el token generado en el servicio desplegado de InfluxDB.

Con la conexión al cliente establecido se definieron funciones para ejecutar consultas específicas en formato texto basadas en flux, ejecutar un ping básico para verificar la conexión y algunas utilidades adicionales como convertir las fechas a formato local.

En la figura 18 se especifica la consulta principal para obtener los datos requeridos para el consumo y procesamiento.

Figura 18

Fragmento de código para obtención de datos de InfluxDB

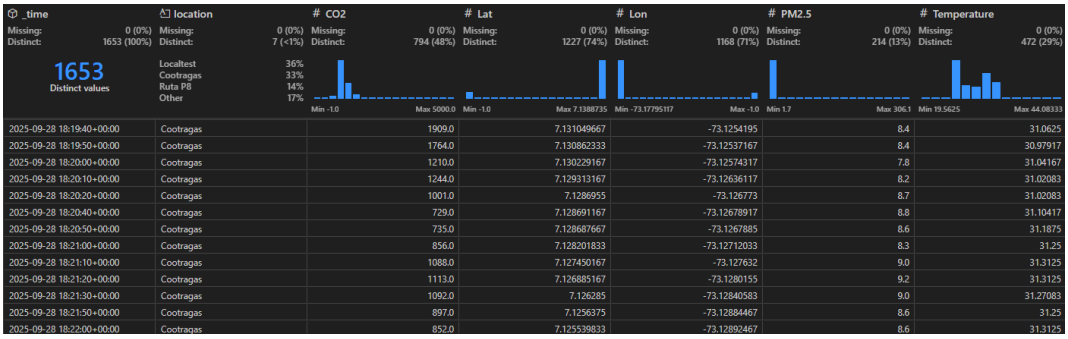
```
from(bucket: "{bucket}")
|> range(start: {start})
|> filter(fn: (r) =>
  r._measurement == "CO2" or
  r._measurement == "PM2.5" or
  r._measurement == "Temperature" or
  r._measurement == "Lat" or
  r._measurement == "Lon"
)
|> aggregateWindow(every: 10s, fn: last, createEmpty: false)
|> pivot(
    rowKey: ["_time","location"],
    columnKey: ["_measurement"],
    valueColumn: "_value"
)
|> keep(columns: ["_time","location","Lat","Lon","CO2","Temperature","PM2.5"])
|> sort(columns: ["location","_time"])
'''
```

Nota. Consulta escrita en flux para la obtención de datos del bucket de InfluxDB.

Al ejecutar la consulta a InfluxDB, usando las funciones del cliente, se especifica que retorne un *DataFrame* de Pandas para facilitar el procesamiento de datos necesario para su visualización y análisis, como se muestra en la figura 19.

Figura 19

DataFrame resultante de la consulta a InfluxDB



_time	location	CO2	Lat	Lon	PM2.5	Temperature
2025-09-28 18:19:40+00:00	Cootragas	1909.0	7.131049667	-73.1254195	8.4	31.0625
2025-09-28 18:19:50+00:00	Cootragas	1764.0	7.130862333	-73.12537167	8.4	30.97917
2025-09-28 18:20:00+00:00	Cootragas	1210.0	7.130229167	-73.12574317	7.8	31.04167
2025-09-28 18:20:10+00:00	Cootragas	1244.0	7.129313167	-73.12636117	8.2	31.02083
2025-09-28 18:20:20+00:00	Cootragas	1001.0	7.1286955	-73.126773	8.7	31.02083
2025-09-28 18:20:40+00:00	Cootragas	729.0	7.128691167	-73.12678917	8.8	31.10417
2025-09-28 18:20:50+00:00	Cootragas	735.0	7.128687667	-73.1267885	8.6	31.1875
2025-09-28 18:21:00+00:00	Cootragas	856.0	7.128201833	-73.12712033	8.3	31.25
2025-09-28 18:21:10+00:00	Cootragas	1088.0	7.127450167	-73.127632	9.0	31.3125
2025-09-28 18:21:20+00:00	Cootragas	1113.0	7.126885167	-73.1280155	9.2	31.3125
2025-09-28 18:21:30+00:00	Cootragas	1092.0	7.126285	-73.12840583	9.0	31.27083
2025-09-28 18:21:50+00:00	Cootragas	897.0	7.1256375	-73.12884467	8.6	31.25
2025-09-28 18:22:00+00:00	Cootragas	852.0	7.125539833	-73.12892467	8.6	31.3125

Nota. Visualización del DataFrame resultante de la consulta a InfluxDB, en la parte superior se encuentran las columnas y debajo de cada una los datos asociados.

5.4.3 Procesamiento de Datos

Para el procesamiento de principal, se plantea de acuerdo con los requerimientos poder visualizar segmentos de ruta en el mapa interactivo, para ello, se definieron las siguientes funciones de transformación de datos para entregar al componente y realizar una correcta visualización.

Figura 20

Fragmento de código para el etiquetado de rutas según el rango de concentración de PM2.5

```
PM25_THRESHOLDS = [
    (0.0, 12.0, "Buena", "#00e400"),          # Verde
    (12.1, 35.4, "Moderada", "#ffff00"),      # Amarill
    (35.5, 55.4, "Dañina para sensibles", "#ff7e00"),
    (55.5, 150.4, "Dañina", "#ff0000"),       # Rojo
    (150.5, 250.4, "Muy dañina", "#8f3f97"),  # Púrpur
    (250.5, 500.4, "Peligrosa", "#7e0023")    # Rojo o
]

def get_pm25_color_and_category(pm25_value):
    # Convierte valor PM2.5 → [R, G, B, A] + Categoría
    for low, high, category, color_hex in PM25_THRESHOLD
        if low <= pm25_value <= high:
            r = int(color_hex[1:3], 16)  # Hex → RGB
            g = int(color_hex[3:5], 16)
            b = int(color_hex[5:7], 16)
            return [r, g, b, 180], category
```

Nota. Función para el etiquetado de rutas basado en valores estandarizados de calidad del aire según las concentraciones del material particulado 2.5.

Con esta función de etiquetado, ahora lo que se busca es armar conjuntos de fragmentos de ruta basados en los puntos obtenidos a través de la información de los sensores. En la figura 21 se especifica la lógica y validaciones necesarias para generar los fragmentos de ruta en donde se consideraron posibles errores de los datos, por tanto, se valida la distancia espacial mediante la definición de distancia euclidiana entre dos puntos y temporal usando la diferencia de fechas para poder unir puntos.

Figura 21

Fragmento de código para la construcción de fragmentos de ruta

```
def _build_for_subset(sub: pd.DataFrame):
    # 1. Ordenar por tiempo
    sub = sub.sort_values("_time").copy()

    # 2. Para cada par de puntos consecutivos:
    for i in range(len(sub) - 1):
        current_point = sub.iloc[i]
        next_point = sub.iloc[i + 1]

        # 3. Validación temporal (máximo 3 minutos)
        time_diff = abs((next_time - current_time).total_seconds())
        if time_diff > 180:  # 3 minutos = 180 segundos
            continue  # Saltar este segmento

        # 4. Validación espacial (máximo ~1km)
        lat_diff = abs(next_point['Lat'] - current_point['Lat'])
        lon_diff = abs(next_point['Lon'] - current_point['Lon'])
        distance_degrees = (lat_diff**2 + lon_diff**2)**0.5
        if distance_degrees > 0.01:  # ~1km
            continue  # Saltar este segmento
```

Nota. Función para la construcción de fragmentos de ruta, en otros términos, la conexión entre puntos para formar una línea.

Mediante la iteración de puntos se configuran los campos mostrados en la figura 22 necesarios para la representación visual de cada ruta. Cada asignación se guarda en una lista que posteriormente se transforma en un DataFrame para consumirlo desde el componente del mapa.

Para la visualización de la información de cada fragmento de ruta se decidió usar la información del punto inicial para evitar posibles inconsistencias o promedios inconsistentes.

Figura 22

Fragmento de código para el cálculo de la representación visual de fragmentos de ruta

```
path = {
    "start_lon": current_point["Lon"],
    "start_lat": current_point["Lat"],
    "start_elevation": 10,
    "end_lon": next_point["Lon"],
    "end_lat": next_point["Lat"],
    "end_elevation": 10,
    "R": path_color[0],
    "G": path_color[1],      You, 6 days ago • Refactor data queries in
    "B": path_color[2],
    "A": opacity,  # Store opacity separately for easier access
    "pm25_category": current_category,
    "co2_value": current_point.get("co2_value", 0),
    "pm25_value": current_point.get("pm25_value", 0),
    "temperature": current_point.get("temperature", 0),
    "timestamp": (
        format_colombia_time(current_point["_time"])
        if "_time" in current_point and pd.notna(current_point["_time"])
        else "No disponible"
    ),
    "location": current_point.get("location", "No disponible"),
```

Nota. Campos calculados necesarios para la representación visual de cada fragmento de ruta en PyDeck.

5.4.4 Desarrollo del Dashboard

Para el desarrollo del dashboard, como se mencionó en la arquitectura, se decidió usar Streamlit, un framework en Python especializado en aplicaciones de datos, con capacidades de diseño responsive y basado en una single page application cliente-servidor.

Primero se determinó una librería de Python que permitiera visualizar puntos geográficos en un mapa interactivo, y presentar las rutas de transporte de una manera intuitiva, en este caso PyDeck fue elegido por sus capacidades de visualización gráfica y funciones que permiten la interacción en axis 3D. PyDeck además permite que, al acercar el cursor al punto de las rutas, se pueda visualizar los datos que contiene este punto, en este caso la concentración de contaminantes. El código completo del proyecto se encuentra alojado en el repositorio:

https://github.com/diegortega98/SI_Calidad_del_Aire_SCUIS

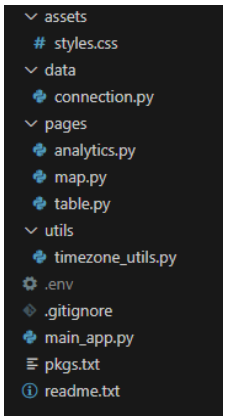
Por defecto, la interfaz de ejemplo del dashboard en Streamlit se acerca en cierta parte a la diseñada en la maquetación, incluyendo la barra lateral que se puede desplegar o minimizar para acceder a otras páginas.

Sin embargo, el framework usa componentes por defecto, por razones del diseño y por tener la posibilidad de insertar elementos que no vienen incluidos por defecto en el framework, se hace uso de un documento de estilos de CSS que se inyecta al proyecto, y permite añadir atributos como flexbox y box shadow a los distintos elementos de la aplicación.

Los elementos creados con CSS se desarrollan con el diseño responsive en mente, haciendo uso de flexbox para que las gráficas que en una pantalla en orientación horizontal normalmente se encuentran una al lado de la otra, se posicionan una debajo de la otra con el cambio de orientación, así como el uso del atributo media para cambiar el tamaño de las fuentes dependiendo de la resolución del dispositivo. El proyecto con base en el framework Streamlit se organizó como se muestra en la figura 23:

Figura 23

Captura de la estructura del proyecto en Visual Code



Nota. Estructura del proyecto usando Python / Streamlit dentro de Visual Code.

Assets. Contiene archivos estáticos que usa la aplicación, como una hoja de estilos personalizada para dar formato a la interfaz (colores, fuentes, tamaños, etc.).

Data. Contiene un archivo para centralizar y configurar la conexión a InfluxDB.

Pages. Organiza el proyecto en páginas navegables, cada una con un enfoque de visualización y análisis específico (p. ej., mapa interactivo, tablas e indicadores). Esto permite separar funcionalidades, facilitar la navegación y mantener vistas optimizadas para cada caso de uso.

Utils. Contiene archivos auxiliares que soportan funciones específicas del sistema de información como la conversión de fechas.

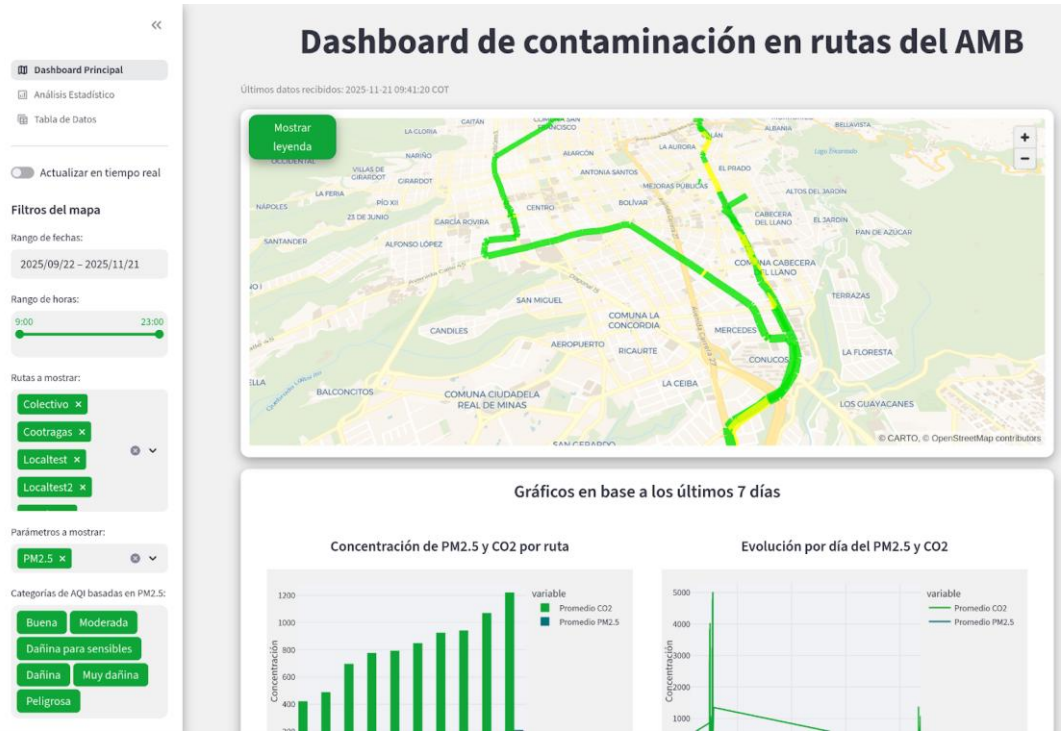
Archivo Principal. Ubicado en la raíz del proyecto, define la estructura base y contiene la lógica necesaria para inicializar y ejecutar la aplicación. Desde este archivo se configuran los componentes principales y se gestiona el levantamiento del sistema.

Para dar una mejor organización y más utilidad al dashboard, se desarrollaron tres páginas, cada una con función específica:

Dashboard Principal: Esta página principal cuenta con una vista del mapa que puede mostrar la concentración de cada contaminante criterio, así como la temperatura, con una leyenda apropiada para cada contaminante y los niveles de concentración.

Figura 24

Captura del dashboard principal



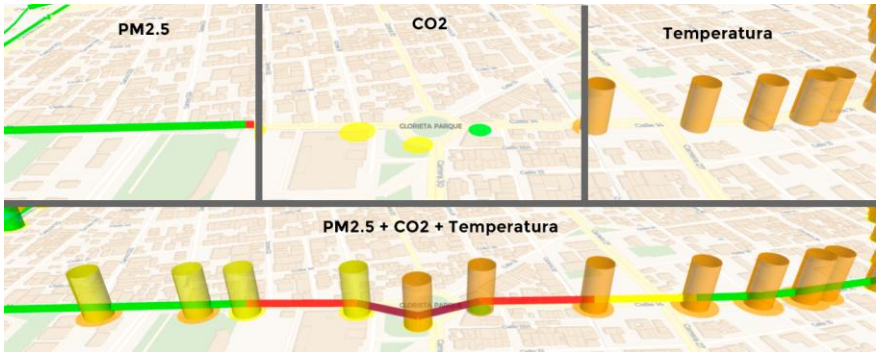
Nota. Captura del dashboard donde se puede visualizar el mapa, filtros y gráficos.

El mapa se puede filtrar por rutas, contaminantes, categorías de concentración de PM2.5 y por fecha y hora que se encuentran ubicados en el sidebar. Además, se cuenta con un toggle para que se pueda actualizar automáticamente en tiempo real cada diez segundos los datos del mapa usando el motor de PyDeck.

El componente del mapa está configurado con base en capas de datos, es decir, para cada tipo de parámetro se configuró una capa distinta para visualizar de forma clara los valores recopilados como se muestra en la figura 25, esta configuración por capas permite deshabilitar y habilitar usando los filtros de la interfaz como los que se muestran en figura 26.

Figura 25

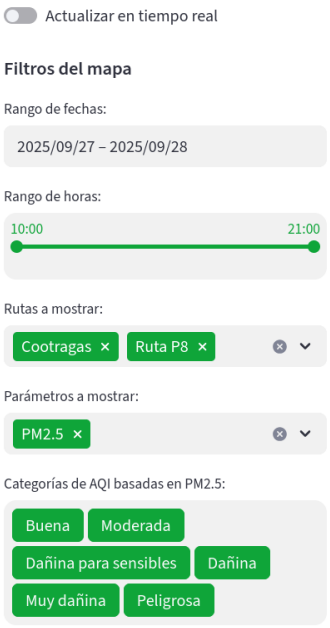
Colección de capturas de cada capa en el mapa por atributo



Nota. Diferencia entre capas configuradas en PyDeck, la captura en el inferior selecciona los tres elementos en el filtro.

Figura 26

Captura del panel lateral del dashboard principal

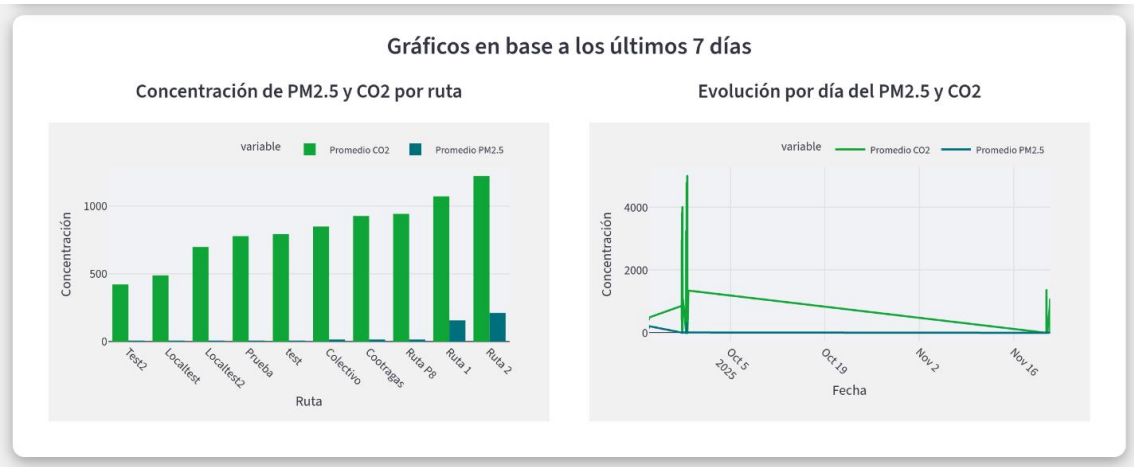


Nota. Este panel contiene varios filtros aplicables a los datos por mostrar en el dashboard.

También se encuentran dos gráficas basadas en los últimos 7 días que muestran las rutas con mayor contaminación en promedio del PM2.5 y el CO_2 y la evolución por hora del PM2.5 y el CO_2.

Figura 27

Captura de los gráficos del dashboard principal con base en los últimos 7 días



Nota. Gráficos obtenidos con los datos recopilados en los últimos 7 días a la fecha.

Además, haciendo uso de los estilos de CSS y de las funciones st.markdown y st.html, se convierte la leyenda en forma de tabla para mostrar al usuario cómo se clasifican los rangos de las distintas concentraciones de contaminantes criterio, de manera apropiada las medidas en el mapa se grafican basado en colores estándares para estas clasificaciones, también cuenta con un botón para refrescar el contenido de la página y para ocultar o mostrar la leyenda.

Figura 28

Captura de la leyenda del mapa

Nota. La leyenda sigue los rangos estándares para cada contaminante criterio.

Análisis Estadístico: Esta página es creada para apoyar con el análisis de los datos obtenidos en las rutas, incluyendo una sección con varios elementos de métricas cómo la hora con mayor concentración de PM2.5 o la ruta con mayor concentración del PM2.5, así como otro container con un gráfico de pastel para representar la distribución de clasificaciones de AQI basándose en la concentración del PM.5 y varias estadísticas para cada día en específico que contenga datos, permitiendo ver una evolución de cada parámetro a lo largo del tiempo.

Además, se incluyen algunos mapas y gráficos que permiten visualizar los puntos con concentraciones más altas de PM2.5, las concentraciones de CO_2 en el mapa y también se encuentran las mismas gráficas del dashboard principal, pero en esta sección incluyen los datos de todas las fechas y no solo los últimos 7 días.

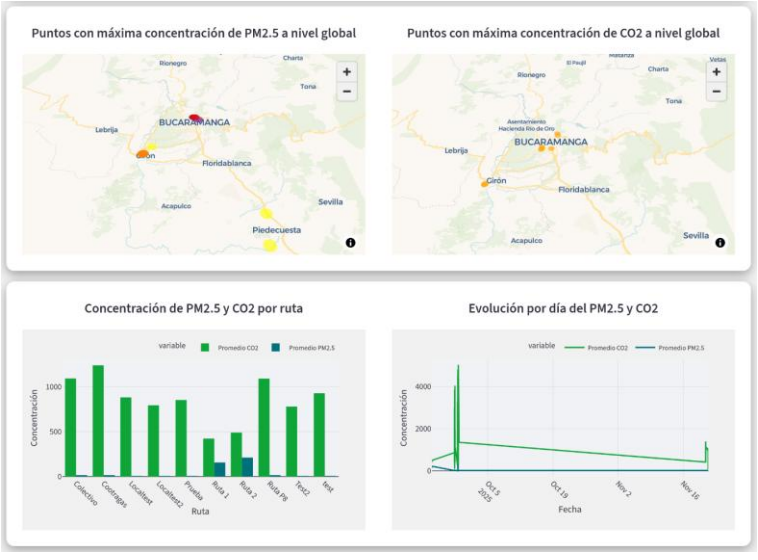
Figura 29

Captura de la página de análisis estadístico con las primeras secciones

Nota. En la captura se puede apreciar la sección con métricas y la sección con la distribución de clasificaciones de PM2.5 y estadísticas diarias.

Figura 30

Captura de las otras secciones de la página de análisis estadístico



Nota. En la captura se puede apreciar la sección con los mapas con los puntos de concentraciones y las gráficas de concentración de PM2.5 y CO_2.

Tabla de Datos: Con el fin de dar más herramientas al usuario se implementa una página principalmente compuesta por una tabla con todos los datos del DataFrame, así como varios indicadores respecto a los datos seleccionados y también la inclusión de filtros para las rutas y el rango de fechas, usando al elemento de Streamlit DataFrame tiene funciones de descarga a csv, de búsqueda y de pantalla completa.

Figura 31

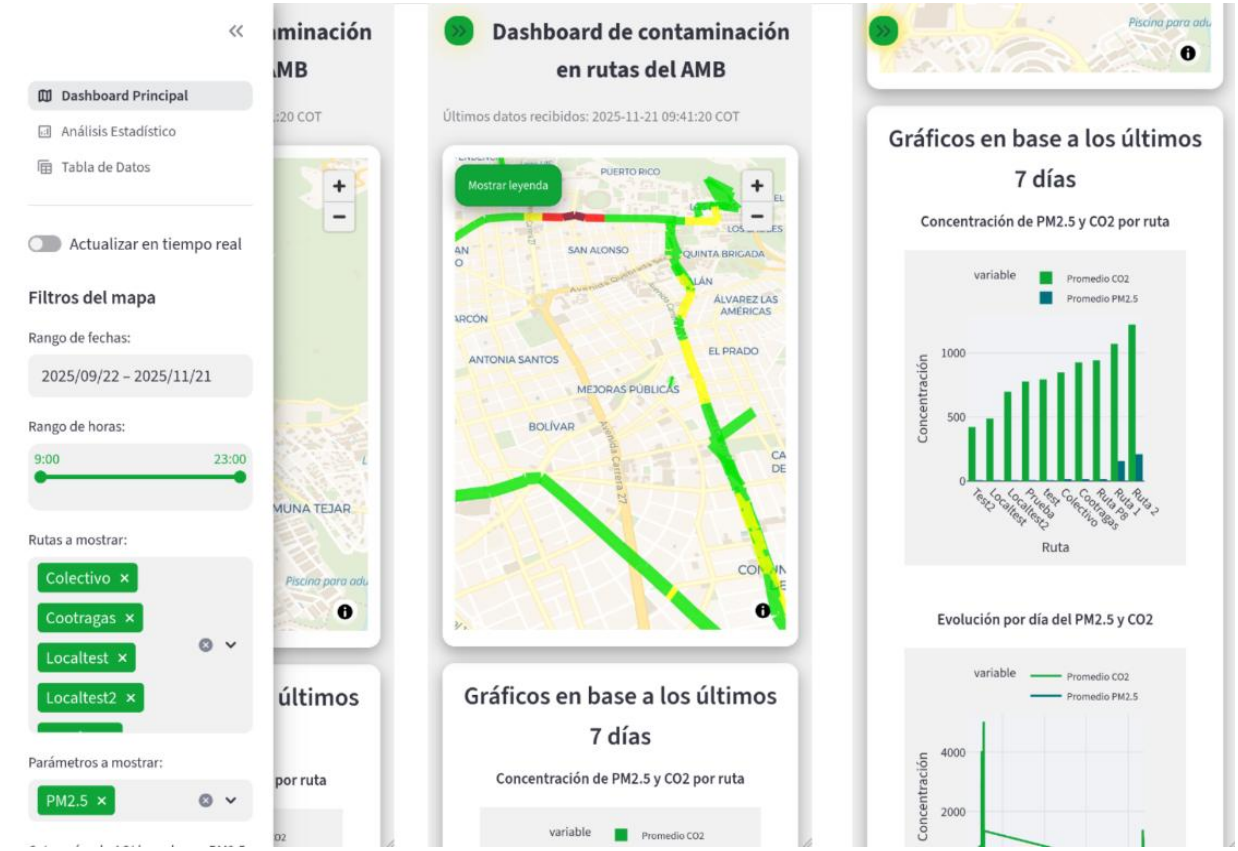
Captura de la página de tabla de datos

Nota. La página contiene filtros en el panel lateral, métricas de los datos y una tabla con los datos del DataFrame.

Se realizó un diseño con capacidades responsive, los contenedores con múltiples elementos hacen uso de un display flex, que permite que las gráficas o los elementos se puedan mover el uno por debajo del otro cuando la resolución horizontal es menor a 1100 px, así como el incremento o reducción en escala de los elementos de texto y la leyenda del mapa, en las siguientes figuras se pueden apreciar algunos de estos comportamientos:

Figura 32

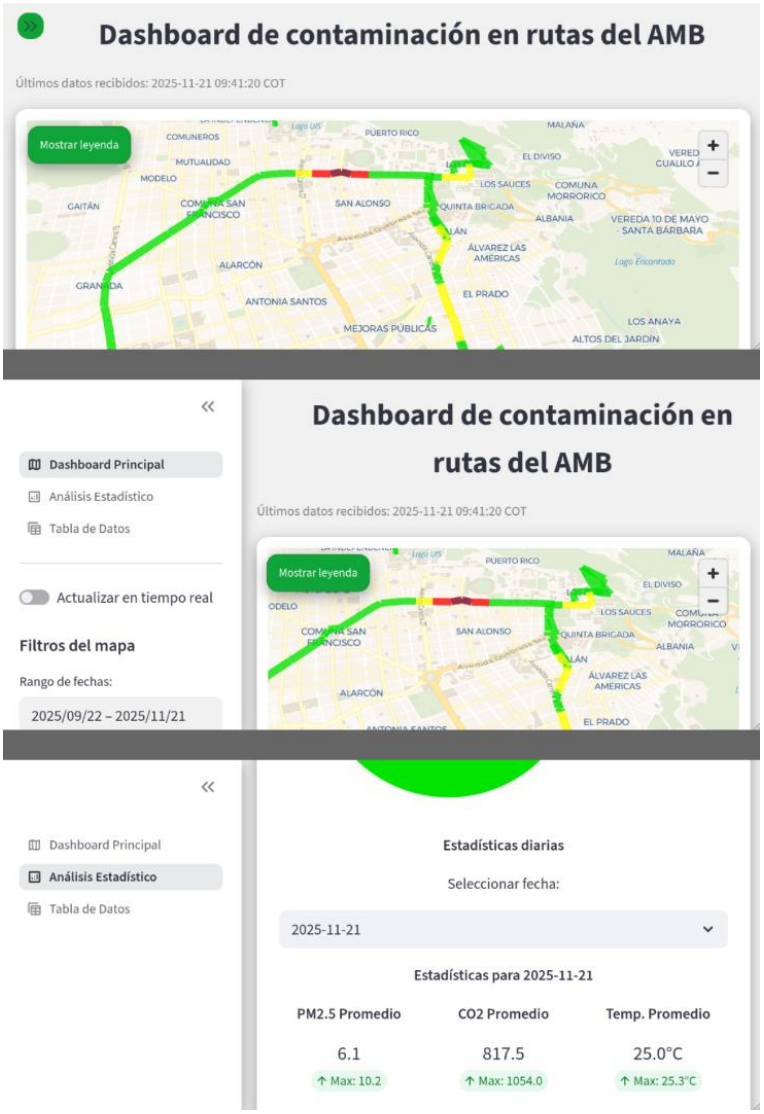
Capturas del diseño responsive del dashboard en orientación vertical con resolución de dispositivo móvil



Nota. Cuando la resolución horizontal no permite ver de manera proporcional las gráficas, la segunda gráfica pasa a estar debajo de la primera.

Figura 33

Capturas del diseño responsive del dashboard en orientación horizontal con resolución de dispositivo móvil



Nota. El mapa y las gráficas se ajustan según la resolución del dispositivo.

5.4.5 Montaje del Prototipo del Dispositivo de Captura de Datos

Para el montaje del dispositivo se utilizaron los componentes descritos en la tabla 5:

Tabla 5

Lista de componentes para el montaje del dispositivo

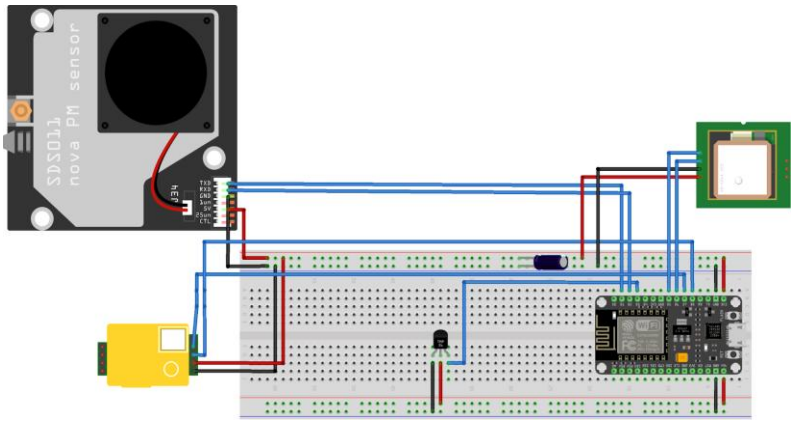
Nombre	**Función**
ESP8266	Microcontrolador con conexión Wifi
SDS011	Sensor de material particulado
MH-Z19B	Sensor de concentración de CO_2
DS18B20 + Antena	Temperatura
NEO-M8N	GPS
Capacitor de 1000 microfaradios	Complemento para evitar picos de energía
Protoboard y cables	Conexión entre dispositivos
PowerBank 20000 Amp	Fuente de alimentación

Nota: Incluye el nombre de cada componente y la función que tiene dentro del dispositivo.

Se ensambló el dispositivo de acuerdo con la representación gráfica de la figura 34, utilizando el esp8266 para la comunicación vía wifi al Broker de EMQX. Para la configuración del dispositivo se programó mediante el IDE de Arduino especificando el payload del mensaje con la estructura en la figura 35. Además, se incluye una imagen del montaje físico del dispositivo, véase el Apéndice A.

Figura 34

Diagrama de conexión de los componentes



Nota. Diagrama creado en un simulador de electrónica mostrando la conexión de los componentes para la toma de datos.

Figura 35

Payload del mensaje en IDE de Arduino

```
{
  "header": {
    "userUUID": {{default}},
    "deviceId": {{default}},
    "timeStamp": "{{timeStamp}}",
    "topic": "device/messages",
    "shouldRequeue": true,
    "location": {{location}}
  },
  "metrics": [
    { "measurement": "CO2",       "value": {{co2}} },
    { "measurement": "PM2.5",     "value": {{pm2_5}} },
    { "measurement": "Temperature","value": {{temperature}} },
    { "measurement": "Lat",      "value": {{Latitude}}},
    { "measurement": "Lon",      "value": {{Longitude}}}
  ]
}
```

Nota. Los datos de los sensores se combinan en el payload, creando un solo mensaje.

Para satisfacer el requerimiento funcional RF13, se implementó un servidor web embebido en el dispositivo con el fin de poder cambiar el campo de la localización de forma ágil y no recurrir a cargar un script en cada toma de datos. El servidor web es accesible desde el

dispositivo o red que esté vinculado, la configuración de este servidor se detalla en el código de la figura 36 y se visualiza con html sencillo con el ingreso de datos a través de una solicitud post como se muestra en la figura 37.

Figura 36

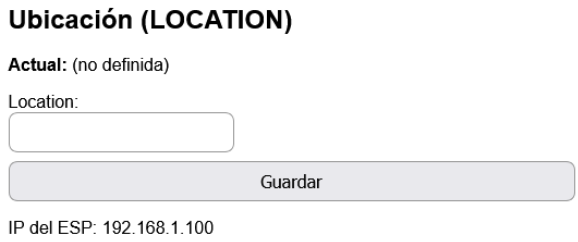
Fragmento de código de la configuración del servidor web

```
// ===== Web/EEPROM (LOCATION) =====
void saveLocationToEEPROM(const String& loc) {
  char buf[LOCATION_MAX]; memset(buf, 0, sizeof(buf));
  loc.substring(0, LOCATION_MAX - 1).toCharArray(buf, LOCATION_MAX);
  EEPROM.put(EEPROM_ADDR_LOC, buf);
  EEPROM.commit();
}

void loadLocationFromEEPROM() {
  char buf[LOCATION_MAX]; EEPROM.get(EEPROM_ADDR_LOC, buf);
  buf[LOCATION_MAX - 1] = '\0';
  if ((uint8_t)buf[0] == 0xFF) buf[0] = '\0';
  if (buf[0] != '\0') LOCATION = String(buf);
}

String htmlIndex() {
  String s;
  s  = F("<!doctype html><html><head><meta charset='utf-8'><meta name='viewport' content
  s += F("<title>Device Config</title><style>body{font-family:sans-serif;max-width:520px
  s += F("<h2>Ubicación (LOCATION)</h2><p><b>Actual:</b> ");
  s += (LOCATION.length() ? LOCATION : String("(no definida)"));
  s += F("</p><form method='POST' action='/set'>");
  s += F("<label>Location:<br><input name='location' required value='");
  String esc = LOCATION; esc.replace("\"","&quot;"); s += esc;
  s += F("'></label><button type='submit'>Guardar</button></form><p>IP del ESP: ");
  s += WiFi.localIP().toString();
  s += F("</p></body></html>");
  return s;
}

void handleRoot(){ server.send(200, "text/html", htmlIndex()); }
void handleSet(){
  String loc = server.hasArg("location")? server.arg("location") : server.arg("loc");
  loc.trim(); if (loc.length() > 0) { LOCATION = loc; saveLocationToEEPROM(LOCATION); }
  server.send(200, "text/html", "<html><body style='font-family:sans-serif'><h3>Guardado
}
```

Nota. Este servidor permite seleccionar el campo de localización con el cual se enviarán los datos.

Figura 37

Captura del formulario HTML para ingresar el campo de localización



Nota. Formulario HTML para el ingreso del campo location (destinado para el nombre de la ruta).

Para más detalles de la configuración del sensor, se almacenó el script en el repositorio de Github en el apartado de utils.

5.5 Fase de Pruebas

5.5.1 Pruebas Funcionales

Estas pruebas permiten comprobar el funcionamiento a partir de la realización de escenarios comunes que se esperan por parte del usuario, con base en los requisitos funcionales.

Siguiendo esta explicación y una plantilla para la creación y ejecución de estos casos de prueba, se genera la siguiente tabla:

Tabla 6

Formato de pruebas funcionales

Categoría	Descripción	Pasos para la prueba	Condiciones requeridas	Resultado Esperado	Estatus
Funcional	Filtrado de rutas	Se interactúa con el filtrado de rutas	El usuario se encuentra en la página del dashboard y tiene desplegado el sidebar	Los datos de cada ruta seleccionada aparecen	OK
Funcional	Filtrado por fecha	Se modifica el rango de la fecha	El usuario se encuentra en la página del dashboard y tiene desplegado el sidebar	Los datos se filtran acordemente por la fecha	OK
Funcional	Filtrado de contaminantes	Se interactúa con el filtrado de contaminantes	El usuario se encuentra en la página del dashboard y tiene desplegado el sidebar	Los datos se filtran acordemente por contaminante	OK

Funcional	Actualización automática	No se requieren pasos	El usuario se encuentra en la página del dashboard	Los datos se actualizan cada cinco segundos	OK
Funcional	Actualización manual	Se presiona el botón para actualizar los datos	El usuario se encuentra en la página del dashboard	Los datos se actualizan	OK
Funcional	Gráficas de compañía	Se interactúa con las gráficas al posterior del mapa	El usuario se encuentra en la página del dashboard	Funciona sin problema	OK
Funcional	Sidebar	Se presiona el botón para desplegar o minimizar el sidebar	El usuario se encuentra en la página del dashboard	El sidebar se despliega o minimiza correctamente	OK
Funcional	Leyenda	Se presiona el botón de mostrar u ocultar leyenda	El usuario se encuentra en la página del dashboard	La tabla de la leyenda se oculta o despliega de manera apropiada	OK

Nota. Tabla para documentar el estado de las pruebas funcionales.

5.5.2 Pruebas No Funcionales

Estas pruebas permiten comprobar el funcionamiento a partir de la realización de escenarios comunes que se esperan por parte del usuario, con base a los requisitos funcionales.

Siguiendo esta explicación y una plantilla para la creación y ejecución de estos casos de prueba, se genera la siguiente tabla:

Tabla 7

Formato de pruebas no funcionales

Categoría	**Descripción**	**Pasos para la prueba**	**Condiciones requeridas**	**Resultado Esperado**	**Estatus**
Responsive	Todo el	Abrir la	No hay	En todos los	OK

	contenido se puede visualizar correctamente independiente del dispositivo	página del dashboard en un computador, un celular y una tableta	condiciones requeridas	dispositivos se puede ver el sidebar, el mapa y las gráficas	
Responsive	Transición correcta entre modo landscape y vertical en dispositivos móviles	En dispositivo móvil y tableta, cambiar la orientación del dispositivo	La opción de orientación no está bloqueada en el dispositivo	El cambio de orientación no lleva errores y el contenido se ajusta correctamente	OK
Responsive	El tamaño de letra es apropiado para todos los dispositivos	Abrir la página del dashboard en un computador, un celular y una tableta	No hay condiciones requeridas	El tamaño de letra es apropiado	OK
Responsive	Las gráficas en flexbox se ajustan correctamente dependiendo en el tamaño de pantalla	Abrir la página del dashboard en un computador, un celular y una tableta	No hay condiciones requeridas	Dependiendo de la resolución horizontal, las gráficas pasan de estar orientadas horizontalmente a verticalmente	OK
Responsive	La navegación es posible en dispositivos móviles	Abrir la página del dashboard en un celular y una tableta	No hay condiciones requeridas	Es posible navegar todo el contenido en dispositivos móviles	OK
Responsive	El botón del sidebar se puede identificar en todo tipo de dispositivo	Abrir la página del dashboard en un computador, un celular y una tableta	No hay condiciones requeridas	En todo dispositivo se puede visualizar el botón para desplegar o minimizar el sidebar	OK
Responsive	El sidebar funciona correctamente en	Abrir la página del dashboard en	No hay condiciones requeridas	En todo dispositivo se puede desplegar y minimizar el	OK

	todos los dispositivos	un computador, un celular y una tableta	sidebar usando el botón

Nota. Tabla para documentar el estado de las pruebas no funcionales.

Experiencia UX. Para evidenciar la intuitividad de la interfaz del dashboard se llevó a cabo una encuesta por medio de Google Forms dirigida a usuarios finales adeptos al uso de dispositivos modernos y sin previo conocimiento del sistema donde se le proporcionó a cada individuo un conjunto de instrucciones por realizar y posteriormente calificar de 1 a 5 (Siendo 1 un valor poco favorable y 5 un valor altamente favorable) la facilidad de uso de cada caso.

Se tomó en cuenta el tipo de dispositivo de cada usuario con el fin de medir la usabilidad tanto en un escritorio cómo en un móvil.

Los resultados, los cuales se pueden evidenciar en el Apéndice D, así como una foto de la actividad en el Apéndice E, fueron principalmente positivos, en móvil el promedio de los puntajes fue de 4.8 y en escritorio de 5, permitiendo evidenciar que la experiencia UX es conveniente para los usuarios, independiente del dispositivo.

Escalabilidad. El sistema, al estar basado en el proyecto Smart Campus UIS, se contemplan opciones de escalabilidad basada en el consumo de los datos consultados a través de InfluxDB, es decir, de manera sencilla es posible adaptar nuevos sensores para consumir a través de Smart Campus UIS, y a su vez procesar la información en el dashboard.

5.5.3 Pruebas de Simulación y Captura de Datos

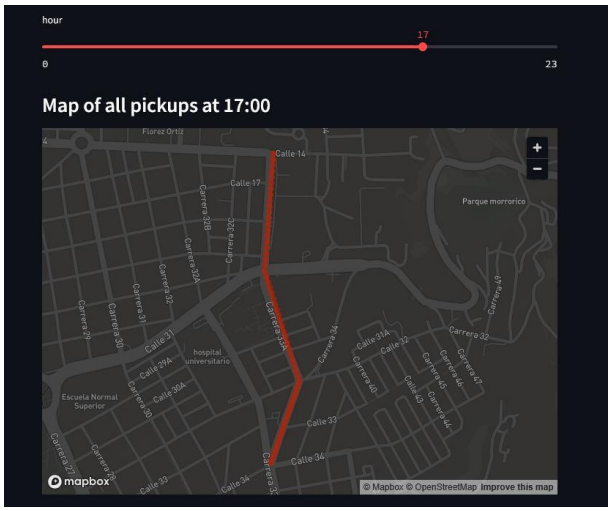
Estas pruebas permiten comprobar el funcionamiento a partir de la realización de escenarios comunes que se esperan por parte del usuario, con base en los requerimientos

funcionales. Se consideraron tres etapas del proyecto que fueron escalando de acuerdo con el avance del proyecto, como la plataforma Smart Campus UIS:

Fase Temprana del Proyecto. Para esta fase, se utilizó un script de Python sencillo para poder interpolar datos de puntos geolocalizados del AMB, con estos puntos fueron generados datos aleatorios exportados a un archivo plano legible para experimentación inicial con los componentes del sistema como se muestra en la figura 38.

Figura 38

Captura del prototipo inicial con datos en un mapa 2D



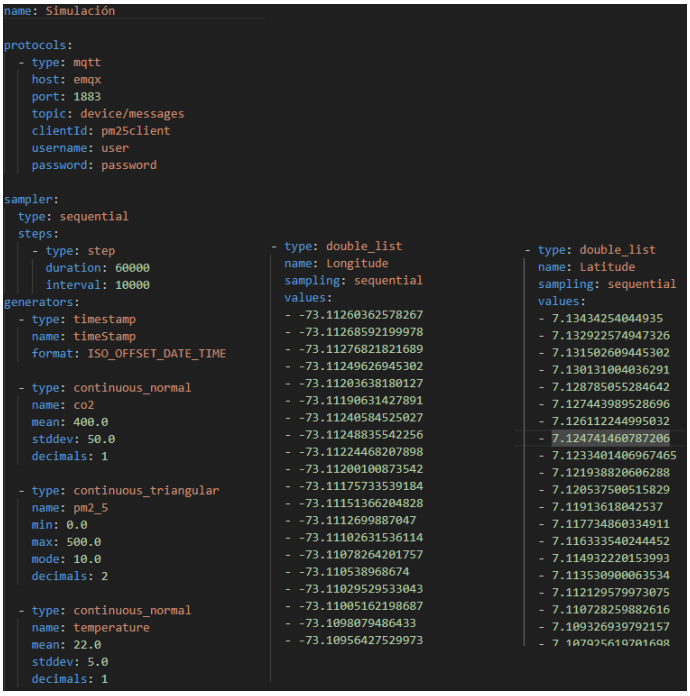
Nota. Prototipo inicial experimental con datos extraídos de un archivo plano.

Fase de Simulación. Al avanzar en los requerimientos funcionales, con base en la herramienta Mocker desarrollada por Peñaloza Torres y Robayo Nieto (2025), una solución práctica para la definición de escenarios de simulación de sensores fue posible realizar pruebas en tiempo real a través de datos simulados basados en distribuciones estadísticas y listas predefinidas.

Para personalizar la simulación se utilizó el DSL (Domain Specific Language) documentado para el proyecto, como se muestra en la figura 39.

Figura 39

Captura del fragmento de código del DSL para Mocker



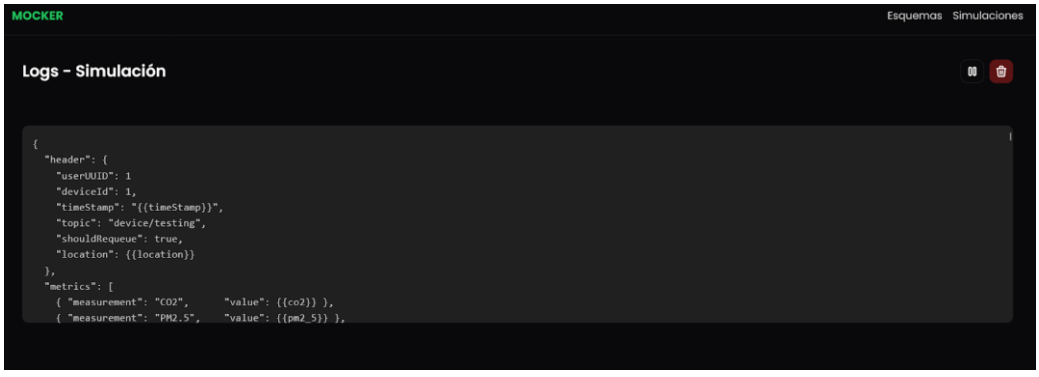
```
name: Simulación

protocols:
  - type: mqtt
    host: emqx
    port: 1883
    topic: device/messages
    clientId: pm25client
    username: user
    password: password

sampler:
  type: sequential
  steps:
    - type: step
      duration: 60000
      interval: 10000
generators:
  - type: timestamp
    name: timeStamp
    format: ISO_OFFSET_DATE_TIME

  - type: continuous_normal
    name: co2
    mean: 400.0
    stddev: 50.0
    decimals: 1

  - type: continuous_triangular
    name: pm2_5
    min: 0.0
    max: 500.0
    mode: 10.0
    decimals: 2

  - type: continuous_normal
    name: temperature
    mean: 22.0
    stddev: 5.0
    decimals: 1

- type: double_list
  name: Longitude
  sampling: sequential
  values:
  - -73.11260362578267
  - -73.11268592199978
  - -73.11276821821689
  - -73.11249626945302
  - -73.11203638180127
  - -73.11190631427891
  - -73.11240584525027
  - -73.11248835542256
  - -73.11224468207898
  - -73.11200100873542
  - -73.11175733539184
  - -73.11151366204828
  - -73.1112699887047
  - -73.11102631536114
  - -73.11078264201757
  - -73.110538968674
  - -73.11029529533043
  - -73.11005162198687
  - -73.1098079486433
  - -73.10956427529973

- type: double_list
  name: Latitude
  sampling: sequential
  values:
  - 7.13434254044935
  - 7.132922574947326
  - 7.131502609445302
  - 7.130131004036291
  - 7.128785055284642
  - 7.127443989528696
  - 7.126112244995032
  - 7.124741460787206
  - 7.1233401406967465
  - 7.121938820606288
  - 7.120537500515829
  - 7.11913618042537
  - 7.117734860334911
  - 7.116333540244452
  - 7.114932220153993
  - 7.113530900063534
  - 7.112129579973075
  - 7.110728259882616
  - 7.109326939792157
```

Nota. Campos personalizados para la simulación de datos de sensores por medio de Mocker.

A su vez, para establecer la estructura del mensaje, se define una plantilla en un plano de texto con la asignación de los campos, muy similar a la estructura mostrada en la figura 40.

Figura 40

Captura de logs de Mocker corriendo la simulación



Nota. Logs de Mocker mostrando la ejecución de la simulación de datos personalizada.

Siguiendo esta explicación y una plantilla para la creación y ejecución de estos casos de prueba, se genera la siguiente tabla:

Tabla 8

Formato de pruebas de captura de datos

Categoría	Descripción	Pasos para la prueba	Condiciones requeridas	Resultado Esperado	Estatus
Captura de Datos	Al seleccionar la ruta para la captura de datos, se asigna correctamente en el mensaje	Seleccionar diferentes rutas y enviar una multitud de mensajes	No hay condiciones requeridas	Los mensajes se envían con la ruta seleccionada	OK
Captura de Datos	Después de capturar datos se pueden visualizar en el dashboard	Al mismo tiempo que se envían mensajes, actualizar el dashboard y ver los cambios	No hay condiciones requeridas	Datos recién agregados aparecen en el dashboard	OK

Nota. Tabla para documentar el estado de las pruebas de captura de datos.

Fase de Pruebas Local: En este escenario se utiliza la infraestructura de Smart Campus UIS con EMQX corriendo en un equipo local (Docker/host). Para permitir que los dispositivos conectados por wifi fuera de la red local y puedan publicar mensajes, se estableció un túnel seguro con ngrok que expone de forma controlada el servicio MQTT de EMQX a Internet, sin requerir apertura directa de puertos en el router. Así, con esta configuración se permite la obtención de datos solamente con el dispositivo de forma remota y solo usando la conexión a la red del dispositivo móvil.

Como aspecto adicional, también se conectó a una VPN (Virtual Private Network) el host y el dispositivo móvil con el fin de poder conectar en tiempo real el sistema de información en la toma de datos en la ruta del transporte público.

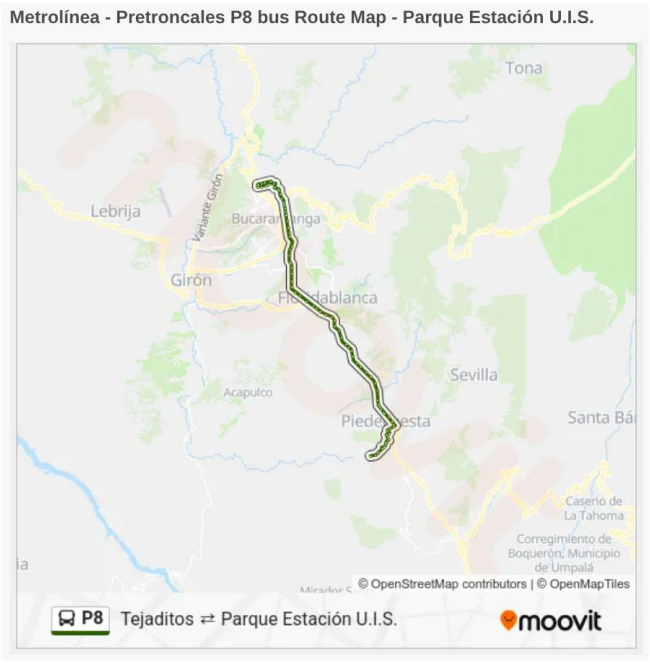
5.5.4 Definición de Rutas de Transporte

Para facilitar la captura de datos y la categorización correcta de los datos, antes de empezar a capturar datos se debe seleccionar la ruta correspondiente, y se puede filtrar por cada ruta de transporte.

Según el AMB, en Bucaramanga existen 56 rutas de transporte público, debido al alcance del proyecto y para las pruebas del dispositivo se hace la captura de datos en las siguientes rutas, que se escogen debido a su accesibilidad:

Figura 41

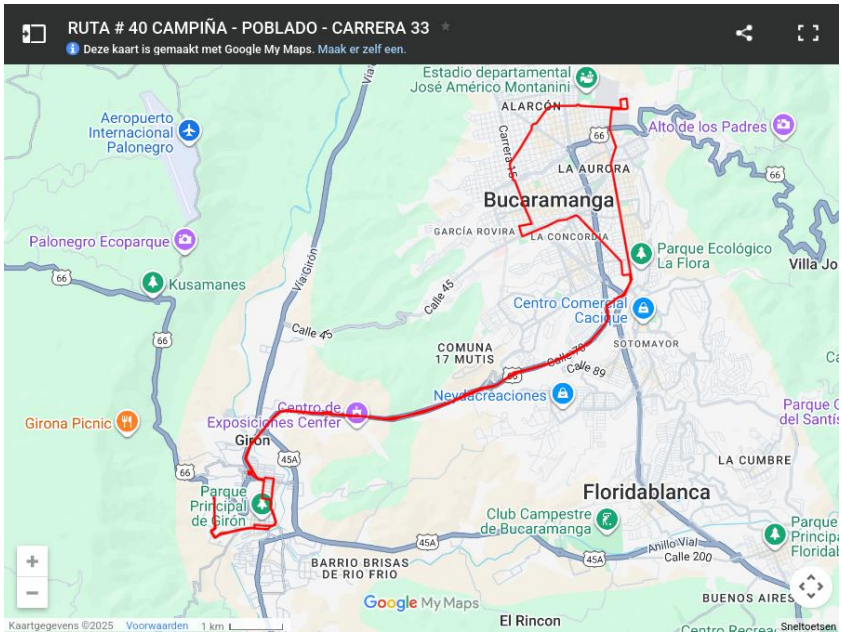
Captura del mapa oficial de la ruta de Metrolínea P8



Nota. Adaptado de Moovit. (s. f.). Metrolínea - Pretroncales - Schedules, Routes and Stops.

Figura 42

Captura del mapa oficial de la ruta de autobús número 40



Nota. Adaptado de AMB. (s. f.). Rutas Transporte Público Colectivo Complementario [Ruta #40].

Teniendo en cuenta estas rutas, se realizan pruebas en el entorno real para probar las capacidades del dispositivo y la conectividad con las redes de internet y los microservicios, así como la transmisión de datos constante sincronizada con el dashboard.

5.5.5 Recolección Real de Datos

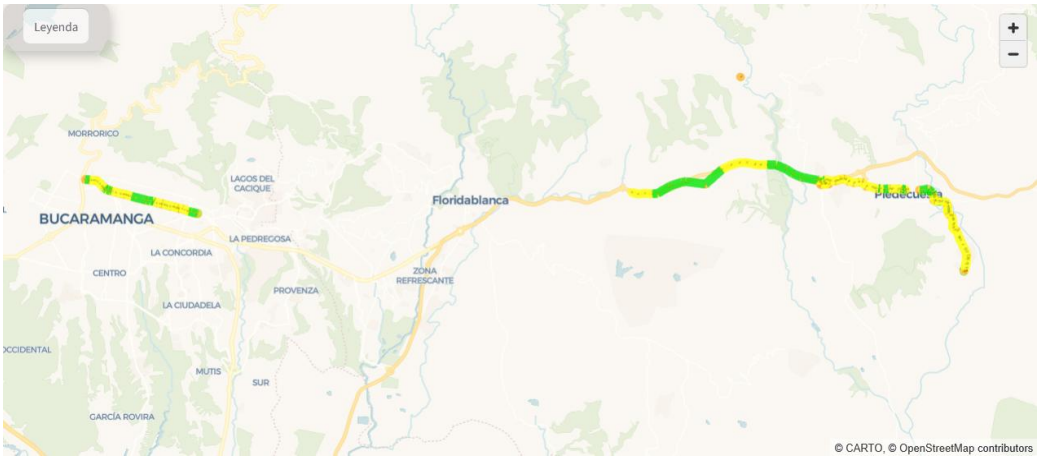
Para la recolección de datos reales, se hizo una conexión a través de un dispositivo móvil como un punto de acceso para el esp8266, en el cual se configuró previamente la dirección de ngrok donde se expone públicamente el servicio de EMQX para recolectar datos desde cualquier lugar.

Con la configuración previa, se procedió a la recolección de datos en las rutas especificadas como se muestran en los Apéndices B y C.

Después de capturar los datos de las dos rutas, dentro del dashboard se pueden visualizar con su concentración de PM2.5. Como se muestra en la figura 43 y 44, también se puede apreciar la similitud entre las rutas oficiales y las creadas a partir de datos reales en el dashboard.

Figura 43

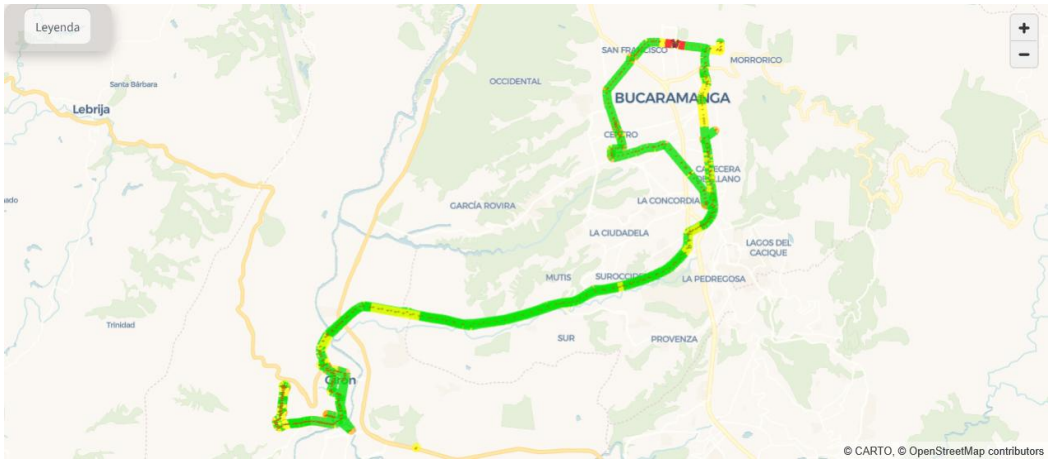
Captura del dashboard de la ruta de Metrolínea P8



Nota. Con el filtro de la ruta P8 activado se *visualiza* solo los datos de esta ruta, debido a una caída de conexión hay una sección faltante entre Bucaramanga y Floridablanca.

Figura 44

Captura del dashboard de la ruta de autobús número 40



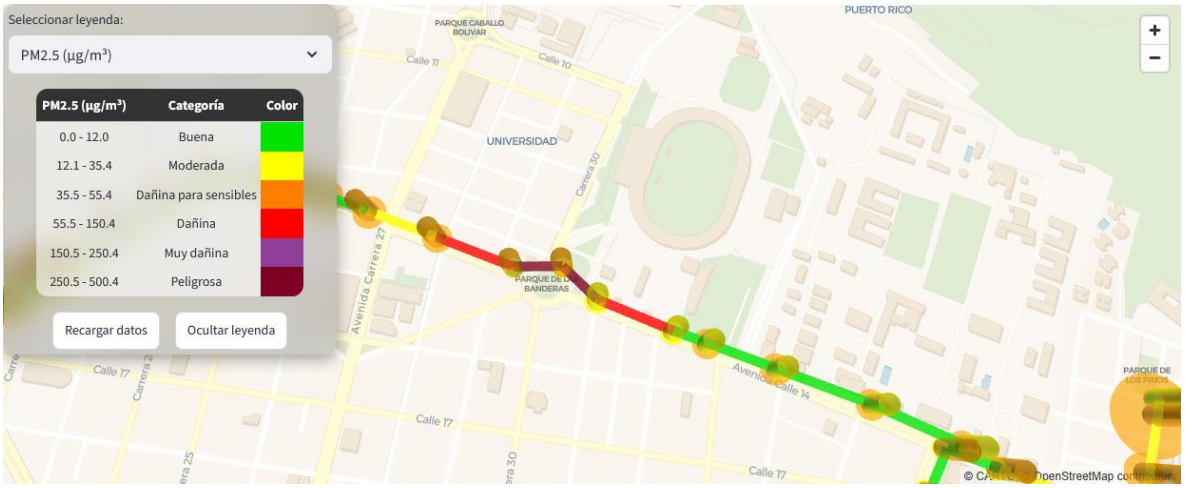
Nota. Con el filtro de la ruta 40 activado se visualiza solo los datos de esta ruta.

5.5.6 Análisis de Datos

Como fase final con base en los datos obtenidos, el sistema permitió realizar las siguientes observaciones:

Figura 45

Captura del dashboard de condiciones de alta exposición en la calle 14



Nota. La concentración de categoría Peligrosa se evidencia en el tramo del mapa.

En el mapa interactivo se identificaron condiciones de alta exposición a PM2.5 en la ruta Cootragas mientras se recorría la calle 14 entre la carrera 32 a la carrera 28.

Figura 46

Captura de sección del dashboard de métricas globales

Nota. Se evidencian los datos con mayores concentraciones y otras mediciones.

Aunque se identificara un pico de alta exposición en la ruta Cootragas, según los indicadores del sistema se indica que en promedio hubo una exposición mayor en la ruta Metrolínea P8.

Figura 47

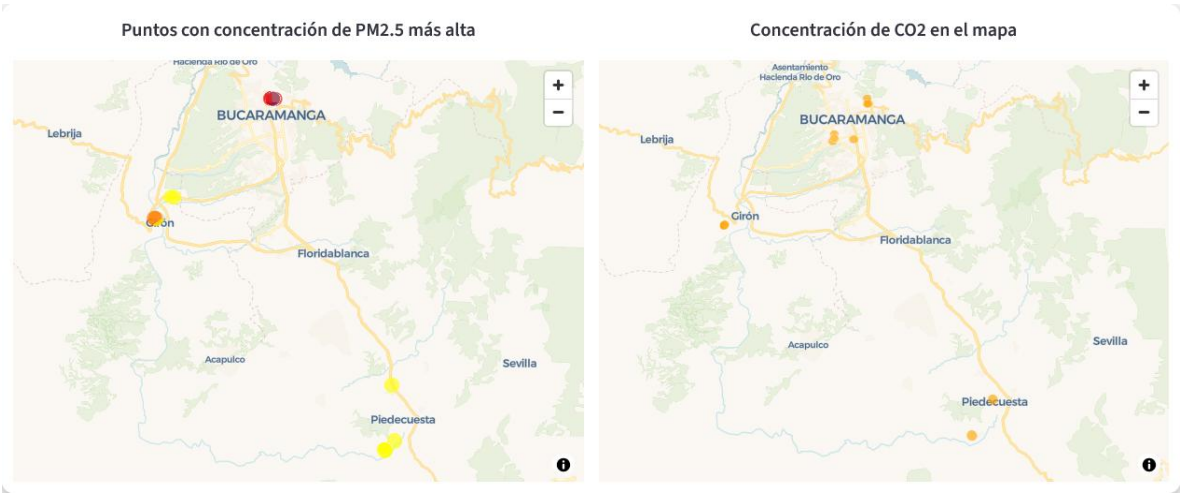
Captura del dashboard con distribución de clasificaciones a nivel global de PM2.5

Nota. Gráfica circular con distribución de clasificaciones de AQI basadas en PM2.5.

En promedio, entre todos los datos se evidencia concentraciones de PM2.5 con un 85.6% clasificadas como “Buena”, un 14.1% como “Moderada" y menos de 1% de “Dañina”, “Dañina para sensibles” y “Peligrosa”.

Figura 48

Captura del dashboard de mapas con valores máximos de concentraciones



Nota. La gráfica en la izquierda denota el PM2.5 y en la derecha el CO2.

También el sistema nos permitió identificar puntos de mayor concentración de PM2.5 y CO2 respectivamente en varios sectores de los municipios que conforman el AMB, pero principalmente en la ciudad de Bucaramanga.

Figura 49

Captura del dashboard de gráficas para análisis de datos

Nota. La gráfica izquierda denota la concentración por ruta y la derecha la evolución.

El sistema también nos permitió identificar en promedio las concentraciones de cada ruta, siendo principalmente la ruta de Metrolínea P8 y Cootragas de mayor contaminación, y las pruebas locales teniendo concentraciones mucho menores.

6. Conclusiones

El presente trabajo enmarcado dentro del macroproyecto Smart Campus UIS, consolida una propuesta tecnológica orientada al análisis de calidad del aire en rutas de transporte público del AMB. A lo largo de todas las fases de trabajo de investigación, diseño, implementación y pruebas, se logró el alcance de los objetivos propuestos basado en los siguientes aportes:

Una definición clara de requerimientos funcionales y no funcionales que permiten identificar una solución con respecto al análisis de contaminantes criterio, con una visualización que permite identificar patrones o puntos críticos de información espaciotemporal.

Una arquitectura soportada en el funcionamiento e integración de distintos componentes que soportan la escalabilidad y el mantenimiento, con flujos de información claros y adaptables que permiten la implementación de diversas soluciones.

La implementación de un prototipo funcional y robusto, capaz de gestionar el proceso desde la captura de datos, el procesamiento y el consumo por el usuario final, asegurando que la información sea confiable y aporte valor para la toma de decisiones.

Ejecución de pruebas en condiciones reales, realizadas en el medio de transporte más utilizado en el AMB. Estas pruebas permitieron obtener una medición objetiva del comportamiento de distintos parámetros en el interior de un autobús, evidenciando la exposición de la población a posibles condiciones críticas de contaminación.

Con base en los datos recolectados, también el sistema permitió realizar un análisis de los datos con su localización, promedio e indicadores de valores máximos y mínimos.

Por último, cabe resaltar que la ejecución del presente proyecto fue una gran oportunidad de integración y aplicación de todos los conocimientos adquiridos en el programa de Ingeniería de sistemas e informática. Esta experiencia no solo permitió afianzar habilidades técnicas, sino explorar soluciones a problemáticas reales en contextos locales que pueden ser fácilmente escalables a macroproyectos que involucren distintos aspectos de desarrollo e implementación tecnológica.

7. Recomendaciones y Trabajo a Futuro

Basándose en las experiencias y dificultades al desarrollar el proyecto, como recomendaciones a futuro se identifican cuatro puntos clave que mejoran y solidifican diversos aspectos del sistema:

Con la adición de un caché o empaquetado al dispositivo IoT, se podrá conseguir un envío de mensajes más robusto y resistente a pérdidas o errores de conexiones, ya que el dispositivo podría guardar los mensajes JSON en caso de fallo en la recepción de la información por los microservicios para luego enviar los datos nuevamente cuando las condiciones hayan mejorado, preservando de esta manera la calidad e integridad de las mediciones.

Teniendo en cuenta que el dispositivo IoT tomará datos desde un bus de transporte público, es recomendable una cubierta y un montaje en una placa de circuito que permita un diseño compacto y disimulado para mejorar la longevidad de los componentes y aportar protección de condiciones externas.

Es importante notar que, aunque el desarrollo del proyecto se hizo teniendo en cuenta sólo el uso interno, la posibilidad de desplegar a la nube tanto los microservicios, cómo la conexión del dispositivo y el dashboard.

Como punto final, gracias a la obtención y consumo de datos en tiempo real, se abre la posibilidad de poder enviar alertas a través de distintos medios a distintas entidades o usuarios en el momento en el que se mida una concentración de algún contaminante criterio que pueda ser potencialmente nocivo para el ambiente y el ser humano.

Referencias Bibliográficas

AMB. (s. f.). *Rutas Transporte Público Colectivo Complementario [Ruta #40]*. Recuperado el 10 de septiembre, 2025, de https://www.amb.gov.co/rutas-publico-colectivo-complementario/

Barrio Andrés, M., & Guglieri, A. (2020). *Internet de las cosas* (Segunda edición). Editorial REUS.

Buitrago Mesa, D. A., & Rodriguez, R. J. (2022). *Estado del arte del uso de sensores de bajo costo para el monitoreo y seguimiento de la calidad del aire por Material Particulado (PM10 y PM2.5) en el territorio colombiano* [Trabajo de grado especialización]. Universidad de Antioquia, Medellín, Colombia.

C., H. G. (2017). *MQTT essentials - A lightweight IoT protocol: Send and receive messages with the MQTT protocol for your IoT solutions*. Packt Publishing, Limited.

CDMB. (s. f.). *Fuentes móviles*. http://caracoli.cdmb.gov.co/cai/cai2/fmoviles.html

CDMB. (s. f.). *Red Aire*. http://caracoli.cdmb.gov.co/cai/cai2/redaire.html

Cirani, S., Ferrari, G., Picone, M., & Veltri, L. (2019). *Internet of Things: Architectures, Protocols and Standards* (First edition). Wiley.

IDEAM. (s. f.). *Atmósfera*. www.ideam.gov.co/web/tiempo-y-clima/atmosfera

IDEAM. (s. f.). *Calidad aire*. http://www.ideam.gov.co/web/siac/calidadaire

IDEAM. (s. f.). *Contaminación Atmosférica*. www.ideam.gov.co/web/contaminacion-y-calidad-ambiental/contaminacion-atmosferica

Jiménez, H., Cárcamo, E., & Pedraza, G. (2020). Plataforma software extensible para campus inteligente basada en microservicios. *Dialnet*. https://dialnet.unirioja.es/servlet/articulo?codigo=8576444

Laoyan, S. (2024). *Qué es la metodología waterfall y cuándo utilizarla*. Asana. https://asana.com/es/resources/waterfall-project-management-methodology

McGrath, M. J., & Scanaill, C. N. (2013). *Sensor Technologies*. Apress eBooks. https://doi.org/10.1007/978-1-4302-6014-1

Ministerio del Ambiente. (2017). *Resolución 2254 de 2017*. www.minambiente.gov.co/wp-content/uploads/2021/10/Resolucion-2254-de-2017.pdf

Moovit. (s. f.). *Metrolínea - Pretroncales - Schedules, Routes and Stops*. Recuperado el 10 de septiembre, 2025, de https://moovitapp.com/index/en/public_transit-lines-Bucaramanga-4324-926178

Peñaloza Torres, C. D., & Robayo Nieto, M. (2025). *Diseño de una solución para la definición de escenarios de simulación de sensores en la plataforma Smart Campus UIS* [Trabajo de grado, Universidad Industrial de Santander]. Universidad Industrial de Santander.

Schwaber, K., & Sutherland, J. (2020). *La guía definitiva de Scrum: las reglas del juego*.

Sebastian, P. L., & Pablo, C. P. J. (2023). *Sistema de información basado en IoT para medición de la calidad del aire y seguimiento de rutas en sistemas de transporte público* [Trabajo de grado, Universidad Industrial de Santander]. Universidad Industrial de Santander.

Streamlit. (s. f.). *Streamlit documentation*. https://docs.streamlit.io/

U.S. Environmental Protection Agency. (2024). *Final updates to the Air Quality Index (AQI) for particulate matter–Fact Sheet and Common Questions*. https://www.epa.gov/system/files/documents/2024-02/pm-naaqs-air-quality-index-fact-sheet.pdf

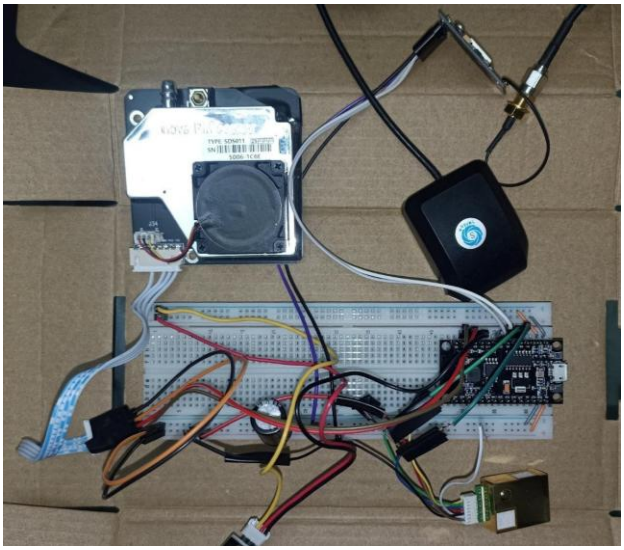
United Nations Environment Programme. (s. f.). *¿Cómo se mide la calidad del aire?* https://www.unep.org/es/noticias-y-reportajes/reportajes/como-se-mide-la-calidad-del-aire

Van Rossum, G. (1997). *Comparing Python to other languages*. Python Software Foundation. https://www.python.org/doc/essays/comparisons/

Waseem, M. (2022). *Building IRIS Responsive dashboard with Python Flask Web Framework*. https://user-images.githubusercontent.com/18219467/155213244-803fecb4-5c76-4805-9bb0-ca5a11411129.png

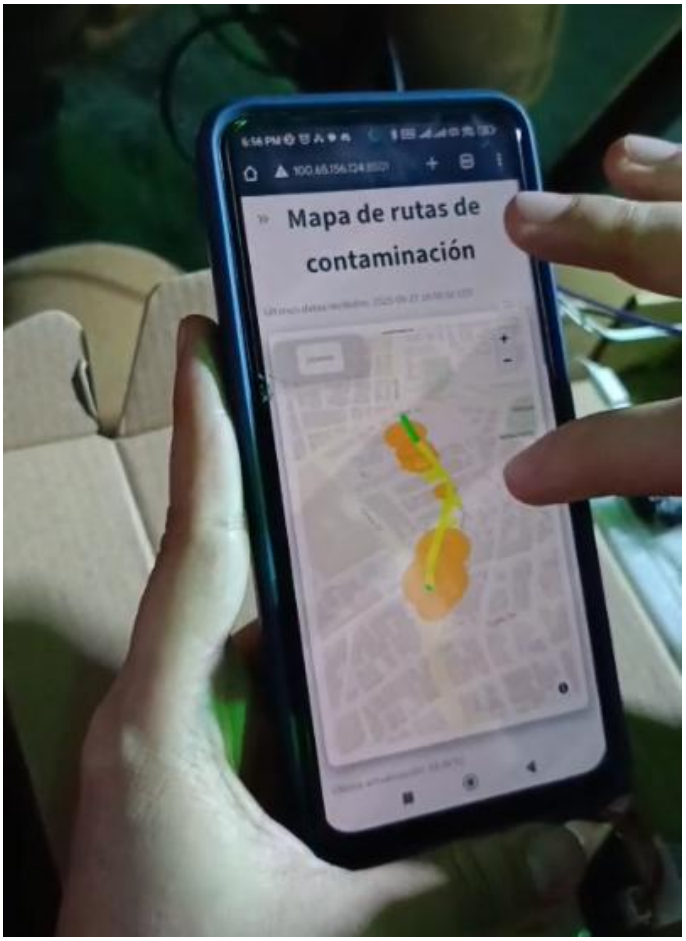
Apéndices

Apéndice A. Foto del montaje físico del dispositivo IoT



Apéndice B. Foto de la toma de datos con visualización en tiempo real de los datos desde un dispositivo móvil en ruta de Cootragas.

Apéndice C. Foto de la Toma de datos con visualización en tiempo real de los datos desde un dispositivo móvil en ruta de Metrolínea P8.



Apéndice D. Captura de los resultados de la encuesta UX a usuarios finales.

Encuesta UX - Interfaz de Dashboard Calidad del Aire (respuestas)

Guardado en Drive

Archivo Editar Ver Insertar Formato Datos Herramientas Extensiones Ayuda

Form_Responses

	Marca temporal	Ingresa tu	Dispositiv	Califique	Califique	Califique	Califique	Califique	Califique	Califique	Califique	Califique	Califique
2	9/12/2025 19:44:03	Silvia	Escritorio	5	5	5	5	5	5	5	5	5	5
3	9/12/2025 19:54:54	Juan	Escritorio	5	5	5	5	5	5	5	5	5	5
4	9/12/2025 13:27:48	Daniel	Móvil	4	5	5	4	5	5	5	5	5	5
5	10/12/2025 11:35:36	Chloé	Móvil	4	5	5	5	5	5	5	5	5	4
6													
7			Promedio Móvil		4,8								
8			Promedio Escritorio		5								
9													

Apéndice E. Foto de la actividad de encuesta UX a usuarios finales.