UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

MODELADO PREDICTIVO CON TÉCNICAS DE CIENCIA DE DATOS E INTERNET DE LAS COSAS PARA MONITOREO DE CALIDAD DEL AIRE EN EL CASCO URBANO DEL DISTRITO DE SAN SALVADOR

PRESENTADO POR:
CARLOS MANUEL RAMOS ZEPEDA
JONATHAN SALOMÓN REYES DEL CID
LUIS RODRIGO ROMERO ESCOBAR

PARA OPTAR POR EL TÍTULO DE:
INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, ENERO DE 2026

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. JUAN ROSA QUINTANILLA

SECRETARIO GENERAL:

LIC. PEDRO ROSALÍO ESCOBAR CASTANEDA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

MSC. LUIS SALVADOR BARRERA MANCÍA

SECRETARIO:

ARQ. RAÚL ALEXANDER FABIÁN ORELLANA

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR:

ING. WERNER DAVID MELÉNDEZ VALLE

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

TRABAJO DE GRADUACION PREVIO A LA OPCION AL GRADO DE:
INGENIERO ELECTRICISTA

TITULO:

MODELADO PREDICTIVO CON TÉCNICAS DE CIENCIA DE DATOS E INTERNET DE LAS COSAS PARA MONITOREO DE CALIDAD DEL AIRE EN EL CASCO URBANO DEL DISTRITO DE SAN SALVADOR

PRESENTADO POR:

CARLOS MANUEL RAMOS ZEPEDA
JONATHAN SALOMÓN REYES DEL CID
LUIS RODRIGO ROMERO ESCOBAR

TRABAJO DE GRADUACION APROBADO POR:

DOCENTE ASESOR:

PhD. CARLOS OSMIN POCASANGRE JÍMENEZ

SAN SALVADOR, ENERO DE 2026

TRABAJO DE GRADUACION APROBADO POR:

DOCENTE ASESOR:

PhD. CARLOS OSMIN POCASANGRE JÍMENEZ

NOTA Y DEFENSA FINAL

En esta fecha, jueves 4 de diciembre de 2025, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 10:00 a.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Werner David Meléndez Valle
   Director

   Firma

2. MSc. José Wilber Calderón Urrutia
   Secretario

   Firma

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

- DR. CARLOS OSMIN POCASANGRE JIMÉNEZ
  (Docente Asesor)

  Firma

- MSC. JOSE WILBER CALDERON URRUTIA

  Firma

- ING. LUIS ERNESTO ESCOBAR BRIZUELA

  Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

MODELADO PREDICTIVO CON TÉCNICAS DE CIENCIA DE DATOS E INTERNET DE LAS COSAS PARA MONITOREO DE CALIDAD DEL AIRE EN EL CASCO URBANO DEL DISTRITO DE SAN SALVADOR

A cargo de los Bachilleres:

- RAMOS ZEPEDA CARLOS MANUEL
- REYES DEL CID JONATHAN SALOMÓN
- ROMERO ESCOBAR LUIS RODRIGO

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 9.4
(Nueve punto cuatro)

AGRADECIMIENTOS

Agradezco a Dios por concederme la fortaleza, la sabiduría y la constancia necesarias para culminar esta etapa de mi formación académica, así como por guiar mi camino durante todo el proceso.

Agradezco a mi madre, Claudia María Zepeda Castañeda, quien hoy se encuentra en el cielo, por su ejemplo de vida, sus valores y enseñanzas, que han guiado cada paso de mi formación personal y profesional. Todo el esfuerzo realizado a lo largo de este camino ha sido un homenaje a su nombre y a su profundo deseo de verme convertido en ingeniero. Desde el primer día de mi vida universitaria me acompañó con amor incondicional, apoyo constante y fe en mis capacidades, permaneciendo a mi lado hasta el último día de su vida. Este logro es reflejo de su entrega y de la huella imborrable que dejó en mí.

A mi familia, especialmente a mis abuelos Manuel Zepeda y Ana Gladis Castañeda, por el respaldo y la orientación brindados a lo largo de mi vida; a mi padre, Juan Carlos, por sus enseñanzas y ejemplo; a mis tíos Verónica, Guadalupe, Manuel, Julio y Carmencita, por su constante acompañamiento y palabras de aliento durante mi formación académica; a mis hermanos Lisseth y César, por su compañía y apoyo; y a Maricel Castillo, por el apoyo incondicional brindado durante este proceso.

A mi amada novia, Jennifer Portillo, le expreso mi más profundo agradecimiento por su amor, comprensión y paciencia incondicional. Su apoyo constante, su disposición para escucharme y su compañía en cada momento de este proceso fueron un refugio y una fortaleza, convirtiéndose en un pilar fundamental para alcanzar la culminación de esta importante etapa de mi vida

Agradezco a mi asesor, PhD. Carlos Osmín Pocasangre Jiménez, por la orientación académica y el acompañamiento brindado en la elaboración de esta tesis. De igual manera, extiendo mi reconocimiento a los catedráticos de la Facultad de Ingeniería y Arquitectura por los conocimientos impartidos, así como a Reina Videz por su colaboración y valiosos consejos.

Carlos Ramos

AGRADECIMIENTOS

Agradezco a Dios primeramente por permitirme culminar este largo recorrido, guiándome siempre con salud, perseverancia, sabiduría y fortaleza para afrontar los distintos desafíos que surgieron.

Expreso mi más sincero agradecimiento a mi madre Beufali Del Cid y mi padre Daniel Reyes, quienes me han apoyado y guiado de manera incondicional, por sus consejos que han sido un pilar fundamental para mi crecimiento personal, y por su compañía constante en los momentos difíciles a lo largo de mi vida. A mis hermanas Abigail, Astrid y Heydi; y a mi hermano Juan, por sus palabras de ánimo y que puedo contar con ellos en cualquier momento.

A Jesús Vasquez, cuya amistad sincera fue un pilar de apoyo durante estos años de estudio, por haberme ayudado cuando lo he necesitado, siendo un privilegio haber contado con su presencia en la universidad.

Expreso mi más sincera gratitud a mis jefes al Piloto Aviador Francisco Samayoa y a la Arquitecta Lidia Liang. Gracias por otorgarme el tiempo y el espacio necesario para la culminación de este proyecto, por la confiaza que fue depositada en mi.

Mi agradecimiento al PhD. Carlos Pocasangre e MSc. Omar Flores-Cortez, los cuales a través de sus conocimientos y enseñanzas se logró ejecutar y culminar este trabajo, a pesar de las dificultades y problemáticas que se presentaron. A Reina Vides por su orientaciòn y su ayuda; así como a la Facultad de Ingeniería y Arquitectura que por medio de su financiamiento hice participación en el congreso de CONESCAPAN 2025.

Finis coronat opus.

Salomón Reyes

AGRADECIMIENTOS

Doy gracias a Dios todopoderoso y eterno, quien en su infinita bondad me brindó toda sabiduría y diligencia, paciencia y perseverancia a lo largo de todo este proceso de tesis, sin las cuales no podría haber logrado avanzar ni siquiera un poco.

Agradezco a mi amada madre Cecilia Escobar y a mi querido padre Raul Romero, quienes desde pequeño me han dado la oportunidad de estudiar y me han brindado los ánimos y el apoyo constante para poder llegar hasta este punto de culminar mi carrera universitaria. Pocos no fueron los momentos durante la universidad en los que me encontraba desanimado, desesperado o estresado por problemas, y en cada uno de ellos, estuvieron ahí para poder darme ese amor y consuelo que solo una madre y un padre pueden dar. Palabras de ánimo, consejos, compañía y plato de comida caliente eran cosas que no hacían falta cuando más las necesitaba.

Agradezco a mi amada novia Josselyn Alvarenga, quien estuvo a mi lado desde el bachillerato, quien se convirtió en mi motivación, mi visión para el futuro y la razón por la cual puse mi empeño en terminar mis estudios. Ella me enseñó a amar y a buscar a Dios, me enseñó a tener paciencia en Él. Me prestó un oído a quien contarle mis preocupaciones, mis ambiciones, mis metas, mis alegrías y mis tristezas, me dió consejos para saber actuar en mi día a día.

Agradezco a nuestro asesor de tésis, el PhD, Carlos Pocasangre quien nos brindó la orientación y, nos dió la oportunidad de aportar nuestros conocimientos a una investigación con el potencial de ser aplicado en la ciencia meteorológica de nuestro país. Nos permitió conocer además a dos expertos quienes también nos brindaron guía y asesoría durante este proyecto.

Ad Maiorem Dei Gloriam.

Luis Romero

ÍNDICE

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capitulo 1. PERFIL DEL PROYECTO

1.1. Introducción

La afectación en la calidad del aire es uno de los principales desafíos ambientales y de salud pública a nivel nacional. En respuesta a la problemática, se realiza el presente trabajo, el cual busca hacer un estudio de las condiciones actuales del aire en zonas estratégicas del país, modernizando las estaciones de bajo costo anteriormente realizadas y mejorandolas para una mayor efectividad de medición, además de aumentar las variables de estudio.

Se presenta el diseño de una arquitectura IoT que conecta la estación móvil con una infraestructura en la nube para procesar, almacenar y visualizar los datos en tiempo real.

Con los datos registrados se implementan técnicas de preprocesamiento y modelos de aprendizaje automático basados en series de tiempo, con el fin de pronosticar concentraciones futuras de contaminantes atmosféricos. Estos resultados permiten calcular el Índice de Calidad del Aire (ICA) conforme a los estándares establecidos por el MARN.

1.2. Descripción del tema

Este proyecto de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador (UES) busca desarrollar un sistema de monitoreo de calidad del aire innovador y de bajo costo, complementando el existente en el Ministerio de Medio Ambiente y Recursos Naturales (MARN). El sistema ha evolucionado a lo largo de varias versiones desarrolladas en proceso de graduación desde 2021: a) V1.0: Dispositivo loT fijo que mide partículas contaminantes (PM) y temperatura, con datos almacenados y visualizados en una plataforma web; b) V2.0: Incorporación de conectividad GSM y sensores avanzados para medir $PM_{1.0}$, $PM_{2.5}$, PM_{10}, formaldehído ($HCHO$). temperatura y humedad; c) V3.0: Introducción de geolocalización (GPS) para monitoreo móvil y generación de mapas de calidad del aire; finalmente d) V3.5 implementación de una plataforma loT centralizada (Thingsboard) para monitoreo en tiempo real de múltiples dispositivos.

Esta propuesta (V4.0) busca mejorar las capacidades tecnológicas y analíticas del sistema, incluyendo: Integración de nuevos sensores: Medición de ozono (O_3), monóxido de carbono (CO), dióxido de carbono (CO_2) y compuestos orgánicos volátiles totales ($TVOC$), además de los sensores existentes. Actualización del hardware: Rediseño del sistema de procesamiento y conectividad loT con tecnologías actuales y de bajo costo. Analítica avanzada de datos:

Implementación de técnicas de ciencia de datos y aprendizaje por computadora para procesamiento, análisis y predicción de contaminantes.

1.3. Objetivos

1.3.1. Objetivo general

1. Implementar un modelado predictivo con técnicas de ciencia de datos e Internet de las Cosas para monitoreo de calidad del aire en el casco urbano del Distrito de San Salvador.

1.3.2. Objetivos específicos

1. Evaluar las nuevas tecnologías aplicables al monitoreo de calidad de aire que cumplan con las normativas ambientales en Centroamérica, índice Centroamericano de Calidad de Aire (ICA).
2. Desarrollar un prototipo electrónico con tecnologías de procesamiento reciente y conectividad loT, que garantice un bajo costo, eficiencia energética y escalabilidad para su implementación en múltiples ubicaciones.
3. Elaborar una plataforma loT que permita el monitoreo en tiempo real de múltiples dispositivos, facilitando la visualización y gestión de los datos recopilados.
4. Realizar pruebas de campo en zonas geográficas específicas del casco urbano del distrito de San Salvador, recopilando datos de calidad del aire y validando la precisión y eficacia del sistema propuesto en condiciones reales.
5. Aplicar técnicas de preprocesamiento, aprendizaje por computadora y análisis estadístico para predecir concentraciones futuras de contaminantes y calcular el ICA según los estándares establecidos por el MARN, contribuyendo a la toma de decisiones informadas en materia ambiental.

1.4. Alcances

1. Prototipo funcional del dispositivo de monitoreo de calidad de aire: El dispositivo contará con sensores de PM_{10}, $PM_{2.5}$, CO, O_3 temperatura, humedad y formaldehído. Con capacidad de registrar coordenadas gps y conectividad a internet móvil.
2. Diseño y construcción de un prototipo electrónico basado en el microcontrolador ESP32, que integre sensores para medir O_3, CO, CO_2, $TVOC$, $PM_{1.0}$, $PM_{2.5}$, PM_{10}, $HCHO$,

temperatura y humedad, con conectividad GSM y GPS para la transmisión de datos y geolocalización.

3. Configuración de una plataforma IoT utilizando el servicio en línea Thingsboard, compatible con las versiones anteriores del sistema (V3.5).
4. Desarrollo de una infraestructura como servicio en la nube para alojar los modelos entrenados de ML, así como los servicios requeridos para visualizar los pronósticos.
5. Desarrollo de algoritmos de ciencia de datos para el preprocesamiento, análisis y predicción de datos de calidad del aire.
6. Pruebas del sistema en al menos tres zonas geográficas de El Salvador, seleccionadas por su relevancia ambiental y representatividad de condiciones atmosféricas diversas. Recopilación y análisis de datos en condiciones reales durante un período mínimo de 1 mes.
7. Implementación de técnicas de machine learning para la predicción de concentraciones de contaminantes y el cálculo del Índice de Calidad del Aire (ICA).
8. Generación de visualizaciones gráficas y mapas de calidad del aire utilizando herramientas como QGIS.
9. Redacción y entrega de documento tesis que detalla la investigación preliminar realizada, así como también la metodología y el proceso de construcción del dispositivo y procesamiento de datos.
10. Redacción y publicación de un artículo científico en una revista indexada.
11. Presentación de los resultados en un congreso científico nacional o internacional.

1.5. Antecedentes

El presente proyecto se enmarca dentro de las iniciativas de investigación de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador (UES), orientadas a brindar soluciones tecnológicas innovadoras y de bajo costo para el monitoreo de la calidad del aire. Este esfuerzo busca complementar y fortalecer el sistema principal de monitoreo ambiental gestionado por el Ministerio de Medio Ambiente y Recursos Naturales (MARN) de El Salvador.

La primera versión del sistema (V1.0), desarrollada como parte de un trabajo de graduación, consistió en un dispositivo IoT fijo capaz de medir concentraciones de partículas contaminantes

(PM) y variables ambientales como la temperatura. Los datos recopilados se almacenan automáticamente en una base de datos y se visualizaron mediante gráficas en una plataforma web. En su segunda iteración (V2.0), se incorporó un módulo de conectividad GSM, eliminando la dependencia de redes WiFi y permitiendo la transmisión de datos mediante una tarjeta SIM. Además, se integraron sensores avanzados para medir concentraciones de $PM_{1.0}$, $PM_{2.5}$, PM_{10}, formaldehído (HCHO), temperatura y humedad.

La tercera versión (V3.0) introdujo un sistema de geolocalización (GPS), que permitió la movilidad del dispositivo y la recopilación de datos en trayectos específicos. Esta versión también incluyó la generación de mapas de calidad del aire mediante el uso de herramientas de sistemas de información geográfica (QGIS). Posteriormente, en la versión 3.5, se implementó una plataforma IoT centralizada utilizando el servicio en línea Thingsboard, permitiendo el monitoreo en tiempo real de múltiples dispositivos desplegados.

1.6. Planteamiento del problema

La contaminación del aire es uno de los principales desafíos ambientales en El Salvador, con implicaciones directas en la salud pública, la calidad de vida y los ecosistemas. Actualmente, el Ministerio de Medio Ambiente y Recursos Naturales (MARN) cuenta con solo tres estaciones fijas de monitoreo de calidad del aire, todas ubicadas en San Salvador. Sin embargo, dos de estas estaciones presentan fallas recurrentes y, en ocasiones, dejan de funcionar, lo que limita significativamente la capacidad del país para monitorear y gestionar la contaminación atmosférica. Esta falta de cobertura geográfica y confiabilidad en los datos impide una evaluación precisa de la calidad del aire en otras regiones del país, así como la implementación de medidas preventivas y correctivas efectivas. Además, los altos costos asociados con el mantenimiento y la expansión de las estaciones existentes dificultan la modernización del sistema. Ante esta situación, se hace urgente desarrollar soluciones tecnológicas innovadoras, escalables y de bajo costo que complementen la infraestructura actual, permitan una recolección de datos más flexible y confiable, y faciliten el análisis predictivo de la calidad del aire mediante el uso de herramientas avanzadas como el Internet de las Cosas (IoT) y la ciencia de datos.

1.7. Justificación

Este proyecto se justifica por la urgente necesidad de mejorar el monitoreo de la calidad del aire en El Salvador, donde el sistema actual del MARN cuenta con solo tres estaciones fijas en San Salvador, dos de las cuales presentan fallas recurrentes. Esta limitación impide una evaluación precisa y representativa de la contaminación atmosférica en otras regiones del país, afectando la salud pública y la gestión ambiental. El desarrollo de un sistema IoT actualizado, basado en tecnologías de bajo costo y ciencia de datos, permitirá una recolección de datos más flexible, confiable y predictiva, complementando la infraestructura existente y apoyando la toma de decisiones informadas. Además, el proyecto tiene un alto valor formativo para los estudiantes involucrados y contribuirá a fortalecer las capacidades tecnológicas y ambientales del país.

1.8. Resultados esperados

1. Un dispositivo electrónico actualizado, basado en el microcontrolador ESP32, que integre sensores para medir O_3, CO, CO_2, $TVOC$, $PM_{1.0}$, $PM_{2.5}$, PM_{10}, $HCHO$, temperatura y humedad, con módulos de conectividad GSM y GPS.
2. Una plataforma configurada en Thingsboard que permita el monitoreo en tiempo real de múltiples dispositivos, con visualización de datos y alertas sobre la calidad del aire.
3. Modelos predictivos entrenados para analizar y pronosticar concentraciones de contaminantes, así como para calcular el Índice de Calidad del Aire (ICA) según los estándares del MARN.
4. Datos recopilados y analizados en al menos tres zonas geográficas de El Salvador, que demuestren la eficacia y precisión del sistema en condiciones reales.
5. Documento técnico de tesis que detalle las etapas de diseño e implementación del sistema propuesto.
6. Un artículo científico publicado en una revista indexada y una presentación en un congreso científico.

Capitulo 2. DEFINICIÓN DE CONCEPTOS, REGULACIONES RELACIONADAS CON LA CALIDAD DEL AIRE Y *MACHINE LEARNING*.

2.1. Calidad de aire

El aire es una mezcla de gases que conforman la atmósfera terrestre, compuesta principalmente por nitrógeno y oxígeno, y en menor medida dióxido de carbono, vapor de agua y otros gases. La calidad del aire hace referencia a la condición del aire en relación con la concentración de contaminantes que pueden afectar la salud humana (Generalitat Valenciana, 2025).

2.1.1. Índice Centroamericano de Calidad de Aire

El índice de calidad de aire (ICA) demuestra el estado del aire, y los riesgos a la salud de acuerdo al valor que se obtenga que va de 0 a 500, significando que a menor sea su valor menor contaminación se tiene (AirNow, 2025). Los contaminantes regulados son: ozono (O_3), monóxido de carbono (CO), óxido de nitrógeno (NO), partículas suspendidas (PM_{10} y $PM_{2.5}$) y dióxido de azufre (SO_2).

Para el cálculo del índice de calidad de aire se hace uso de la siguiente fórmula:

$$AQI = \frac{AQI_{High} - AQI_{Low}}{C_{High} - C_{Low}} * (C_x - C_{Low}) + AQI_{Low} \quad (2.1)$$

Donde:

Para el cálculo del valor de ICCA para cada una de las concentraciones se ocupa la siguiente fórmula. Donde:

- AQI_{High}: Valor superior del ICCA del rango.
- AQI_{Low}: Valor inferior del ICCA del rango.
- C_{High}: Valor superior de concentración del rango.
- C_{Low}: Valor inferior de concentración del rango.
- C_x: Valor de concentración medido.

Para conocer el nivel en el que se encuentran los contaminantes existe el índice de calidad de aire, el cual determina la categoría en la que se encuentra.

Tabla 2.1. *Valor del ICA.*

Estado del aire	Rango del índice de calidad de aire
Bueno	0-50
Moderado	50-100
Perjudicial para grupos sensibles	100-150
Perjudicial	150-200
Tóxico	200-300
Altamente tóxico	300-500

Nota: La tabla muestra los distintos rangos del índice de calidad del aire de acuerdo al estado que se encuentre. (Air Quality Planning, 2025).

2.1.2. Decreto N°40 Reglamento Especial de Normas Técnicas de calidad ambiental.

En el país se presenta una regulación respectiva a la concentración y el periodo de medición, los contaminantes del aire de estudio en el país son los siguientes: dióxido de azufre (SO_2), monóxido de carbono(CO), nitratos (NO_x), ozono (O_3), plomo (Pb), partículas totales suspendidas (PM_{10} y $PM_{2.5}$). En el decreto No 40, los artículos 9, 10, 11, 12, 13, 14, 15 y 16; se regulan las fuentes fijas y móviles, exigiendo la instalación de sistemas para el control de las emisiones (Asamblea Legislativa de la República de El Salvador, 2000).

2.2. Internet de las Cosas

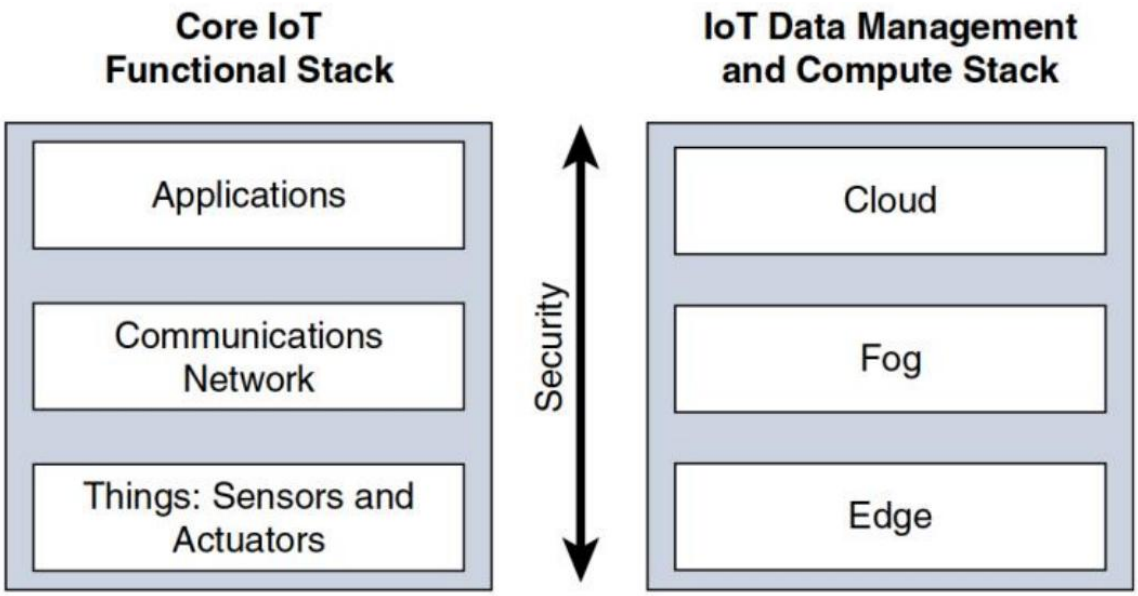
El Internet de las Cosas (IoT) se define como una transición tecnológica que busca conectar lo desconectado. Esto significa que los objetos que antes no estaban unidos a una red, como el Internet, ahora pueden comunicarse e interactuar con personas y otros objetos.

El IoT representa la fase más reciente en la evolución de Internet, avanzando hacia la digitalización del mundo al conectar personas, procesos, datos y, fundamentalmente, cosas. (David Hanes et al. 2017)

2.3. El Stack Tecnológico del Internet de las Cosas

El IoT es un "paraguas de varios conceptos, protocolos y tecnologías"

Figura 2.1. *Diagrama simplificado del Stack Tecnológico del Internet de las Cosas.*



Nota: El diagrama muestra los diferentes niveles de computación relacionados con las tecnologías IOT. En el borde están los sensores, en el "fog" se encuentran los nodos que interceptan los datos creados por los sensores, y en la nube se encuentran las aplicaciones que procesan estos datos. Obtenido de IoT Fundamentals (Rohini College. 2021).

2.3.1. El Stack Funcional del IoT (Core IoT Functional Stack)

- **Primera capa:** "Things", Sensores y Actuadores

Esta capa es el bloque fundamental del IoT y que está compuesto por smart objects. Los sensores miden una cantidad física y la convierten en datos digitales mientras que los actuadores reciben una señal de control y provocan una acción física, como un movimiento o fuerza. Un objeto inteligente típicamente incluye un procesador, sensores y actuadores, un dispositivo de comunicación y una fuente de energía.

- **Segunda capa**: Redes de comunicación.

Esta capa conecta los objetos inteligentes e incluye subcapas esenciales para el transporte de datos como las redes de acceso que de entre las cuales se tienen las tecnologías de radiofrecuencia (RF), redes cableadas, IEEE 802.15.4 (ZigBee), LoRaWAN y NB-IoT.

Se tienen además las Gateways y Backhaul. El gateway es crucial, ya que reenvía la información recolectada por los objetos a través de un medio de mayor alcance (backhaul) a una estación central para su procesamiento.

Protocolos de comunicación: Son un conjunto de reglas que dictan el formato y el método con el que se transmiten datos. En el IOT, comúnmente se encuentran los protocolos MQTTS y HTTP/HTTPS

- **Tercera capa:** Aplicaciones y Analítica

En este nivel, se procesan los datos recolectados. Las aplicaciones de control modifican el comportamiento de los objetos, mientras que las aplicaciones de analítica procesan los datos para proporcionar una compresión del sistema (Rohini College. 2021.).

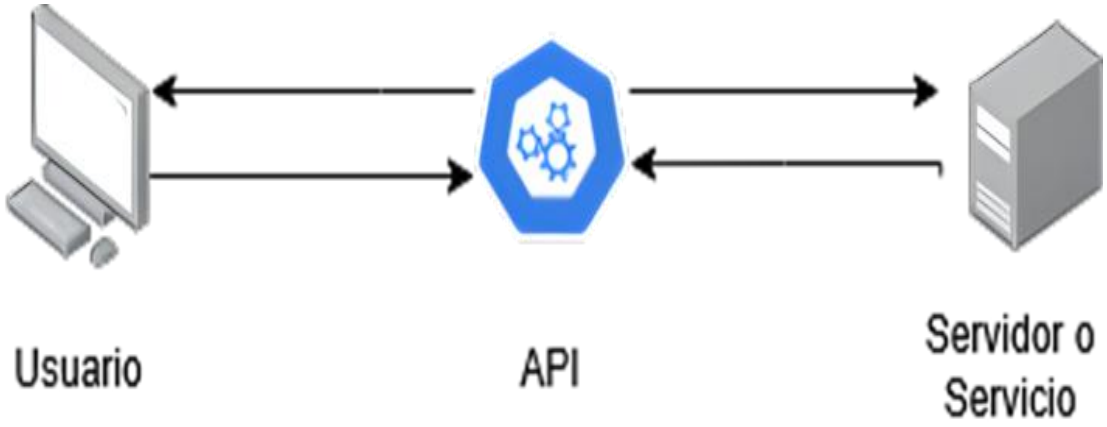
2.3.2. Stack de Cómputo y Gestión de Datos de IoT (IoT Data Management and Compute Stack)

Este stack aborda la latencia y el volumen de datos (big data) mediante la distribución de la capacidad de procesamiento (Rohini College. 2021.). Se comprenden tres aspectos en este stack:

- Edge Computing: Procesamiento de datos realizado directamente en el sensor o dispositivo IoT
- Fog Computing: Un nivel distribuido que utiliza los *gateways* o nodos de red cercanos al borde para realizar análisis y control con baja latencia y para filtrar datos, reduciendo el volumen de tráfico enviado a la nube
- Cloud Computing: Procesamiento centralizado para el análisis de *big data* histórico

Application Programming Interface (API)

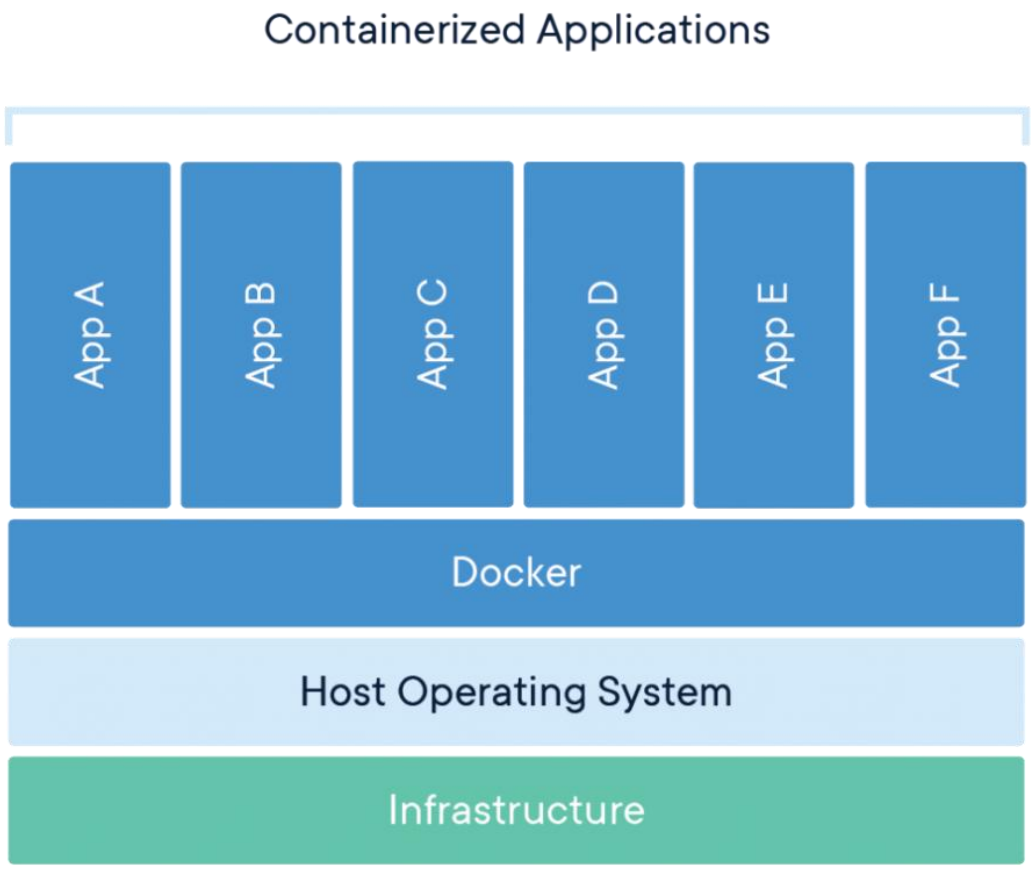
Figura 2.2. *Diagrama de representación de una API.*



Nota: Como se observa en el diagrama, el usuario recibe datos del servidor a través de la API sin interactuar directamente con este. Fuente: elaboración propia usando Diagrams.net.

Una API es un conjunto de reglas y protocolos que permiten la comunicación entre diferentes aplicaciones. Un servidor está compuesto de procesos con información delicada que no puede exponerse a la red pública. Una API permite exponer una ruta hacia una función que, por ejemplo, obtiene un dato en específico de una base de datos. De esta forma, el servidor puede ofrecer a un cliente un medio para llamar datos de esa base de datos sin exponer el proceso con el cual esos datos fueron generados (Masse, 2011).

Figura 2.3. *Stack de una aplicación contenerizada.*



Nota: Docker, Inc. (2025) muestra este sistema de contenedores como un conjunto de aplicaciones en contenedores que se ejecutan en la plataforma de contenerizado (Docker en este caso) y que la cual se ejecuta sobre el sistema operativo en el que se aloja.

Un contenedor es un paquete que incluye todas las librerías y archivos que necesita una aplicación para poder correr. Los contenedores utilizan procesos comunes del sistema en el que son desplegados, pero funcionan de forma aislada del resto de las aplicaciones, de tal forma que utilizan su propia versión de cada archivo y librería necesario. De esta forma, si un contenedor funciona correctamente en la máquina en la que fue desarrollado, entonces podrá correr en cualquier otra máquina, sin importar el hardware y software que tenga (Merkel, D. 2014).

2.4. Reverse proxy

Un reverse proxy es un servidor web que se ubica frente a los servicios expuestos a la red, este intercepta toda solicitud realizada por clientes y los redirige hacia el servidor o servicio correspondiente, actuando como un intermediario y equilibra la carga global del servidor. Permiten únicamente el paso de tráfico HTTP/HTTPS. (Nginx, Inc, 2025).

2.5. Machine learning

Machine Learning (aprendizaje automático) constituye un subcampo de la inteligencia artificial cuyo propósito consiste en la utilización de algoritmos capaces de descubrir patrones en conjuntos de datos, los cuales pueden ser numéricos, textuales, imágenes, etc., con el resultado de ser capaz de imitar el modo en que aprenden los humanos para realizar tareas de forma autónoma. Posee áreas de aplicación en diferentes ámbitos, de los cuales se puede mencionar el sector financiero, marketing y publicidad, salud y medicina, entre otros. Los modelos de machine learning utilizan datos para poder descubrir lo que sucedió, predecir lo que sucederá y realizar recomendaciones de lo que puede pasar (Bergmann, 2023; DataScientest, 2023; Red Hat, 2022)

La clasificación de los algoritmos de machine learning dependerá del conjunto de datos que se les proporcione, y puede dividirse en tres categorías principales: aprendizaje supervisado, no supervisado y semi-supervisado.

2.5.1. Aprendizaje supervisado

Son algoritmos que aprenden de conjuntos de datos debidamente etiquetados; en otras palabras, el desarrollador proporciona pares de datos entrada-salida, de modo que el algoritmo tratará de establecer una relación o reconocer patrones entre ambos, para posteriormente generar una salida correspondiente a nuevos datos de entrada. Este tipo de es habitualmente utilizado para el análisis predictivo (Müller & Guido, 2016).

El aprendizaje supervisado dispone de los siguientes algoritmos:

- **Algoritmos de regresión:** buscan establecer una relación matemática entre valores reales, analizando patrones y tendencias para generar predicciones e ideas que ayuden en la toma de decisiones.
- **Algoritmos de clasificación:** estos identifican patrones en los datos etiquetados para asignar elementos individuales a categorías predefinidas.

2.5.2. Aprendizaje no supervisado

A lo contrario del aprendizaje supervisado, el aprendizaje no supervisado son algoritmos que aprenden de datos no etiquetados. Este tipo de aprendizaje es el más común en el análisis de conjuntos conglomerados ya que tiene la capacidad de descubrir patrones, estructuras y relaciones ocultas dentro de los datos (Kpucha, 2023; Müller & Guido, 2016).

De los algoritmos más comunes dentro del aprendizaje no supervisados están:

- **K-medias:** tiene como objetivo particionar el conjunto de datos en subconjuntos no solapados. Su idea principal se basa en que escoge aleatoriamente un punto central para cada subconjunto “k” (Kpucha, 2023; StatDeveloper, 2025).
- **Agrupación jerárquica:** este construye una jerarquía multinivel de conglomerados, que a diferencia de otros métodos como K-medias, no necesita definir el número de conglomerados al inicio, y tiene como resultado una estructura jerárquica que muestra como los datos se van agrupando o dividiendo (DataCamp, 2025).
- **Reducción de dimensionalidad:** la reducción de dimensionalidad comprende un conjunto de técnicas cuyo objetivo es simplificar los datos al disminuir la cantidad de variables originales, manteniendo al mismo tiempo la mayor cantidad posible de información relevante (Kpucha, 2023).

2.5.3. Aprendizaje semi supervisado

El aprendizaje semi supervisado es un enfoque híbrido que combina elementos del aprendizaje supervisado y no supervisado. Su principal característica es que utiliza simultáneamente una pequeña cantidad de datos etiquetados y una gran cantidad de datos no etiquetados para entrenar modelos de machine learning. Este tipo de aprendizaje resulta especialmente útil cuando obtener etiquetas es costoso, lento o requiere conocimientos especializados, pero existe abundancia de datos sin anotar.

La idea central del aprendizaje semi supervisado es que los datos no etiquetados contienen información valiosa sobre la estructura interna del conjunto de datos. Al incorporarlos en el proceso de entrenamiento, el modelo puede aprender límites de decisión más precisos y generalizar mejor que si solo se entrenara con los pocos datos etiquetados disponibles (IBM, 2025; Oracle, 2024).

Entre las técnicas más comunes del aprendizaje semi supervisado se encuentran:

- **Propagación de etiquetas:** Asigna etiquetas a los datos no etiquetados basándose en su similitud con los datos etiquetados, bajo la idea de que puntos cercanos en el espacio de características deberían compartir la misma clase (Brownlee, 2019).
- **Autoentrenamiento:** El modelo inicial, entrenado únicamente con datos etiquetados, genera "pseudo etiquetas" para los datos no etiquetados. Luego, estas pseudo etiquetas se incorporan al entrenamiento para mejorar progresivamente el modelo (GeeksforGeeks, 2025).
- **Coentrenamiento:** Utiliza dos o más modelos que aprenden de diferentes vistas o subconjuntos de los datos. Cada modelo genera pseudo etiquetas que sirven para entrenar a los otros, reduciendo el riesgo de reforzar errores (Scikit-learn, 2025).
- **Métodos basados en pre entrenamiento:** Emplean técnicas no supervisadas (como auto codificadores o reducción de dimensionalidad) para aprender representaciones útiles de los datos no etiquetados antes de entrenar un modelo supervisado final (Scikit-learn, 2025).
- **Modelos intrínsecamente semi supervisados:** Incorporan directamente los datos no etiquetados en la función de pérdida, como ocurre en las máquinas de vectores de soporte semi supervisadas (S3VM) o en arquitecturas de deep learning diseñadas específicamente para este propósito (Scikit-learn, 2025).

2.6. Deep learning

El deep learning es una subárea del aprendizaje automático que se basa en el uso de redes neuronales profundas para modelar relaciones complejas dentro de los datos. A diferencia de los modelos tradicionales de machine learning, que suelen trabajar con arquitecturas poco profundas, el deep learning emplea redes con múltiples capas ocultas capaces de aprender representaciones jerárquicas y de alto nivel a partir de datos brutos. Esto permite que los modelos identifiquen patrones complejos sin necesidad de una ingeniería manual de características (IBM, 2025).

Una de las principales ventajas del deep learning es su capacidad para trabajar con grandes volúmenes de datos no estructurados, como imágenes, audio, texto o series temporales. Gracias a procesos como la propagación hacia adelante y la retropropagación, las redes neuronales ajustan sus parámetros de forma iterativa hasta mejorar su precisión. Sin embargo, este tipo de modelos

requiere una alta capacidad computacional, por lo que suelen entrenarse utilizando unidades de procesamiento gráfico (GPU) o entornos de computación en la nube (Data Center Market, 2025). Dentro del deep learning existen diferentes tipos de arquitecturas, cada una diseñada para resolver problemas específicos. Entre las más utilizadas se encuentran:

- **Redes Neuronales Convolucionales (CNN):** Se emplean principalmente en tareas de visión por computadora, como clasificación de imágenes, detección de objetos y reconocimiento facial. Su estructura permite extraer características visuales de manera automática y eficiente.
- **Redes Neuronales Recurrentes (RNN):** Están orientadas al procesamiento de datos secuenciales, como texto, audio o series temporales. Su característica principal es que incorporan memoria, permitiendo que la salida dependa de entradas previas dentro de la secuencia.
- **LSTM (Long Short-Term Memory):** Son una variante mejorada de las RNN, diseñadas para superar problemas como el desvanecimiento del gradiente. Las LSTM pueden aprender dependencias a largo plazo, lo que las hace especialmente útiles en predicción de series temporales, modelado de secuencias y análisis de datos ambientales.
- **Auto Codificadores:** Se utilizan para aprender representaciones comprimidas de los datos, lo que resulta útil en reducción de dimensionalidad, detección de anomalías y pre entrenamiento no supervisado.
- **GAN (Generative Adversarial Networks):** Permiten generar nuevos datos similares a los originales mediante la interacción entre un generador y un discriminador. Se emplean en síntesis de imágenes, aumento de datos y aplicaciones creativas.
- **Modelos transformadores:** Han revolucionado el procesamiento del lenguaje natural al permitir el análisis paralelo de secuencias y el aprendizaje de dependencias a largo plazo. Son la base de los modelos de lenguaje actuales.

2.7. Redes LSTM para predicción de series de tiempo

Las redes LSTM (Long Short-Term Memory) resultan ser una variante de las redes neuronales recurrentes (RNN) diseñadas específicamente para procesar datos secuenciales. Con respecto a las RNN tradicionales presentan una diferencia la cual es que las LSTM incorporan mecanismos internos llamados compuertas, que permiten controlar qué información se mantiene, cuál se

actualiza y cuál se descarta. Esto permite que las LSTM pueden aprender dependencias a largo plazo dentro de una secuencia de datos, evitando problemas comunes como el desvanecimiento del gradiente.

Este tipo de redes se ha convertido en una de las herramientas más utilizadas dentro del deep learning para realizar predicciones sobre series de tiempo, ya que son capaces de identificar patrones temporales complejos y generar pronósticos precisos a partir del comportamiento histórico de los datos.

Las LSTM resultan especialmente útiles en las series de tiempo porque pueden aprender cómo evoluciona una variable a lo largo del tiempo y utilizar esa información para predecir su comportamiento futuro. La configuración del modelo dependerá tanto del número de variables disponibles como del horizonte de predicción deseado (GeeksforGeeks, 2025).

2.7.1. Tipos de configuraciones LSTM para series de tiempo

Las diferentes configuraciones de LSTM puede adaptarse a distintos escenarios de predicción de las cuales se encuentra:

- **Modelos univariados-unistep:** En esta configuración se utiliza una sola variable de entrada y se predice un único valor futuro. Un ejemplo de esta configuración puede ser la siguiente: si se dispone del registro de temperatura de las últimas 24 horas, el modelo puede aprender a predecir la temperatura de la siguiente hora.
- **Modelos multivariados-unistep:** Se emplean múltiples variables de entrada, pero la predicción sigue siendo un único valor futuro. Siguiendo el ejemplo anterior, además de la temperatura se pueden incluir variables como humedad, presión atmosférica o precipitación para mejorar la predicción de la temperatura una hora adelante.
- **Modelos univariados-multistep:** Se utiliza una sola variable de entrada, pero se predicen varios valores futuros. Por ejemplo, a partir de 24 horas de temperatura, el modelo puede estimar cómo se comportará la temperatura durante las próximas 5 horas.
- **Modelos multivariados-multistep:** Utiliza múltiples variables de entrada y genera predicciones para varios instantes de tiempo futuros. Por ejemplo, usando registros históricos de temperatura, humedad y presión, el modelo puede predecir la temperatura para las próximas 5 horas (GeeksforGeeks, 2025).

2.7.2. Funcionamiento general de una celda LSTM

Las celdas LSTM están formadas por distintos componentes internos que trabajan en conjunto para almacenar, actualizar y controlar la información a lo largo del tiempo. Gracias a esta estructura, las LSTM son capaces de aprender relaciones de largo plazo en datos secuenciales, superando las limitaciones de las redes neuronales recurrentes tradicionales.

- **Puerta de olvido:** La puerta de olvido se encarga de decidir qué información almacenada en el estado de la celda ya no es relevante y puede ser descartada. De esta forma, la red evita conservar datos innecesarios que podrían afectar el aprendizaje.
- **Puerta de entrada:** La puerta de entrada determina qué información nueva proveniente de la entrada actual debe incorporarse al estado de la celda. Esta puerta analiza tanto la entrada del instante actual como el estado oculto anterior, generando un nivel de activación que controla la cantidad de información que será añadida.
- **Estado de la celda candidata:** El estado de la celda candidata representa la nueva información que podría incorporarse a la memoria de la celda. Este valor se obtiene aplicando la función de activación *tanh* a una combinación de la entrada actual y el estado oculto previo. Su función principal es proponer nueva información sin eliminar la que ya se encuentra almacenada.
- **Actualización del estado de la celda:** El estado de la celda se actualiza combinando la información que se decide conservar del estado anterior con la nueva información propuesta por la celda candidata. Este proceso se realiza mediante operaciones elemento a elemento, donde la puerta de olvido controla lo que se mantiene y la puerta de entrada regula lo que se agrega.
- **Puerta de salida:** La puerta de salida define qué parte de la información contenida en el estado de la celda actual se utilizará para generar el nuevo estado oculto. Para ello, considera la entrada actual y el estado oculto anterior, permitiendo que solo la información relevante continúe hacia el siguiente paso de tiempo (Kumar, 2025).

2.7.3. LSTM para predicción de series temporales en la práctica

Para utilizar una LSTM en predicción de series temporales, es necesario transformar la serie en ventanas de tiempo. Cada ventana contiene un conjunto de observaciones consecutivas que sirven

como entrada del modelo, mientras que el valor o valores siguientes actúan como objetivo de predicción. Este enfoque permite que la red aprenda la dinámica temporal de la serie.

En frameworks como PyTorch o TensorFlow, una red LSTM suele complementarse con una capa completamente conectada al final, encargada de generar la predicción final a partir del estado oculto de la última celda.

Capitulo 3. METODOLOGÍA DE LA MEDICIÓN DE LA CALIDAD DE AIRE Y EL ENTORNO DE DESARROLLO DE MACHINE LEARNING.

3.1. Estación móvil para monitoreo de calidad de aire

La estación móvil a mejorar ha tenido una evolución de versiones la cuales son las siguientes:

- En la versión 1.0, el dispositivo medía las partículas en suspensión (PM) y la temperatura, que se visualizaban y almacenaban en una plataforma web (Flores-Cortez et al, 2019).
- En la versión 2.0, el dispositivo incorpora conexión GSM, además de medir las concentraciones de $PM_{1.0}$, $PM_{2.5}$, PM_{10} y formaldehído ($HCHO$), la temperatura y la humedad mediante sensores avanzados (Flores-Cortez et al, 2023).
- En la versión 3.0, se introdujeron la geolocalización (GPS) y la generación de mapas de calor de la calidad del aire para la monitorización móvil (Flores-Cortez et al, 2024).
- En la versión 3.5, se implementó una plataforma IoT centralizada (Thingsboard) para el seguimiento en tiempo real (Castillo-Rosales et al, 2024).

Ahora, en la versión 4.0, se busca mejorar las capacidades tecnológicas y analíticas de los sistemas anteriores, además de integrar nuevos sensores para medir el ozono (O_3), el monóxido de carbono (CO), el dióxido de carbono (CO_2) y los compuestos orgánicos volátiles totales ($TVOC$), así como los sensores utilizados anteriormente en las otras versiones.

3.1.1. Componentes de la estación móvil

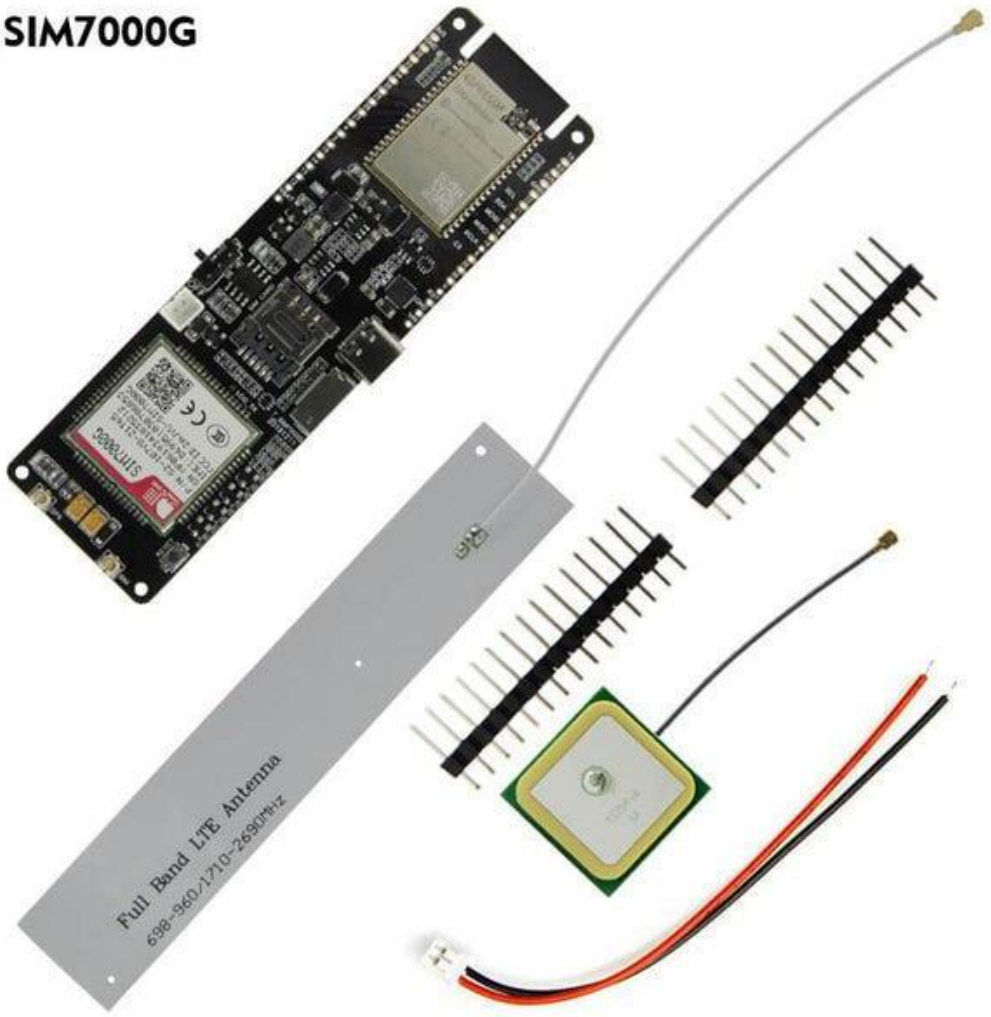
- **Placa de desarrollo LILYGO® TTGO T-SIM7000G**

La placa cuenta con una placa de desarrollo ESP32, con un chip SIM 7000 G. Permitiendo la conexión por Wi-Fi y Bluetooth, así como el envío de SMS, la hora GMT y la ubicación por medio del GPS (Lilygo, 2025).

Características:

- Marca: LILYGO
- Nombre del modelo: CH9102F 16MB
- Capacidad de almacenamiento de memoria: 16 MB
- PSRAM: 8 MB
- Voltaje de salida: 3.3 V o 5 V
- Interfaces periféricas: I2C, SPI, UART, SDIO, I2S, CAN

Figura 3.1. *Placa de desarrollo.*



Nota: La placa desarrolladora cuenta con pines, un cable, una antena gps y una antena LTE. Lilygo (2025)

- **Sensor láser PMS5003ST**

El sensor permite la medición de $PM_{1.0}$, $PM_{2.5}$ y PM_{10}, formaldehído, temperatura y humedad (Plantower, 2025).

Características:

- Tasa cero de alarmas falsas.
- Diámetro mínimo distinguible de partícula de hasta 0.3 um.
- Voltaje de alimentación: 5 V
- Comunicación con la placa: UART

Figura 3.2. *Sensor láser PMS5003ST.*

Nota: Imagen referencial del sensor láser utilizado para la medición del material particulado. Plantower (2025)

- **Sensor de gases MEMS**

El sensor de concentración de gas MEMS proporciona la detección de distintas concentraciones de gases como lo son el CO, $C2H5OH$ (Alcohol), H_2, NO_2 y NH_3 (SGX Sensortech, 2025).

Características:

- Permite la detección de diversos gases nocivos.
- Salida digital I2C.
- Integra el cálculo por fórmulas de diferentes concentraciones de gases.
- Compatible con voltaje de alimentación 3.3-5.5V.

Figura 3.3. *Sensor de gases MEMS.*

Nota: Imagen referencial del sensor medidor de gases. DFRobot (2025)

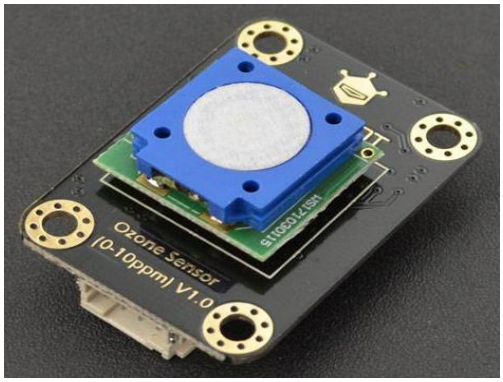
- **Sensor de ozono.**

El sensor basado en principios electroquímicos mide la concentración de ozono en el ambiente, tiene una alta capacidad anti interferencia, alta estabilidad y sensibilidad (Winsen, 2025).

Características:

- Altamente sensible.
- De bajo consumo.
- Salida digital I2C.
- Protección de polaridad.
- Compatible con voltaje de alimentación 3.3-5.5V.

Figura 3.4. *Sensor de ozono.*



Nota: Imagen referencial del sensor medidor de ozono. DFRobot (2025)

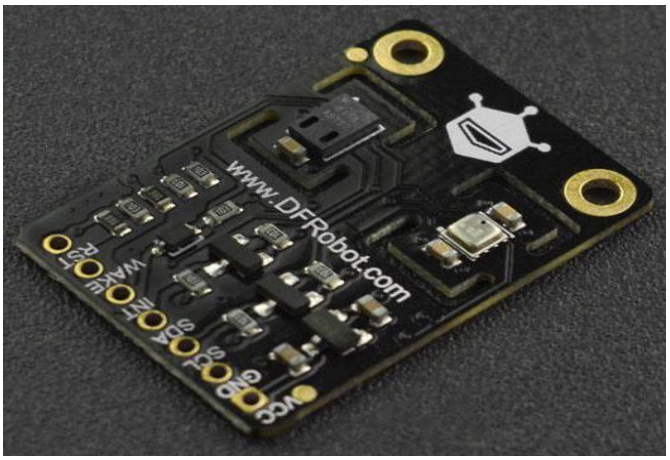
- **Sensor de multifunción Ambiental ENS 160 y BME 280**

El sensor hace la combinación del chip ENS 160+BME 280, el sensor ENS 160 permite la detección de múltiples datos como $TVOC$, eCO_2, AQI; el sensor BME 280 combina el sensor de temperatura, sensor de humedad y barómetro, proporcionando un rendimiento estable (Bosch, 2025).

Características:

- Apilable con otros módulos de Fermión.
- De bajo consumo.
- Salida digital I2C.
- Medida simultánea de los distintos parámetros.
- Compatible con voltaje de alimentación 3.3-5.5V.

Figura 3.5. *Sensor ENS 160+BME 280.*



Nota: Imagen referencial del sensor medidor de TVOC y CO_2. DFRobot (2025)

- **Pantalla táctil capacitivo SSD1963 7”**

El módulo utiliza un controlador LCD SSD1963 con un módulo LCD de 7 pulgadas con pantalla táctil. Esta pantalla LCD tiene una calidad de visualización superior y un ángulo de visión súper amplio. Se puede utilizar en cualquier sistema integrado que requiera la visualización de imágenes coloridas de alta calidad (Solomon Systech, 2025).

Características:

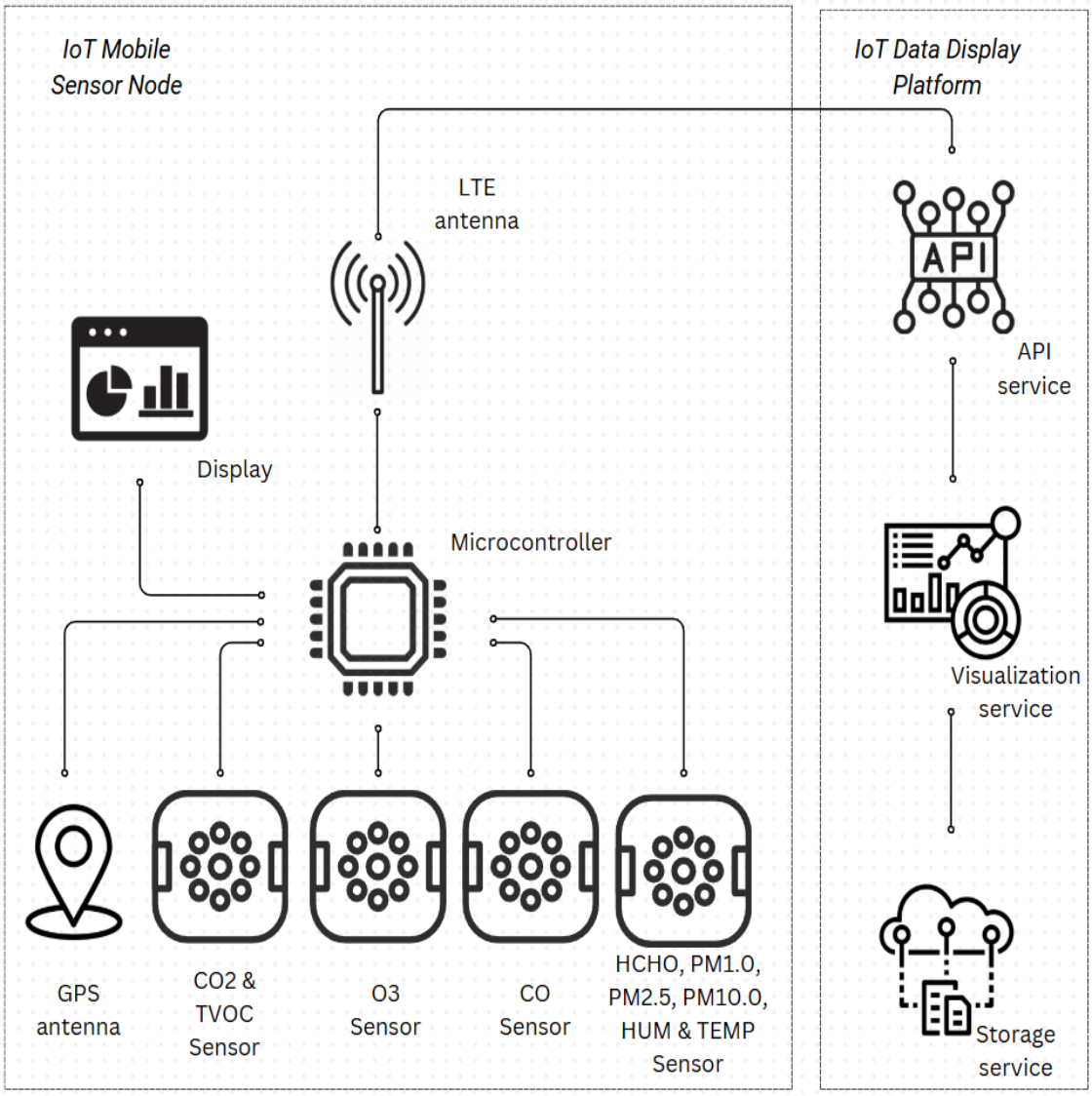
- Módulo LCD TFT de 7,0", resolución 800x480, controlador SSD1963.
- Tipo de LCD: TFT transmisivo blanco normal

- Interfaz de bus paralelo de 16 bits
- Regulador elevador CC-CC de 400 mA integrado para proporcionar alimentación a la retroiluminación de la pantalla LCD.
- Área activa: 154 mm x 86 mm.
- Distancia entre píxeles: 0,179 mm x 0,179 mm.
- Conector estándar 2x20 de 2,54 mm para la conexión a la MCU/placa de desarrollo.

Figura 3.6. *Pantalla SSD1963***.**

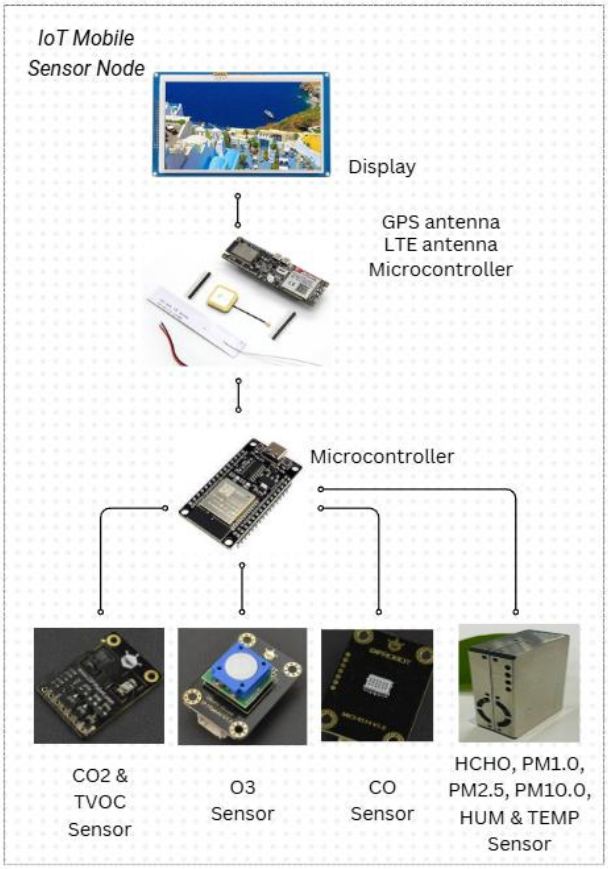
Nota: Imagen referencial de la pantalla utilizada para la visualización de datos. World Chips (2025)

Figura 3.7. *Esquema de la estación móvil medidora de calidad del aire.*



Nota: Funcionamiento general de los dispositivos y su conexión con la plataforma, explicación específica en la sección 3.2. Fuente: Elaboración propia utilizando canva.com.

Figura 3.8. *Esquema de la estación móvil medidora de calidad de aire.*



Nota: Esquema de la estación móvil con los tipos de dispositivos ocupados en cada segmento. Fuente: Elaboración propia utilizando canva.com.

3.2. Funcionamiento del dispositivo

3.2.1. Pasos operativos de la estación móvil

En primer lugar, cuando se enciende el dispositivo, se inicia la placa de desarrollo, lo que activa la pantalla que muestra los procesos que se están llevando a cabo. A continuación, se procede al calentamiento de los sensores, que estarán listos para realizar mediciones transcurridos tres minutos. Se establece la conexión a Internet a través del APN y comienza el proceso de posicionamiento con los satélites cercanos. Las mediciones se toman a intervalos de 4,5 segundos y luego se envían a ThingsBoard (Thingsboard, 2025), donde se muestran y almacenan los datos.

Figura 3.9. *Estación móvil vista frontal.*

Nota: La imagen muestra como se encuentran distribuidos los valores en la pantalla, encontrándose con fecha y hora en la parte superior derecha, información del GPS y GSM en la izquierda, y los contaminantes con su valor ICA en la parte inferior. Fuente: Elaboración propia, fotografía.

3.2.2. Región a medir

Las mediciones se realizan entre las 5:00 am y las 8:00 pm, abarcando tanto las horas pico como las horas de menor tráfico. El área de interés para este estudio fueron espacios como hospitales, clínicas y centros de salud, donde se encuentran personas con salud vulnerable y donde estas concentraciones son perjudiciales para su bienestar y recuperación. Además, se incluyeron áreas

de espacios públicos, como plazas. La ruta comienza en el Hospital Infantil Benjamín Bloom, siguiendo la Avenida 25 Norte hasta llegar al Hospital Nacional Rosales, donde se encuentran diferentes hospitales y clínicas, luego la ruta continúa por la calle Rubén Darío hasta llegar a la plaza Libertad, luego continúa por la Avenida 6a Sur hasta llegar a la calle 1ª Este, finalmente la ruta termina llegando a la intersección con la Avenida España, como se muestra en la Fig. 3.10.

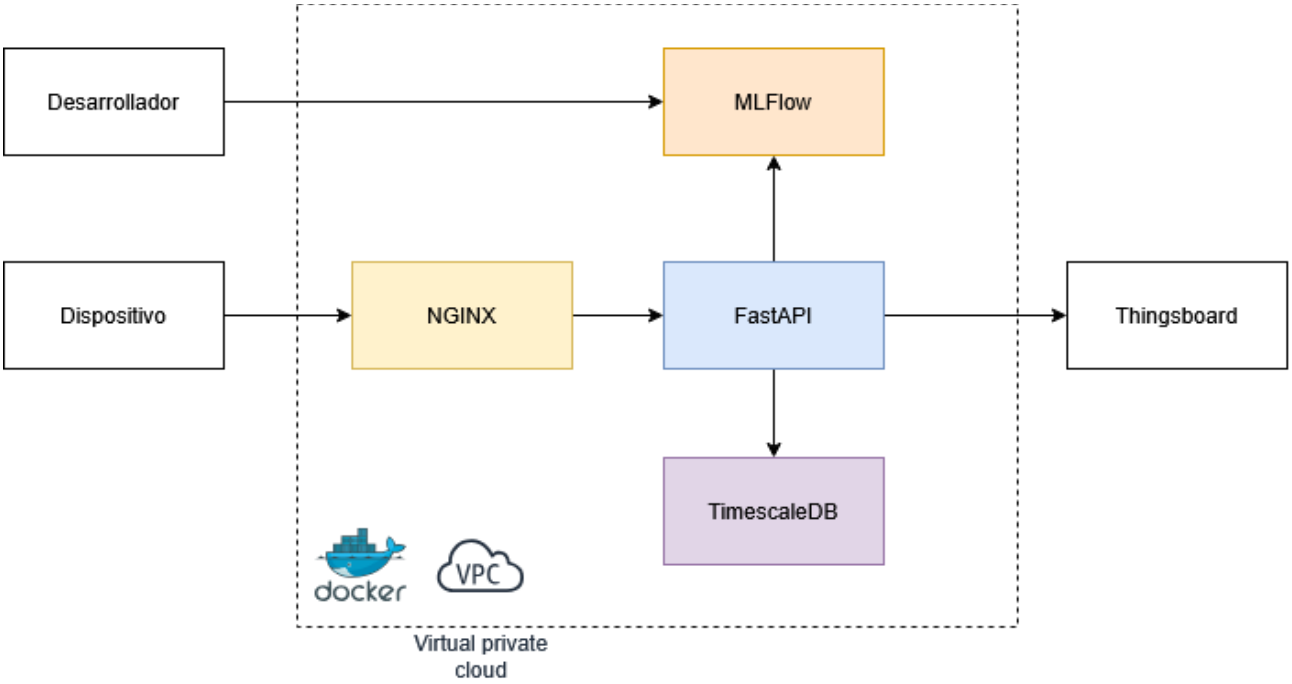
Figura 3.10. *Ruta de medición en el centro de San Salvador.*

Nota: Recorrido en el cual se harán las tomas de las mediciones de la calidad de aire. Fuente: Elaboración propia usando QGIS, OpenStreetMap.

3.3. Diseño de la arquitectura

El servidor IOT se centra en un servicio de API que maneja todos los datos enviados por la estación móvil. Todos los servicios dentro de la nube son contenerizados. En este apartado se explicará cada componente que comprende la nube, los cuales se observan de forma resumida en la Figura 3.11.

Figura 3.11. *Diagrama de cajas del servidor IOT resumido.*



Nota: Las aplicaciones son desplegadas en contenedores dentro de un servidor de Contabo, el cual es un proveedor de hosting de nubes privadas de bajo costo con el cual tenemos acceso a un entorno virtual donde podemos desplegar aplicaciones de forma remota (Contabo, 2025). Fuente: Elaboración propia utilizando Diagrams.net.

3.4. Componentes de la arquitectura

3.4.1. FastAPI

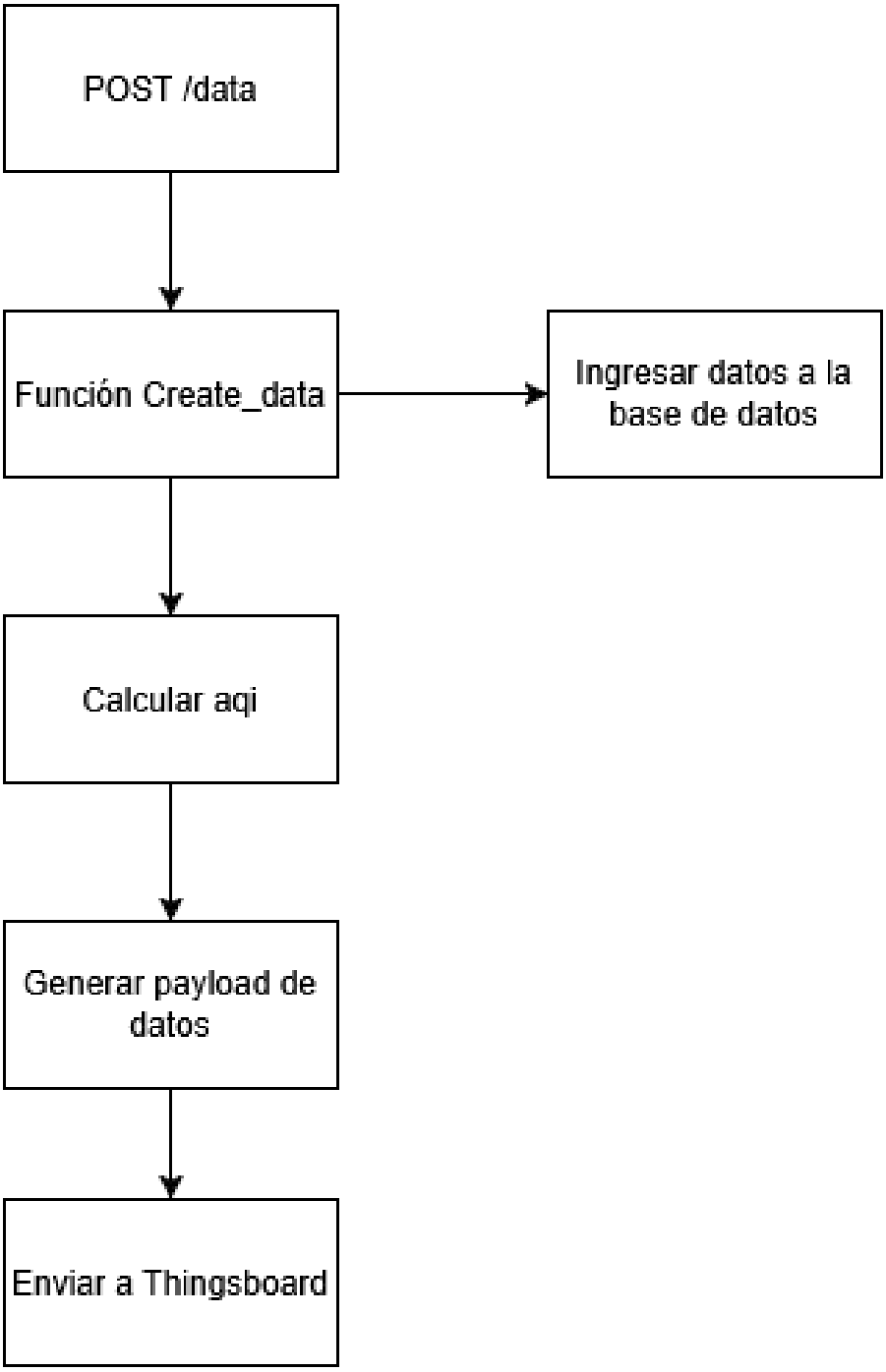
FastAPI es un framework de alto rendimiento para construir APIs con Python. Se basa en Starlette y Pydantic, lo que permite definir endpoints RESTful de forma sencilla, validar automáticamente los datos de entrada y generar una documentación interactiva. (Ramírez S., 2025).
Utilizando FastAPI, hemos definido una ruta “/data” (Figura 3.12), que la estación móvil utiliza para enviar datos hacia el servidor mediante HTTP.

Figura 3.12. *Código de definición de la ruta principal*

```
app = FastAPI(lifespan=lifespan)
@app.post("/data", response_model=schemas.SensorDataResponse)
def create_data(data: schemas.SensorDataCreate, db: Session =   Depends(get_db)):
    result = crud.create_sensor_data(db, data)
    aqi_co2 = aqi_utils.calcular_aqi_individual(data.co2, "co2")
    telemetry = {
        "aqi": aqi_co2,
        "ozono": data.ozono,
        "particulas": data.particulas,
        "co2": data.co2,
        "co": data.co,
        "pms1": data.pms1,
        "pms25": data.pms25,
        "pms10": data.pms10,
        "formaldehido": data.formaldehido,
        "temperatura": data.temperatura,
        "humedad": data.humedad,
        "latitude": data.latitude,
        "longitude": data.longitude,
       "altitud": data.altitud,
        "timestamp": datetime(
            data.year, data.month, data.day,
            data.hour, data.minute, data.second
        ).isoformat()
    }
    # enviar a ThingsBoard Cloud (no bloqueante: solo intento)
    tbcloud.send_to_thingsboard(telemetry)
    return result
```

Nota: Todos los datos a visualizar en el dashboard de Thingsboard se recopilan en un solo objeto llamado “telemetry”. Fuente: Elaboración propia.

Figura 3.13. *Diagrama de bloques de la función "Create_data"*



Nota: La función Create_data se ejecuta automáticamente al acceder a la ruta. Fuente: Elaboración propia utilizando Diagrams.net.

Como se observa en la Figura 3.12, se define el objeto “data” en el cual se guardan temporalmente los datos recién recibidos por la API, estos datos se comparan con un modelo que establece qué nombre deben de tener las variables en la solicitud, cuantas variables tienen que ser y qué formato se espera que contenga cada variable. Luego de ser validados los datos, se guardan dentro de la base de datos por medio de la función “create_sensor_data”.

Luego de esto, de los datos guardados en el objeto “data” se toma el valor de “co2” para poder calcular el Índice de calidad del aire.
Además, se construye un string con cada uno de los datos recolectados, convirtiendo además los datos de tiempo en un solo formato de “timestamp”. Luego de esto se envía hacia thingsboard por medio de una solicitud HTTP POST hacia Thingsboard. Este proceso se resume en el diagrama de la Figura 3.13.

Figura 3.14. *Código de definición del modelo de mensaje JSON.*

```
1. class SensorDataCreate(BaseModel):
2.     ozono: float
3.     particulas: float
4.     co2: float
5.     co: float
6.     pms1: float
7.     pms25: float
8.     pms10: float
9.     formaldehido: float
10.    temperatura: float
11.    humedad: float
12.    latitude: float
13.    longitude: float
14.    altitud: float
15.    year: int
16.    month: int
17    day: int
18.    hour: int
19.    minute: int
20.    second: int
```

Nota: Todos los mensajes que reciba la ruta /data serán comparados con este esquema. Fuente: Elaboración propia.

Figura 3.15. *Código de definición de funciones para generar entradas de base de datos.*

```
1. def create_sensor_data(db: Session, data: schemas.SensorDataCreate):
2.     try:
3.         timestamp = datetime(
4.             year=data.year,
5.             month=data.month,
6.             day=data.day,
7.             hour=data.hour,
8.             minute=data.minute,
9.             second=data.second
10.         )
11.     except ValueError as e:
12.         raise HTTPException(status_code=400, detail=f"Invalid datetime: {e}")
13.
14.     sensor_entry = models.SensorData(
15.         ozono=data.ozono,
16.         particulas=data.particulas,
17.         co=data.co,
18.         co2 = data.co2,
19.         pms1 = data.pms1,
20.         pms25 = data.pms25,
21.         pms10 = data.pms10,
22.         formaldehido = data.formaldehido,
23.         temperatura = data.temperatura,
24.         humedad = data.humedad,
25.         latitude = data.latitude,
26.         longitude = data.longitude,
27.         altitud = data.altitud,
28.         timestamp=timestamp
29.     )
30.     try:
31.         db.add(sensor_entry)
32.         db.commit()
33.         db.refresh(sensor_entry)
34.     except Exception as e:
35.         db.rollback()
36.         raise HTTPException(status_code=500, detail=f"DB insert error: {e}")
37.     return sensor_entry
```

Nota: Al verificar que el tiempo esté en el formato de timestamp entonces se ingresa a la base de datos con db.add y db.commit. Fuente: Elaboración propia.

Figura 3.16. *Código de definición de función para envío de datos a thingsboard.*

```
1. import os
2. import requests
3.
4. THINGSBOARD_URL = os.getenv("THINGSBOARD_URL", "https://thingsboard.cloud")
5. ACCESS_TOKEN = os.getenv("THINGSBOARD_ACCESS_TOKEN", "")
6.
7. def send_to_thingsboard(payload: dict) -> bool:
8.     if not ACCESS_TOKEN:
9.         print("THINGSBOARD token not set; skipping send")
10.        return False
11.    url = f"{THINGSBOARD_URL}/api/v1/{ACCESS_TOKEN}/telemetry"
12.    try:
13.        resp = requests.post(url, json=payload, timeout=5)
14.        resp.raise_for_status()
15.        return True
16.    except requests.RequestException as e:
17.        print(f"Error sending telemetry to ThingsBoard Cloud: {e}")
18.        return False
```

Nota: Para asegurar una buena privacidad y protección contra ataques, el token se encuentra en un archivo .env, del cual se extrae la información usando os.getenv. Fuente: Elaboración propia.

3.4.2. Nginx

Lanzado el 04 de octubre de 2004, Nginx es un servidor web de uso libre y código abierto. Este componente se sitúa en la entrada del servidor, en el puerto 80, recibiendo todas las solicitudes de entrada que se envían desde el internet hacia la IP de nuestro servidor y las redirige hacia el servicio correcto. (Nginx, Inc., 2025)

Figura 3.17. *Archivo de configuración de Nginx.*

```
1. server {
2.     listen 80;
3.     location / {
4.         client_max_body_size 100M;  # Increase limit here
5.         proxy_pass http://web:8000;
6.         proxy_set_header Host $host;
7.         proxy_set_header X-Real-IP $remote_addr;
8.         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
9.     }
10.
11.    location /mlflow/ {
```

```
12.         client_max_body_size 100M;  # Increase limit here
13.         proxy_pass http://mlflow:5000/;
14.         proxy_http_version 1.1;
15.         proxy_set_header Upgrade $http_upgrade;
16.         proxy_set_header Connection keep-alive;
17.         proxy_set_header Host $host;
18.         proxy_cache_bypass $http_upgrade;
19.     }
20. }
```

Nota: la ruta "/" es la ruta base y "/mlflow/" es la ruta al servicio de control de modelos de machine learning. Fuente: Elaboración propia.

Como se observa en la figura 3.17, el servidor de nginx maneja dos bloques de servicios, el servicio de la API y el servicio de Mlflow. En la sintaxis se define el puerto 80 para recibir peticiones http, y luego se empieza a definir cada bloque de servicio comenzando con "location /" el cual seria el directorio base de la ruta del servidor, por ejemplo, http://192.0.2.0/. Dentro de este bloque se define un tamaño máximo de información que recibirá el servidor en cada solicitud. Luego con "proxy_pass" se indica a nginx hacia donde se necesita que se redirija las solicitudes hechas a la ruta http://192.0.2.0/. A través de esta configuración, nginx es capaz de redirigir las solicitudes hechas hacia los endpoints de FastAPI usando la ruta base del servidor mas la ruta del endpoint, por ejemplo, http://192.0.2.0/data redirige hacia http://web:8000/data. El "proxy_set_header X-Fowarded-For" se encarga de guardar un registro de todas las ips asociadas a una solicitud http entrante.

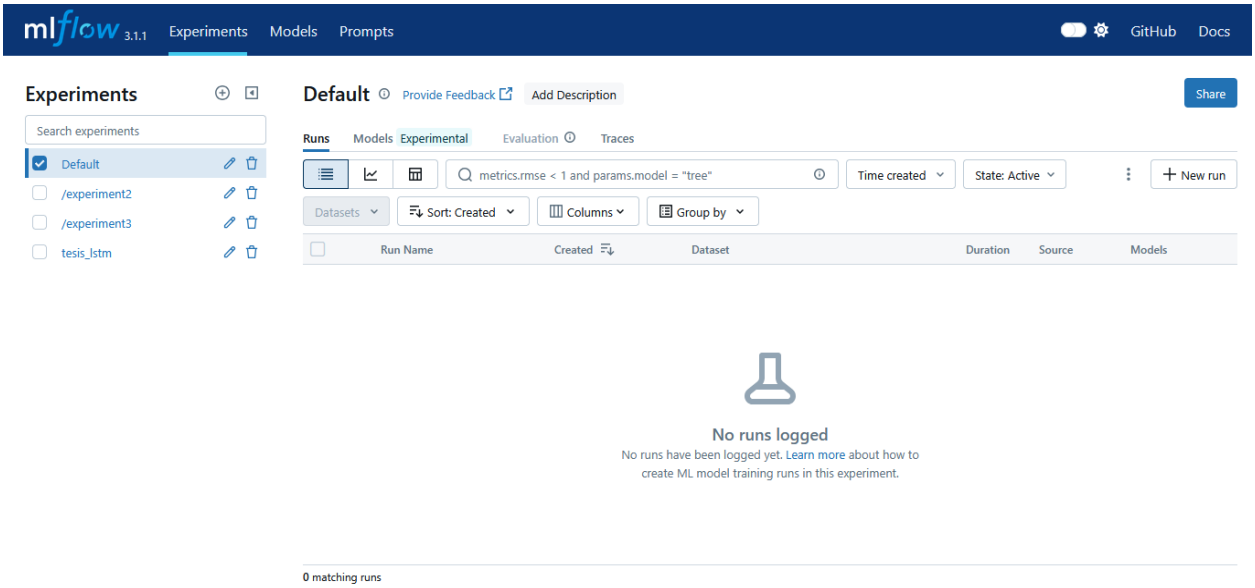
El segundo bloque de servicio es de Mlflow, definiendo la ruta http://192.0.2.0/mlflow como el enlace al que un usuario hace la solicitud para acceder al recurso http://mflow:5000/, en cual es la ruta hacia la interfaz de navegador de Mlflow.

3.4.3. Mlflow

Mlflow es una Plataforma de código abierto para el desarrollo de modelos de machine learning, Deep learning e inteligencia artificial. Ofrece en un solo lugar una interfaz (Figura 3.18) funciones para guardar como artefactos las métricas de entrenamiento, datasets, modelos entrenados diferenciados por versiones los cuales pueden ser descargados de forma remota desde otro dispositivo. Cuenta con integración de diversos proveedores de infraestructura de computación en la nube tales como Amazon, Microsoft, Databricks, entre otros. Dentro del servidor se desplegó un sistema de registro de modelos llamado Mlflow Tracking server el cual utiliza un bucket de

objetos de amazon AWS S3 (o cualquier otra solución de bucket de objetos) como almacenamiento de los artefactos del modelo de machine learning junto con una base de datos local de SQLite para el manejo de metadatos (Zaharia, M., et al, 2018).

Figura 3.18. *Interfaz de usuario de mlflow.*



Nota: En esta página principal se observan los diferentes experimentos que se van creando con cada entrenamiento del modelo. Captura de pantalla de plataforma web del contenedor descargado de Mlflow.org. Fuente: Elaboración propia.

En el siguiente bloque de la Figura 3.19, mediante “mlflow.set_tracking_uri” se configura el enlace del servidor al que el compilador intentará conectarse para acceder al Tracking server de mlflow. Luego con “mlflow.set_experiment” se define la carpeta interna de mlflow donde se registrará todos los parámetros del entrenamiento actual que se hayan definido dentro de “mlflow.start_run() as run:”. La librería de mlflow contiene soporte para diferentes paquetes de modelos de ml. Puesto que nuestro modelo a utilizar es un modelo de pytorch, utilizamos “mlflow.pytorch.log_model” para el registro del modelo entrenado, marcándolo con el nombre de “test1cpu”. Este proceso se reduce al diagrama de la Figura 3.20.

Figura 3.19. *Código utilizado para registrar modelo en mlflow.*

```
1. # Configurar URI del tracking server remoto
2. mlflow.set_tracking_uri("http://example.com/mlflow/")
3. mlflow.set_experiment("/experiment2")
4. with mlflow.start_run() as run:
5.   mlflow.log_param("epochs", 50)
6.   mlflow.log_param("hidden_dim", 32)
7.   mlflow.log_metric("final_loss", loss.item())
8.   # Guardar modelo en artefactos sin invocar el registry
9.   mlflow.pytorch.log_model(
10.      pytorch_model=model,
11.      name="test1cpu"
12.  )
13.  print(f"Modelo registrado en run_id={run.info.run_id}")
```

Nota: La dirección real del enlace al servicio de mlflow fue reemplazada con “example.com” para mantener la privacidad del servidor. Fuente: Elaboración propia.

Figura 3.21. *Uso de mlflow para el descargado del modelo.*

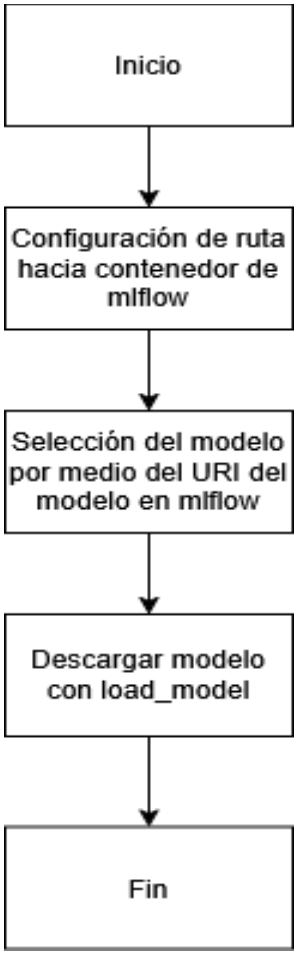
```
1.  MLFLOW_TRACKING_URI = "http://mlflow:5000"  # o IP externa si no está
en el mismo docker network
2.  model_name = os.getenv("MODEL_NAME", "test1cpu")
3.  model_version = "latest"
4.  model_uri = f"models:/{model_name}/{model_version}"
5.
6.  FEATURE_COLS = [
7.      'particles', 'ozone', 'co', 'pms_1_0', 'pms_2_5', 'pms_10',
8.      'formaldehyde', 'temperature', 'humidity'  # Variables que usarás
como entrada
9.  ]
10.
11. mlflow.set_tracking_uri(MLFLOW_TRACKING_URI)
12.
13. model = mlflow.pytorch.load_model(model_uri,
map_location=torch.device('cpu'))
```

Nota: torch.device(‘cpu’) asegura que el modelo se ejecute usando la versión de cpu del modelo, puesto que el servidor no cuenta con hardware de GPU. Fuente: Elaboración propia.

Figura 3.20. *Diagrama de cajas para el registro de un modelo en mlflow.*

Nota: La configuración de la ruta hacia el servidor puede realizarse también antes de empezar a entrenar el modelo. Fuente: Elaboración propia utilizando Diagrams.net.

Figura 3.22. *Diagrama de cajas para el proceso de descarga del modelo.*



Nota: La ruta hacia el contenedor de mlflow (http://mlflow:5000) es diferente a la ruta hacia el servicio del servidor (example.com/mlflow). Fuente: Elaboración propia utilizando Diagrams.net.

Una vez registrado el modelo dentro del Tracking Server, se puede utilizar de forma remota desde cualquier máquina o servidor mediante la función “mlflow.load_model”, pasando como parámetro la ruta del modelo registrado dentro del Tracking Server, con el formato de “models:/{nombre_del_modelo}/{version_del_modelo}". Al ejecutarse el modelo se descarga y se guarda dentro de la variable “model” (Figura 3.21). Este proceso de resume en el diagrama de la Figura 3.22.

Teniendo el modelo descargado, puede luego ser utilizado para generar predicciones. En la porción de código siguiente se observa una entrada X, la cual es un vector que contiene los últimos 12 datos de CO_2 enviados por la estación móvil. Estos datos pasan por un escalador para convertirlos

a una escala de 0 a 1 y posteriormente convertidos a un tensor de pytorch. Con esta secuencia se utiliza el modelo ya entrenado con la función “model(inp)” (Figura 3.23). El resultado de esto se convierte de regreso a la escala original y este valor predicho de CO_2 se utiliza como parámetro para la función de cálculo del índice de calidad de aire para obtener un pronóstico del valor del índice, el cual luego es enviado hacia thingsboard junto con el valor de CO_2 obtenido del modelo. Este proceso se resume en el diagrama de la Figura 3.24.

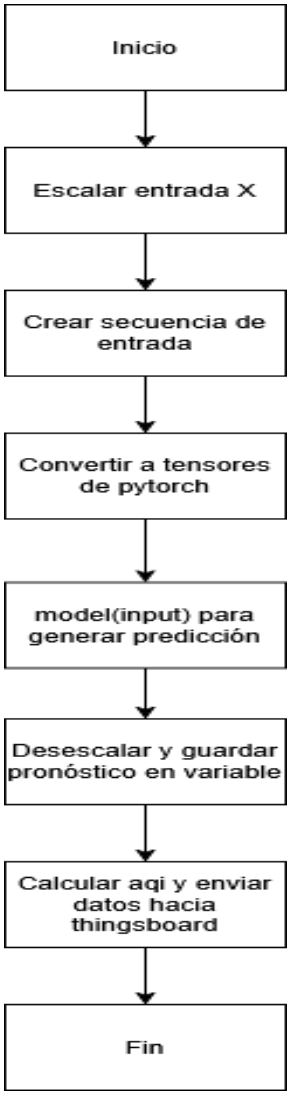
Figura 3.23. *Uso del modelo descargado de mlflow*

```
1.  X_scaled = scaler_X.transform(X)
2.  y_scaled = scaler_y.transform(y)
3.  #Crear secuencia de entrada
4.  input_seq = X_scaled[-12:]  # últimas 12 horas
5.  inp = torch.tensor(input_seq, dtype=torch.float32).unsqueeze(0).to(device)

6.  #Predicción
7.  model.eval()
8.  with torch.no_grad():
9.      pred_scaled = model(inp).cpu().numpy()[0]
10. #Desescalar
11. pred_real = scaler_y.inverse_transform(pred_scaled.reshape(1, -1)).flatten()
12. # # Obtener la predicción del horizonte
13. ozone_prediction = float(pred_real[0])
14.
15. ##Calcular AQI
16. aqi_prediction = float((aqi_utils.calcular_aqi_individual(ozone_prediction,
"co2")))
17.
18. ###Envío de datos
19. telemetry = {"future_co2" : round(ozone_prediction,2),
                    "future_aqi" : round(aqi_prediction,2)}
20.
21. tbcloud.send_to_thingsboard(telemetry)
```

Nota: model.eval() activa el modo de evaluación del modelo de machine learning, lo que evita que utilice funciones y procesos relacionados con el entrenamiento de este, evitando el uso innecesario de recursos. Fuente: Elaboración propia.

Figura 3.24. *Diagrama de cajas de proceso de uso del modelo de machine learning*



Nota: El escalador utilizado en este proceso será el mismo que se utilizó durante el proceso de entrenamiento del modelo de machine learning. Fuente: Elaboración propia utilizando Diagrams.net.

3.4.4. TimescaleDB

TimescaleDB es una extensión de PostgreSQL, que permite agrupar datos según la fecha y hora en la que se crearon en tablas llamadas Hypertables, las cuales son tablas que contienen más tablas. El constante flujo de datos que envía el dispositivo a lo largo del tiempo terminan generando cientos de miles de entradas en una base de datos tradicional; TimescaleDB ofrece funciones,

operadores e índices que permiten trabajar con esos datos según el intervalo de tiempo que se necesita, facilitando la lectura de los mismos.

Para crear la hypertable se utiliza primero la sintaxis normal de SQL para crear una base de datos con “CREATE EXTENSION” (figura 3.25). Luego se crea una tabla normal con “CREATE TABLE” la cual contendrá todas las variables medidas por la estación móvil. Es importante incluir en esta tabla una columna con el formato TIMESTAPTZ, la cual es la que se utiliza para poder crear la hypertable, utilizando la función de timescaledb “create_hypertable”. Esto se logra observar en el siguiente bloque de código:

Figura 3.25. *Código de creación de una base de datos PostgreSQL y TimescaleDB.*

```
1.  CREATE EXTENSION IF NOT EXISTS timescaledb;

2.  -- Esto se ejecuta automáticamente al levantar la base de datos
3.  CREATE TABLE IF NOT EXISTS sensor_data (
4.    id SERIAL,
5.    timestamp TIMESTAMPTZ NOT NULL,
6.    ozono DOUBLE PRECISION,
7.    particulas DOUBLE PRECISION,
8.    co2 DOUBLE PRECISION,
9.    co DOUBLE PRECISION,
10.   pms1 DOUBLE PRECISION,
11.   pms25 DOUBLE PRECISION,
12.   pms10 DOUBLE PRECISION,
13.   formaldehido DOUBLE PRECISION,
14.   temperatura DOUBLE PRECISION,
15.   humedad DOUBLE PRECISION,
16.   latitude DOUBLE PRECISION,
17.   longitude DOUBLE PRECISION,
18.   altitud DOUBLE PRECISION
19.  );
20.
21.  SELECT create_hypertable('sensor_data', 'timestamp', if_not_exists => TRUE);
```

Nota: El proceso generará automáticamente la base de datos al arrancar el proceso del contenedor de la API. Si esta ya existe, procede a levantarla sin generar una nueva. Fuente: Elaboración propia.

Una vez generada la base de datos, se puede acceder a ella utilizando el URL de formato “postgresql://{usuario}:{contraseña}@{db_host}:{db_port}/{db_name}” (figura 3.26), la cual es utilizada para crear el motor de base de datos dentro de la aplicación de fastapi, y al crear este motor, se utiliza para crear sesiones donde se establece conexión temporalmente con la base de datos.

Figura 3.26. *Código para establecer conexión con la base de datos dentro del servidor.*

```
DB_USER = os.getenv("POSTGRES_USER", "postgres")
DB_PASS = os.getenv("POSTGRES_PASSWORD", "postgres")
DB_HOST = os.getenv("POSTGRES_HOST", "db")
DB_PORT = os.getenv("POSTGRES_PORT", "5432")
DB_NAME = os.getenv("POSTGRES_DB", "sensordata")
DATABASE_URL = f"postgresql://{DB_USER}:{DB_PASS}@{DB_HOST}:{DB_PORT}/{DB_NAME}"
engine = create_engine(DATABASE_URL, pool_pre_ping=True)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()
def get_db():
       db = SessionLocal()
       try:
       yield db
       finally:
       db.close()
```

Nota: Por privacidad y seguridad, las credenciales de la base de datos se guardan en un archivo. env las cuales son recuperadas utilizando os.getenv. Fuente: Elaboración propia.

Dentro de la ruta /data, los datos enviados por la estación móvil se guardan en la base de datos utilizando la función “create_sensor_data” (figura 3.27). Puesto que los datos de tiempo generados por el módulo de GPS son guardados y enviados en variables separadas de año, mes, día, hora, minuto, segundo, es necesario convertirlos en formato timestamp antes de introducirlos a la base de datos. Luego de esto se recopilan las variables de las mediciones y junto con el timestamp se envían a la base de datos usando las funciones “add” y “commit”.

Figura 3.27. *Definición de la función “Create_sensor_data”*

```
def create_sensor_data(db: Session, data: schemas.SensorDataCreate):
       try:
         timestamp = datetime(
             year=data.year,
             month=data.month,
             day=data.day,
             hour=data.hour,
             minute=data.minute,
                second=data.second
       )
       except ValueError as e:
       raise HTTPException(status_code=400, detail=f"Invalid datetime: {e}")
```

```
        sensor_entry = models.SensorData(
        ozono=data.ozono,
        particulas=data.particulas,
        co=data.co,
        co2 = data.co2,
        pms1 = data.pms1,
        pms25 = data.pms25,
        pms10 = data.pms10,
        formaldehido = data.formaldehido,
        temperatura = data.temperatura,
        humedad = data.humedad,
        latitude = data.latitude,
          longitude = data.longitude,
        altitud = data.altitud,
          timestamp=timestamp
        )
        try:
          db.add(sensor_entry)
          db.commit()
          db.refresh(sensor_entry)
        except Exception as e:
          db.rollback()
        raise HTTPException(status_code=500, detail=f"DB insert error: {e}")
        return sensor_entry
```

Nota: Al verificar que el tiempo esté en el formato de timestamp entonces se ingresa a la base de datos con db.add y db.commit. Fuente: Elaboración propia.

Al momento de generar predicciones con el modelo de machine learning, se necesita poder recuperar los últimos datos guardados en la base de datos. Creamos una función llamada "get_sensor_data" (figura 3.28) para este propósito. La cual recupera las últimas 12 filas (se puede ajustar con la variable "limit") de la base de datos en orden descendente y las guarda en un dataframe de pandas.

Figura 3.28. *Definición de la función "get_sensor_data"*

```
def get_sensor_data(db: Session, limit: int = 12, ascending: bool = False):
        """
        Obtiene los últimos registros de la tabla sensor_data.
        """
        try:
        query = db.query(models.SensorData).order_by(
```

```
            models.SensorData.timestamp.asc() if ascending else
models.SensorData.timestamp.desc()
        ).limit(limit)
       data = query.all()

       if not data:
              raise HTTPException(status_code=404, detail="No hay datos almacenados
en la base de datos.")

       df = pd.DataFrame([d.__dict__ for d in data])
       df = df.drop(columns=["_sa_instance_state"], errors="ignore")
       return df

       except Exception as e:
       raise HTTPException(status_code=500, detail=f"Error al obtener datos: {e}")
```

Nota: Cuando la variable ascending es verdadera, los datos se recuperan en orden ascendente, por lo que siempre recuperará los primeros registros, cuando es falsa, recupera los últimos registros. Fuente: Elaboración propia.

3.4.5. APScheduler

Advanced Python Scheduler (APScheduler) es una librería de python que permite automatizar la ejecución de funciones agendándolas para repetirse en un intervalo de tiempo en concreto ya sea 1 segundo, 1 minuto, 1 hora, etc. FastAPI permite una integración sencilla con esta librería, utilizando la función de @asyncontextmanager (figura 3.29), dentro de la cual se define la función "lifespan" de APScheduler. Se define el modelo de ejecución, la función a ejecutar, sus parámetros y el intervalo de tiempo en el que se ejecutará esta tarea.

Luego al inicializar la aplicación de fastAPI, arranca también APScheduler con las tareas que se definieron y se empieza a ejecutar en el intervalo establecido. Se crea una tarea de la siguiente forma:

Figura 3.29. *Creación de una tarea de APScheduler.*

```
@asynccontextmanager
async def lifespan(app:FastAPI):
    scheduler = AsyncIOScheduler()
    scheduler.add_job(predict_co2, "interval", args=[model],  minutes = 1)
    scheduler.start()
    yield
```

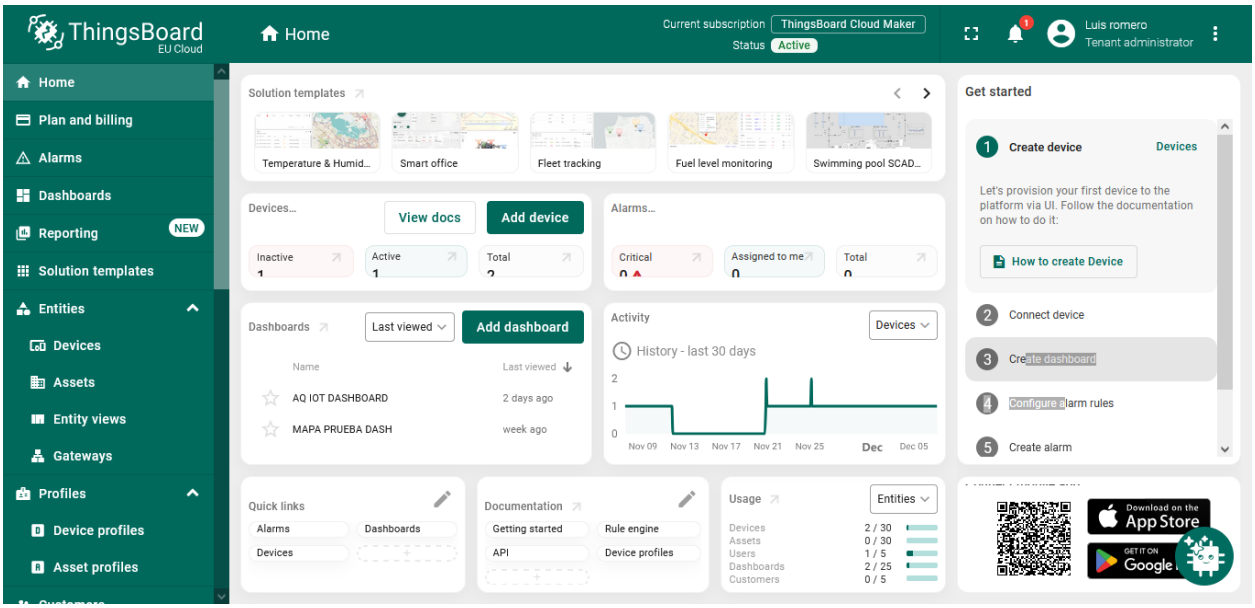
```
app = FastAPI(lifespan=lifespan)
```

Nota: El argumento “lifespan” permite que una función (en este caso, la función “lifespan”) pueda ser ejecutada al levantar el servicio de Fast Api. Fuente: Elaboración propia.

3.4.6. Thingsboard

Thingsboard es una plataforma de código abierto para sistemas IOT que permite recolectar datos, procesarlos y visualizarlos, así como también clasificar y manejar dispositivos y usuarios. La visualización de datos se realiza por medio de un dashboard, en el que se permite utilizar widgets que reciben los datos enviados por los dispositivos y mostrar con ellos gráficas, tablas y mapas. La edición de Thingsboard community permite de forma gratuita desplegar la plataforma de forma local y luego una edición llamada Thingsboard Cloud (Figura 3.30) ofrece acceso a la plataforma desde internet por medio del navegador, lo cual facilita su uso, por el costo de una suscripción de $10 al mes. (Thingsboard, Inc., 2025)

Figura 3.30. *Página de inicio de Thingsboard Cloud.*

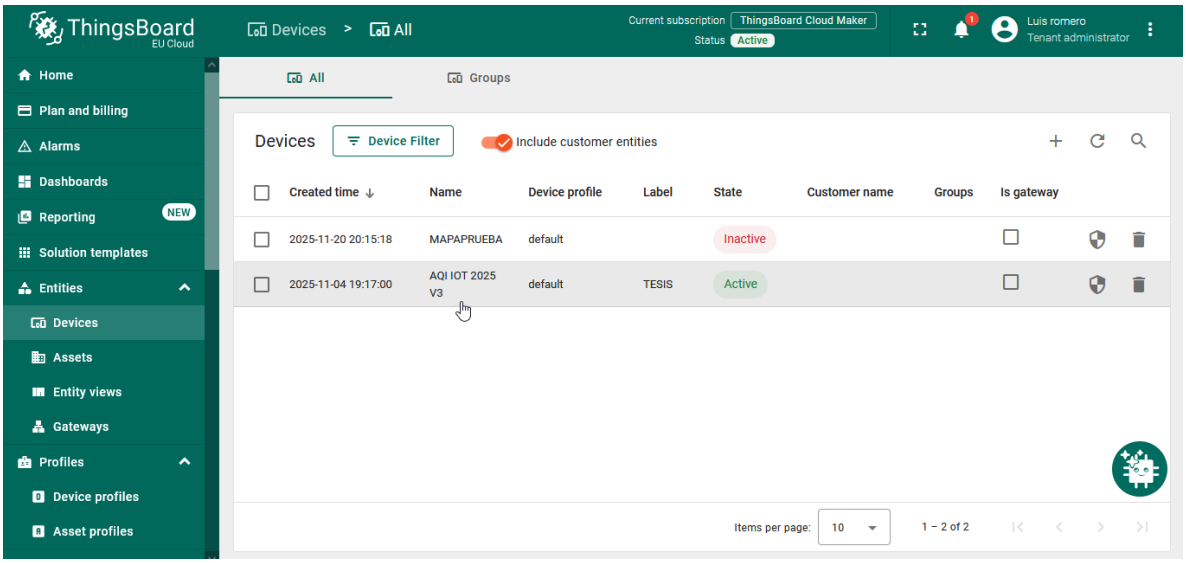


Nota: La versión Cloud de Thingsboard cuenta con una prueba gratuita de un mes. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

3.5. Creación de un dispositivo

Nos dirigimos hacia la pestaña de “Devices” (Figura 3.31) y encontramos una lista de todos los dispositivos configurados en la plataforma.

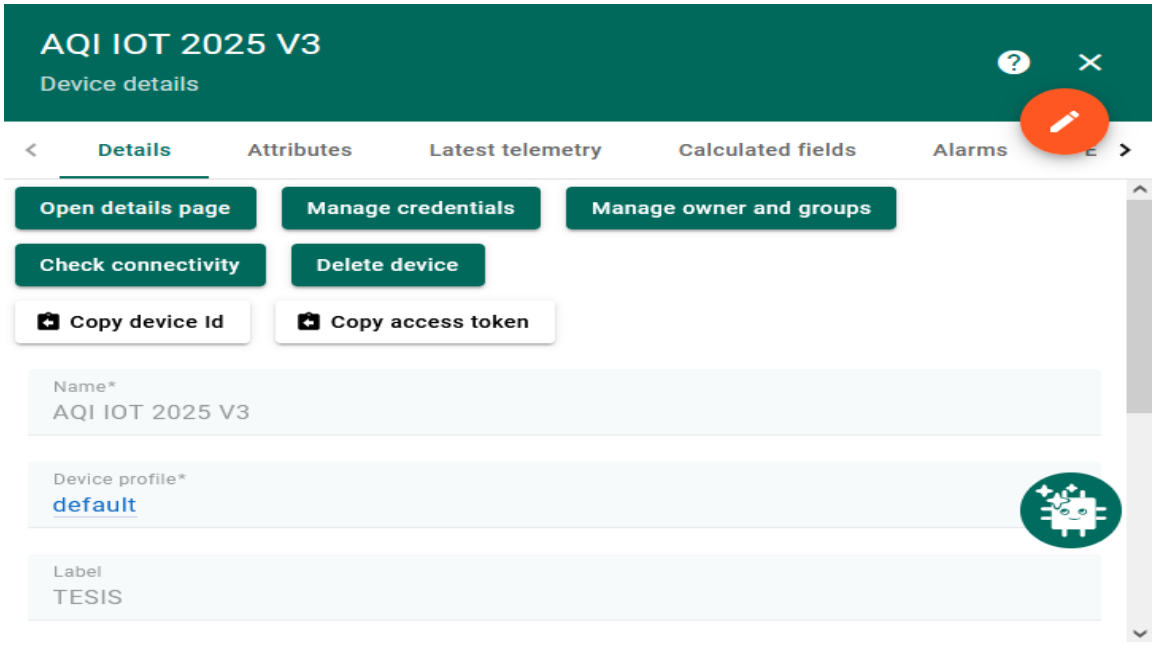
Figura 3.31. *Página de manejo de dispositivos en Thingsboard.*



Nota: Cada dispositivo nos muestra si este se encuentra activo (enviando datos) o inactivo (sin enviar datos). Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

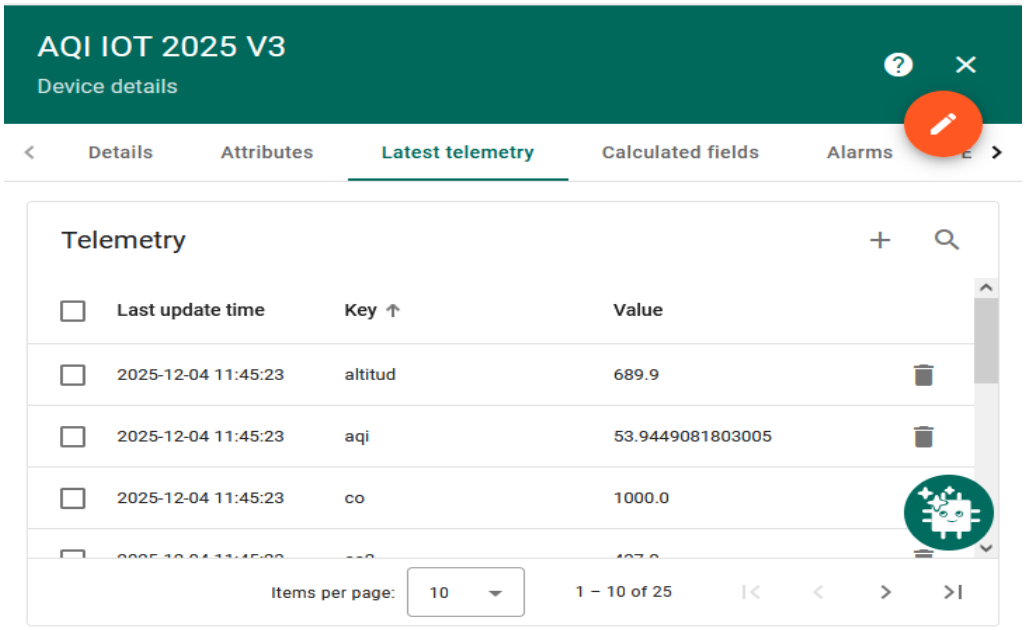
Al seleccionar uno, se nos muestran detalles de credenciales, id del dispositivo, token de dispositivo, opción para verificar la conectividad con el dispositivo (Figura 3.32). y además se pueden ver los últimos datos recibidos (Figura 3.33).

Figura 3.32. *Ventana de información de un dispositivo.*



Nota: Desde esta ventana podemos copiar el token de acceso del dispositivo. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

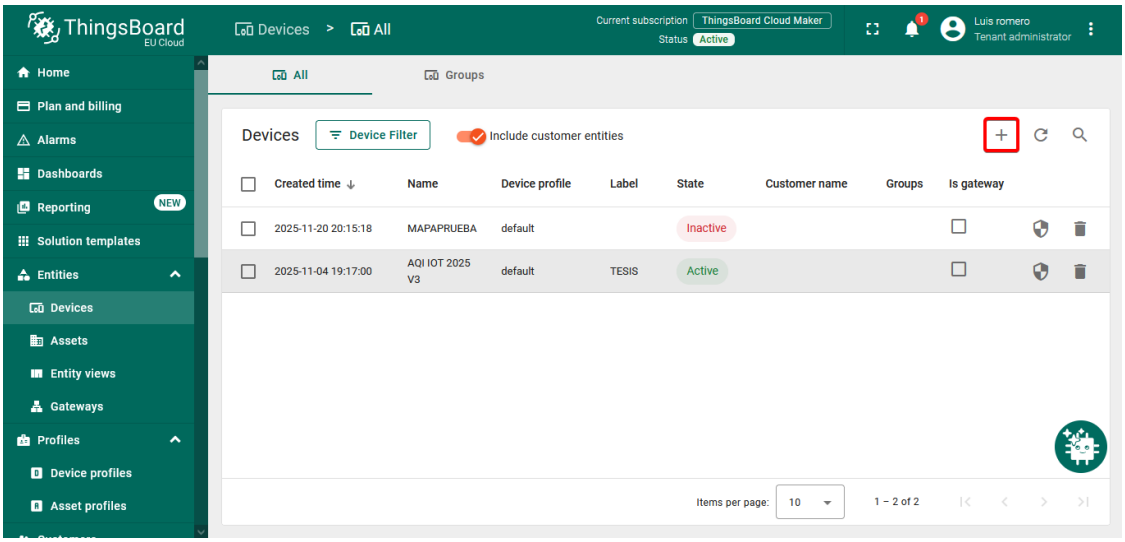
Figura 3.33. *Ventana para observar los últimos datos recibidos en thingsboard.*



Nota: Al enviar variables por primera vez, estas quedan registradas automaticamente. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

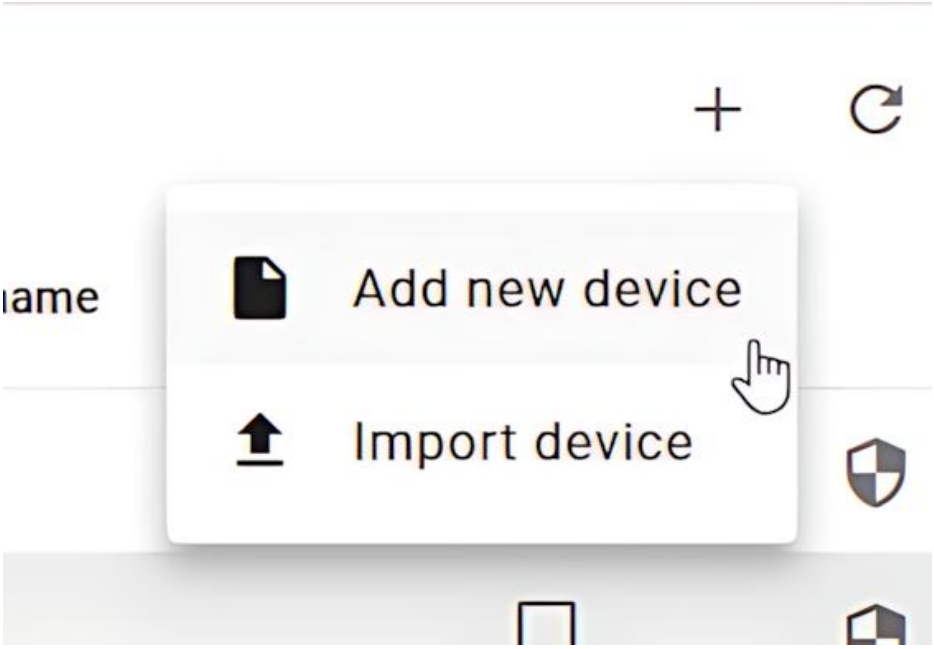
Para crear un nuevo dispositivo, nos dirigimos a la página de los dispositivos, se presiona el botón "+" (Figura 3.34) y luego se elige la opción "add new device" (Figura 3.35). Esto nos lleva a la ventana de creación del dispositivo (Figura 3.36), donde podemos asignar un nombre al dispositivo y una etiqueta que lo identifique. Además de esto podemos tomar nota del token único para este dispositivo (Figura 3.37).

Figura 3.34. *Primer paso de la creación de un nuevo dispositivo.*



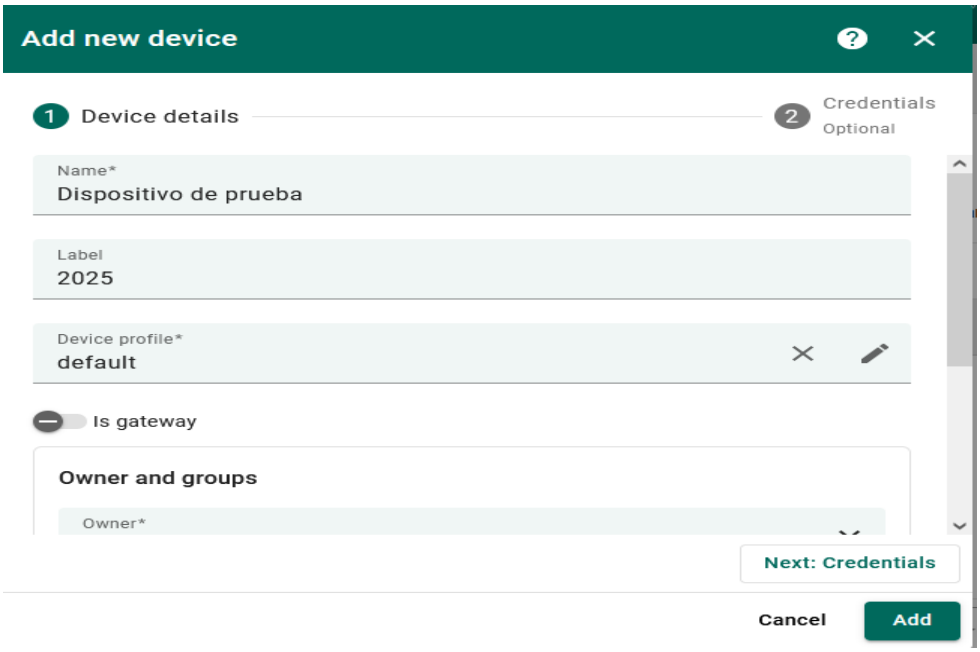
Nota: En este paso nos dirigimos a la pestaña de dispositivos. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.35. *Segundo paso para la creación de un dispositivo.*



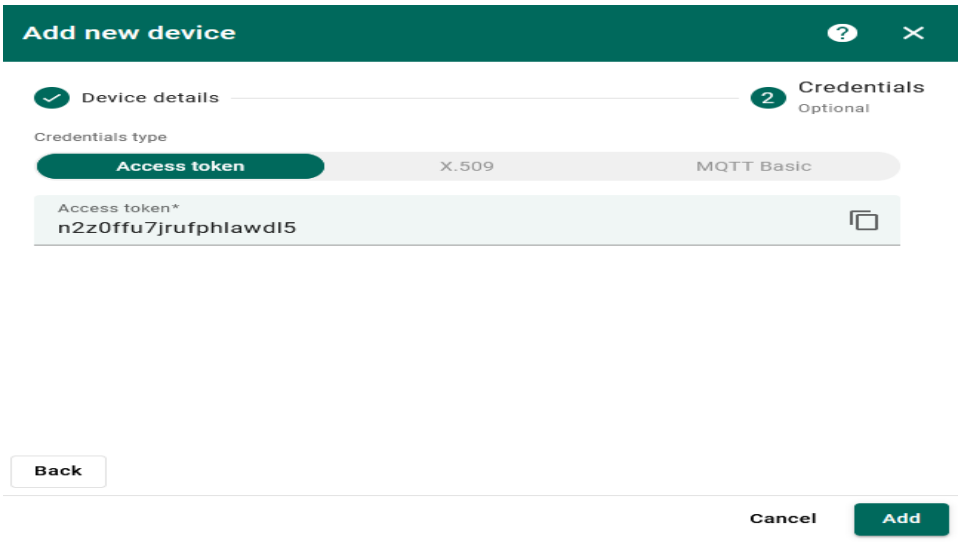
Nota: En este paso se selecciona la opción de añadir nuevo dispositivo. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.36. *Tercer paso de la creación de un dispositivo.*



Nota: En este paso se le da un nombre al dispositivo. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

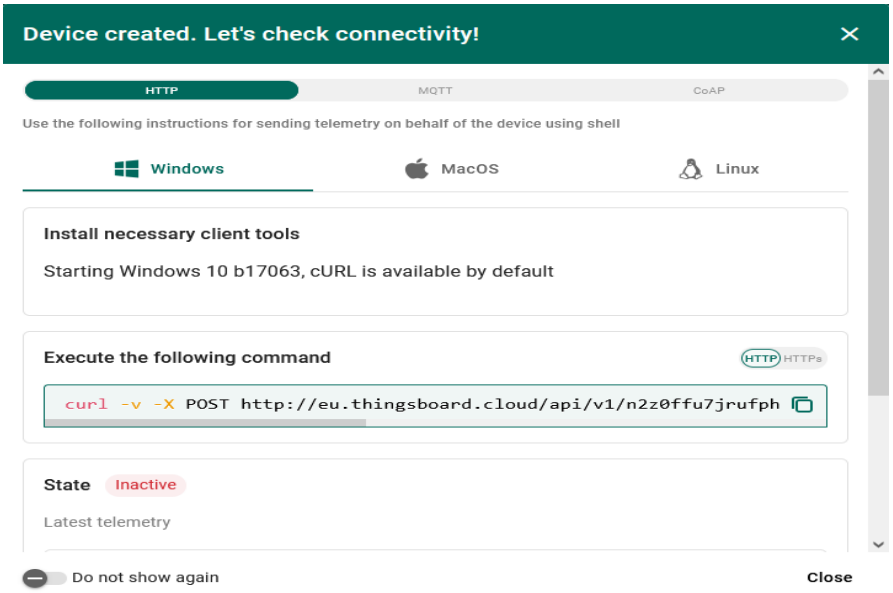
Figura 3.37. *Cuarto paso de la creación de un dispositivo.*



Nota: En este paso se puede editar el token de acceso al dispositivo. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Una vez configurado, se presiona el botón “add” y esto nos lleva a una ventana confirmando que el dispositivo fue creado. En esta ventana se nos da la opción también de verificar que si se puede enviar datos hacia thingsboard utilizando el token de este dispositivo mediante una petición HTTP/HTTPS (Figura 3.38).

Figura 3.38. *Ventana de confirmación de creación del dispositivo.*



Nota: Se muestra una opción de comando de prueba para tres sistemas operativos diferentes: Windows, MacOS y Linux. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.39. *Resultado de prueba de telemetría.*

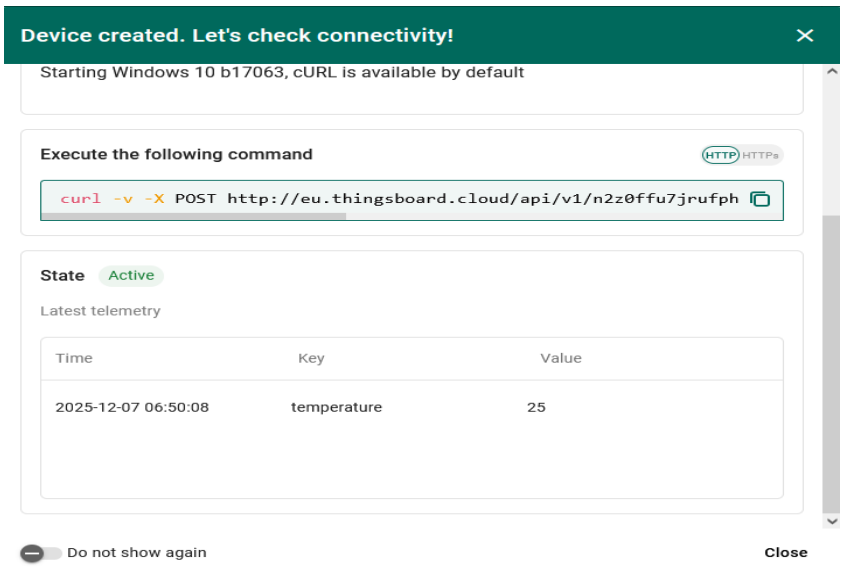
```
C:\Users\luisr>curl -v -X POST
http/eu.thingsboard.cloud/api/v1/n2z0ffu7jrufphlawdl5/telemetry --header Content-
Type:application/json --data "(temperature:25)"
Note: Unnecessary use of -X or --request, POST is already inferred.
* Host eu.thingsboard.cloud:80 was resolved.
* IPv6: (none)
* IPv4:
*   Trying
* Connected to eu.thingsboard.cloud (        ) port 80
* using HTTP/1.x
> POST /api/v1/n2z0ffu7jrufphlawd15/telemetry HTTP/1.1
> Host: eu.thingsboard.cloud
> User-Agent: curl/8.13.0
> Accept: */*
> Content-Type:application/json
> Content-Lenght: 16
>
* upload completely sent off: 16 bytes
< HTTP/1.1 200
< vary: Origin
```

```
< vary: Access-Control-Request-Method
< vary: Access-Control-Request-Headers
< x-content-type-options: nosniff
< x-xss-protection: 0
< cache-control: no-cache, no-store, max-age=0, must-revalidate
< pragma: no-cache
< expires: 0
< x-frame-options: DENY
< content-lenght: 0
< date: Sun, 07 Dec 2025 12:50:08 GMT
<
* Connection #0 to host eu.thingsboard.cloud left intact
```

Nota: Ejecución del comando propuesto por thingsboard para verificar la conectividad con la plataforma. Por privacidad, las ips han sido censuradas. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Al ejecutar el comando propuesto por thingsboard (Figura 3.39), se observa en la misma ventana el valor de temperatura enviado en la petición HTTP POST (Figura 3.40).

Figura 3.40. *Confirmación del recibimiento del valor enviado hacia thingsboard*

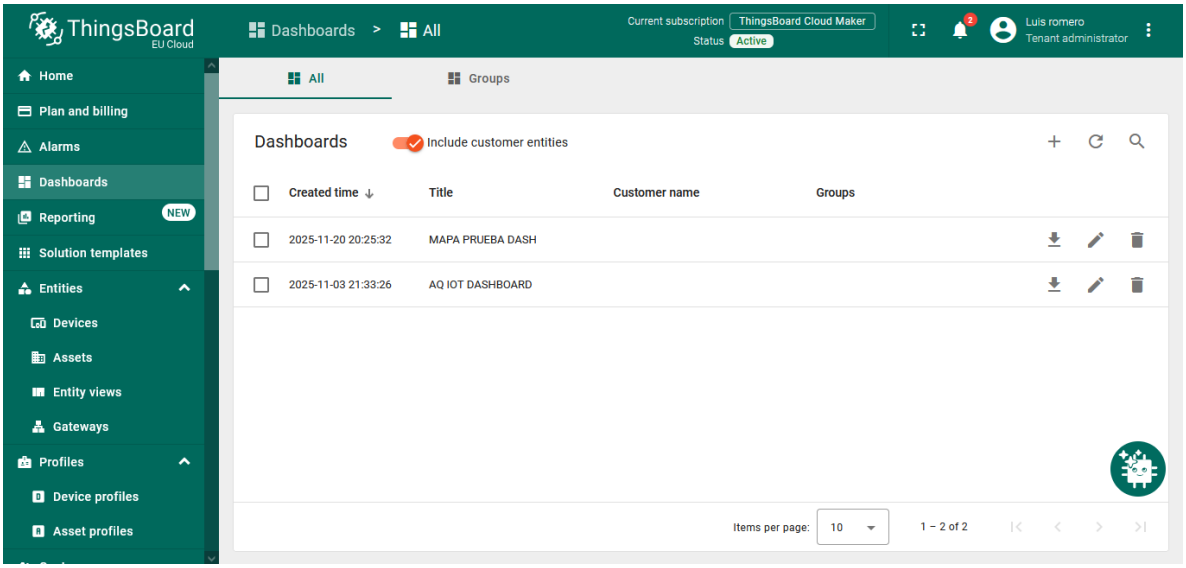


Nota: El mensaje JSON enviado a través del comando de prueba de la Figura 3.38 contiene una variable de temperatura con un valor de 25 unidades. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

3.6. Creación de un Dashboard

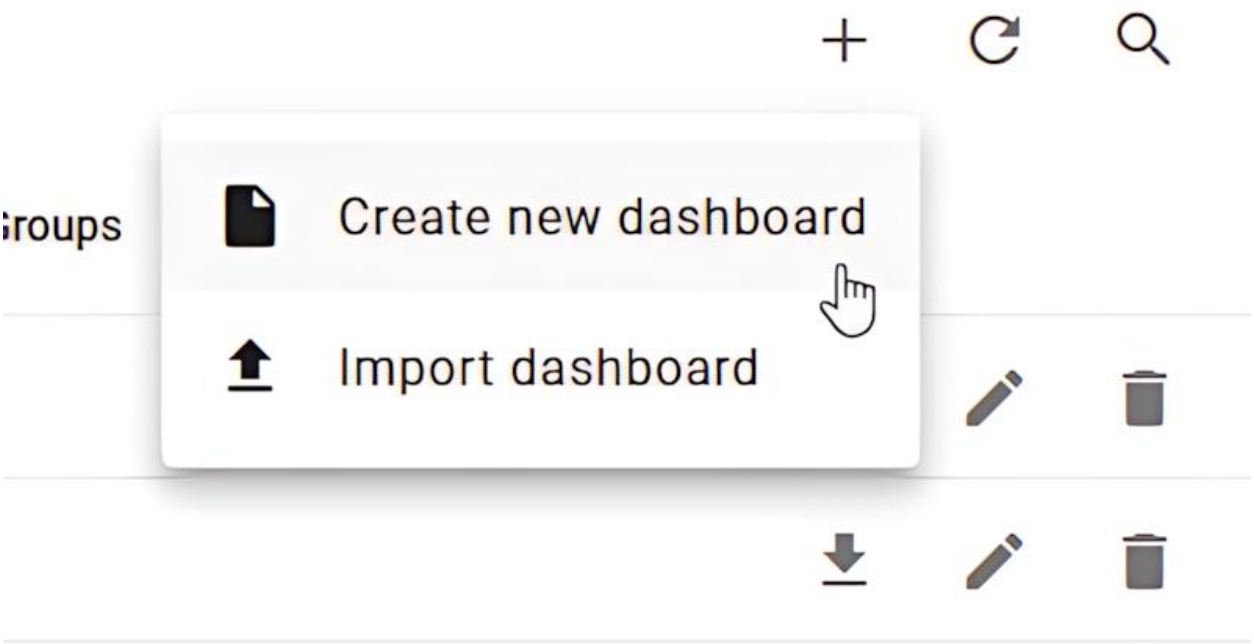
Comenzamos en la pestaña de manejo de Dashboards (Figura 3.41), en esta página encontramos una lista de todos los dashboards creados. Para crear uno nuevo, nos dirigimos al botón de “+” y elegimos la opción “create new dashboard” (Figura 3.42).

Figura 3.41. *Página de manejo de Dashboards en Thingsboard.*



Nota: En este paso nos dirigimos a la pestaña de Dashboards. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

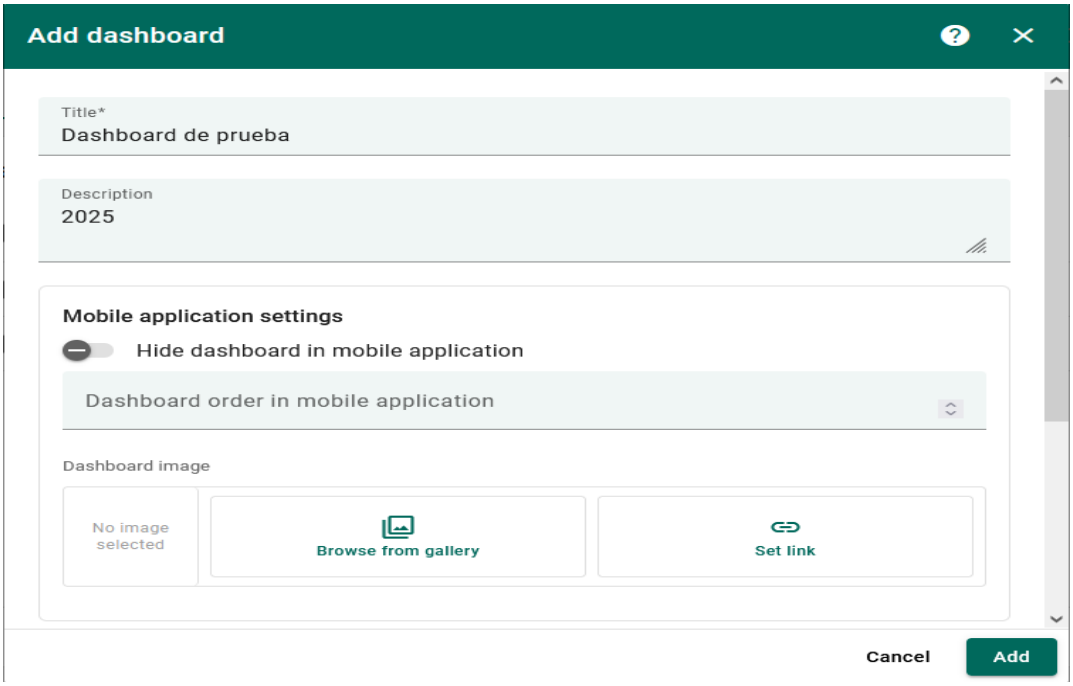
Figura 3.42. *Primer paso de creación de un Dashboard.*



Nota: En este paso seleccionamos la opción de crear nuevo dashboard. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

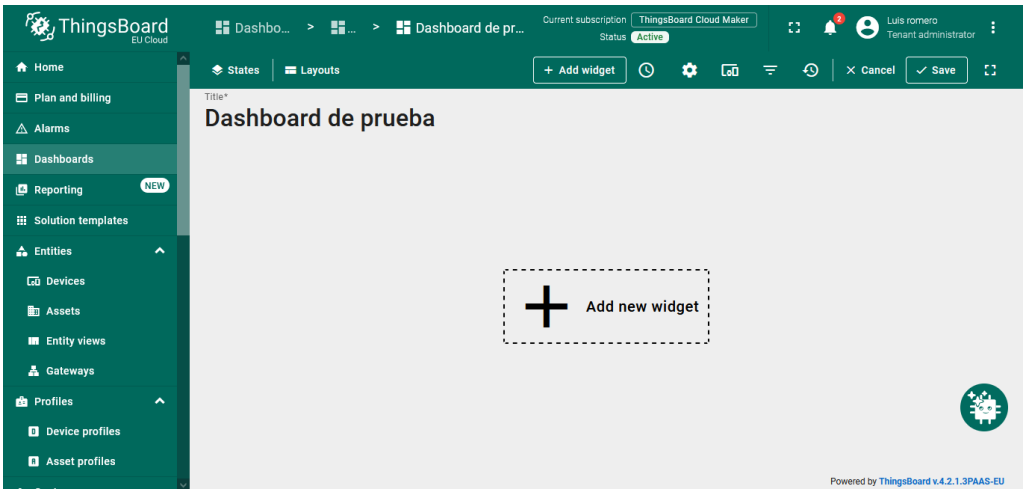
Esto nos lleva a la ventana de creación de un dashboard, en la cual podemos asignarle un nombre, una descripción asi como también una imagen que lo identifique (Figura 3.43). Una vez configurado esto, se presiona el botón “add” y se nos lleva directamente a la página de edición del dashboard (Figura 3.44).

Figura 3.43. *Segundo paso de creación de un Dashboard.*



Nota. En este paso podemos darle un nombre al dashboard, una descripción, una imagen representativa y un enlace personalizado. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.44. *Tercer paso de creación de un dashboard, presionar "Add Widget"().*



Nota: En este paso, nos encontramos dentro de la página de edición del dashboard, donde al no tener ningún widget, se nos muestra en grande la opción de añadir un nuevo widget. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

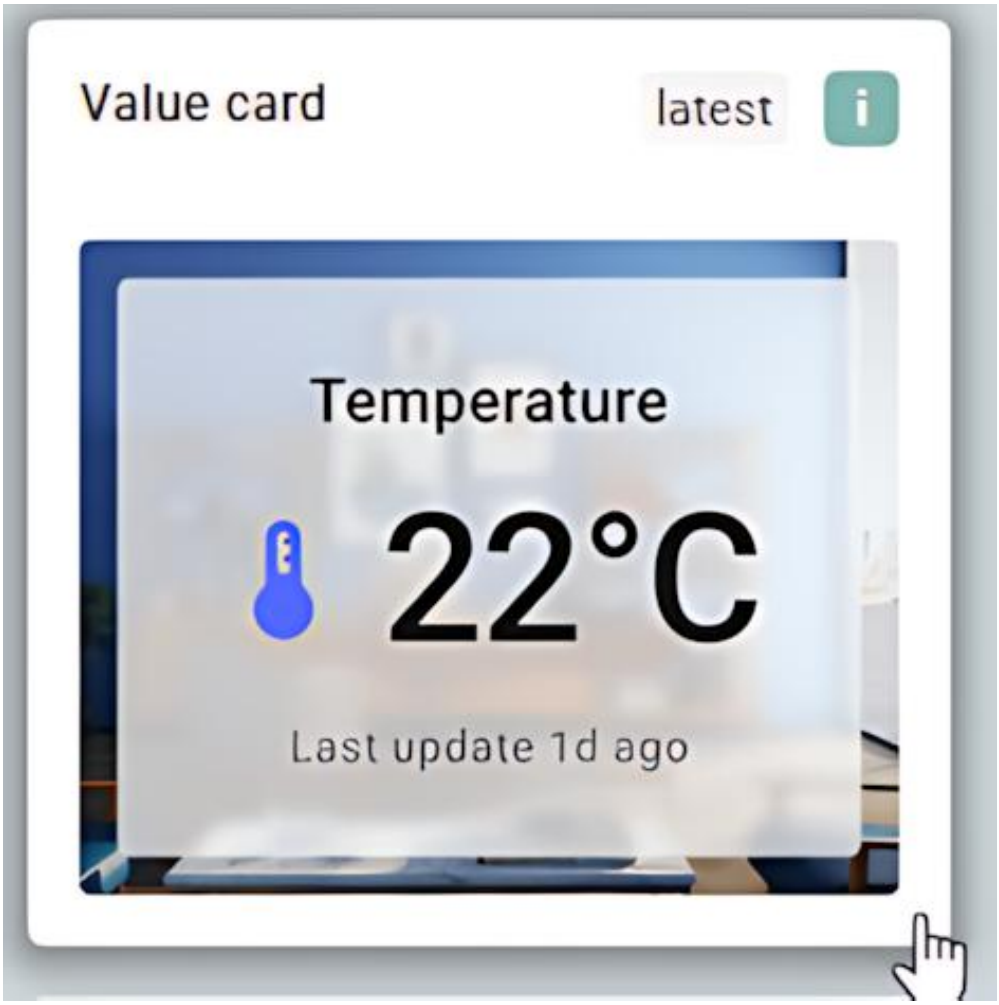
Dentro de esta página del dashboard, tenemos la opción de crear los objetos (llamados “widgets”) donde se mostrarán los datos de telemetría. Para añadir un nuevo widget presionamos la opción “Add widget”. Esto nos lleva a una ventana donde encontramos una amplia selección de widgets ordenadas según su tipo y su uso. Tenemos por ejemplo gráficos, tarjetas que muestran un valor, tablas, mapas, etc (Figura 3.45).

Al seleccionar un widget se abre la ventana de configuración del widget, donde podemos elegir el dispositivo de donde se tomará el dato (Figura 3.47) y elegir cual dato que envía ese dispositivo se mostrará en el widget. Para la estación móvil, podría ser cualquiera de los datos de contaminantes, temperatura y humedad o datos de gps (Figura 3.48). Una vez elegida la variable, se elige un título para el widget, se puede elegir un ícono que represente el dato así como también poner las unidades que le corresponden (Figura 3.49). Una vez se termina la configuración, se agrega automáticamente el widget al layout del dashboard. (Figura 3.50)

Figura 3.45. *Ventana de selección de widgets del dashboard.*

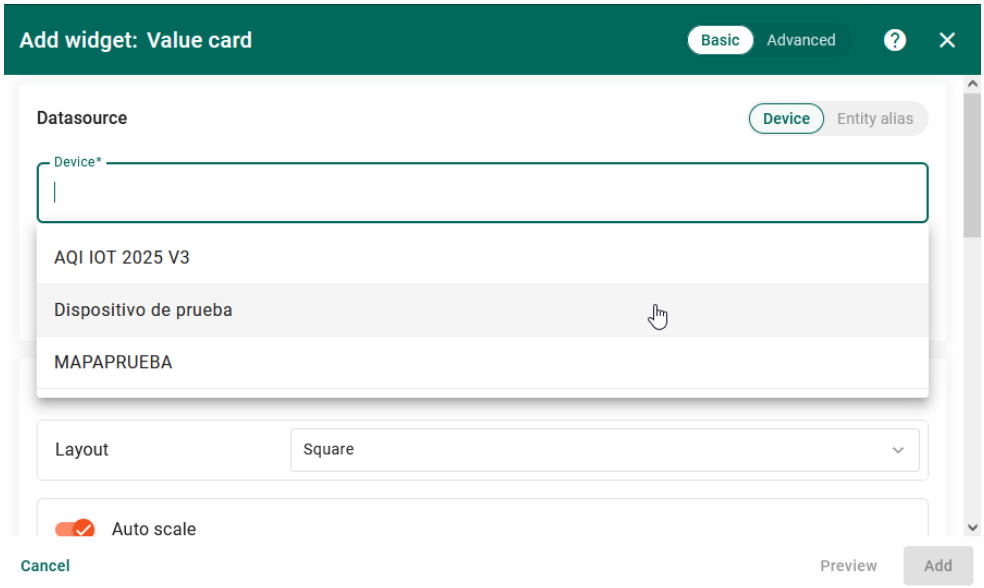
Nota: Los widget se pueden visualizar según su categoría, o también todos los widgets disponibles en una sola lista Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.46. *Cuarto paso, seleccionar widget.*



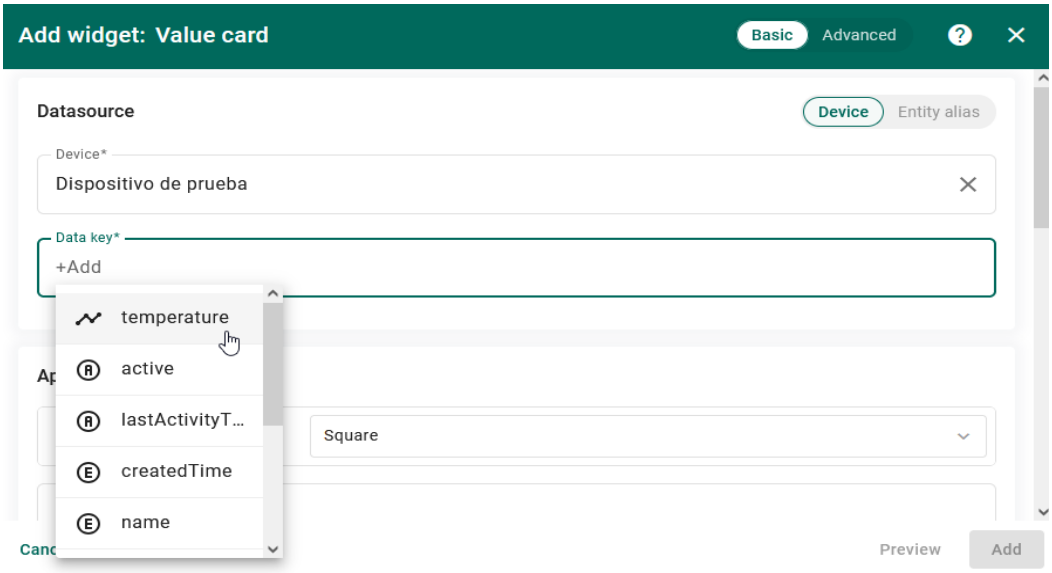
Nota: Este es un widget sencillo que muestra el valor actual de temperatura. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.47. *Quinto paso, seleccionar el dispositivo de donde se obtendrán los datos para el widget.*



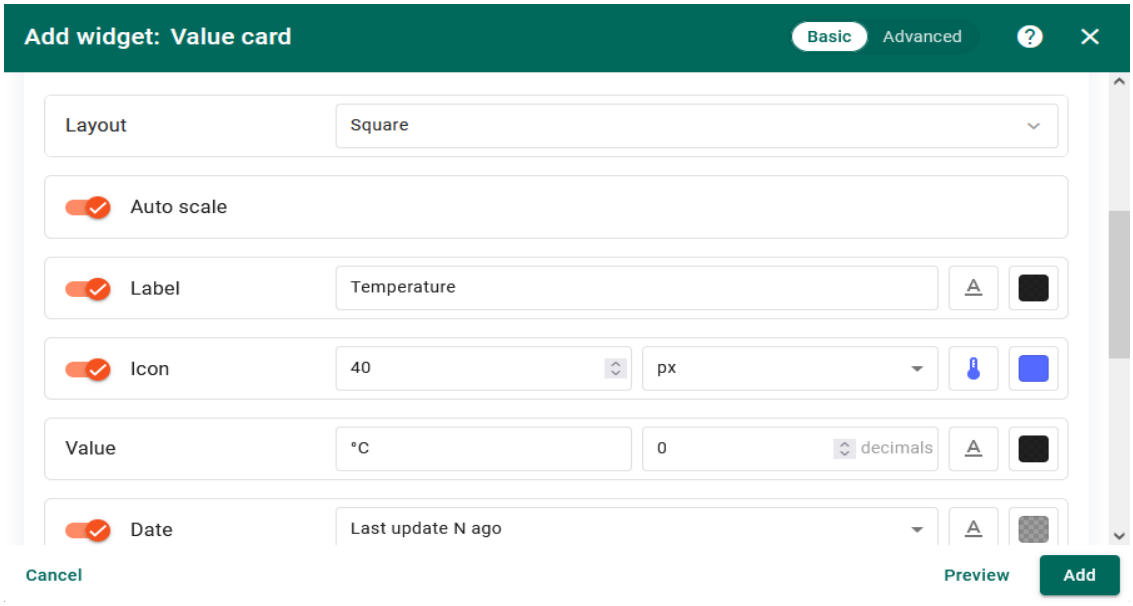
Nota: En esta ventana podremos elegir de cual dispositivo se recibirán datos para mostrar en el widget. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.48. *Sexto paso, seleccionar el dato que se mostrará en el widget.*



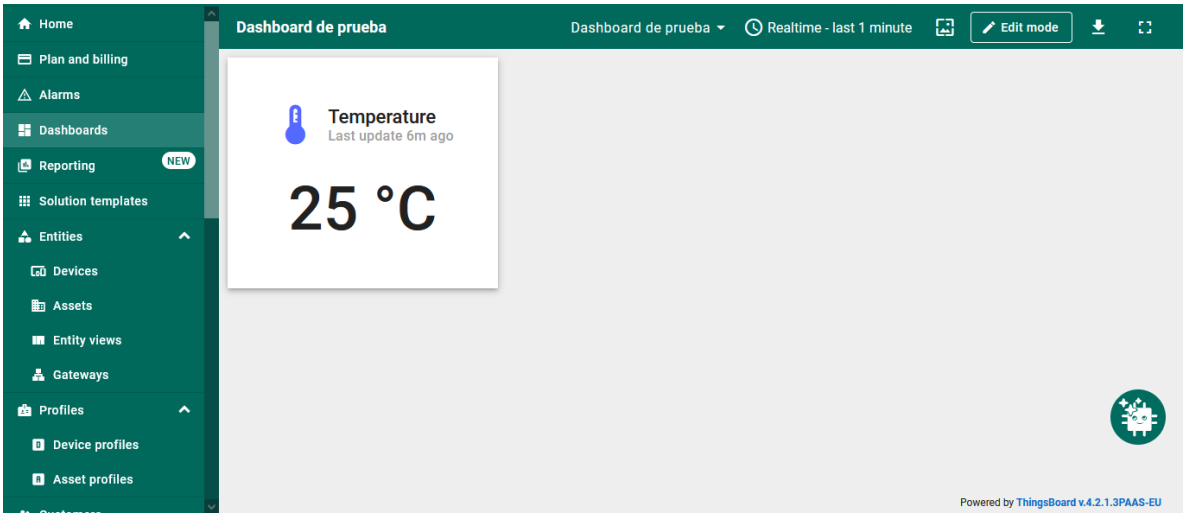
Nota: Estos parámetros se registran automáticamente al enviar datos a thingsboard. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.49. *Séptimo paso, editar título, icono del widget y unidades.*



Nota: En esta ventana se puede configurar el formato en el que se muestra el dato del parámetro seleccionado en la Figura 3.48 Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 3.50. *Ubicación del widget creado en el dashboard.*



Nota: Al finalizar la configuración del widget, este se coloca automáticamente en la ventana del dashboard. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

3.6.1. Contabo

Todos los servicios de nuestro sistema IOT están desplegados en una nube privada de Contabo, la cual ofrece servidores virtuales a un precio económico. Como especificaciones del equipo tenemos un procesador de 2 GHz Intel de 4 núcleos, con 8GB de RAM y 250GB de almacenamiento. Como sistema operativo se utiliza una distribución de Linux Ubuntu. (Contabo, 2025)

3.6.2. Docker

Docker es una plataforma open source para el desarrollo y despliegue de aplicaciones en contenedores, los cuales pueden correr en simultáneo en el mismo host(servidor o máquina virtual). Docker ofrece la conveniencia de contar con un repositorio de imágenes que contienen los componentes necesarios para hacer funcionar un contenedor, permitiendo a un desarrollador enfocarse principalmente en la creación de sus servicios (Docker, Inc., 2025.).

3.6.3. Ciberseguridad

En la red pública del internet, a cada hora del día se encuentran bots y escáneres que envían constantes peticiones a rangos específicos del directorio de IPs, probando encontrar puertos vulnerables de donde pueden extraer información importante. Por tanto, es necesario en un servidor configurar un firewall, pues los contenedores de Docker exponen sus puertos para facilitar la comunicación entre sí. Un ataque común es hacia servicios de bases de datos que no han sido configuradas apropiadamente. En una etapa inicial del despliegue de los contenedores en el servidor recibimos un ataque malicioso en el que el bot ejecutó un script dentro del contenedor de TimescaleDB, con el cual descargó e instaló un malware llamado “kinsing” (figura 3.51), el cual es utilizado para minar criptomonedas. Al haber identificado esto se apagaron los servicios y se formateó todo el servidor, instalando desde cero todos los servicios. El firewall UFW fue configurado para bloquear todas las solicitudes entrantes, excepto por los puertos 80/443 (para http y https) y el puerto 22 (para conexión remota al servidor). De esta forma, solamente los contenedores expuestos al puerto 80/443 (como nginx) reciben todas las solicitudes, las cuales analiza y bloquea o acepta (figura 3.52).

Figura 3.51. *Detalles de registro del contenedor de TimescaleDB.*

```
2025-09-29 02:28:36.623 UTC [2545] DETAIL: Connection matched pg_hba.conf line 100:
"host all all all scram-sha-256" md5sum: can't open '/tmp/kinsing': No such file or
directory /tmp/kinsing is not b3039abf2ad5202f4a9363b418002351, actual chmod:
/tmp/kinsing: No such file or directory
Connecting to 78.153.140.66 (78.153.140.66:80)
saving to '/tmp/kinsing'
2025-09-29 02:28:37.065 UTC [2670] FATAL: password authentication failed for user
"postgres"
2025-09-29 02:28:37.065 UTC [2670] DETAIL: Connection matched pg_hba.conf line 100:
"host all all all scram-sha-256"
2025-09-29 02:28:37.476 UTC [2672] FATAL: password authentication failed for user
"postgres"
2025-09-29 02:28:37.476 UTC [2672] DETAIL: Connection matched pg_hba.conf line 100:
"host all all all scram-sha-256"
kinsing 63% |******************** | 3672k 0:00:00 ETA
kinsing 100% |********************************| 5828k 0:00:00 ETA
'/tmp/kinsing' saved
```

Nota: Se observa que el atacante logró establecer una conexión remota con su servidor desde el interior del contenedor para descargar el malware. Fuente: Elaboracion propia.

Figura 3.52. *Registro del contenedor de Nginx.*

```
zilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KNTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36" "-"
192.241.188.156 - - (08/Dec/2025:02:12:13 +0000] "GET /form.hcml HTTP/1.1" 404 2 2
"-" "curl/8.1.2" "-"
192.241.188.156 - - (08/Dec/2025:02:12:13 +0000] "GET /upl.php HTTP/1.1" 404 22 "-"
"Mozilla/5.0" "-"
192.241.188.156 - - (08/Dec/2025:02:12:13 +0000] "GET /c4 HTTP/1.1" 404 22 "-" "
MOZilla/5.0" "-"
192.241.188.156 - - (08/Dec/2025:02:12:14 +0000] "GET /geoip/ HTTP/1.1" 404 22 "
-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/108.0.0.0 Safari/537.36' "-" 192.241.188.156 - -
(08/Dec/2025:02:12:14 +0000] "GET /favicon.ico HTTP/1.1" 404 22 "-" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36" "-"
192.241.188.156 - - (08/Dec/2025:02:12:14 +0000] "GET /1.php HTTP/1.1" 404 22 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeblac/537.36 (1011741., like
Gecko) Chrome/108.0.0.0 Safari/537.36" "-"
192.241.188.156 - - (08/Dec/2025:02:12:14 +0000) "GET /syscembc/password.php
HTTP/1.1" 404 22 "-" "MOZ41141/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36"-"
```

```
192.241.188.156 - - (08/Dec/2025:02:12:14 +0000] "GET /password.php HTTP/1.1" 40 4
22 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/108.0.0.0 Safari/537.36" '-'
```

Nota: Se observa el constante flujo de peticiones hacia el servidor intentando obtener archivos comunes mal protegidos por medio de un GET. Captura de ventana de CMD de Windows. Fuente: Elaboración propia.

3.7. Diagrama completo de la arquitectura del servidor IOT

Al unir cada uno de estos componentes, nos resulta el sistema descrito en el diagrama de cajas de la Figura 3.53. El proceso inicia con la toma de mediciones por medio del dispositivo de calidad de aire, el cual recolecta todos los valores y los prepara en un objeto JSON, el cual es enviado por medio de una petición POST hacia un endpoint /data definido en el servicio de FastAPI.

Al ser enviados los datos hacia el endpoint /data, la solicitud es recibida por el contenedor de Nginx y luego redirigida hacia el contenedor principal de Python, donde se encuentra nuestro servicio de FastAPI. La aplicación recibe los datos y los valida para asegurarse de que está recibiendo la cantidad de variables, el nombre de variables y el tipo de datos correctos para cada variable, para evitar que un tercero envíe datos potencialmente dañinos hacia el servidor.

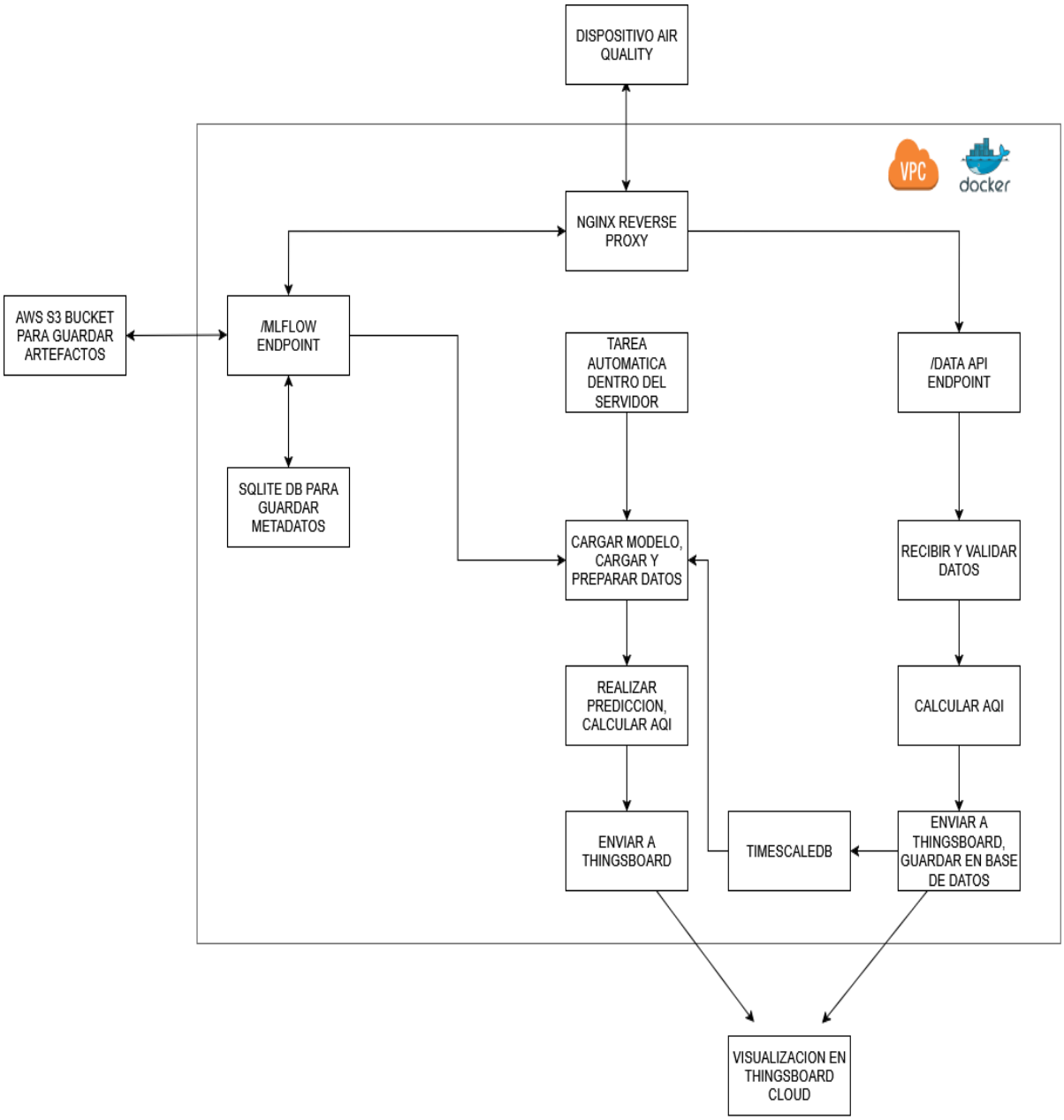
Una vez validado los datos, se toman los datos de fecha y hora y se convierten a un formato de timestamp, para luego poder almacenarlos correctamente en la base de datos de TimescaleDB. Al mismo tiempo se toma el dato de CO_2 y se calcula el Indice de calidad de aire.

Una vez terminado este proceso, se recolectan todos los datos en un JSON y se envían hacia thingsboard, haciendo una petición POST hacia la API de Thingsboard para poder ser visualizados por medio del dashboard.

Para el desarrollo del modelo de machine learning, el desarrollador utiliza el endpoint /mlflow con la función log model para subir y registrar el modelo en el servidor de Mlflow Tracking Server. Dentro de la nube IOT, se llama y se carga dicho modelo y automáticamente se ejecuta la función predict_co2, la cual llama los últimos datos registrados en la base de datos, los guarda en un dataframe con nombres de columnas iguales a los nombres de las features definidas en el entrenamiento del modelo y se separan los valores en X (que contiene los valores de las features) y en Y (que contiene los valores de CO_2).

Luego se utiliza la función minMaxScaler para convertir los datos de entrada en valores entre -1 y 1, que son los que espera el modelo de LSTM.

Figura 3.53. *Diagrama completo de cajas de la arquitectura del servidor IOT.*



Nota: Nginx no sólo monitorea las entradas hacia el servidor, sino también las conexiones que salen del servidor hacia otros servidores. Fuente: Elaboración propia utilizando Diagrams.net.

Al tener los datos preparados, se generan con estos una secuencia y se convierte en un objeto de tensores de Pytorch, con el cual se genera el pronóstico de CO_2. Luego, que está en formato de tensor, se convierte de regreso a su escala normal. Una vez obtenido este dato, se calcula el Indice

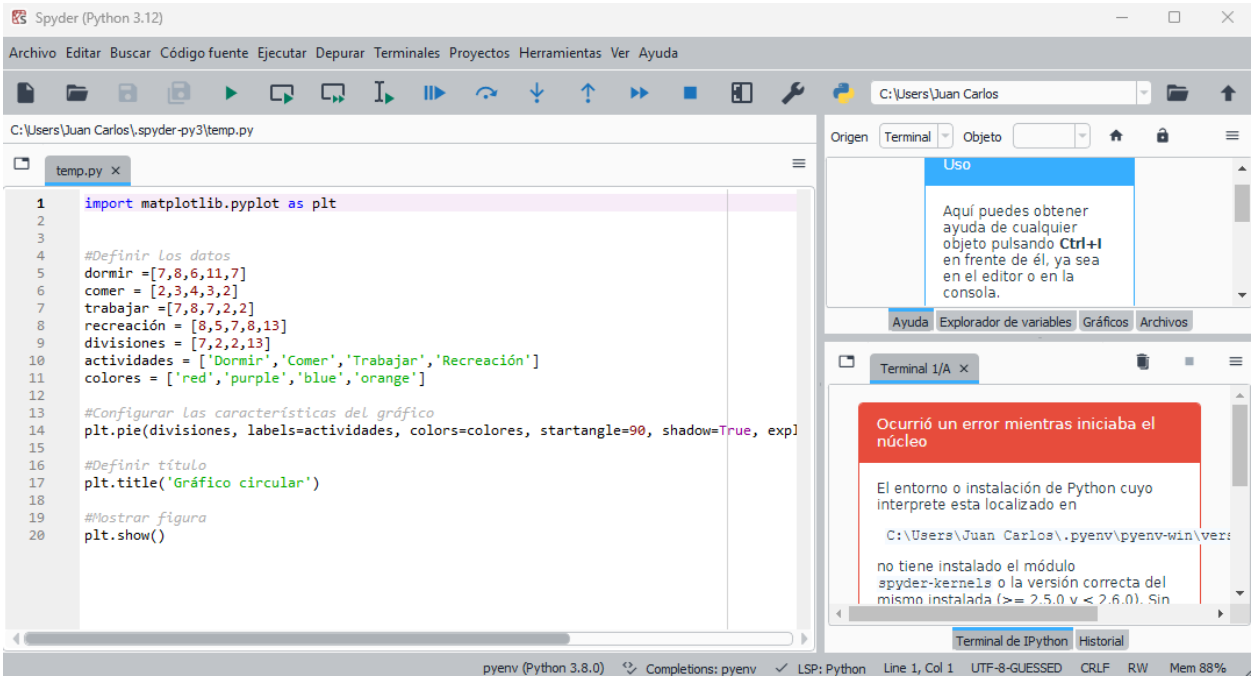
de Calidad de Aire y se recolecta junto con el pronóstico de CO_2 para luego ser enviados hacia thingsboard.

3.8. Entorno de desarrollo para Machine Learning

3.8.1. Spyder

Spyder es un entorno de desarrollo de código abierto para programación en lenguaje Python, integrado con varios paquetes para el análisis de datos, tales como NumPy, SciPy, Matplotlib, pandas, IPython, SymPy, Cython, entre otros. Dispone de una interfaz gráfica e incluye herramientas tales como: editor de código, la consola de Python, explorador de variables, visualización de datos y gráficos entre otras herramientas que facilitan el desarrollo de programas. (Spyder IDE, 2025)

Figura 3.54. *Interfaz del entorno de desarrollo Spyder.*



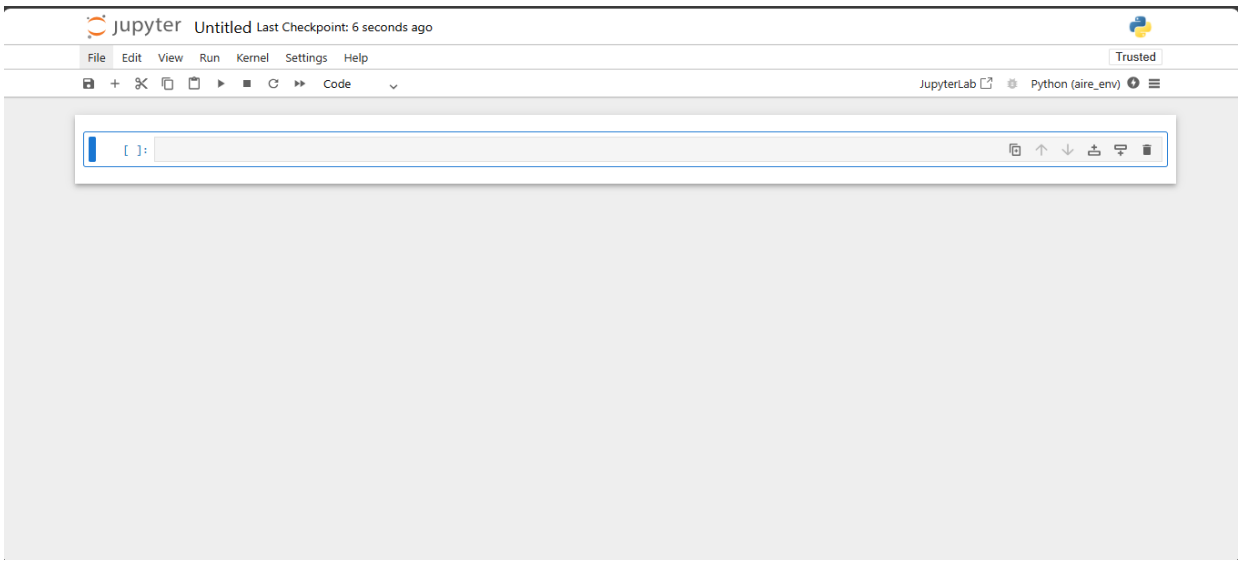
Nota: Se muestra la interfaz gráfica del entorno de desarrollo integrado Spyder, utilizado para la ejecución, depuración y visualización de código en Python durante el desarrollo del modelo. Elaboración propia, hecho en Spyder.

3.8.2. Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto, la cual permite que el desarrollador pueda dividir su código por partes y trabajar en celdas de acuerdo a la necesidad que el desarrollador requiera y el orden de ejecución que le quiera dar. Es compatible con lenguajes de programación como: Python, Ruby, Julia, Perl, Matlab y R.

Jupyter Notebook puede ejecutarse tanto de forma local, en una computadora personal, como en la nube. Para su funcionamiento, basta con disponer de un servidor con acceso mediante SSH o HTTP. (Project Jupyter, 2025).

Figura 3.55. *Interfaz gráfica de IDE Jupyter Notebook.*



Nota: Interfaz del entorno de desarrollo Jupyter Notebook empleada para el desarrollo y análisis de código en Python. Elaboración propia, hecho en Jupyter Notebook.

3.9. Librerías utilizadas

3.9.1. Pandas

Una librería de código abierto orientada a la ciencia de datos, la cual proporciona estructuras de datos como DataFrame y Series para el análisis de información, así mismo, permite leer y escribir en diversos formatos como lo son CSV, Excel y entre otros, limpiar, transformar, agrupar y analizar datos de manera rápida. (Pandas development team, 2025).

3.9.2. Pytorch

Librería de código abierto para Python que se utiliza en aplicaciones de aprendizaje automático (Machine Learning), dispone de características como tensores, grafos computacionales dinámicos, diferenciación automática (Autograd), y un ecosistema robusto. (PyTorch Foundation, 2025).

3.9.3. Matplotlib

Matplotlib es una librería de código abierto para la generación de gráficos y visualizaciones en Python. Permite representar datos de forma gráfica mediante gráficos de líneas, barras, dispersión e histogramas, entre otros, lo cual facilita el análisis visual de tendencias, patrones y comportamientos presentes en los datos. (Hunter & Matplotlib Development Team, 2025).

3.9.4. Seaborn

Es una librería de visualización de datos construida sobre Matplotlib, enfocada en la creación de gráficos estadísticos más atractivos y descriptivos. Proporciona una interfaz de alto nivel para la visualización de distribuciones, correlaciones y relaciones entre variables, mejorando la interpretación de los resultados obtenidos durante el análisis exploratorio de datos. (Waskom & Seaborn Development Team, 2025).

3.9.5. Numpy

Una librería fundamental para la computación científica en Python, la cual proporciona soporte para arreglos multidimensionales y operaciones matemáticas de alto rendimiento. Además, ofrece funciones para álgebra lineal, estadísticas y manipulación numérica, siendo una herramienta esencial para el procesamiento eficiente de datos y el desarrollo de algoritmos de aprendizaje automático. (Harris, Millman, van der Walt, & NumPy Development Team, 2025).

3.10. Recolección y limpieza de datos

Tras completar las campañas de mediciones, se recopilaron los datos medidos en un archivo CSV para poder utilizarlos en el código.

Figura 3.56. *Muestra del conjunto de datos recopilados.*

Ozono	TVOC	CO2	CO	PM1.0	PM2.5	PM10	Formaldehido	Temperatura	% Humedad	Latitud	Longitud	Altitud	Año	Mes	Dia	Hora	Minutos	Segundos
28895	36	427	0	22	33	33	0.06	30.94	61.7	0	0	0	0	0	0	0	0	0
28895	40	436	0	22	33	33	0.05	30.95	61.4	0	0	0	0	0	0	0	0	0
28895	52	461	0	23	34	34	0.04	30.95	61.3	0	0	0	0	0	0	0	0	0
28895	30	414	0	23	33	33	0.04	30.94	61.1	0	0	0	0	0	0	0	0	0
28895	29	412	0	21	30	31	0.03	30.95	60.7	0	0	0	0	0	0	0	0	0
24079	18	400	0	21	30	30	0.03	30.96	60.5	0	0	0	0	0	0	0	0	0
20639	29	412	0	22	30	30	0.03	30.98	60.5	13.7	-89.2	714.6	2025	5	23	17	24	43
21671	45	446	0	21	31	31	0.02	31.01	60.5	13.7	-89.2	705.2	2025	5	23	17	24	48
22473	40	437	0	20	31	31	0.02	31.02	60.5	13.7	-89.2	701.6	2025	5	23	17	24	52
23144	45	446	0	20	30	30	0.02	31.05	60.6	13.7	-89.2	700.6	2025	5	23	17	24	57
21040	65	487	0	19	31	31	0.02	31.07	60.4	13.7	-89.2	700.7	2025	5	23	17	25	2
21695	39	433	0	20	30	31	0.02	31.09	60.4	13.7	-89.2	700.7	2025	5	23	17	25	7
22249	38	431	0	19	28	29	0.02	31.13	60.6	13.7	-89.2	700.8	2025	5	23	17	25	11
20659	65	487	0	20	28	28	0.01	31.15	61	13.7	-89.2	700.8	2025	5	23	17	25	16
21208	32	419	0	19	28	29	0.01	31.16	61.2	13.7	-89.2	703.3	2025	5	23	17	25	21
21689	46	448	0	19	28	29	0.01	31.18	61	13.7	-89.2	704.9	2025	5	23	17	25	26

Nota: Se presenta un extracto de los datos obtenidos durante la campaña de monitoreo ambiental de la fecha veintitrés de mayo del dos mil veinticinco, organizados en un archivo de hoja de cálculo para su posterior análisis y procesamiento. Elaboración propia, hecho en Excel.

Estos datos contienen filas con datos erróneos y falsos que por desperfecto de los sensores del dispositivo se habrán tomado, por tanto, es necesario realizar una limpieza de estos datos. Utilizaremos la librería Pandas del lenguaje de programación Python para este propósito. Comenzamos importando los datos del archivo de Excel a un dataframe de Pandas.

Figura 3.57. *DataFrame original del conjunto de datos.*

Out[5]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Un
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
1	NaN	NaN	OZONO	PARTICULAS	CO2	CO	PMS 1.0	PMS 2.5	PMS 10	FORMALDEHIDO	...	/
2	NaN	NaN	28895	36	427	0	22	33	33	0.06	...	
3	NaN	NaN	28895	40	436	0	22	33	33	0.05	...	
4	NaN	NaN	28895	52	461	0	23	34	34	0.04	...	
...	...	...	...	...	...	...	...	...	...	...	...	
6355	NaN	NaN	0	581	890	0	18	24	25	0.01	...	657
6356	NaN	NaN	0	426	843	0	19	25	25	0.01	...	657
6357	NaN	NaN	0	772	944	0	20	27	27	0.01	...	657
6358	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
6359	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	

Nota: Se muestra el DataFrame que contiene la totalidad de las muestras recopiladas durante las campañas de monitoreo, incluyendo valores faltantes y registros inconsistentes, previo al proceso de limpieza y depuración de los datos. Elaboración propia, hecho en Jupyter Notebook.

Una vez cargado los datos a un dataframe, se eliminan todas las filas y columnas vacías y se define el nombre de cada variable como encabezado de cada columna del dataframe. Además de esto, se convierten los datos de año, mes, día, hora, minuto y segundo a un solo timestamp para poder utilizarlo como índice de cada fila. Esto permite identificar cada fila de datos tomados con su respectiva hora en la que fueron tomados.

Figura 3.58. *Resultado de las primeras medidas de limpieza del conjunto de datos.*

timestamp	OZONO	PARTICULAS	CO2	CO	PMS 1.0	PMS 2.5	PMS 10	FORMALDEHIDO	TEMPERATURA
2025-06-14 08:36:02+00:00	141	39	434	0	18	32	33	0.01	29.73
2025-06-14 08:36:06+00:00	145	43	442	0	20	32	33	0.01	29.76
2025-06-14 08:36:11+00:00	148	44	445	0	19	33	34	0.01	29.79
2025-06-14 08:36:16+00:00	150	48	452	0	19	30	31	0.01	29.81
2025-06-14 08:36:21+00:00	150	56	469	0	19	30	31	0.01	29.84
...	...	...	...	...	...	...	...	...	...
2025-06-22 09:41:00+00:00	130	0	0	1000	8	11	12	0.01	26.95
2025-06-22 09:41:05+00:00	128	0	0	1000	8	10	11	0.01	26.75
2025-06-22 09:41:10+00:00	123	0	0	1000	6	9	10	0.01	26.69
2025-06-22 09:41:14+00:00	118	0	0	1000	6	9	10	0.01	26.68
2025-06-22 09:41:19+00:00	108	0	0	1000	5	8	8	0.01	0

Nota**:** Cada columna corresponde a una variable de contaminante ambiental, mientras que cada fila representa una medición individual, cuyo índice indica el instante de tiempo en el que fue tomada la muestra. Elaboración propia, hecho en Jupyter Notebook.

3.11. Análisis exploratorio de los datos

Inicialmente se cargó el archivo resultante del proceso de limpieza de los datos y, mediante el uso de la librería pandas, junto al orden temporal de los datos, se orientó el análisis hacia las variables ambientales más relevantes tales como: ozono, TVOC, dióxido de carbono (CO2), monóxido de carbono (CO), material particulado (PMS 1.0, PMS 2.5 y PMS 10) formaldehido, temperatura y humedad relativa.

Como primer paso, se realizó un análisis estadístico descriptivo, calculando datos de interés como la media, la desviación estándar, valores mínimos y máximos, con el objetivo de obtener información sobre el comportamiento de cada variable.

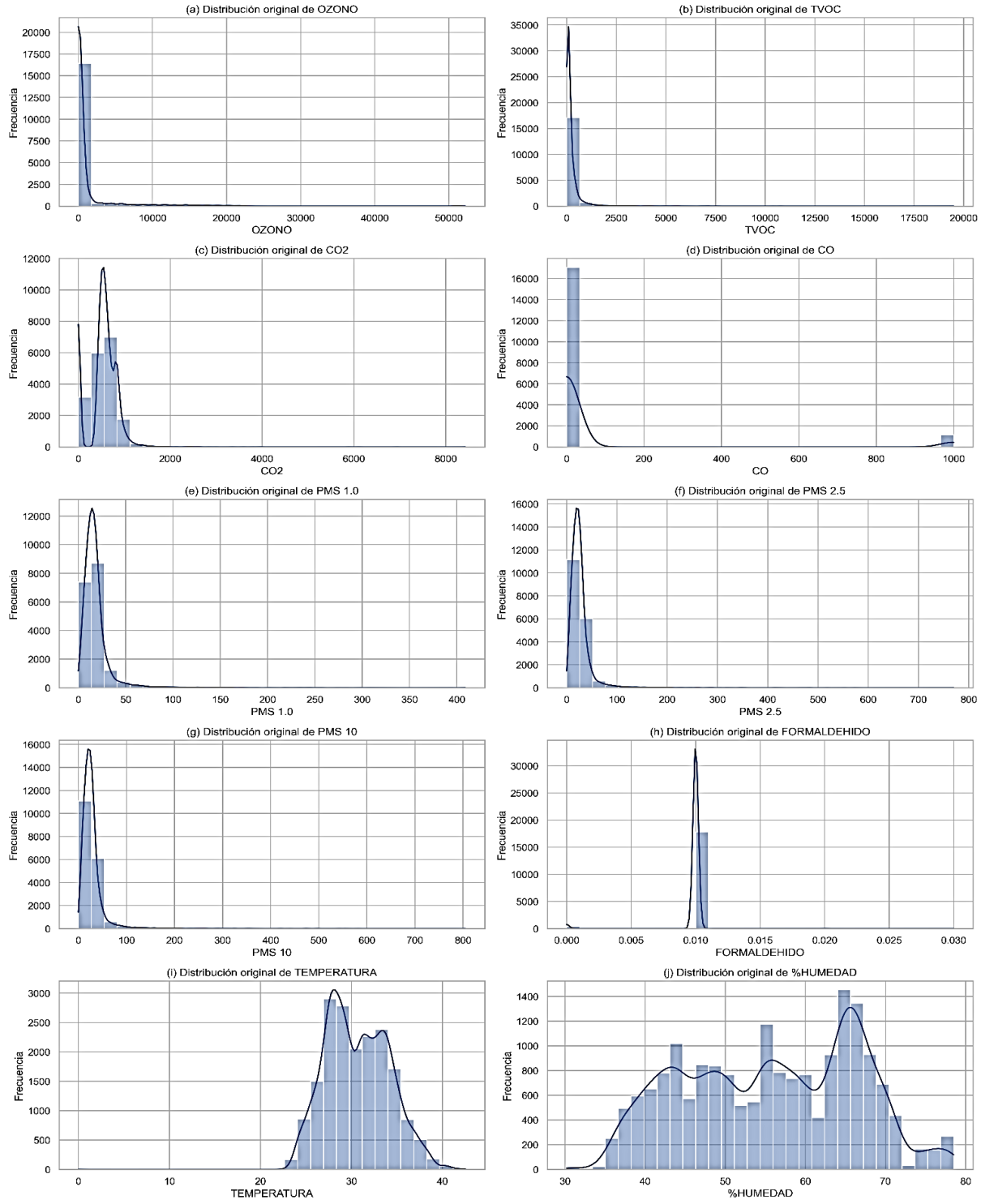
Del primer paso se obtiene la siguiente tabla resumen:

Tabla 3.1. *Resumen estadístico de las variables ambientales.*

Variable	count	mean	std	min	25%	50%	75%	max
OZONO	18194.00	1076.19	3674.55	0.00	17.00	47.00	158.00	52237.00
TVOC	18194.00	226.54	553.16	0.00	47.00	107.00	215.00	19499.00
CO2	18194.00	537.78	322.89	0.00	450.00	560.00	712.00	8429.00
CO	18194.00	62.38	241.86	0.00	0.00	0.00	0.00	1000.00
PM 1.0	18194.00	18.36	18.62	0.00	10.00	15.00	20.00	409.00
PM 2.5	18194.00	26.33	25.59	0.00	15.00	22.00	30.00	770.00
PM 10	18194.00	27.39	26.37	0.00	16.00	23.00	32.00	805.00
FORMALDEHÍDO	18194.00	0.01	0.00	0.00	0.01	0.01	0.01	0.03
TEMPERATURA	18194.00	30.65	3.45	0.00	27.92	30.42	33.30	42.52
%HUMEDAD	18194.00	55.68	10.69	30.20	46.70	56.00	65.20	78.50

Nota: Se muestran las medidas estadísticas básicas de las variables ambientales obtenidas a partir del conjunto de datos. Elaboración propia.

Figura 3.59. *Distribución estadística de las variables antes del tratamiento de datos.*



Nota: (a) O_3, (b) TVOC, (c) Dióxido de carbono (CO_2), (d) Monóxido de carbono (CO), (e) $PM_{1.0}$, (f) $PM_{2.5}$, (g) PM_{10}, (h) Formaldehído(HCHO), (i) Temperatura y (j) Humedad. Elaboración propia, , hecho en Jupyter Notebook.

A partir de este análisis, se observa que las variables presentan distribuciones asimétricas y la presencia de valores extremos, sobre todo en las variables como ozono, CO_2 y material particulado. Estas características son visualizadas mediante los histogramas antes del tratamiento. Las influencias de valores atípicos pueden tener un impacto negativo sobre el desempeño del modelo de aprendizaje automático, por lo cual se aplicó un proceso de filtrado de outliers utilizando el método del rango intercuartílico (IQR). Dicho proceso permite eliminar observaciones alejadas del comportamiento general, conservando valores dentro de un rango estadísticamente aceptables para cada variable.

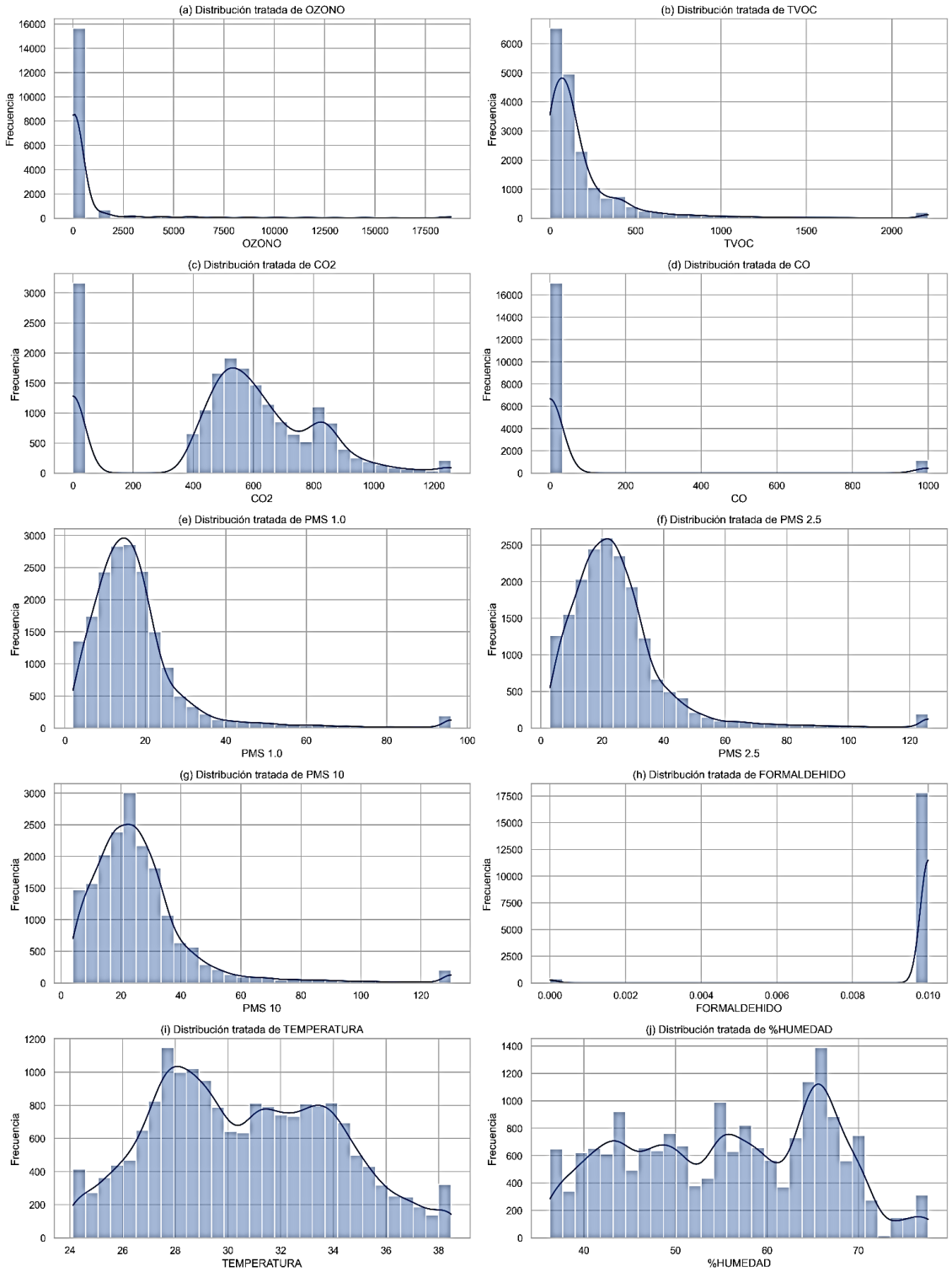
Ya filtrados y transformados los datos, se calcularon nuevamente las estadísticas descriptivas del conjunto de datos.

Tabla 3.2. *Resumen estadístico de las variables ambientales tratadas.*

Variable	count	mean	std	min	25%	50%	75%	max
OZONO	18194.00	1021.48	3192.16	0.00	17.00	47.00	158.00	18781.00
TVOC	18194.00	205.28	329.47	0.00	47.00	107.00	215.00	2212.14
CO2	18194.00	532.73	293.11	0.00	450.00	560.00	712.00	1257.00
CO	18194.00	62.38	241.86	0.00	0.00	0.00	0.00	1000.00
PM 1.0	18194.00	17.79	13.64	2.00	10.00	15.00	20.00	96.00
PM 2.5	18194.00	25.52	18.27	3.00	15.00	22.00	30.00	126.00
PM 10	18194.00	26.58	18.96	4.00	16.00	23.00	32.00	130.07
FORMALDEHÍDO	18194.00	0.01	0.00	0.00	0.01	0.01	0.01	0.01
TEMPERATURA	18194.00	30.64	3.41	24.11	27.92	30.42	33.30	38.47
%HUMEDAD	18194.00	55.69	10.66	36.30	46.70	56.00	65.20	77.60

Nota: Se muestran las medidas estadísticas básicas de las variables ambientales obtenidas a partir del conjunto de datos tratado, posterior al proceso de limpieza y eliminación de valores erróneos. Elaboración propia.

Figura 3.60. *Distribución estadística de las variables después del tratamiento de datos.*



Nota: (a) O_3, (b) $TVOC$, (c) Dióxido de carbono (CO_2), (d) Monóxido de carbono (CO), (e) $PM_{1.0}$, (f) $PM_{2.5}$, (g) PM_{10}, (h) Formaldehído ($HCHO$), (i) Temperatura y (j) Humedad. Elaboración propia, hecho en Jupyter Notebook.

Los resultados obtenidos muestran una reducción en la dispersión de las variables, reflejada en valores menores de desviación estándar y acotación en los valores máximos, con ello se ha logrado una mejora sustancial en el conjunto de datos.

Los histogramas después del tratamiento confirmaron visualmente dichosos resultados, mostrando distribuciones más compactas comparadas a las distribuciones originales.

En el caso del formaldehído, la distribución resultante se encuentra altamente concentrada en un rango estrecho de valores, lo cual concuerda con su baja variabilidad observada tras el filtrado. Por su parte, la temperatura y la humedad relativa muestran distribuciones más suaves y cercanas a una forma aproximadamente normal, sin la presencia de valores extremos significativos.

Además, parte del análisis exploratorio de las series temporales, se evaluó la estacionariedad de la variable dióxido de carbono (CO_2), ya que esta propiedad es fundamental para el correcto desempeño de los modelos de predicción basados en series de tiempo, como las redes neuronales recurrentes tipo LSTM.

Para este propósito, se aplicó la prueba Dickey-Fuller aumentada (ADF), la cual contrasta la hipótesis nula de que la serie presenta una raíz unitaria, es decir, que no es estacionaria. Los resultados obtenidos para la serie de CO_2 se presentan a continuación:

Estadístico ADF: -5.9182

p-value: 0.0000

Valores críticos: $\{ 1\%: -3.4307, 5\%: -2.8617, 10\%: -2.5669 \}$

El valor del estadístico ADF es menor que los valores críticos en todos los niveles de significancia considerados, y el p-value es inferior a 0.05. Por lo tanto, se rechaza la hipótesis nula, concluyendo que la serie temporal de CO2 es estacionaria.

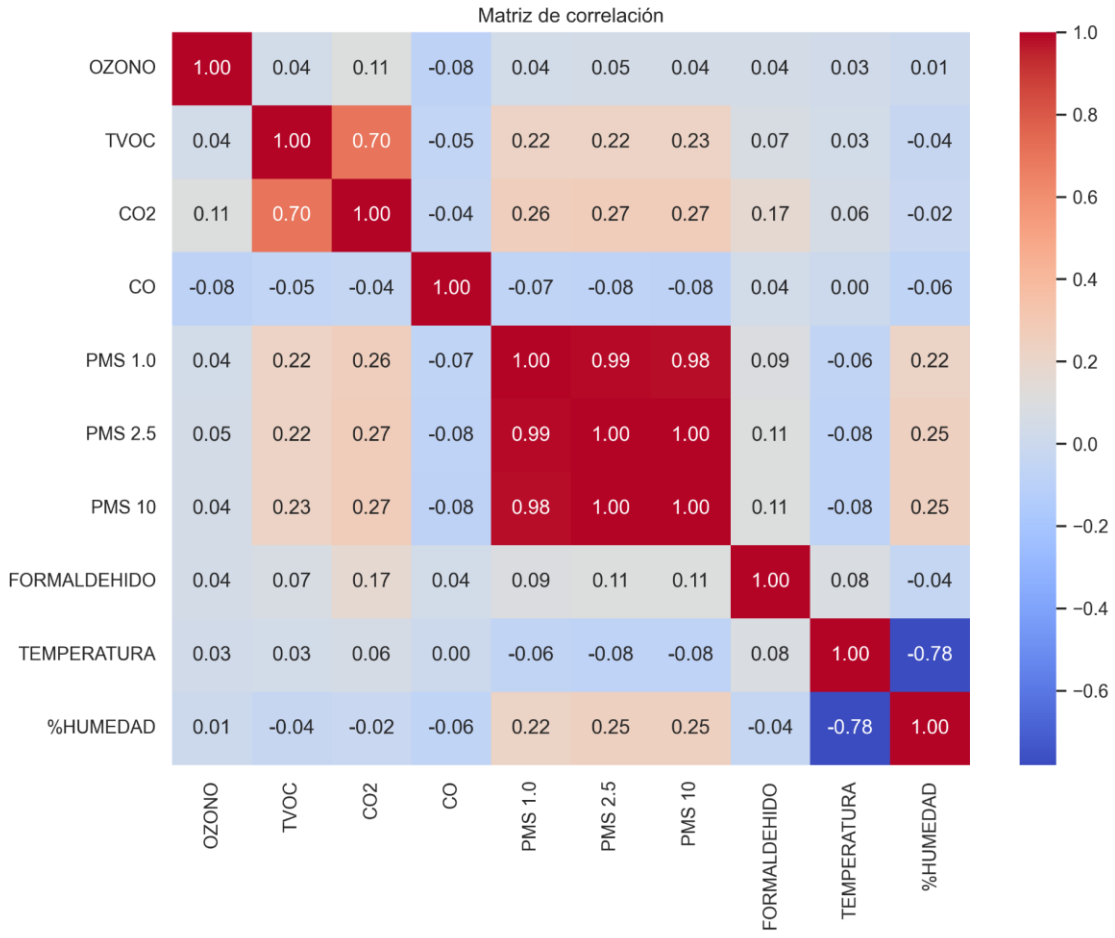
Este resultado indica que la serie no presenta tendencias ni variaciones sistemáticas a lo largo del tiempo que puedan afectar negativamente el proceso de entrenamiento del modelo. En consecuencia, no fue necesario aplicar técnicas adicionales de diferenciación para esta variable, permitiendo su uso directo dentro del conjunto de datos preprocesado.

Análisis de correlación entre variables ambientales

El análisis de correlación es una técnica estadística utilizada para medir el grado de relación lineal entre dos variables cuantitativas. Uno de los coeficientes más empleados es el coeficiente de correlación de Pearson, el cual toma valores en el intervalo [−1,1]. Un valor cercano a 1 indica una correlación positiva fuerte, un valor cercano a -1 indica una correlación negativa fuerte, mientras que valores próximos a 0 sugieren una relación lineal débil o inexistente.

En el contexto del análisis de datos ambientales, la correlación permite identificar posibles dependencias entre contaminantes y variables meteorológicas, así como detectar redundancia entre variables, lo cual resulta fundamental para la selección adecuada de variables de entrada en modelos de aprendizaje automático.

Figura 3.61. *Matriz de correlación de las variables ambientales.*



Nota: La figura muestra los coeficientes de correlación de Pearson entre las variables ambientales analizadas, permitiendo identificar relaciones lineales positivas y negativas, así como el grado de asociación entre ellas. Elaboración propia, hecho en Jupyter Notebook.

El análisis de correlación se realizó sobre el conjunto de datos previamente tratado, con el objetivo de evaluar las relaciones lineales entre las variables ambientales consideradas en el estudio. Para ello, se calculó la matriz de correlación de Pearson utilizando las variables: ozono, $TVOC$, dióxido de carbono (CO_2), monóxido de carbono (CO), material particulado ($PM_{1.0}$, $PM_{2.5}$ y PM_{10}), formaldehído, temperatura y humedad relativa.

Los resultados obtenidos fueron representados mediante un mapa de calor (heatmap), lo que permite una interpretación visual clara de la intensidad y el signo de las correlaciones existentes entre las variables.

A partir de la matriz de correlación obtenida, se observa que la variable dióxido de carbono (CO_2), definida como variable objetivo del estudio, presenta una correlación positiva fuerte con la variable $TVOC$, con un coeficiente de 0.70, lo que indica una relación lineal significativa entre ambas variables.

Asimismo, las variables correspondientes al material particulado ($PM_{1.0}$, $PM_{2.5}$ y PM_{10}) presentan correlaciones moderadas con el CO_2, con valores cercanos a 0.26 – 0.27. Sin embargo, se evidencia una correlación extremadamente alta entre estas fracciones de material particulado, con coeficientes superiores a 0.98, lo cual indica una alta redundancia entre ellas.

Por otra parte, variables como la temperatura y la humedad relativa muestran una correlación lineal baja con el CO_2. No obstante, se identifica una fuerte correlación negativa entre la temperatura y la humedad relativa (-0.78), lo cual es coherente con el comportamiento físico de estas variables ambientales.

Variables como el ozono, el formaldehído y el monóxido de carbono presentan correlaciones débiles con el CO_2, indicando una relación lineal limitada en el conjunto de datos analizado.

3.12. Desarrollo del modelo LSTM

Inicialmente se importaron las librerías necesarias en el código (Arévalo, 2025) mostradas la figura 3.62 para el procesamiento de datos, construcción del modelo y evaluación de resultados. También se emplearon funciones y clases personalizadas que tienen como objetivo la creación de secuencias temporales, la evaluación del modelo y generación de gráficos.

Figura 3.62. *Código de importación de librerías para el modelo LSTM.*

```
1. import numpy as np
2. import pandas as pd
```

```
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import os
```

Nota: El código muestra las librerías empleadas para el procesamiento de datos, el entrenamiento del modelo LSTM, la evaluación del desempeño y la visualización de resultados. Código modificado de Arévalo, 2025, hecho en Jupyter Notebook.

Se cargó el conjunto de datos preprocesados desde el archivo dataset_tratado.csv, donde se verificó las dimensiones de este y se seleccionaron las variables más importantes con el fin de simplificar el manejo de los datos durante el modelado.

Se seleccionó como variable objetivo para la predicción de CO_2, tal como se muestra en la figura 3.63, mientras que como variables de entrada se utilizaron partículas volátiles totales, material particulado en sus diferentes fracciones ($PM_{1.0}$, $PM_{2.5}$ y PM_{10}), temperatura y humedad relativa. De esta forma, el modelo aprende a predecir el comportamiento del CO_2 a partir de otras variables ambientales relacionadas.

Figura 3.63. *Código de definición de la variable objetivo y de entrada del modelo LSTM.*

```
target_cols = ['co2']

feature_cols = [
    'tvoc', 'pms_1_0', 'pms_2_5', 'pms_10', 'temperature', 'humidity'
]

# Separar X e y
y = data[target_cols]
X = data[feature_cols]
```

Nota: En el código se define el dióxido de carbono (CO_2) como variable objetivo del modelo, mientras que las variables $TVOC$, material particulado, temperatura y humedad relativa se utilizan como variables de entrada para el entrenamiento del modelo LSTM. Código modificado de Arévalo, 2025, hecho en Jupyter Notebook.

Posteriormente se dividió el conjunto de datos de manera secuencial, en subconjuntos los cuales son entrenamiento, validación y prueba, utilizando un porcentaje de 70%, 15% y 15%, respectivamente.

Con el objetivo de mejorar la estabilidad del entrenamiento del modelo, se aplicó un proceso de escalamiento mediante la técnica Min-Max Scaling. Este escalamiento se realizó utilizando únicamente los datos de entrenamiento para evitar sesgos, y posteriormente se aplicó a los conjuntos de validación y prueba. Tanto las variables de entrada como la variable objetivo fueron escaladas de forma independiente.

Una vez escalados los datos, se construyeron con el código de la figura 3.64 secuencias temporales deslizantes, donde cada muestra de entrada está compuesta por una ventana de 12 pasos de tiempo consecutivos y la salida corresponde a un valor futuro de CO_2. Este enfoque permite que la red LSTM capture patrones temporales y dependencias entre observaciones pasadas y futuras.

Figura 3.64. *Código de creación de secuencias temporales para el modelo LSTM.*

```
input_len, output_len = 12, 1

X_train_seq, y_train_seq = create_sequences(
    X_train_s, y_train_s, input_len, output_len
)
X_val_seq, y_val_seq = create_sequences(
   X_val_s, y_val_s, input_len, output_len
)
X_test_seq, y_test_seq = create_sequences(
    X_test_s, y_test_s, input_len, output_len
)
```

Nota: El código muestra la generación de secuencias deslizantes a partir de los datos escalados, utilizando una ventana de entrada de 12 pasos temporales y un horizonte de predicción de un paso, para los conjuntos de entrenamiento, validación y prueba. Código modificado de Arévalo, 2025, hecho en Jupyter Notebook.

Posteriormente, las secuencias generadas fueron convertidas a tensores y organizadas en DataLoaders, lo que facilita el entrenamiento por lotes (batch training) y mejora la eficiencia computacional del proceso de aprendizaje.

El modelo LSTM fue inicializado utilizando una arquitectura definida previamente, ajustada al número de variables de entrada y al horizonte de predicción seleccionado. Para el entrenamiento,

se empleó el optimizador Adam, junto con la función de pérdida de error cuadrático medio (MSE), debido a su idoneidad para problemas de regresión.

El proceso de entrenamiento se llevó a cabo durante un máximo de 50 épocas, incorporando un mecanismo de Early Stopping que permite detener el entrenamiento de forma anticipada cuando la pérdida de validación deja de mejorar, evitando así el sobreajuste del modelo. Durante cada época se calcularon las pérdidas de entrenamiento y validación, las cuales fueron almacenadas para su posterior análisis.

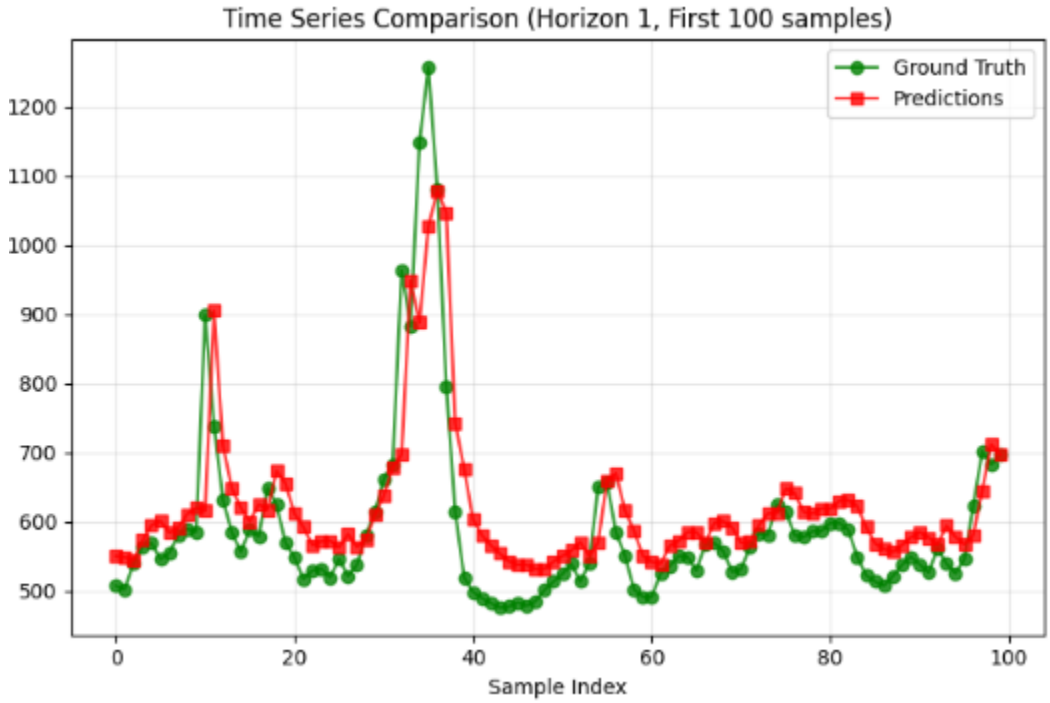
Figura 3.65. *Gráfico de pérdida de entrenamiento y validación del modelo.*

Nota: El modelo fue entrenado durante 25 épocas, mostrando una disminución progresiva en la pérdida MSE para los conjuntos de entrenamiento y validación, lo que indica aprendizaje efectivo y evaluación de la capacidad de generalización, hecho en Jupyter Notebook.

Finalizado el entrenamiento, se visualizaron las curvas de pérdida como la presentada en la figura 3.65 correspondientes a los conjuntos de entrenamiento y validación, lo que permitió evaluar gráficamente el comportamiento del aprendizaje del modelo. Posteriormente, los pesos del modelo entrenado fueron guardados para su reutilización futura.

Finalmente, el modelo fue evaluado utilizando el conjunto de prueba. Se generaron predicciones y se calcularon métricas de desempeño para analizar la calidad del pronóstico.

Figura 3.66. *Comparación temporal de predicciones (Horizonte 1)*



Nota: El modelo predice los valores de CO_2 para los primeros 100 casos del conjunto de prueba. La línea roja representa las predicciones y la verde los valores reales. Elaboración propia, hecho en Jupyter Notebook.

La Figura 3.66 muestra la comparación entre los valores reales de CO_2 y las predicciones generadas por el modelo para las primeras 100 muestras del horizonte de predicción inmediato (Horizonte 1). La línea verde representa los valores reales (Ground Truth), mientras que la línea roja indica las predicciones. Se observa que el modelo logra seguir la tendencia general de la serie temporal, aunque presenta desviaciones en ciertos picos, lo cual evidencia áreas de mejora en la capacidad de seguimiento de variaciones abruptas.

Esta visualización permite evaluar de forma directa la calidad del pronóstico, complementando el análisis cuantitativo realizado mediante métricas como MAE, RMSE y R^2.

La gráfica mostrada en la Figura 3.67 presenta la evolución histórica de los niveles de CO_2 (línea azul) justo antes del momento de pronóstico. En el punto marcado como “Forecast Start”, el modelo genera una predicción (punto rojo) que se compara con el valor real medido (cuadro verde).

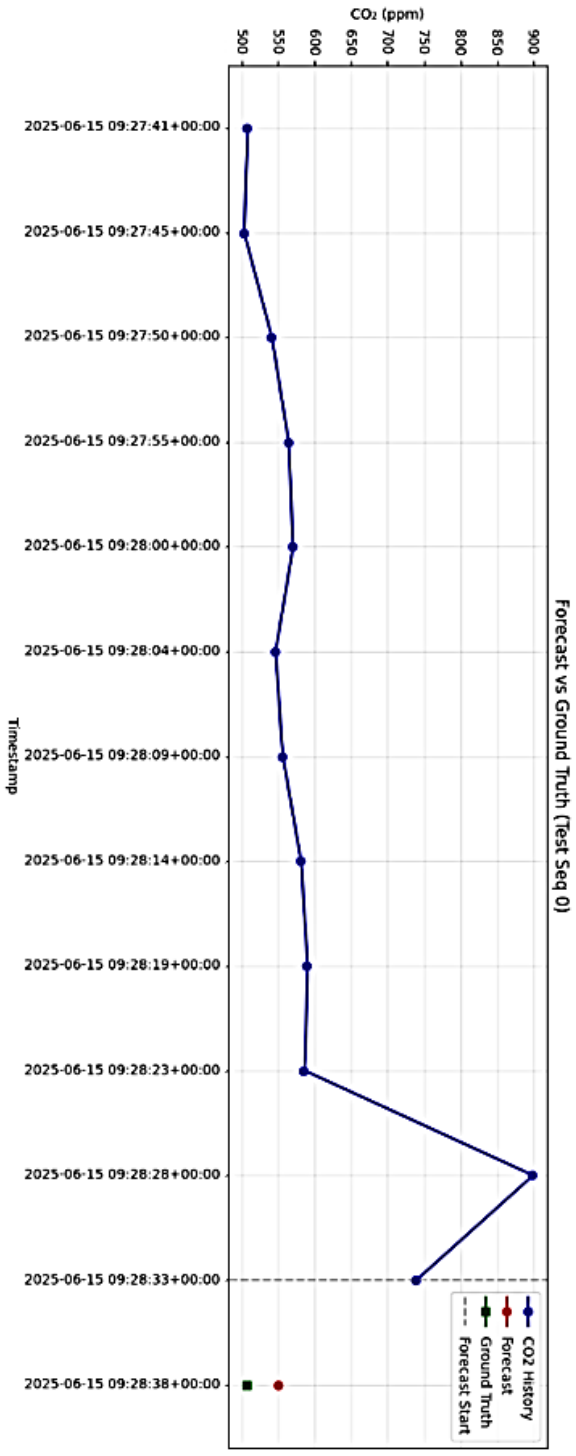
En esta secuencia, la predicción se aproxima al valor real, lo que indica un buen desempeño del modelo en este caso específico.

La gráfica mostrada en la Figura 3.68 muestra los valores históricos de CO_2 cercanos a cero (línea azul), seguidos por una predicción abruptamente elevada (punto rojo) en el instante de pronóstico. El valor real medido (cuadro verde) permanece cerca de cero, evidenciando un error notable del modelo en esta instancia. Este caso ilustra una limitación en la capacidad del modelo para generalizar correctamente en ciertos contextos del conjunto de prueba.

Se visualiza en la Figura 3.69 la evolución histórica de los niveles de CO_2 (línea azul) antes del instante de pronóstico. En el punto marcado como "Forecast Start", el modelo genera una predicción (punto rojo) que se compara con el valor real medido (cuadro verde). En este caso, la predicción se encuentra relativamente cerca del valor real, lo que indica un desempeño aceptable del modelo en esta instancia.

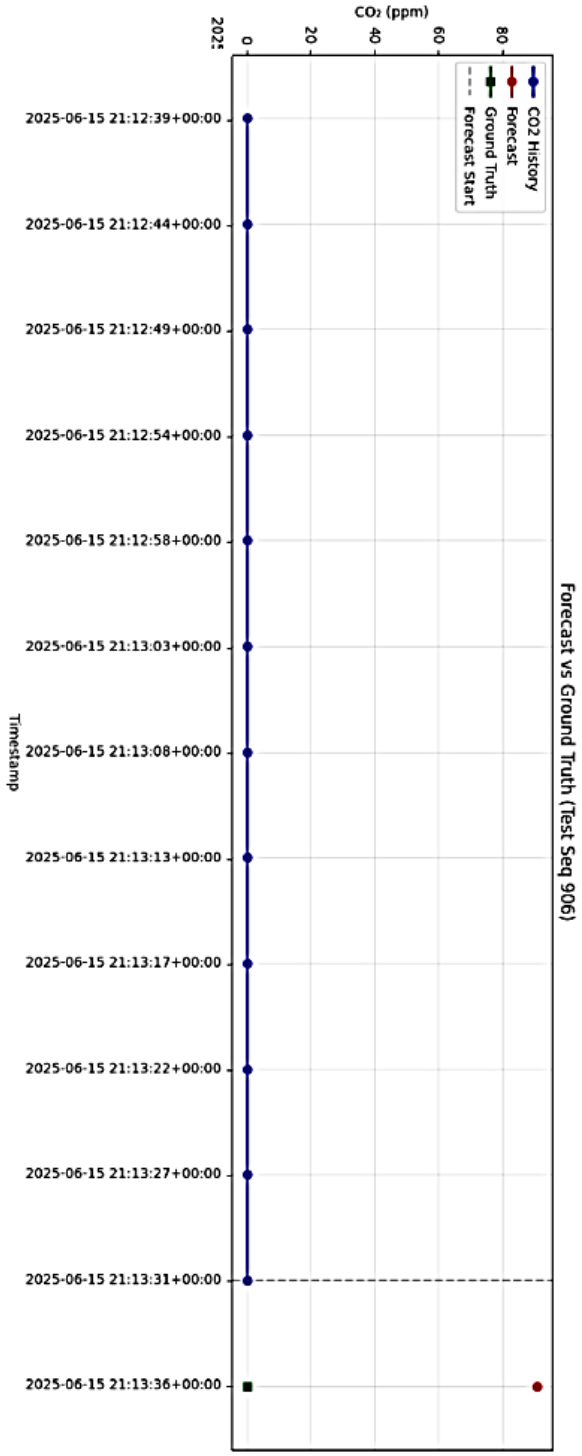
La gráfica de la figura 3.70 muestra cómo se distribuyen las predicciones del modelo respecto a los valores reales de CO_2. La línea roja discontinua representa la predicción perfecta (y = x). Aunque muchas predicciones siguen la tendencia general, se observan desviaciones, especialmente en valores bajos y altos, lo que indica variabilidad en la precisión del modelo.

Figura 3.67. *Comparación puntual entre predicción y valor real (Test Seq 0)*



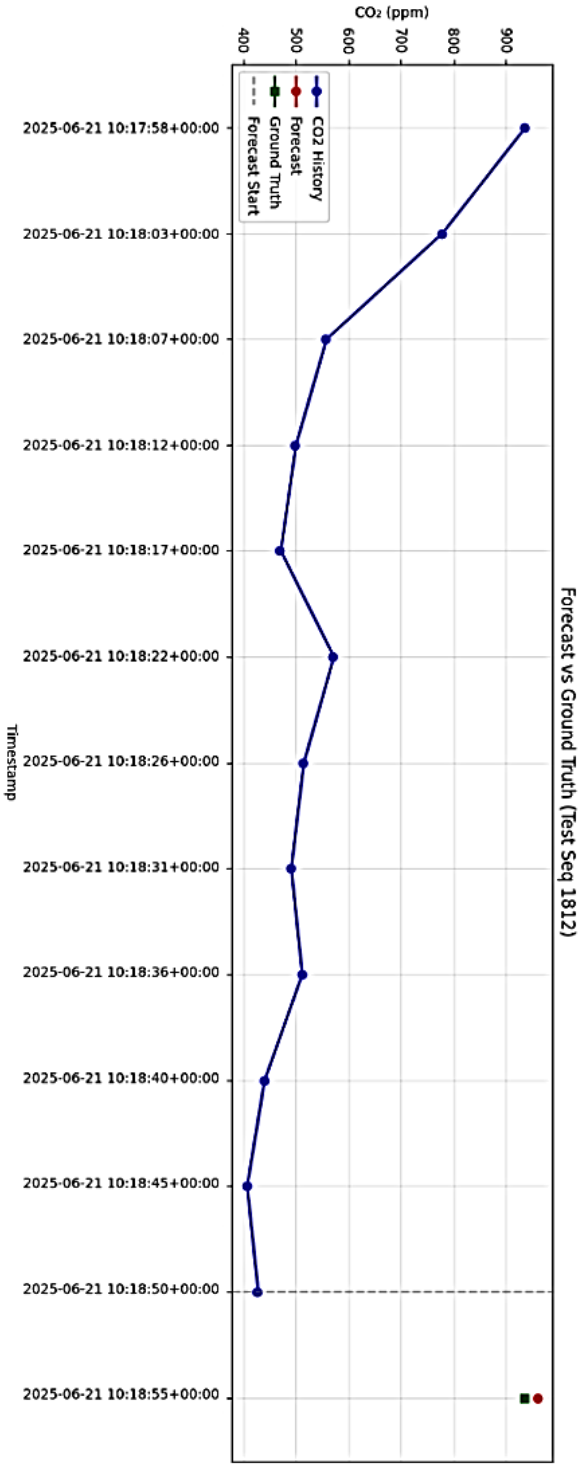
Nota: Se muestra la predicción de CO_2 para una secuencia de prueba específica, comparada con el valor real en el mismo instante. Elaboración propia, hecho en Jupyter Notebook.

Figura 3.68. *Error significativo en la predicción (Test Seq 906)*



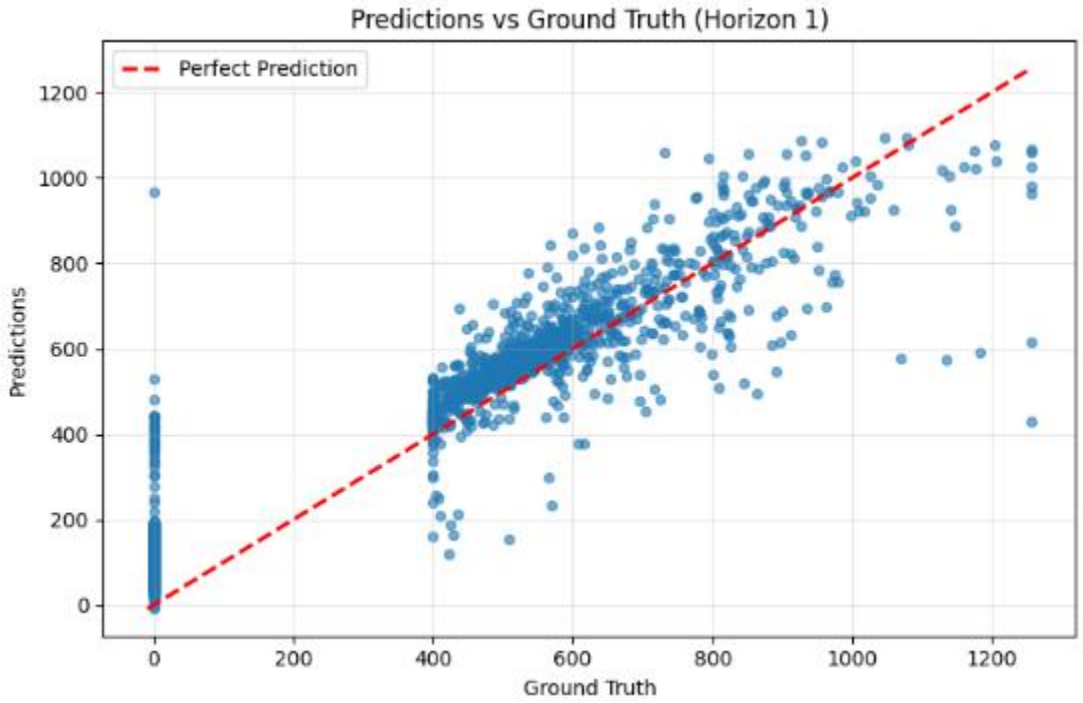
Nota: La predicción del modelo se desvía considerablemente del valor real de CO_2 en esta secuencia. Elaboración propia, hecho en Jupyter Notebook.

Figura 3.69. *Predicción cercana al valor real (Test Seq 1812)*



Nota**:** La predicción del modelo se aproxima al valor real de CO_2 en esta secuencia. Elaboración propia, hecho en Jupyter Notebook.

Figura 3.70. *Relación entre predicciones y valores reales (Horizonte 1)*



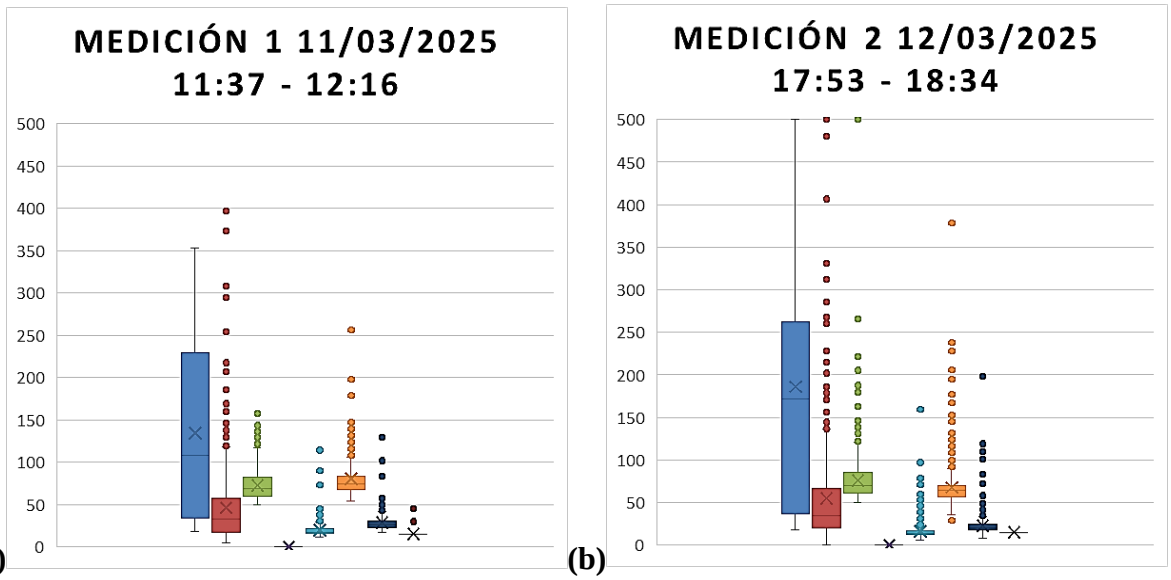
Nota: Cada punto representa una predicción comparada con su valor real correspondiente. Elaboración propia, hecho en Jupyter Notebook.

Capitulo 4. RESULTADOS Y DISCUSIÓN PERFIL DEL PROYECTO

4.1. Calidad de aire de acuerdo a las mediciones.

Los valores de las mediciones dieron valores muy similares teniendo un leve incremento en las concentraciones en los momentos en que se contaba con más congestionamiento vehicular. De acuerdo a las mediciones que se tomaron las mediciones en el ozono rondaba de 0 a 300, teniendo una mediana de calidad de aire de perjudicial para grupos sensibles; en partículas orgánicas volátiles totales en un rango de 0 a 100, teniendo una mediana en el estado de aire moderado; el material particulado y el dióxido de carbono de 50 a 100, igualmente siendo moderado; el formaldehído y el material particulado 1.0 y 10 en el rango de 0 a 50, presentando una calidad de aire buena.

Figura 4.1. a) b) *Diagrama de caja de dos muestras de las mediciones tomadas.*



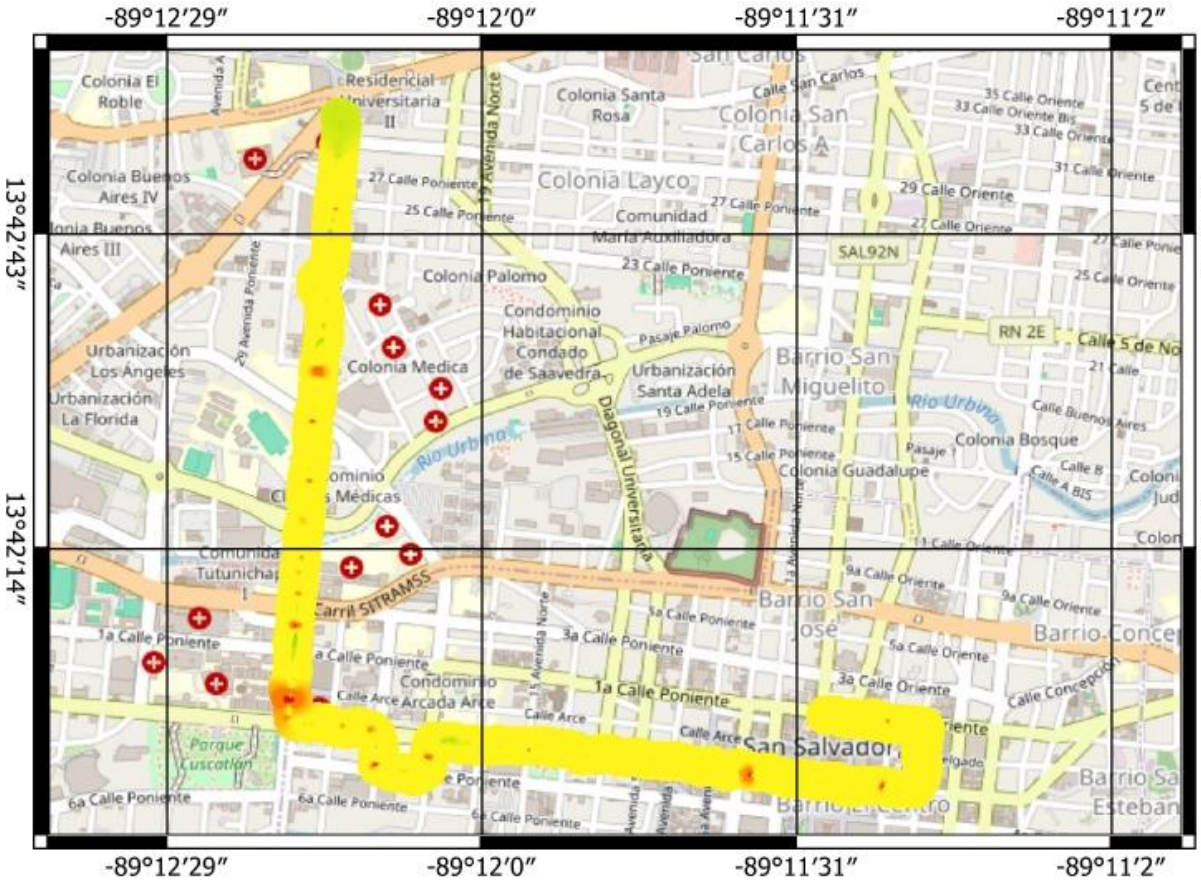
(a) **(b)**

Nota: El diagrama del literal a) muestra las mediciones realizadas en un rango de tiempo donde se encuentra mayor congestionamiento vehicular el día 11/03/2025 de 11:37 a 12:16 y el literal b) cuando la congestión vehicular es más reducida el día 12/03/2025 de 17:53 a 18:34. Fuente: Elaboración propia utilizando la aplicaciòn de Excel.

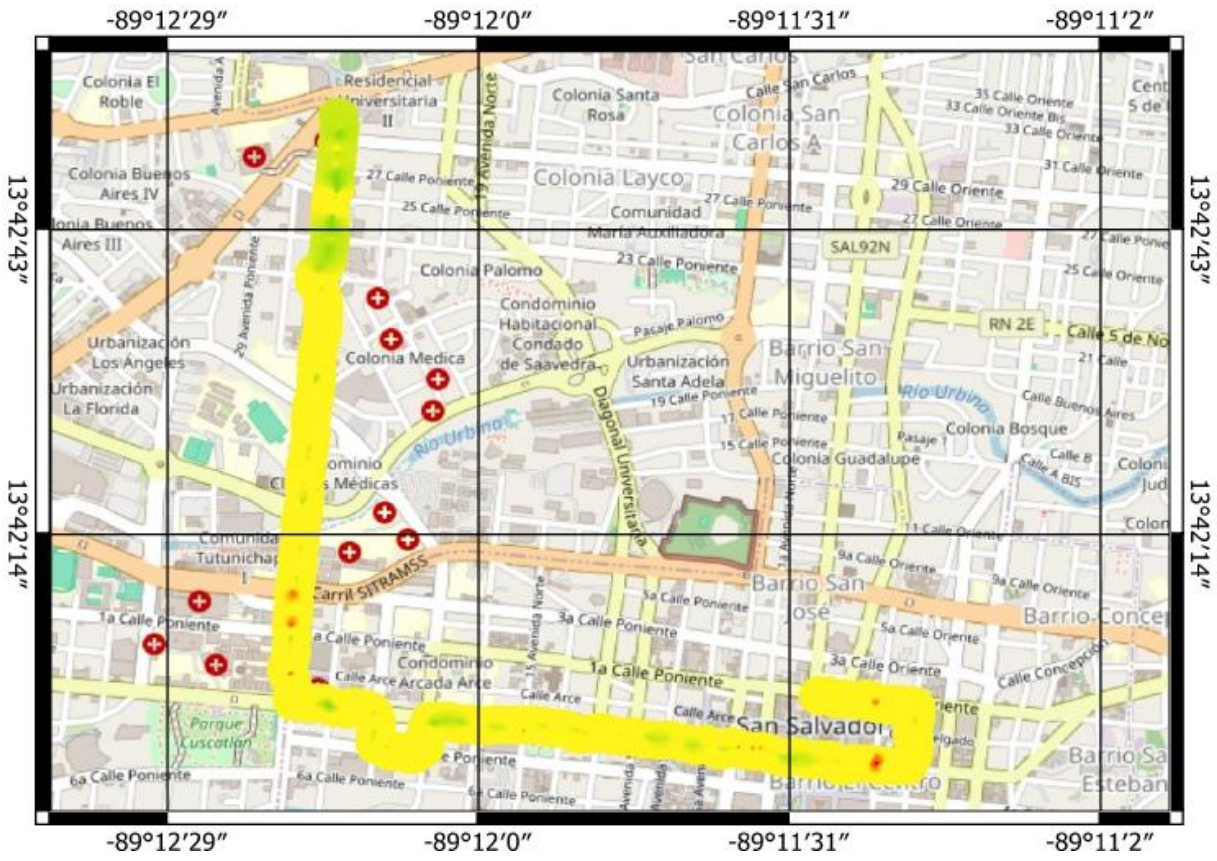
Al pasar los datos de las mediciones tomadas a QGIS, y para visualizarlas de mejor forma, se graficaron en mapas de calor, teniendo desde el color verde indicando una calidad de aire buena hasta el color rojo siendo una calidad de aire perjudicial.

Las zonas en las que se encontraban con mejor calidad de aire como lo muestra la Figura 4.2 a) y b) fue el Palacio Nacional, la Plaza Gerardo Barrios y Hospital de Niños Benjamín Bloom, las zonas donde se la calidad de aire era perjudicial fueron el Hospital General ISSS y el Hospital de Especialidades, las posibles causas de que en estos lugares hayan tenido esa terrible calidad de aire es por estar en renovación de las edificaciones las cuales generaban polvo y el uso de maquinarias pesadas.

Figura 4.2. a) b) *Mapa de calor de dos muestras de las mediciones tomadas.*



(a)



(b)

Nota: En la comparación de los dos literales anteriores, las zonas de color rojo se mantienen siendo más visibles cuando hay más congestionamiento vehicular.En el inciso a) mediciones realizadas el día 11/03/2025 de 11:37 a 12:16 y b) el día 12/03/2025 de 17:53 a 18:34. Fuente: Elaboración propia utilizando QGIS, OpenStreetMap.

El armado de la estación móvil tuvo un costo de acuerdo a la tabla 4.1.

Tabla 4.1. *Costo estimado de la estación móvil.*

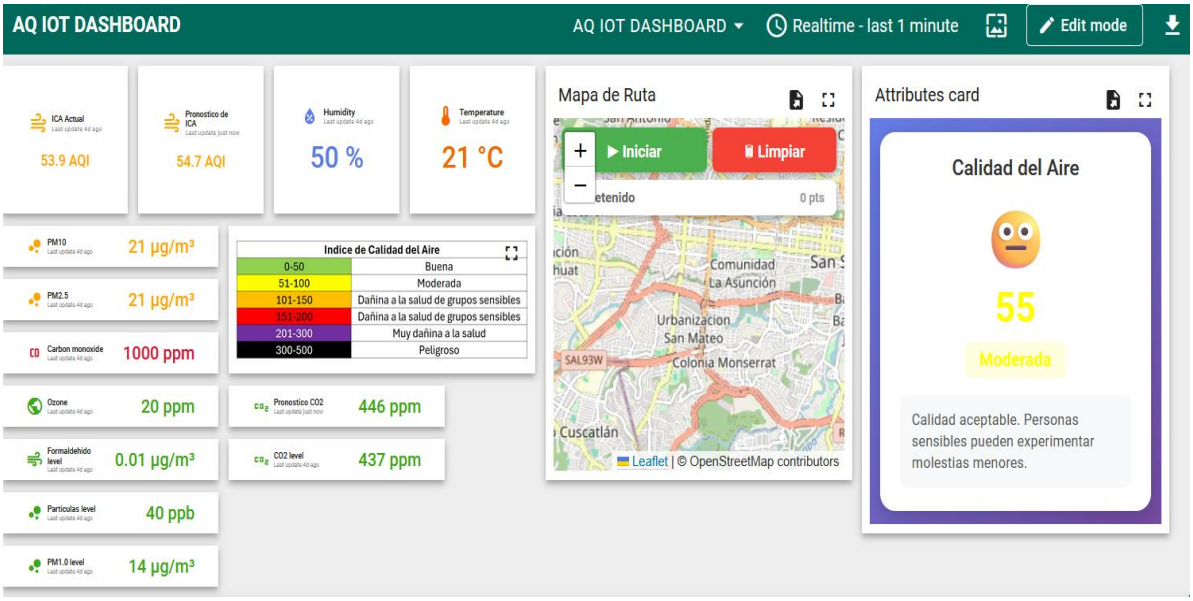
Componente	Precio
Liligo T-Sim7000G	$52.50
Sensor de O3	$49.00
Sensor de CO2 y TVOC	$29.90
Sensor de CO	$39.90
PMS5003ST	$36.18
Pantalla SSD1963	$35.99
Filamento PET	$19.95
Esp32 CH340	$20.00
TOTAL	$289.42

Nota: Los precios pueden variar si el envío presenta un costo.

4.2. Dashboard final en la plataforma Thingsboard

Como parte del layout del dashboard de Thingsboard (Figura 4.3) tenemos un widget para cada uno de los contaminantes, como el CO2, CO, O3, Formaldehido, TVOC, PM 1, 2.5 y 10. También se muestra el valor actual del Índice de Calidad del Aire, así como también un pronóstico del nivel del CO2 y del ICA futuro. Incluye además un mapa con el cual se puede realizar tracking de la ruta en la que se realizó la medición y un widget que muestra un comentario breve acerca del ICA pronosticado. Este dashboard se puede visualizar desde el sitio web de Thingsboard cloud, tanto en el escritorio como en un smartphone (Figura 4.4).

Figura 4.3. *Layout de escritorio del Dashboard de Thingsboard.*



Nota: Los datos mostrados en esta captura de pantalla son parte de una secuencia de prueba tomada por la estación móvil de calidad del aire. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

Figura 4.4. *Porción del layout de smartphone del Dashboard de Thingsboard.*

Nota: Esta configuración de pantalla se genera automáticamente a medida que se agregan y editar los widgets en la configuración normal. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

La figura 4.5. muestra un instante durante una campaña de medición. Se observa los siguientes valores para ese momento:

CO2: 640ppm

CO: 0 mg/m3

O3: 0 ug/m3

Formaldehido: 0.01ppm

PM1: 22 ug/m3

PM2.5: 31 ug/m3

PM10: 32 ug/m3

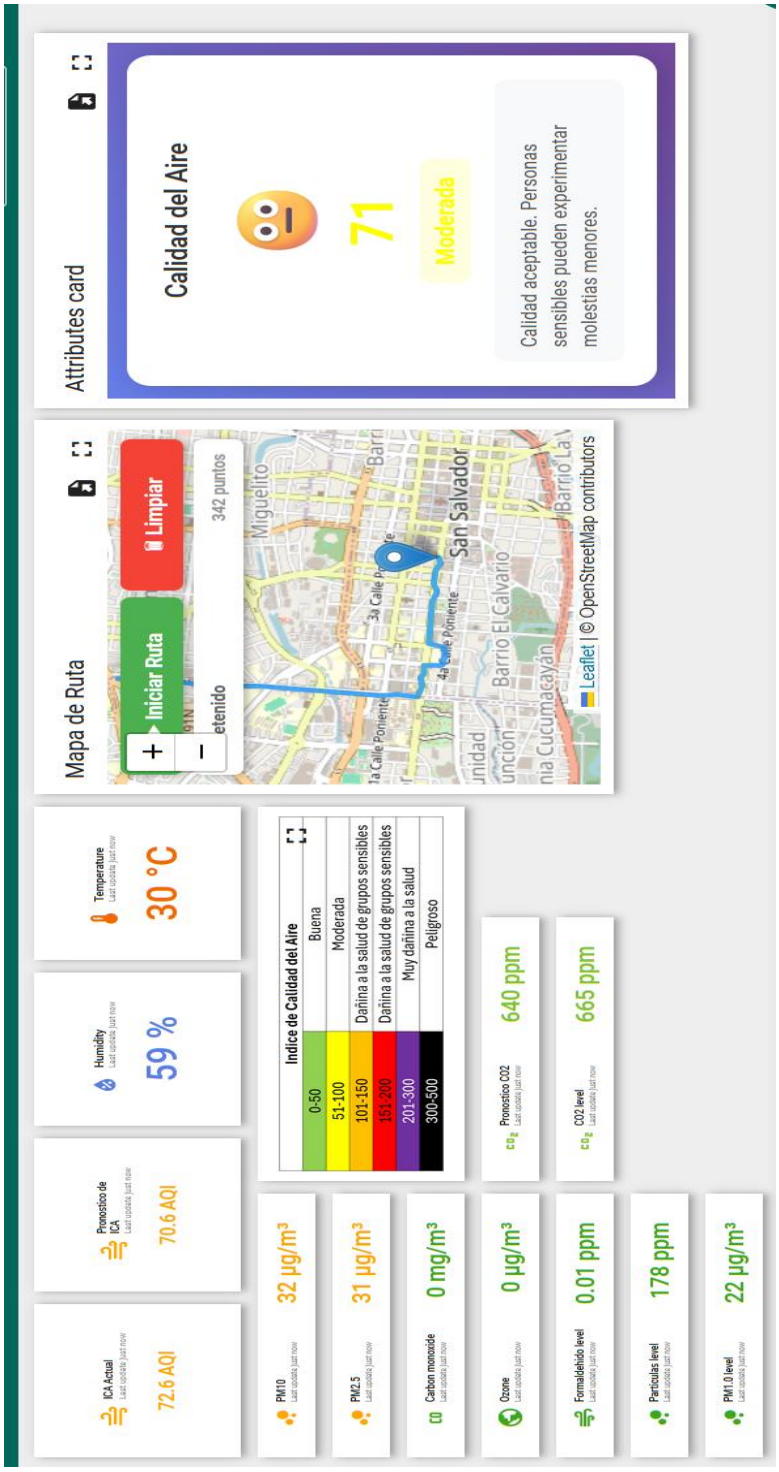
Temperatura 30°C

Humedad 59%

Pronóstico de CO2: 665 ppm

Con los cuales tenemos un ICA de 72.6, el cual entra en el rango de calidad moderada del aire. Luego también, con el dato de CO2 predicho, se calcula el índice de calidad del aire, Se observa también cómo a lo largo de todo el trayecto la ruta se iba trazando sobre el mapa.

Figura 4.5. *Elementos del Dashboard durante una campaña de medición.*



Nota: En este momento exacto de la campaña de medición, se muestra una cantidad moderada de concentración de partículas en el aire. Captura de pantalla de sitio web de Thingsboard (2025). Fuente: Elaboración propia.

4.3. Resultados Machine Learning

Tabla 4.2. *Resultado de métricas del algoritmo de machine learning*

Métrica	¿Qué mide?	Valor idóneo	Resultado
RMSE (Root Mean Square Error)	Magnitud promedio de los errores, penalizando más los grandes.	→ 0	124.66
MAE (Mean Absolute Error)	Magnitud promedio de los errores absolutos.	→ 0	105.7
R^2 (Coeficiente de determinación)	Proporción de la varianza explicada por el modelo.	→ 1	0.8341
NSE (Nash-Sutcliffe Efficiency)	Eficiencia respecto a usar la media como predictor.	→ 1	0.8341

Nota: Se muestra una tabla resumen con el resultado de las métricas resultantes que evalúan el desempeño del modelo.

En la Tabla 4.2 se presentan las métricas utilizadas para evaluar el desempeño del modelo predictivo: RMSE, MAE, coeficiente de determinación (R^2) y eficiencia de Nash-Sutcliffe (NSE). Estas métricas permiten analizar la precisión, consistencia y capacidad explicativa del modelo frente a los valores observados. El Error Cuadrático Medio (RMSE) obtuvo un valor de 124.66, lo que indica que, en promedio, las predicciones del modelo presentan desviaciones de esta magnitud respecto a los valores reales. Dado que esta métrica penaliza con mayor peso los errores grandes, su valor refleja la presencia de algunas predicciones con errores relativamente elevados, aunque sin comprometer el desempeño global del modelo. Por su parte, el Error Absoluto Medio (MAE) registró un valor de 105.7, lo que representa el error promedio absoluto entre los valores observados y los valores estimados. Al no elevar los errores al cuadrado, esta métrica proporciona una medida más directa y fácil de interpretar del error típico del modelo, evidenciando una diferencia moderada entre las predicciones y los datos reales. El coeficiente de determinación (R^2) alcanzó un valor de 0.8341, lo que significa que aproximadamente el 83.41 % de la variabilidad presente en los datos observados es explicada por el modelo. Este resultado indica una alta capacidad explicativa y un buen ajuste entre los valores predichos y los valores reales. De manera consistente, la Eficiencia de Nash-Sutcliffe (NSE) presentó un valor de 0.8341, lo que confirma que el modelo ofrece un desempeño considerablemente superior al de utilizar la media de los datos observados como predictor. Un valor cercano a 1 en esta métrica indica un buen nivel de eficiencia predictiva.

Capitulo 5. CONCLUSIONES

En la búsqueda de medidores de calidad de aire con alta confiabilidad respecto a las medidas se encontraron de diferentes rangos de precios están los que rondaban de $50 a $300 los cuales solo contaban con lo básico como era el sensor de PM, y otros de $500 a $1500, donde traían una más variedad de contaminantes a medir. La estación móvil que se creó, el precio rondó en los $289.42, siendo un precio de bajo costo a comparación de los medidores que tenían una similar función. Otro de los beneficios extras que cuenta la estación móvil es la geolocalización de los datos y el envío de esta información a una plataforma para su visualización en paralelo.

De acuerdo a las mediciones realizadas aplicando la fórmula para del ICCA, se observó que hubo tendencias muy cercana, como es el caso de las concentraciones de ozono, las cuales oscilaban entre el valor de 0 a 250, teniendo su media rondando por el valor de 100, que según la tabla lo ubica en un nivel de calidad de aire moderado a insalubre para grupos sensibles.

Para el valor de las partículas orgánicas volátiles totales se encuentra en el rango de 0 a 100 teniendo su media abajo del valor de los 50, el valor de $PM_{1.0}$ en el rango de 0 a 50, el valor de PM_{10} en el rango de 0 a 50 y formaldehído en el rango de 0 a 50, indicando un valor de calidad de aire bueno. En la concentraciones de dióxido de carbono el rango oscilaba entre los 50 a 100 al igual que las concentraciones de $PM_{2.5}$, señalando un nivel de calidad de aire moderado.

A través de la arquitectura de nube IOT desarrollada se logró recopilar muestras de concentraciones de contaminantes en el aire tomadas por la estación móvil de calidad del aire durante las campañas, las cuales fueron enviadas automáticamente por medio del servidor hacia la plataforma de visualización de datos. Al mismo tiempo, estos datos se guardaron dentro de una base de datos y con estas entradas se generaron pronósticos de calidad de aire utilizando el modelo de machine learning entrenado.

Los datos de concentraciones de contaminantes en el aire recopilados en las campañas fueron procesados a través de métodos de ciencia de datos, con los cuales el modelo LSTM fue entrenado y luego registrado dentro del servidor utilizando la plataforma de Mflow. Con este servicio se desplegó el modelo y se creó una tarea repetitiva que recibe los últimos datos de la estación móvil en tiempo real. Esta tarea se repetía en un intervalo de 1 minuto para generar pronósticos sobre las concentraciones de CO_2 futuras.

La configuración del firewall ufw, reforzó la seguridad del servidor IOT, bloqueando todos los puertos, excepto los puertos 80 y 443. De esta forma se evitó que escáneres automáticos y bots de

internet lograran acceder a los servicios expuestos por Docker, únicamente el servicio de Nginx recibe peticiones, las cuales son filtradas y rechazadas al intentar acceder a recursos no autorizados.

Los modelos LSTM aplicados demostraron una alta capacidad para capturar las tendencias temporales de los contaminantes atmosféricos, alcanzando métricas de desempeño satisfactorias (R^2 y NSE cercanos a 0.83). Estos resultados confirman la confiabilidad del enfoque de aprendizaje automático para la predicción de concentraciones futuras y el cálculo del Índice de Calidad del Aire (ICA) conforme a los estándares establecidos por el MARN, sentando una base sólida para futuras investigaciones orientadas a la mejora y expansión del sistema propuesto.

El preprocesamiento de los datos, incluyendo la normalización y la estructuración temporal de las series, fue un factor determinante en el desempeño del modelo LSTM. La correcta preparación de la información contribuyó a la estabilidad del entrenamiento y a la precisión de las predicciones obtenidas, confirmando la importancia de esta etapa dentro del proceso metodológico aplicado.

El desempeño alcanzado por el modelo LSTM demuestra que las técnicas de aprendizaje automático son herramientas efectivas para el análisis de series temporales ambientales, ya que lograron aprender los patrones subyacentes de los datos sin necesidad de supuestos lineales. Esto confirma que el enfoque seleccionado es apropiado para la predicción de concentraciones de contaminantes y para el análisis dinámico de su comportamiento en el tiempo.

El análisis del desempeño del modelo pone en evidencia que la efectividad de las técnicas de aprendizaje automático aplicadas depende directamente de la estructura temporal de los datos y de su correcta preparación. A pesar de las limitaciones propias del conjunto de datos utilizado, los resultados obtenidos confirman que el modelo LSTM se ajustó adecuadamente a la información disponible, logrando predicciones coherentes y estables.

Capitulo 6. RECOMENDACIONES

1. La pantalla hace uso del modo de comunicación paralelo de 8 bits, el cual ocupa una mayor cantidad de pines para la conexión, lo que reduce las posibilidades de adaptar más dispositivos a una misma placa desarrolladora. Por lo cual, se propone hacer uso de otros modos de comunicación como es el protocolo SPI, que solamente hace uso de 4 pines para su comunicación.

2. Los datos atmosféricos con los que se trabajaron están promediados a periodos de 24 horas, siendo estos periodos muy largos si los comparamos con los periodos de los datos de frecuencia. Para obtener resultados más claros y precisos se necesitaría de datos atmosféricos en periodos más cortos que se acerquen al periodo de los datos de frecuencia.

3. La estación móvil con frecuencia genera datos incompletos y erróneos o datos que no son de utilidad para el desarrollo del modelo de machine learning (por ejemplo: al encender el dispositivo, este empieza a enviar datos hacia el servidor inmediatamente, pero estos son valores nulos que generan entradas llenas de ceros en la base de datos). Para evitar popular la base de datos con estas entradas, se recomienda aplicar un filtro automático de datos ya sea en el punto de origen, la estación móvil, o, dentro del servidor antes de crear nuevas entradas a la base de datos.

4. Actualmente la nube IOT cuenta con una configuración básica de seguridad. Se utiliza un firewall que únicamente permite conexiones con puertos 80 y 443. Se recomienda robustecer la seguridad mediante tokens de API que aseguran que solamente los dispositivos con acceso a los tokens son capaces de hacer uso de los endpoints expuestos por la API.

5. Como medida de seguridad adicional, se sugiere implementar un certificado SSL utilizando Let’s Encrypt, la cual es una autoridad certificadora que emite certificados SSL gratuitos. Nginx ofrece por defecto soporte para el uso de esto. Como requisito se necesita tener un dominio web (enlace personalizado tipo ejemplo.com)

6. El modelo predictivo hace uso de una arquitectura LSTM estándar, la cual logra un desempeño aceptable pero presenta limitaciones en la captura de dependencias de largo plazo y en la estabilidad de las métricas obtenidas. Esto reduce las posibilidades de mejorar la precisión en escenarios más complejos o con mayor variabilidad de datos. Por lo cual se propone explorar arquitecturas híbridas, como LSTM con mecanismos de atención o modelos basados en Transformers, que permiten un mejor aprovechamiento de las secuencias temporales y una reducción significativa en los errores de predicción, contribuyendo a mejorar métricas como RMSE, MAE y R^2.

Capitulo 7. REFERENCIAS

Arévalo, F. (2025). “Desarrollo de código LSTM” https://www.researchgate.net/profile/Fernando-Arevalo-4, https://orcid.org/0000-0001-8664-2626

Asamblea Legislativa de la República de El Salvador. (2000) *Decreto Nª40 Reglamento Especial de Normas Tecnicas de Calidad Ambiental*

B. A. Castillo-Rosales, L. D. Segovia-López, C. Pocasangre, O. O. Flores, and F. Arevalo-Navas. (2024) Estudio de calidad del aire presente en el área metropolitana y su estado respecto a los índices centroamericanos en el contorno de la Universidad de El Salvador. *Revista Minerva, vol. 7,* pp. 1–10, 10. https://doi.org/10.5377/revminerva.v7i3.18911

Bergmann, D. (2023). “*¿Qué es el machine learning?”*. https://www.ibm.com/mx-es/think/topics/machine-learning.

BOSCH (2025). *BME280 Datasheet.* https://dfimg.dfrobot.com/nobody/wiki/eebf7904aecb84aeebf5af3f6a19533f.pdf

Brownlee, J. (2019). “*Semi-Supervised Learning With Label Propagation”*. https://machinelearningmastery.com/semi-supervised-learning-with-label-propagation/

Canva (2025). *Templates – Canva.* https://www.canva.com/templates

Contabo GmbH. (2025). Hosting VPS. https://contabo.com/es/vps/

Data Center Market. (2025, septiembre 17). “*Qué es el deep learning y cómo funciona. Desafíos y retos”*. https://www.datacentermarket.es/dcm-xl/que-es-el-deep-learning-y-como-funciona-desafios-y-retos/

DataCamp. (2025). “*Clustering jerárquico: definición y ejemplos”*. https://www.datacamp.com/es/tutorial/hierarchical-clustering

DataScientest. (2023). “Machine Learning: definición, funcionamiento, usos”. https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos.

Docker Inc. (2025). What is a container?. https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-a-container/

Docker, Inc. (2025). What is Docker?. https://docs.docker.com/get-started/docker-overview/

GeeksforGeeks. (2025). “*What is LSTM - Long Short Term Memory?”.* https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/

GeeksforGeeks. (2025, julio 23). "*Multivariate Time Series Forecasting with LSTMs in Keras"*. https://www.geeksforgeeks.org/deep-learning/multivariate-time-series-forecasting-with-lstms-in-keras/

GeeksforGeeks. (2025, julio 23). "Self-Training in Semi-Supervised Learning". https://www.geeksforgeeks.org/machine-learning/self-training-in-semi-supervised-learning/

Generalitat Valenciana. (2024) *Composición química.* Vicepresidencia Tercera y Conselleria de Medio Ambiente, Infraestructuras, Territorio y de la Recuperación. https://mediambient.gva.es/es/web/calidad-ambiental/composicion-quimica

Hanes, D., Salgueiro, G., Grossetête, P., Barton, R., & Henry, J. (2017). *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*.

https://img.dfrobot.com.cn/wiki/5b973267c87e6f19943ab3ad/b5b08fe2ea631f0becdfa0c15db88c4a.pdf

Harris, C. R., Millman, K. J., van der Walt, S. J., & NumPy Development Team. (2025). *NumPy: The fundamental package for scientific computing with Python.* https://numpy.org

Hunter & Matplotlib Development Team. (2025). *Matplotlib: Visualization with Python.* https://matplotlib.org

IBM. (2025). "*¿Qué es el aprendizaje semisupervisado?"*. https://www.ibm.com/es-es/think/topics/semi-supervised-learning

IBM. (2025). "*¿Qué es el deep learning?"*. https://www.ibm.com/es-es/think/topics/deep-learning

Jgraph. (2025) Draw.io web version. https://app.diagrams.net/

Kpucha. (2023, agosto 10). "*Algoritmos de aprendizaje no supervisado: fundamentos, aplicaciones y ejemplos"*. https://blog.kpucha.dev/posts/algoritmos-de-aprendizaje-no-supervisado-fundamentos-aplicaciones-y-ejemplos/

Kpucha. (2023, agosto 10). "*Algoritmos de aprendizaje no supervisado: fundamentos, aplicaciones y ejemplos"*. https://blog.kpucha.dev/posts/algoritmos-de-aprendizaje-no-supervisado-fundamentos-aplicaciones-y-ejemplos/

Kpucha. (2023, agosto 10). "*Algoritmos de aprendizaje no supervisado: fundamentos, aplicaciones y ejemplos"*. https://blog.kpucha.dev/posts/algoritmos-de-aprendizaje-no-supervisado-fundamentos-aplicaciones-y-ejemplos/

Kumar, A. (2025, diciembre 16). "*LSTM Architecture Explained: How Gates Control Memory in Neural Networks"*. https://wireunwired.com/lstm-architecture-explained-how-gates-control-memory-in-neural-networks/

Lilygo (2025). *LILYGO-T-SIM7000G-16MB.* https://agelectronica.lat/pdfs/textos/L/LILYGO-T-SIM7000G-16MB.PDF

Masse, M. (2011). REST API design rulebook. O'Reilly Media.

Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment..

Mlflow. (2025) Official Mlflow Docker Image. https://mlflow.org/docs/latest/ml/docker/

Müller, A. C., & Guido, S. (2016). Introduction to machine learning with Python: A guide for data scientists. O'Reilly Media. https://files.addictbooks.com/wp-content/uploads/2024/07/Introduction-to-Machine-Learning-with-Python.pdf

NGINX Inc. (2025). "NGINX Reverse Proxy". https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

O. O. Flores-Cortez, C. O. Pocasangre, F. Arevalo, B. A. Castillo, and L. D. Segovia. (2024) Mobile air quality sensor system for gis mapping, study case in san salvador city. *IEEE Xplore,* pp. 1–6.

O. O. Flores-Cortez, C. Pocasangre Jimenez, F. Arévalo, R. L. López, D. Peña Martínez, and O. P. Rafael. (2023) A low-cost iot mobile system for air quality monitoring in developing countries, a study case in El Salvador. *IEEE Xplore*, pp. 1–8.

O. O. Flores-Cortez, R. Adalberto Cortez, and V. I. Rosa. (2019) A Low-cost IoT System for Environmental Pollution Monitoring in Developing Countries.

Oracle. (2024, octubre 29). "*Explicación del aprendizaje semi-supervisado"*. https://www.oracle.com/latam/artificial-intelligence/machine-learning/semi-supervised-learning/

Pandas development team. (2025). *Pandas: Python data analysis library*. https://pandas.pydata.org

Plantower (2025). *Digital universal particle concentration sensor.* https://homotix_it.e-mind.it/upld/catalogo/doc/PMS5003ST.pdf

Project Jupyter. (2025). *Jupyter Notebook: The classic notebook interface*. https://jupyter.org

PyTorch Foundation. (2025). *PyTorch: An open source machine learning framework*. https://pytorch.org

Ramírez S. (2025). FastAPI Documentation. https://fastapi.tiangolo.com/

Red Hat. (2022, diciembre 21). "*¿Qué es el machine learning?*". https://www.redhat.com/es/topics/ai/what-is-machine-learning

Rohini college of Engineering and Technology. (2021). Simplified IoT Architecture and Core IoT Functional Stack. https://www.rcet.org.in/uploads/academics/regulation2021/rohini_60698112657.pdf

Rohini college of Engineering and Technology. (2021). Simplified IoT Architecture and Core IoT Functional Stack. https://www.rcet.org.in/uploads/academics/regulation2021/rohini_60698112657.pdf

Rohini college of Engineering and Technology. (2021). Simplified IoT Architecture and Core IoT Functional Stack. https://www.rcet.org.in/uploads/academics/regulation2021/rohini_60698112657.pdf

Scikit-learn. (2025). "Semi-supervised learning". https://scikit-learn.org/stable/modules/semi_supervised.html

SGX Sensortech (2025). *SensorMiCS-4514 Datasheet.*

Solomon Systech (2025). *SSD1963.* https://www.seacomp.com/sites/default/files/references/Solomon-Systech-SSD1963.pdf

Spyder IDE. (2025). "Spyder: The Python IDE that scientists and data analysts deserve". https://www.spyder-ide.org.

StatDeveloper. (2025). "*Algoritmo de agrupación por K-medias"*. https://www.statdeveloper.com/agrupacion-por-k-medias/

Thingsboard (2025). *ThingsBoard Open-source IoT Platform.* https://thingsboard.io

Thingsboard, Inc. (2025). What is Thingsboard?. https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/

U.S. Environmental Protection Agency [EPA]. (2024) *Air quality index (aqi) basics.* AirNow. https://web-archive-org.translate.goog/web/20180612115717/

U.S. Environmental Protection Agency [EPA]. (2024) *Technical assistance document for the reporting of daily air quality– the air quality index (aqi).* AirNow. https://www.airnow.gov/publications/air-quality-index/technical-assistance-document-for-reporting-the-daily-aqi/

Waskom, M. L., & Seaborn Development Team. (2025). *Seaborn: Statistical data visualization*. https://seaborn.pydata.org

Winsen (2025). *Ozone Sensor Datasheet.* https://dfimg.dfrobot.com/nobody/wiki/3c1da72292464bac7c2761d932e72b1c.pdf

Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. IEEE Data Eng. Bull., 41, 39–45. https://api.semanticscholar.org/CorpusID:83459546