ANDES BOARD

SISTEMA DE MONITOREO Y SUPERVISIÓN REMOTO MEDIANTE SENSORES IOT PARA DETERMINAR EL NIVEL DE GAS GLP DE USO DOMÉSTICO EN SU ESTADO BIFASICO.

Autor y productor:

Juan Camilo Suárez Soto

Corporación universitaria Minuto de Dios

Rectoría principal

Sede Calle 80 (Principal)

Programa de ingeniería de sistemas

Noviembre de 2025

ANDES BOARD
SISTEMA DE MONITOREO Y SUPERVISIÓN REMOTO MEDIANTE SENSORES IOT PARA DETERMINAR EL NIVEL DE GAS GLP DE USO DOMÉSTICO EN SU ESTADO BIFASICO.

Autor y productor:

Juan Camilo Suárez Soto

Trabajo de grado presentado como requisito para optar al título de ingeniero de sistemas

Asesor:
Juan Carlos Muñoz Vera

Corporación universitaria Minuto de Dios
Rectoría Principal
Sede Calle 80(Principal)
Programa de ingeniería de sistemas
Noviembre de 2025

Tabla de Contenido

Tabla de Ilustraciones

Lista de tablas

Resumen

El proyecto titulado Andes Board, tiene como propósito diseñar e implementar un sistema de monitoreo remoto para cilindros domésticos de gas GLP, utilizando tecnologías IoT, cálculos de física de fluidos y un enfoque web adaptable. Este trabajo surge ante la dificultad que existe para medir con precisión el nivel de gas, especialmente cuando se encuentra en estado bifásico, situación que limita la planificación de recargas y puede generar interrupciones en el suministro.

A través de sensores de presión y temperatura conectados a internet, se capturan variables físicas que luego son transformadas en indicadores como el flujo másico, la masa removida y la masa restante del gas. Dichos cálculos se realizan aplicando modelos de física de fluidos establecidos por la ecuación de Bernoulli que consideran los regímenes de flujo crítico y subcrítico, según las condiciones de presión diferencial y temperatura del cilindro.

Los datos procesados se almacenan en una base de datos y se presentan mediante un dashboard web interactivo desarrollado con Django y TailwindCSS, permitiendo al usuario visualizar el comportamiento del gas en los cilindros. Todo el entorno ha sido configurado y desplegado mediante contenedores Docker, lo que garantiza portabilidad, independencia de entorno y facilidad para la integración de los diferentes servicios, como Redis, Celery y PostgreSQL.

Además, el sistema incluye un módulo de notificaciones automáticas que alerta al usuario ante comportamientos anómalos o niveles críticos de presión, fortaleciendo así la seguridad y la continuidad del suministro. En términos generales, los resultados obtenidos confirman que la propuesta es viable tanto técnica como funcionalmente, logrando una comunicación estable entre los sensores y el dashboard, y una precisión aceptable en los cálculos validados de forma empírica.

Abstract

The project Andes Board presents the design and implementation of a remote monitoring system for domestic LPG cylinders through the integration of IoT technologies, thermodynamic modeling a Software as a Service (SaaS) architecture. The lack of precision in gas-level measurement—particularly in biphasic states—limits users' ability to plan refills and increases the risk of supply interruptions. To address this issue, pressure and temperature sensors connected to microcontrollers collect physical variables of the system, which are then transformed into gas behavior indicators such as mass flow rate, removed mass, and remaining mass. These indicators are derived from thermodynamic models that consider critical (choked) and subcritical (non-choked) flow regimes, depending on the pressure and temperature conditions within the cylinder.

The processed data are transmitted to a database and visualized through an interactive web dashboard developed with Django and Tailwind CSS. In production, the system is deployed on Microsoft Azure, ensuring scalability, high availability, and analytical capabilities. Additionally, an automated email alert system notifies users in case of abnormal consumption patterns or critical pressure levels. The proposed solution not only enhances the domestic user experience but also provides a scalable model for gas distribution companies seeking to manage inventories with greater precision, safety, and automation.

Capítulo I: Introducción

1.1 Contexto y justificación

El gas licuado de petróleo (GLP) es una de las fuentes de energía más utilizadas tanto en zonas rurales como en hogares urbanos de bajos recursos en Colombia. Su almacenamiento en cilindros portátiles ha sido una práctica tradicional durante muchos años; sin embargo, las tecnologías disponibles para monitorear el contenido suelen ser costosas y poco accesibles. Por otro lado, los métodos manuales como estimar el peso, calcular el tiempo de uso o usar indicadores mecánicos básicos no brindan resultados precisos ni garantizan la seguridad del usuario. Frente a esta problemática, surge la necesidad de crear un sistema automatizado, confiable y de bajo costo que permita conocer con exactitud la cantidad de gas disponible en el cilindro, incluso cuando se encuentra en estado bifásico, es decir la mezcla de líquido y vapor (Ilustración 1).

El proyecto Andes Baird nace como respuesta a esa necesidad. Su diseño integra sensores de presión y temperatura conectados a un servidor que ejecuta ecuaciones de flujo másico para estimar el volumen de gas restante. Para ello, se emplean microcontroladores de bajo consumo energético, comunicación mediante el protocolo MQTT y almacenamiento en PostgreSQL. En la etapa de producción, el sistema se desplegará en contenedores Docker, lo que permitirá un entorno más flexible, escalable y fácil de mantener. Además, el sistema incluirá funciones de visualización y envío automático de alertas por correo electrónico.

Este documento presenta la arquitectura del sistema Andes Board, fundamentada en principios termodinámicos, cálculos fisicoquímicos e ingeniería aplicada al monitoreo de GLP. Se describe la selección del hardware y las tecnologías empleadas: Django y Celery para el backend, y Tailwind CSS en el frontend. El desarrollo sigue una metodología ágil basada en SCRUM, priorizando la iteración continua y la mejora del producto.

1.1.1 Detalles del problema a resolver

En la empresa Andes Technologies ha surgido la necesidad de determinar con precisión la cantidad de gas GLP en su estado bifásico (mezcla de líquido y vapor) dentro de los cilindros domésticos, con el fin de mejorar la gestión del recurso y reducir los riesgos asociados a fugas o desabastecimiento.

El uso de cilindros de gas GLP de uso doméstico es común en Colombia y en muchos países de América Latina, especialmente en zonas rurales y estratos socioeconómicos bajos. Sin embargo, la ausencia de sistemas de monitoreo que impide conocer con precisión el volumen restante de gas genera incertidumbre sobre la disponibilidad del recurso y dificulta una adecuada planificación de recarga. Esta falta de visibilidad puede traducirse en interrupciones inesperadas del servicio, afectando actividades esenciales como la preparación de alimentos y la higiene personal.

Actualmente, los usuarios dependen de estimaciones basadas en el peso percibido del cilindro o el tiempo de uso transcurrido desde la última recarga. Esto no permite un monitoreo remoto del nivel de gas, lo cual puede generar situaciones problemáticas, especialmente en contextos donde la continuidad del servicio es esencial, como en entornos rurales o zonas sin acceso a redes de distribución continua. Dado que los cilindros de GLP son herméticos y no permiten observar directamente su contenido, la falta de un método confiable para conocer el nivel de gas puede conducir a interrupciones inesperadas o a recargas innecesarias.

Si bien existen tecnologías IoT que permiten medir parámetros como la presión o la temperatura del gas, estas soluciones suelen ser costosas, poco accesibles o mal adaptadas a la realidad técnica y económica de los usuarios residenciales. Además, muchos sensores del mercado no tienen la capacidad de inferir el volumen total de gas en estado bifásico (Ilustración 1), lo que limita su utilidad práctica.

La incapacidad para detectar patrones anómalos en el consumo o caídas bruscas de

presión también representa un riesgo potencial de seguridad, en tanto que estos eventos podrían ser indicativos de fugas, manipulación indebida o funcionamiento defectuoso del cilindro..

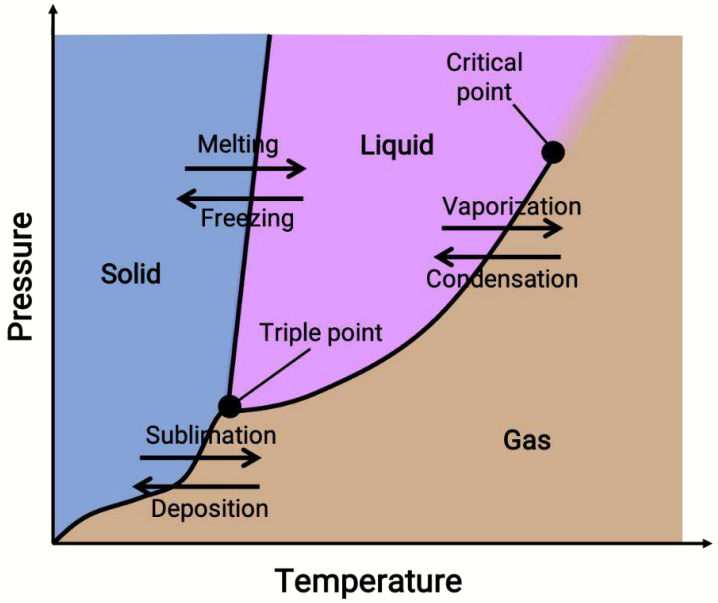


Ilustración 1: Diagrama de fases del gas. Nota. Fuente: LibreTexts Chemistry (2025). https://chem.libretexts.org/Courses/University_of_Toronto/UTSC%3A_First-Year_Chemistry_Textbook_(Fall_2025)/08%3A_Liquids_and_Solids_and_Phase_Changes, 2021

Sumado a esto, la ausencia de un sistema automatizado que notifique al usuario sobre comportamientos anómalos o niveles bajos de gas a través de mecanismos eficientes como correos electrónicos, reduce la posibilidad de actuar con anticipación y mitigar eventos no deseados.

¿Es posible determinar la cantidad de gas GLP en estado bifásico dentro de un cilindro doméstico mediante ecuaciones de flujo másico y sensores de presión y temperatura, e integrarse esta información en una solución tecnológica basada en servicios en la nube que permita su monitoreo remoto, visualización y generación de alertas automatizadas?

1.1.2 Descripción del problema

En la actualidad, los usuarios enfrentan dificultades importantes para conocer el nivel real de gas disponible, debido a que la mayoría de métodos existentes; como estimar el peso, medir el tiempo de uso o emplear indicadores mecánicos básicos; ofrecen un nivel de precisión limitado y no permiten un seguimiento continuo.

Esta falta de información genera riesgos operativos, interrupciones inesperadas del servicio y dificultades logísticas tanto para usuarios residenciales como para distribuidores de GLP. Además, las tecnologías disponibles que permiten medir parámetros como presión o temperatura suelen ser costosas o poco adaptadas al contexto económico local.

La problemática también incluye la incapacidad de los sensores convencionales para estimar la cantidad de gas en estado bifásico, lo que limita el valor real de las lecturas obtenidas. Esta limitación provoca que los usuarios y distribuidores carezcan de métricas confiables para tomar decisiones oportunas sobre recargas, mantenimiento o detección de anomalías. Asimismo, la ausencia de mecanismos automatizados de alerta disminuye la capacidad de prevenir situaciones potencialmente peligrosas, como fugas o comportamientos atípicos en el consumo.

En este contexto, Andes Board se propone integrar sensores de presión y temperatura con cálculos termodinámicos y un sistema web centralizado, con el fin de proporcionar un monitoreo remoto accesible, preciso y orientado a mejorar la seguridad y la experiencia del usuario.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar y probar un sistema de monitoreo remoto para cilindros de gas GLP, que mediante el uso de sensores IoT de presión y temperatura, y la aplicación de modelos de flujo másico, determine con precisión el volumen de gas en estado bifásico; e integre estos datos en una plataforma para su visualización y la generación de alertas automatizadas.

1.2.2 Objetivos específicos

- Definir las especificaciones de requerimientos del software aplicando los lineamientos de las normas IEEE 830/29148, complementado con modelado UML ISO/IEC 19501, con el fin de identificar de manera precisa las necesidades de visualización, monitoreo y generación de alertas asociadas al nivel de gas GLP.

- Diseñar una aplicación web de monitoreo bajo una arquitectura monolítica con el patrón Model-View-Template (MVT), aplicando principios de desarrollo limpio como SOLID y Strategy, y empleando herramientas modernas para la interfaz y el procesamiento asíncrono.

- Configurar e integrar la arquitectura tecnológica del sistema utilizando contenedores Docker y servicios de backend que permitan la recepción, procesamiento y almacenamiento de los datos provenientes de los sensores.

- Realizar una prueba piloto con un cilindro y el sensor de presión y temperatura, con el fin de validar el funcionamiento del sistema, verificar los cálculos termodinámicos y comparar los resultados estimados con mediciones físicas reales.

1.3 Alcance y limitaciones

1.3.1 Alcance del proyecto

El proyecto Andes Board consiste en el diseño, desarrollo e implementación de un sistema de monitoreo remoto para cilindros domésticos de gas GLP, enfocado en un modelo de negocio **B2B**, donde las empresas distribuidoras son las encargadas de instalar y supervisar los sensores en campo. En la parte técnica, el sistema integra sensores comerciales de presión y temperatura **TP2401** con conectividad 4G, junto con la aplicación de ecuaciones de **Bernoulli** para flujo crítico y subcrítico, lo que permite estimar el volumen de gas en estado bifásico. Además, se desarrolla una aplicación web con Django que funciona como dashboard, desde el cual se visualizan las variables medidas y se gestionan alertas automáticas.

El sistema se ejecutará dentro de **contenedores Docker**, utilizando una infraestructura que integra los servicios de **Django, PostgreSQL, Redis y Celery**, con el objetivo de facilitar su despliegue en la nube y garantizar una arquitectura modular, escalable y fácil de mantener.

Quedan fuera del alcance del proyecto el diseño o fabricación de hardware propio, así como cualquier modificación física del sensor TP2401; tampoco se abordará el desarrollo de nuevos protocolos de comunicación o servidores dedicados para la conexión simultánea de sensores. Igualmente, no se incluye la implementación de modelos predictivos basados en machine learning, ni el análisis de mercado sobre las zonas de instalación, dado que el enfoque del proyecto está orientado a empresas distribuidoras (modelo B2B).

Cabe resaltar que, aunque Andes Technologies trabaja actualmente con el sensor Helm HM200SE; el cual ofrece mayor precisión GPS y un costo cercano a 150 USD inferior al TP2401; para este proyecto se utilizará el sensor TP2401, facilitado por la empresa, con el fin de realizar las pruebas de integración y validación del sistema. (ver anexo A-23)

1.3.2 Limitaciones y supuestos

El proyecto se centra exclusivamente en la integración funcional de componentes existentes, evitando desarrollos de hardware o algoritmos propios. Se limita a un entorno experimental y de prueba controlada, en el que se validará la capacidad del sistema para estimar la cantidad de gas en cilindros domésticos mediante el análisis de presión, temperatura y peso.

Se asume que los sensores físicos estarán correctamente calibrados y conectados a internet y al cilindro de gas, así funcionarán sin interrupciones significativas, y que las pruebas se realizarán en territorio colombiano, conforme a las regulaciones nacionales vigentes para dispositivos IoT.

El proyecto corresponde a una investigación aplicada y tecnológica, con enfoque cuantitativo y experimental, orientada a la validación práctica de modelos físico-químicos y termodinámicos que permitan estimar el volumen y densidad del GLP en estado bifásico. Su propósito principal es aplicar conocimiento existente para resolver un problema técnico de la industria, más que generar teoría nueva.

Los resultados esperados buscan demostrar la viabilidad técnica y operativa del sistema, facilitando su futura escalabilidad hacia entornos productivos, una vez superadas las restricciones regulatorias y de comercialización internacional.

1.4 Metodología de investigación

La metodología empleada en este proyecto corresponde a una investigación aplicada, dado que busca desarrollar y validar un prototipo funcional orientado a resolver un problema real: el monitoreo del nivel de gas GLP en cilindros domésticos. Para ello, se implementó un estudio experimental tipo piloto o prueba de concepto, cuyo propósito fue evaluar el comportamiento del modelo físico-matemático desarrollado —basado en ecuaciones derivadas de Bernoulli y en el análisis de flujo crítico— frente a mediciones reales obtenidas sobre un cilindro de GLP en condiciones controladas.

El proceso experimental permitió contrastar los valores estimados por el sistema con los datos obtenidos mediante instrumentos físicos, particularmente una balanza digital. La variable independiente del estudio fue la técnica de estimación implementada por el modelo Andes Board, mientras que las variables dependientes correspondieron a la masa estimada del gas y al error de cálculo. Asimismo, se definieron variables de control, tales como masa del cilindro vacío y lleno, volumen interno, coeficiente de descarga, diámetro del orificio, composición típica del GLP, masa molar, entropía y factor Z. Estos parámetros garantizaron la repetibilidad y la validez del experimento.

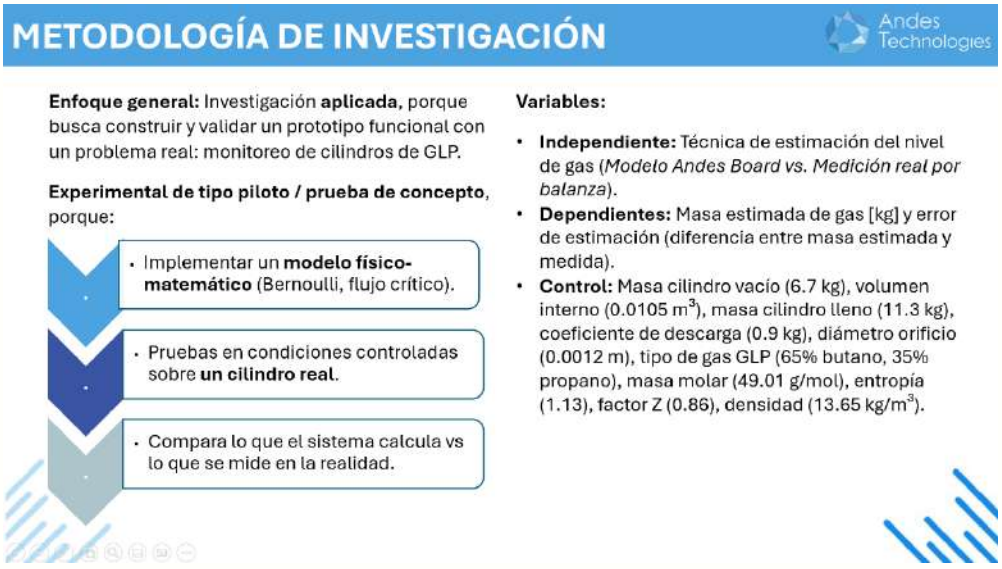


Ilustración 2: Diagrama de herramientas. Fuente: Elaboración Propia

Capitulo II: Revisión de literatura

2.1 Fundamentos de la arquitectura de software

2.1.1 Principios y patrones de la arquitectura de software

En los sistemas de software orientados al Internet de las Cosas (IoT), los principios arquitectónicos constituyen la base para garantizar soluciones robustas, escalables y sostenibles en el tiempo. Conceptos como la modularidad, la reutilización y el bajo acoplamiento permiten dividir el sistema en componentes independientes y fácilmente mantenibles, lo que facilita la incorporación de nuevas funcionalidades sin comprometer la estabilidad general de la aplicación. En el caso de Andes Board, estos principios resultan fundamentales para permitir la integración de nuevos tipos de sensores, algoritmos termodinámicos y servicios en la nube sin alterar la estructura base del sistema.

Asimismo, la arquitectura se guía por los principios SOLID, los cuales aseguran que cada componente cumpla una única responsabilidad, sea fácilmente extensible y mantenga un acoplamiento mínimo con el resto de la aplicación. En particular, el principio Open/Closed (O) se aplica para permitir que el sistema se amplíe con nuevos modelos de sensores o cálculos sin modificar el código existente, garantizando así la estabilidad del núcleo. Complementariamente, el uso del patrón Strategy posibilita definir múltiples estrategias de lectura, procesamiento y cálculo para distintos tipos de sensores, seleccionando dinámicamente la implementación adecuada en tiempo de ejecución. Este enfoque no solo promueve la reutilización de código bajo el principio DRY (Don’t Repeat Yourself), sino que también permite mantener la coherencia y mantenibilidad del sistema a largo plazo.

2.1.2 Modelos arquitectónicos

La solución implementa una arquitectura monolítica modular basada en el patrón MVT (Model–View–Template) de Django, que concentra la lógica de negocio, la gestión de usuarios y la interfaz de visualización de datos en un entorno coherente y de fácil mantenimiento. Aunque inicialmente se consideró una arquitectura basada en microservicios, se determinó que este enfoque generaría una sobrecarga innecesaria en la infraestructura, ya que implicaría crear y mantener un servicio independiente por cada modelo o marca de sensor (por ejemplo, sensores de presión Toprie o Helm Sensors). Dicha fragmentación aumentaría la complejidad y el costo de escalado sin aportar beneficios proporcionales, especialmente considerando que el sistema opera sobre una base de datos centralizada y flexible.

Por ello, se optó por mantener un monolito modular que, combinado con los principios SOLID y el patrón Strategy, ofrece una estructura lo suficientemente extensible para incorporar nuevos sensores, algoritmos o pipelines de cálculo sin afectar el núcleo del sistema. Este modelo facilita la comunicación interna entre módulos, reduce los tiempos de despliegue y simplifica la administración en entornos de contenedores como Docker o Azure Container Apps. De esta manera, Andes Board logra un equilibrio entre mantenibilidad y escalabilidad, garantizando que la arquitectura pueda evolucionar gradualmente sin incurrir en la complejidad de una infraestructura distribuida completa.

2.2 Desarrollo de software

2.2.1 Ciclo de vida de desarrollo de software (SDLC)

El ciclo de vida de desarrollo de software aplicado en Andes Board combinó las fases clásicas del SDLC con la dinámica iterativa del marco Scrum, lo que permitió avanzar en entregas incrementales y validaciones tempranas. (ver Ilustración 3).

Durante la fase de análisis, se definió la viabilidad técnica de emplear ecuaciones de flujo másico para estimar la cantidad de gas a partir de las mediciones de presión y temperatura obtenidas por sensores IoT. Esta etapa también incluyó la identificación de requerimientos funcionales y no funcionales del sistema (ver tabla 3 y tabla 4).

En la fase de diseño, se modeló la arquitectura general del sistema, integrando los módulos de captura de datos, procesamiento de cálculos y visualización mediante un dashboard web. (ver Ilustración 10, Ilustración 11)

La fase de desarrollo se abordó de forma iterativa a través de sprints semanales, donde se construyeron progresivamente el pipeline de cálculo, las estrategias de lectura de sensores y la interfaz del dashboard. (ver Anexo A-10)

Posteriormente, en la fase de pruebas, se compararon los resultados calculados con mediciones reales de cilindros, verificando la precisión del modelo y el desempeño del sistema.

Finalmente, en la etapa de despliegue y mantenimiento, la aplicación se implementó en entornos de producción utilizando Docker, garantizando su disponibilidad y escalabilidad. El mantenimiento continuo se centra en refinar las ecuaciones de Bernoulli y optimizar el rendimiento del sistema con base en la retroalimentación obtenida en cada sprint.

2.2.2 Metodologías ágiles vs. tradicionales

El contraste entre metodologías tradicionales y ágiles permite comprender la pertinencia del enfoque adoptado.

Los modelos tradicionales como el de cascada siguen una secuencia rígida de fases, priorizando la documentación y el control sobre la adaptabilidad, lo que dificulta atender cambios en proyectos con componentes físicos y digitales. Por el contrario, Scrum promueve un desarrollo iterativo e incremental, basado en la entrega continua de valor funcional.

En Andes Board, este enfoque resultó esencial, ya que permitió validar progresivamente la comunicación entre sensores y el dashboard, ajustar las ecuaciones, las variables y resolver incidencias sin esperar al final del ciclo. Gracias a esta flexibilidad, el sistema evolucionó de forma controlada, manteniendo la trazabilidad de las decisiones técnicas y garantizando entregables funcionales en cada sprint.

Aspecto	Metodologías Tradicionales (Cascada)	Metodologías Ágiles (Scrum)
Flujo de trabajo	No da cabida a dificultades con el sensor o el dashboard	permite reorganizar prioridades e iterar la función en el siguiente sprint
Retroalimentación	Al final del proyecto	Continua en cada ciclo
Documentación	Extensa y detallada	Ligera, centrada en lo necesario
Riesgo de error	Alto: problemas se detectan tarde	Bajo: errores se detectan temprano
Entregables	Solo al final del proceso	Parciales en cada sprint
Orientación	A procesos y documentos	A producto funcional y colaboración

Tabla 1: Metodología tradicional vs ágil en Andes Board. Elaboración propia

Por este motivo, se eligió SCRUM como la metodología de Andes Board.

Dentro de este enfoque, se implementó una metodología ágil de desarrollo, utilizando el marco de trabajo Scrum. Esta elección responde a la naturaleza iterativa y experimental del proyecto, en el que es necesario validar continuamente la interacción entre el hardware (sensores), el software (dashboard web) y las ecuaciones de Bernoulli.

El trabajo se estructuró en sprints de una semana, gestionados a través de Trello (ver Anexo A-12), donde se priorizaron las tareas del backlog y se realizó seguimiento al progreso de cada iteración. Al finalizar cada sprint, se llevaron a cabo reuniones de revisión y retrospectiva para analizar los avances, corregir desviaciones y planificar los siguientes objetivos. Este enfoque permitió mantener un flujo de comunicación constante entre los actores involucrados y una validación progresiva del sistema en entornos reales.

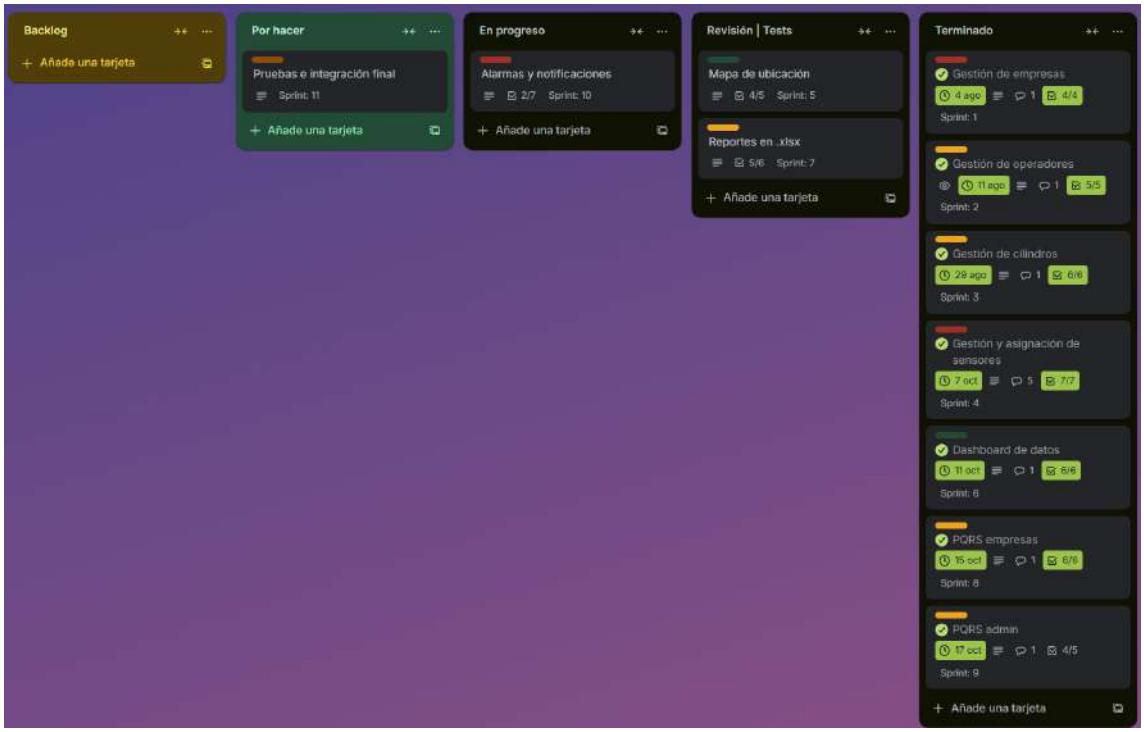


Ilustración 3 : Tablero de Trello. Fuente: Elaboración Propia

2.3 Tecnologías y herramientas

2.3.1 Tecnologías relevantes para el proyecto

El desarrollo de Andes Board se apoya en un conjunto de tecnologías elegidas por su estabilidad y facilidad de integración en entornos de desarrollo reales. En el **backend**, se utilizó **Python** junto con el framework **Django**, conocido por su arquitectura **MVT** (Model–View–Template) que ofrece un sistema seguro, escalable y con un potente **ORM** para garantizar la integridad de los datos. **Django** se eligió frente a otras alternativas como FastAPI o Flask por su madurez, su ecosistema de librerías y su capacidad de integrarse fácilmente con sistemas empresariales.

En el **frontend**, se trabajó con **JavaScript**, **Node.js, Apex Charts** y **TailwindCSS**, herramientas que permiten construir interfaces modernas, responsivas y ligeras. Además, se integró **Webpack** como empaquetador de recursos, lo que permite generar archivos optimizados para la web, facilitando el almacenamiento en caché y mejorando los tiempos de carga, incluso en redes de baja velocidad.

Para la **gestión de datos**, se implementó **PostgreSQL**, una base de datos relacional robusta y confiable, ideal para manejar transacciones complejas y cálculos asociados con las ecuaciones de Bernoulli. El componente de visualización geográfica se desarrolló con la librería **Leafmap**, que permite integrar mapas interactivos y visualizar las ubicaciones de los sensores en el dashboard de monitoreo.

En cuanto a la comunicación con el hardware, los sensores IoT transmiten los valores de presión y temperatura mediante el protocolo MQTT, utilizando autenticación OAuth para proteger el flujo de datos al servidor de Toprie. El sistema actual se conecta al servidor del fabricante TOPRIE (que usa Nginx), el cual no soporta múltiples conexiones concurrentes, lo que limita el número de solicitudes simultáneas. Sin embargo, esta configuración es suficiente para las pruebas del prototipo. Y andes board se conecta a ese servidor a través de una API

El procesamiento asíncrono se maneja con Celery, configurado para ejecutar múltiples tareas en paralelo, como la recepción de datos, la actualización periódica del sistema y la generación de respaldos automáticos. Estas tareas se coordinan mediante Redis, que actúa como broker de mensajes para evitar bloqueos del servidor principal y asegurar la estabilidad del sistema.

La infraestructura de despliegue está completamente dockerizada, con cinco contenedores principales: Django, PostgreSQL, Redis, Celery y Celery Beat, todos gestionados mediante Docker Compose. Este enfoque garantiza modularidad, escalabilidad y un entorno controlado para la ejecución del sistema. (ver Ilustración 3)

Finalmente, aunque durante las pruebas del proyecto se utilizó el sensor TP2401 (prestado por la empresa Andes Technologies), en la práctica el sistema se validó con el sensor Helm HM200SE, el cual demostró mejor rendimiento en la transmisión de datos y mayor precisión en las mediciones GPS, además de un costo aproximado de 50 USD menor al del TP2401.

2.3.2 Herramientas de desarrollo y gestión

El ciclo de desarrollo y despliegue de *Andes Board* se apoya en un conjunto de herramientas que permiten mantener la calidad del software y garantizar la entrega continua. Para el control de versiones se utiliza Git y GitHub (ver Anexo A-24). En la parte de frontend, **Webpack** se encarga de compilar los estilos generados por **Flowbite** y **TailwindCSS**, optimizándolos en un **archivo CDN** que permite a los usuarios acceder a la página de manera más rápida, incluso cuando los archivos estáticos no se cargan en su totalidad. En cuanto a la ejecución de procesos en segundo plano, Celery junto con Redis gestionan tareas asincrónicas sin bloquear el servidor. la gestión del proyecto bajo la metodología Scrum se apoyó en Trello, para lograr mayor trazabilidad y seguimiento de los sprints. El flujo de datos inicia desde una API del sensor, que extrae las mediciones y las

carga al sistema; posteriormente, la nube de Azure recibe esta información y, a través del orquestador de Docker, los contenedores de Celery Beat y Redis reaccionan automáticamente para procesar el archivo JSON y ejecutar los ecuaciones de Bernoulli requeridos. Finalmente, el contenedor de PostgreSQL almacena estos resultados de manera estructurada para su consulta posterior, mientras que el contenedor desplegado en Azure se encarga de exponerlos al dashboard web interactivo. Todo este ecosistema de herramientas, representado en el diagrama correspondiente, asegura un flujo de trabajo ágil, seguro y altamente escalable.

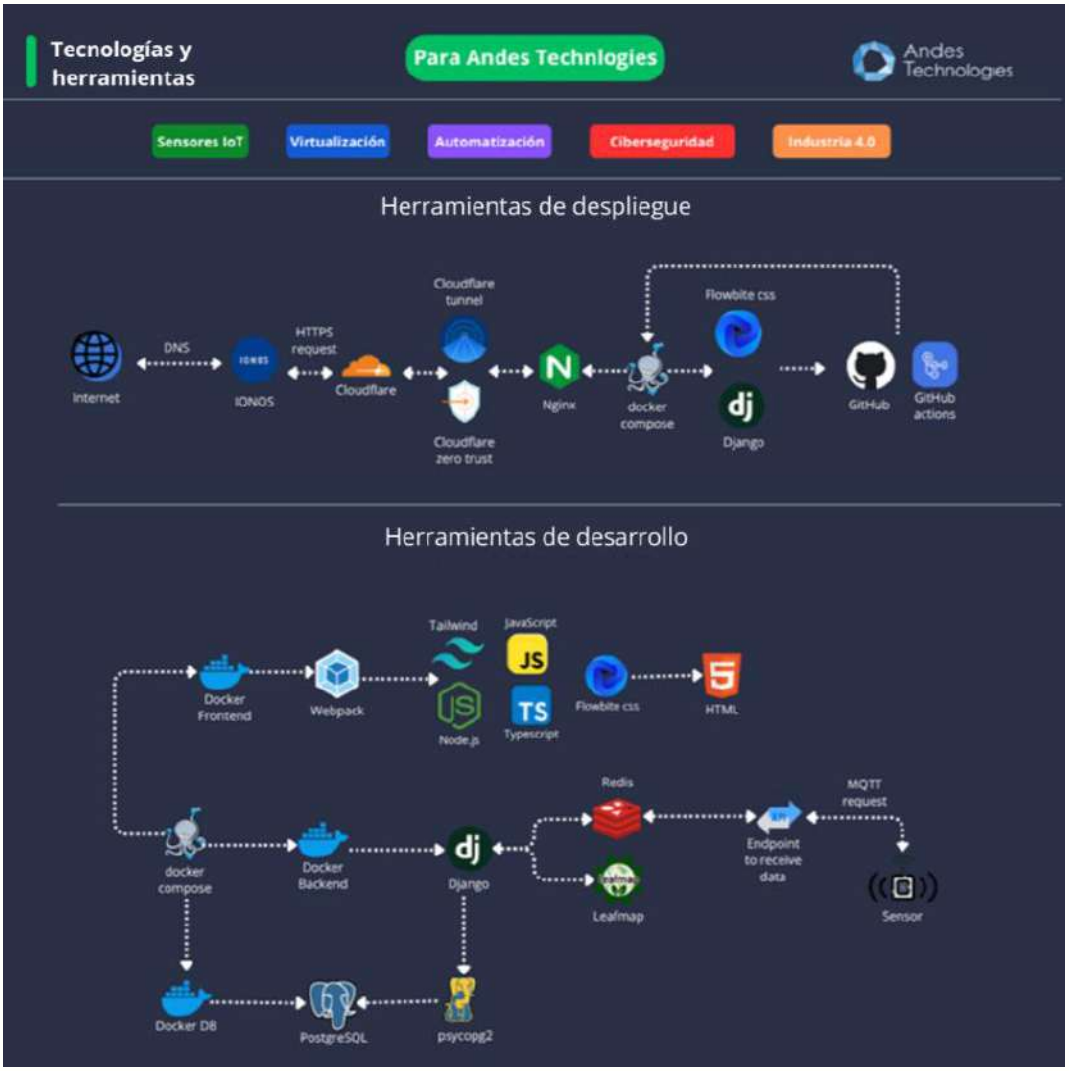


Ilustración 4: Diagrama de tecnologías

Capitulo III: Análisis del problema

3.1 Descripción del problema

3.1.1 Detalles del problema a resolver

La gestión de cilindros de GLP presenta múltiples dificultades técnicas y operativas que afectan tanto a usuarios como a distribuidores. En la industria actual, la medición del nivel de gas depende de métodos manuales o dispositivos mecánicos que no ofrecen precisión suficiente ni permiten un monitoreo continuo. Estas limitaciones limitan la capacidad de planificar recargas, anticipar fallas o detectar situaciones de riesgo.

Para analizar las causas raíz del problema, se elaboró un diagrama de Ishikawa, herramienta que permitió clasificar los factores técnicos, operativos y humanos que originan la problemática. La Ilustración 4 presenta el análisis detallado, del cual se identificaron las siguientes categorías principales:

Usuarios residenciales Los hogares que utilizan cilindros de GLP enfrentan una incertidumbre constante respecto al nivel de gas disponible, debido a que actualmente dependen de métodos empíricos como golpear el cilindro o estimar su peso. Esta falta de información puede provocar desabastecimiento inesperado o, por el contrario, operaciones en condiciones peligrosamente bajas. El proyecto plantea resolver esta limitación mediante un sistema de sensores acoplados a los cilindros, capaces de realizar cálculos de volumen a partir de presión, temperatura y otros factores medidos localmente, enviando esta información a una plataforma centralizada.

Optimización logística para distribuidores Para las empresas distribuidoras, el desconocimiento del nivel de gas almacenado en cada cilindro dificulta la planificación eficiente de las rutas de recolección y despacho. En muchos casos, se realizan visitas innecesarias o, por el contrario, se producen retrasos en la atención de clientes que ya agotaron su suministro. Con Andes Board, los datos serán recolectados

automáticamente y visualizados en dashboards alimentados por servicios de Azure, lo que permitirá optimizar rutas y sincronizar inventarios en función de la demanda real, reduciendo costos operativos y mejorando la atención al cliente.

Limitaciones tecnológicas de los sensores actual: Los sensores comúnmente usados en la industria, como caudalímetros Coriolis o térmicos, presentan dificultades técnicas para medir el gas en fase bifásica (líquido y vapor), lo que los vuelve poco confiables o demasiado costosos para aplicaciones domésticas. Andes Board opta por una estrategia más asequible y práctica: aprovechar sensores de presión y temperatura junto a modelos físicos para estimar el contenido del cilindro, disminuyendo costos y aumentando la replicabilidad del sistema.

Integración tecnológica basada en Azure en producción: Aunque la recolección inicial de datos puede realizarse mediante una API local, el objetivo a nivel productivo es operar completamente sobre servicios de Azure. Esto permite la centralización del sistema, la interoperabilidad con otras soluciones industriales y la activación de funciones clave como alarmas automatizadas, almacenamiento de históricos y escalabilidad para miles de dispositivos. El uso de Azure garantiza confiabilidad, disponibilidad y facilidad de implementación a gran escala.

Seguridad activa y notificaciones inteligentes: Un valor añadido del sistema es su capacidad de generar alertas automatizadas vía correo electrónico ante comportamientos anómalos, como caídas súbitas en el nivel de gas, patrones de consumo fuera de lo habitual o posibles fugas. Esta funcionalidad es fundamental para prevenir accidentes, proteger instalaciones y mejorar la trazabilidad del uso del GLP en el entorno doméstico.

En la Ilustración 4 se presenta un diagrama de Ishikawa, el cual identifica de manera estructurada las principales causas y consecuencias que motivan el desarrollo de esta propuesta tecnológica.

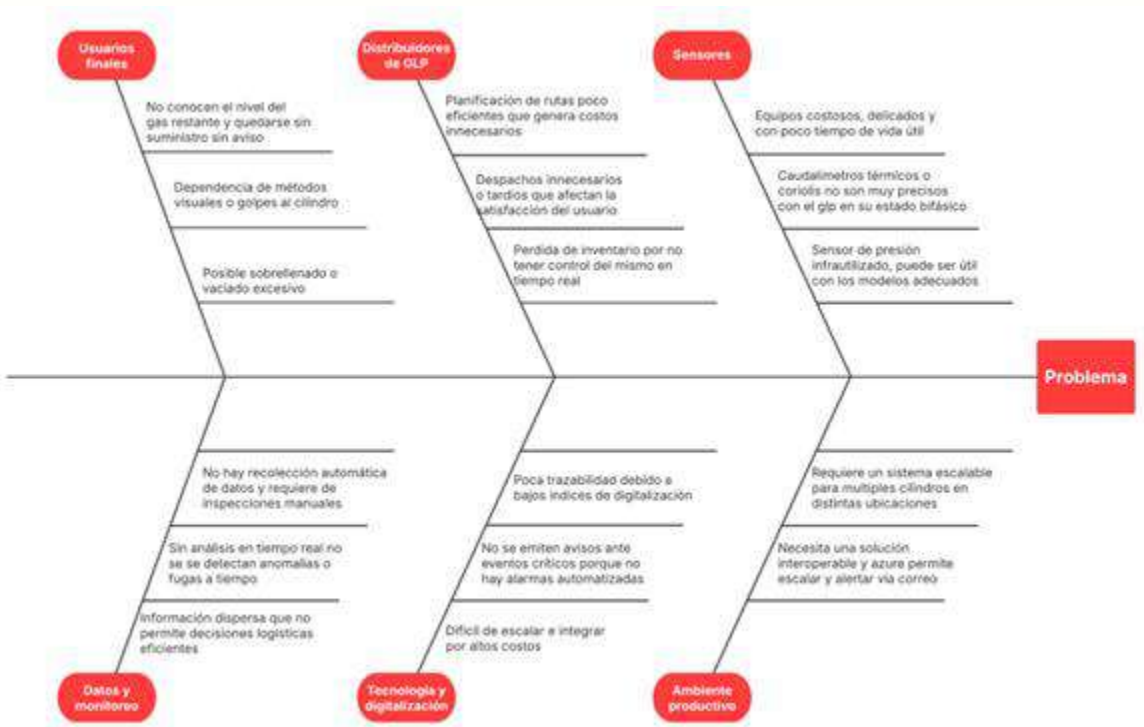


Ilustración 5: Diagrama de ishikawa. Fuente: Elaboración Propia

3.1.2 Análisis de requisitos

La industria del Gas Licuado de Petróleo (GLP), desde la gestión de inventarios de cilindros hasta la distribución minorista, enfrenta deficiencias persistentes en la medición precisa y el control operativo del recurso. Actualmente, la estimación del nivel de gas en los cilindros se realiza mediante métodos manuales o dispositivos mecánicos que carecen de continuidad, precisión y capacidad de adaptación a diferentes condiciones de uso. Estas limitaciones dificultan la planificación logística, la predicción de la demanda y la identificación temprana de **anomalías en el suministro**. Aunque existen soluciones tecnológicas en el mercado, los dispositivos de medición disponibles suelen ser costosos o no aptos para operar bajo las condiciones típicas del GLP. En este escenario, los sensores de presión representan una alternativa viable y equilibrada, siempre que su información sea complementada con un procesamiento adecuado de datos físicos y matemáticos. La revisión técnica evidencia además una escasez de soluciones que integren sensores accesibles, procesamiento avanzado de datos y despliegue en la nube, lo que plantea una oportunidad para desarrollar una solución IoT que combine eficiencia, precisión y accesibilidad.

A partir de este contexto, se identifican los actores principales y sus respectivas necesidades. En primer lugar, los **administradores** —personal de Andes Technologies o personal técnico autorizado— requieren **herramientas** para gestionar **datos maestros**, incluyendo la configuración de empresas, sensores y credenciales asociadas a los servidores de conexión. También deben supervisar los distintos **tipos de sensores** y atender las solicitudes relacionadas con **soporte técnico** (PQRS), garantizando la disponibilidad y estabilidad del sistema. En segundo lugar, las **empresas** distribuidoras o embotelladoras gestionan sus propios usuarios tipo operador, así como los sensores y cilindros asignados. Estas empresas necesitan acceso a la **información capturada por los sensores**, la posibilidad de **analizar el comportamiento del gas en cada cilindro** y mecanismos para tramitar **solicitudes de soporte**. Por último, los **operadores** o técnicos de campo son responsables de registrar las características físicas de los cilindros, realizar configuraciones iniciales y supervisar el comportamiento del gas con el fin de optimizar las operaciones

El sistema debe ofrecer una interfaz web adaptable a distintos dispositivos, con mecanismos de notificación ante comportamientos anómalos detectados en los sensores. En cuanto a las restricciones de negocio, el despliegue inicial del sistema se limita al territorio colombiano hasta obtener las certificaciones necesarias para operar en otros países. El modelo comercial adoptado es de tipo B2B, en el cual Andes Technologies provee los sensores a empresas distribuidoras que los instalan y administran directamente.

Entre los riesgos críticos que condicionan estos requisitos se encuentran la conectividad intermitente en ciertas zonas geográficas, las regulaciones sobre el uso de equipos de medición presurizados, la variabilidad en el desempeño entre diferentes modelos de sensores y los requerimientos de seguridad de los datos en tránsito y almacenamiento. Por eso, las prioridades del sistema se centran en la confiabilidad de las mediciones, la disponibilidad del servicio y la capacidad de escalar para integrar nuevos modelos, sensores y funcionalidades.

Finalmente, las diez historias de usuario **(Anexos A-11 a A-22)** levantadas durante la fase de análisis representan la base funcional del sistema. Estas abarcan la gestión de empresas y operadores, la administración y asignación de sensores, la georreferenciación, la visualización de datos procesados, la generación de reportes, el sistema de soporte técnico y la gestión de alarmas.

Diagrama de flujo de Andes Board

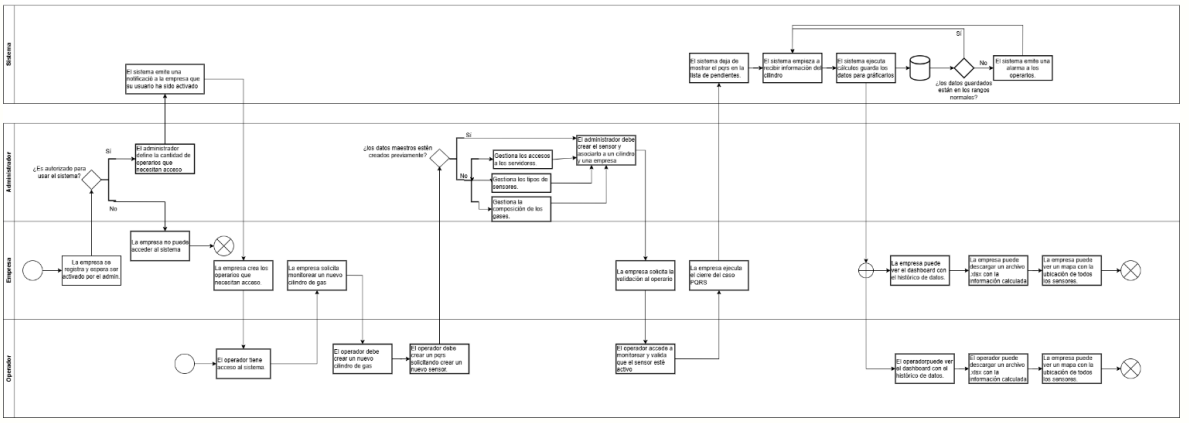


Ilustración 5: Diagrama BPMN de solución propuesta. Elaboración Propia

3.2 Análisis de requisitos

3.2.1 Requisitos funcionales

Requisito Funcional	Descripción	Criterio de aceptación	Prioridad
RF-01 — Gestión de empresas (CRUD por Administrador)	El sistema debe permitir al administrador crear, visualizar, actualizar y eliminar empresas desde el módulo “Empresas”, manteniendo la integridad de la base de datos.	Un administrador autenticado accede al menú “Empresas”, visualiza la lista completa y puede realizar operaciones de creación, actualización o eliminación. Los cambios se ven en la base de datos.	Media
RF-02 — Gestión de operadores (CRUD por Empresa)	Las empresas podrán gestionar sus operadores desde la opción “Operadores”, creando, editando o eliminando usuarios asociados a su identificador único de empresa.	Un usuario autenticado de tipo “empresa” accede al módulo “Operadores”, crea nuevos registros mediante formulario y puede modificarlos o eliminarlos. Los cambios se actualizan en la base de datos.	Media
RF-03 — Gestión de cilindros de gas	El sistema debe permitir al operador gestionar el inventario de cilindros domésticos de gas GLP, registrando sus parámetros físicos y asociándolos de manera unívoca a sensores IoT por el sistema de PQRS.	Un operador autenticado accede a “Cilindros”, registra peso vacío, peso máximo, diámetro del orificio y otras variables para determinar el flujo masico. Posteriormente, solicita la asociación del cilindro mediante un PQRS.	Alta
RF-04 — Gestión y asignación de sensores	El administrador podrá registrar, actualizar, eliminar y asignar sensores (TP2401 o compatibles) a cilindros o tanques.	Un administrador autenticado valida solicitudes de creación o asignación en “PQRS” y, desde el módulo “Sensores”, gestiona los sensores seleccionando su tipo y completando la configuración requerida.	Alta
RF-05 — Visualización de	El operador podrá visualizar la ubicación de	Al seleccionar un sensor y pulsar “Rastrear”, el	Alta

posición geográfica	un sensor activo mediante un mapa integrado en el dashboard.	dashboard muestra el mapa con marcador, timestamp y opción de volver al panel principal.	
RF-06 — Visualización de datos procesados en dashboard	El sistema debe presentar en el dashboard los valores calculados (porcentaje, densidad, caudal, presión, temperatura, timestamp) asociados a un único sensor y cilindro activo.	Al seleccionar un cilindro, el dashboard muestra solo los datos de ese sensor, actualizados y coherentes. No se muestran datos simultáneos de varios sensores.	Crítica
RF-07 — Generación de reportes (.xlsx)	El sistema debe permitir la descarga de reportes históricos en formato .xlsx con los datos procesados por el pipeline.	El operador selecciona un sensor y al hacer clic en “Reporte completo”, se genera un archivo .xlsx con las variables históricas asociadas.	Media
RF-08 — Gestión de PQRS (Empresas)	Las empresas podrán crear, visualizar y eliminar solicitudes de soporte (PQRS) clasificadas por tipo de petición.	Una empresa autenticada accede al módulo “Soporte”, crea una PQRS mediante formulario (tipo, asunto, nivel, estado, texto, adjuntos). No se muestran PQRS cerradas con más de 30 días.	Media
RF-09 — Seguimiento de PQRS (Administrador)	El administrador podrá visualizar todas las PQRS registradas y actualizar su estado (En proceso, Finalizada).	Al acceder al menú “Soporte”, el administrador puede consultar y actualizar las PQRS. Las solicitudes finalizadas y con más de 30 días no se muestran en la vista activa.	Media
RF-10 — Sistema de alarmas y notificación por correo	El sistema debe generar alarmas automáticas cuando un cilindro esté por debajo del 20% de capacidad o supere los límites de presión, notificando a las empresas por correo electrónico.	Cuando el pipeline detecta una condición de alarma, se registra el evento, se envía un correo a la empresa y la alarma aparece en la vista “Alarmas”. Puede marcarse como cerrada y archivarse.	Alta

Tabla 2: Requisitos funcionales de Andes Board: Elaboración propia

3.2.2 Requisitos no funcionales

Requisito No Funcional	Descripción	Criterio de aceptación	Prioridad
RNF-01 — Control de acceso (autenticación y autorización)	Solo usuarios autorizados podrán acceder a funciones según su rol (administrador, empresa, operador).	Un usuario sin rol administrador no puede acceder a “Empresas”. Un administrador autenticado puede realizar operaciones.	Crítica
RNF-02 — Aislamiento por empresa (multi-tenancy)	Cada empresa solo podrá acceder a sus propios datos (operadores, sensores, PQRS).	Los usuarios de una empresa no pueden visualizar ni editar información de otra.	Alta
RNF-03 — Conectividad segura del sensor	Las conexiones con sensores deben realizarse mediante API sobre TLS con reintentos automáticos ante fallos.	Los reintentos se registran en logs. La actualización del estado del sensor ocurre dentro del intervalo de muestreo	Crítica
RNF-04 — Geolocalización precisa y privada	Las coordenadas mostradas deben ser accesibles para los usuarios logueados.	El mapa muestra la última coordenada con acceso restringido por autenticación.	Alta
RNF-05 — Frescura de datos y tiempo de respuesta	Los datos mostrados deben actualizarse en cada ciclo de muestreo sin retrasos perceptibles.	El dashboard carga los datos más recientes en ≤ 2 segundos y con frescura ≤ 10 minutos.	Alta
RNF-06 — Exportación de reportes eficiente	Los reportes .xlsx deben ser compatibles con Excel y generarse en menos de 5 segundos.	El archivo se abre sin errores y se genera en ≤ 5 s con ≤ 10.000 registros.	Media
RNF-07 — Fiabilidad de notificaciones	El sistema debe garantizar la entrega de correos mediante reintentos automáticos y registro de eventos.	En caso de fallo SMTP, el sistema reintenta y registra el evento; las alarmas se notifican dentro de los 10 minutos posteriores.	Alta
RNF-08 — Admisión de archivos adjuntos (PQRS)	El módulo de PQRS debe aceptar archivos binarios con extensiones .docx, .pdf, .png, .jpg, .jpeg, .xlsx.	El sistema valida el formato del archivo antes de subirlo y permite solo extensiones permitidas.	Media
RNF-09 —	El sistema debe evaluar la	Celery ejecuta tareas	Crítica

Evaluación constante de conexión y multitareas	conexión de sensores en segundo plano mediante multihilos gestionados con Celery y Redis.	periódicas de verificación y cálculo sin bloquear el servidor.	

Tabla 3 Requisitos no funcionales de Andes Board: Elaboración propia

3.3 Estudio de viabilidad

3.3.1 Viabilidad técnica

La viabilidad técnica del proyecto Andes Board se considera alta, ya que tanto los componentes de hardware como los de software se apoyan en tecnologías comprobada. En cuanto al hardware, existen sensores de presión y temperatura comerciales capaces de medir con precisión las variables necesarias para aplicar las ecuaciones de Bernoulli. Durante la implementación se utilizó el Helm Sensor HM200SE. Para las pruebas de validación, se empleó el TP2401, facilitado por la empresa Andes Technologies, aprovechando su certificación industrial y robustez operativa. Esta combinación permitió realizar un desarrollo técnicamente sólido y económicamente escalable para futuras fases del proyecto. (ver anexo A-24)

En el área del software e infraestructura, el sistema se construyó sobre tecnologías maduras y ampliamente utilizadas en entornos de desarrollo reales. La arquitectura incorpora una API de ingestión de datos proveniente de los sensores, actualmente alojada en un servidor remoto del fabricante, aunque se planea implementar un servidor propio con soporte de concurrencia que centralice las comunicaciones de múltiples dispositivos. Todo el sistema se ejecuta dentro de contenedores Docker, orquestados mediante Docker Compose, lo que permite reproducir y desplegar fácilmente el entorno en diferentes equipos o servidores. La base de datos PostgreSQL asegura integridad relacional y consultas eficientes, mientras que las tareas asíncronas como el procesamiento de archivos JSON enviados por los sensores, se gestionan con Celery y Redis, lo que habilita ejecución concurrente sin sobrecargar el servidor principal.

Desde el punto de vista arquitectónico, Andes Board adopta una estructura

monolítica modular basada en el patrón MVT (Model–View–Template) de Django, junto con principios de diseño SOLID y el patrón Strategy, que facilitan la incorporación de nuevos modelos de sensores o ecuaciones sin alterar las capas principales. Esta organización reduce el acoplamiento entre módulos y favorece el mantenimiento del código a largo plazo. Además, la arquitectura está pensada para escalar horizontalmente las cargas de trabajo críticas sin necesidad de modificar la capa de usuarios ni comprometer la consistencia del sistema.

En términos de seguridad y operación, la plataforma aprovecha las protecciones integradas de Django, como la autenticación de usuarios, el control de roles y la defensa ante ataques web comunes (CSRF, XSS, inyecciones SQL, entre otros). Para la comunicación con los sensores se utiliza el protocolo MQTT, reforzado con OAuth 2.0 para la autenticación de la API.

3.3.2 Viabilidad económica

El proyecto Andes Board propone un modelo de negocio B2B (Business to Business), en el que Andes Technologies suministra los dispositivos de monitoreo de GLP a empresas distribuidoras, responsables de su instalación, operación y mantenimiento en campo. En esta fase inicial, el proyecto se implementará únicamente en Colombia, ya que la empresa aún no dispone de los permisos necesarios para la comercialización internacional de equipos IoT. Esta delimitación geográfica permite concentrar los esfuerzos de desarrollo y validación bajo el marco normativo local, optimizando costos y garantizando la compatibilidad técnica y legal del sistema.

Los recursos del proyecto se agrupan en tres categorías principales: materiales, equipos e implementos, y servicios operativos. En la primera, se incluye el sensor IoT de presión y temperatura, componente esencial para la captura de datos. Inicialmente se utilizó el TP2401, por su robustez y certificación industrial, aunque el análisis de costos identificó una alternativa más asequible: el Helm Sensor HM200E, con un valor aproximado de USD 130 (incluyendo envío y aduana), manteniendo prestaciones técnicas equivalentes y

reduciendo el costo total del prototipo.

En cuanto a equipos e infraestructura, se considera el uso de una computadora de desarrollo y las licencias de software necesarias para administrar el entorno productivo, con una arquitectura desplegada en contenedores Docker que integran los servicios de Django, PostgreSQL, Redis y Celery. Este entorno garantiza la disponibilidad del sistema, la escalabilidad y la continuidad operativa.

Dentro de la categoría de servicios, se incluye la conexión a internet; indispensable para la comunicación entre el sensor, la API y el servidor; así como los honorarios del personal encargado del diseño, desarrollo y pruebas del sistema. En conjunto, estos recursos aseguran la funcionalidad completa del proyecto, desde la adquisición de datos hasta su visualización en el dashboard web.

Finalmente, el presupuesto total estimado asciende a $5 721 400 COP, distribuidos entre materiales, infraestructura tecnológica y costos operativos. Esta planificación financiera es realista y sostenible, ya que prioriza los elementos clave para garantizar la operatividad del sistema y demuestra la viabilidad económica del proyecto en función de los beneficios esperados: monitoreo automatizado, escalabilidad y reducción de costos para las empresas del sector GLP.

Rubros	Recurso	Cantidad /Unidad	Valor unitario	Valor total	Aporte M1	Aporte M2	Aporte M3	Aporte M4
Materiales	Sensor IoT	1	1’420.000	1’420.000	1’420.000	0	0	0
	Subtotales	1	**$1’420.000**	**$1’420.000**	**$1’420.000**	**$0**	**$0**	**$0**
Equipos e implementos	Computadoras	1	3’300.000	3’300.000	3’300.000	0	0	0
	Licencias de software	3	333.600	1’000.000	0	333.600	333.600	333.600
	Subtotales	4	**$3’633.600**	**$4’320.000**	**$4’720.000**	**$333.600**	**$333.600**	**$333.600**
VALOR TOTAL APORTADO M1								4’720.600
VALOR TOTAL APORTADO M2								333.600
VALOR TOTAL APORTADO M3								333.600
VALOR TOTAL APORTADO M4								333.600
COSTO TOTAL DEL PROYECTO								**$5’721.400**

Tabla 4: Viabilidad económica, elaboración propia

Capitulo IV: Diseño de la solución

4.1 Arquitectura de software

4.1.1 Descripción general de la arquitectura propuesta

La arquitectura propuesta para el sistema Andes Board se fundamenta en la integración de tecnologías IoT, procesamiento asíncrono y servicios en la nube, con el propósito de ofrecer un monitoreo remoto confiable del nivel de gas GLP en cilindros domésticos. El sistema parte de un sensor de presión y temperatura instalado en la válvula del cilindro, encargado de capturar las condiciones físicas del gas en tiempo real.

Los datos generados por el sensor se transmiten hacia una API remota proporcionada por el fabricante, la cual actúa como punto de entrada para el sistema. Desde allí, la aplicación web desarrollada en Django consulta periódicamente la información mediante peticiones HTTP y la procesa en un entorno de contenedores Docker desplegado sobre Azure Container Apps.

En este entorno, un conjunto de servicios coordinados maneja el flujo completo de datos:

- Celery gestiona las tareas asíncronas de procesamiento, mientras que Redis funciona como intermediario (message broker) y sistema de colas.
- Los cálculos se ejecutan en un pipeline de procesamiento que aplica modelos matemáticos y termodinámicos simplificados para estimar indicadores como masa removida, masa restante y flujo másico, considerando el régimen de flujo crítico o subcrítico según la relación de presiones en el cilindro.
- Los resultados obtenidos se almacenan de manera estructurada en una base de datos PostgreSQL, alojada en el mismo entorno de contenedores o en un servicio administrado de Azure.

Adicionalmente, se incorpora un sistema de notificaciones por correo electrónico basado en el protocolo SMTP, el cual alerta automáticamente a los usuarios ante condiciones anómalas, como disminuciones bruscas de presión o niveles críticos de gas.

Finalmente, los datos procesados se visualizan en un dashboard web interactivo desarrollado con Django y Tailwind CSS, que permite el seguimiento histórico y en tiempo real de cada cilindro, facilitando la toma de decisiones operativas y la planificación de recargas.

Esta arquitectura monolítica modular, apoyada en principios SOLID y en el patrón de diseño Strategy para la gestión de sensores y cálculos, garantiza flexibilidad, mantenibilidad y escalabilidad, permitiendo incorporar nuevos modelos de sensores o mejorar los algoritmos sin alterar la estructura base del sistema.

4.1.2 Diagrama de arquitectura

El diagrama de arquitectura de Andes Board representa la integración entre los sensores físicos de presión y la infraestructura de software desplegada en Azure para el monitoreo remoto del nivel de gas GLP. Los sensores instalados en los cilindros capturan en tiempo real variables de presión y temperatura, las cuales son transmitidas mediante el protocolo MQTT hacia una API REST remota provista por el fabricante.

Posteriormente, la aplicación desarrollada en Django consulta dicha API y procesa los datos dentro de un entorno de contenedores Docker alojado en Azure Container Apps. En este entorno, los servicios se organizan de manera modular:

- Redis actúa como intermediario de mensajes y sistema de colas.
- Celery gestiona las tareas asíncronas de procesamiento.
- PostgreSQL almacena los resultados estructurados de las mediciones.
- Módulos internos generan reportes, alertas y visualizaciones.

Los usuarios acceden al sistema a través de un dashboard web interactivo, disponible tanto para clientes móviles como de escritorio mediante conexiones seguras HTTPS (443).

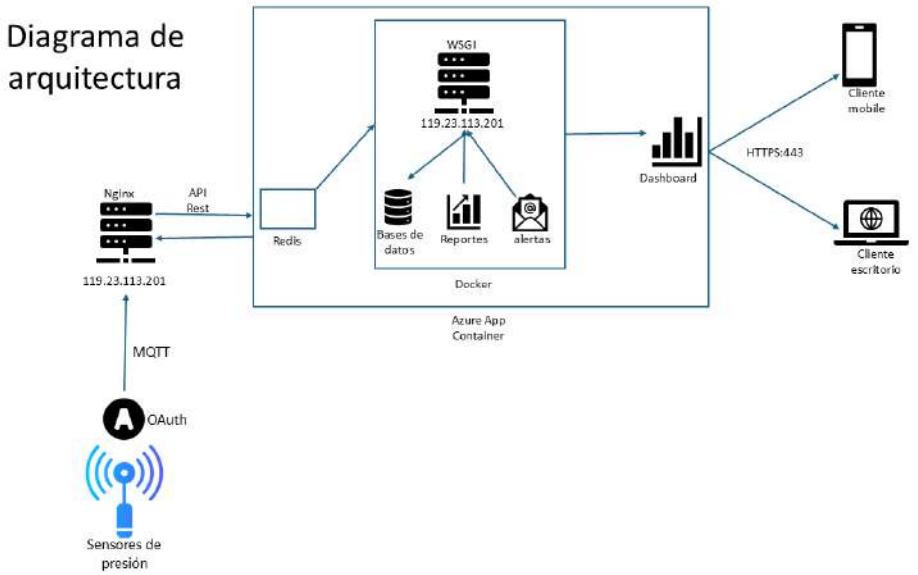


Ilustración 6 Diagrama de arquitectura. Elaboración Propia

4.1.3 Justificación de la elección arquitectónica

La arquitectura adoptada en Andes Board responde a la necesidad de equilibrar simplicidad, escalabilidad y mantenibilidad en el desarrollo de un sistema IoT de monitoreo de gas GLP. Si bien existen enfoques distribuidos como las arquitecturas basadas en microservicios o serverless, su implementación implica una complejidad operativa elevada y un mayor costo de despliegue y mantenimiento, especialmente en las fases iniciales del proyecto.

Por esta razón, se optó por una arquitectura monolítica modular, desarrollada bajo el patrón Model–View–Template (MVT) de Django, complementada con contenedores Docker y principios SOLID. Este enfoque permite mantener una estructura centralizada y coherente, donde la lógica de negocio, el almacenamiento y la presentación coexisten de manera integrada, pero conservan independencia interna a través de módulos especializados (gestión de sensores, ecuaciones de Bernoulli, dashboard y sistema de alertas).

La decisión de no adoptar una arquitectura de microservicios se basa en que, en este tipo de sistemas, cada sensor o fabricante requeriría su propio servicio independiente, lo cual generaría sobrecarga de comunicación, duplicidad de código y costos innecesarios. En cambio, el uso del patrón Strategy dentro de una arquitectura monolítica permite extender el sistema a nuevos tipos de sensores o modelos de cálculo sin modificar la estructura existente, cumpliendo con el principio O de los SOLID (abierto para extensión, cerrado para modificación).

Además, la integración con Azure Container Apps proporciona una capa de escalabilidad horizontal y aislamiento controlado de procesos (como Celery, Redis y PostgreSQL), sin perder la simplicidad de un despliegue centralizado. Esto permite que el sistema crezca progresivamente sin requerir una orquestación compleja ni un rediseño estructural.

4.2 Diseño de componentes

4.2.1 Diseño de módulos y componentes

El diagrama de componentes del sistema Andes Board representa la estructura modular del monolito desarrollado en Django, compuesto principalmente por dos aplicaciones; La aplicación Accounts gestiona los procesos de autenticación, registro y administración de usuarios, integrando formularios `forms`, modelos de datos `models` y vistas personalizadas `views`, las cuales se conectan mediante rutas `urls` que definen la lógica de navegación y acceso. Por su parte, la aplicación App concentra la mayor parte de la funcionalidad del sistema, incluyendo los módulos de gestión de sensores, ecuaciones de Bernoulli, tablero de control `dashboard`, módulo PQRS, y administración de credenciales. En ella se articulan distintos componentes internos:

- `views`, que exponen la lógica de presentación y control como `DashboardView,` etc
- `models`, que definen las entidades y relaciones persistentes;
- `services`, `repositories` y `strategies`, encargados del procesamiento de datos, la aplicación de los principios SOLID y la implementación del patrón Strategy para la gestión dinámica de sensores, cálculos matemáticos;
- `tasks`, donde se orquestan las tareas asíncronas a través de `Celery`;
- `adapters`, permiten la comunicación con la API externa TOPRIE, responsable de proveer los datos de los sensores IoT.

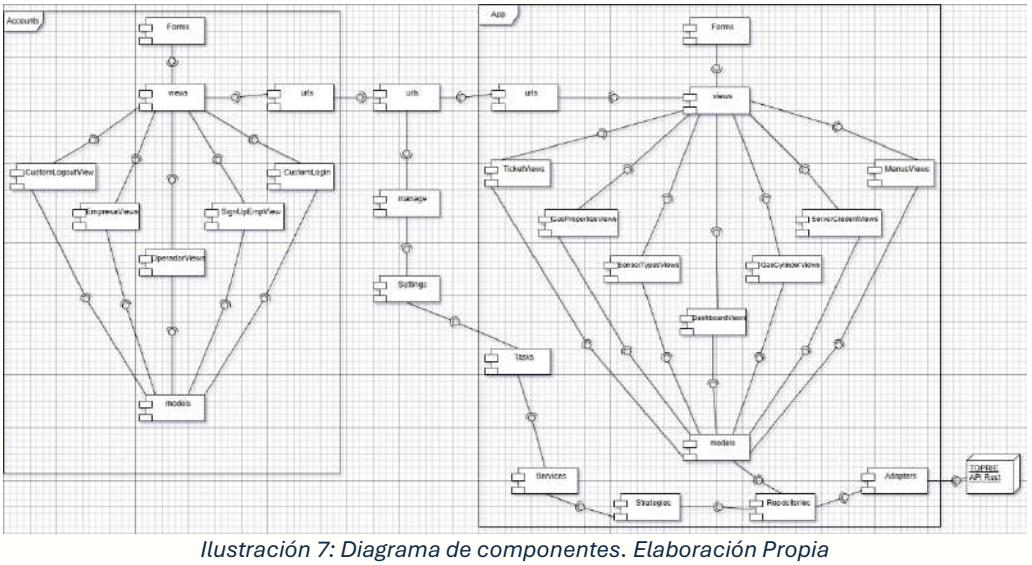


Ilustración 7: Diagrama de componentes. Elaboración Propia

4.2.2 Diagrama de clases y secuencia

El diagrama de clases de la aplicación Accounts ilustra la estructura y relaciones internas de los componentes encargados de la gestión de usuarios, autenticación y control de acceso en el sistema Andes Board. En la capa de modelos, la clase central Usuario, derivada de AbstractUser, extiende la funcionalidad base del framework Django para incluir atributos personalizados como email, tipo_documento, numDocumento, rol y estado, estableciendo relaciones directas con las clases TipoDocumento y Rol, lo que permite una gestión flexible y relacional de la información del usuario.

Asimismo, las clases Operador y Empresa complementan la estructura de usuarios finales, permitiendo diferenciar entre los roles operativos y empresariales dentro del sistema. Estas entidades están vinculadas mediante relaciones de asociación con Usuario, lo que garantiza la integridad de los datos en los procesos de registro y autenticación.

En la capa de vistas, las clases como CustomLoginView, CustomLogoutView y SignUpEmpView implementan los procesos de inicio de sesión, cierre de sesión y registro de nuevos usuarios empresariales, respectivamente, mientras que las vistas OperadorCreateView, OperadorUpdateView, OperadorDeleteView y EmpresaUpdateView gestionan las operaciones CRUD de las entidades correspondientes mediante formularios definidos en forms.py y administrados por el ORM de Django.

Cada vista interactúa con los modelos a través de métodos genéricos de la clase base View y view.generic, reflejando una clara aplicación del patrón MVT (Model–View–Template) y una separación de responsabilidades entre la capa lógica y la de presentación.

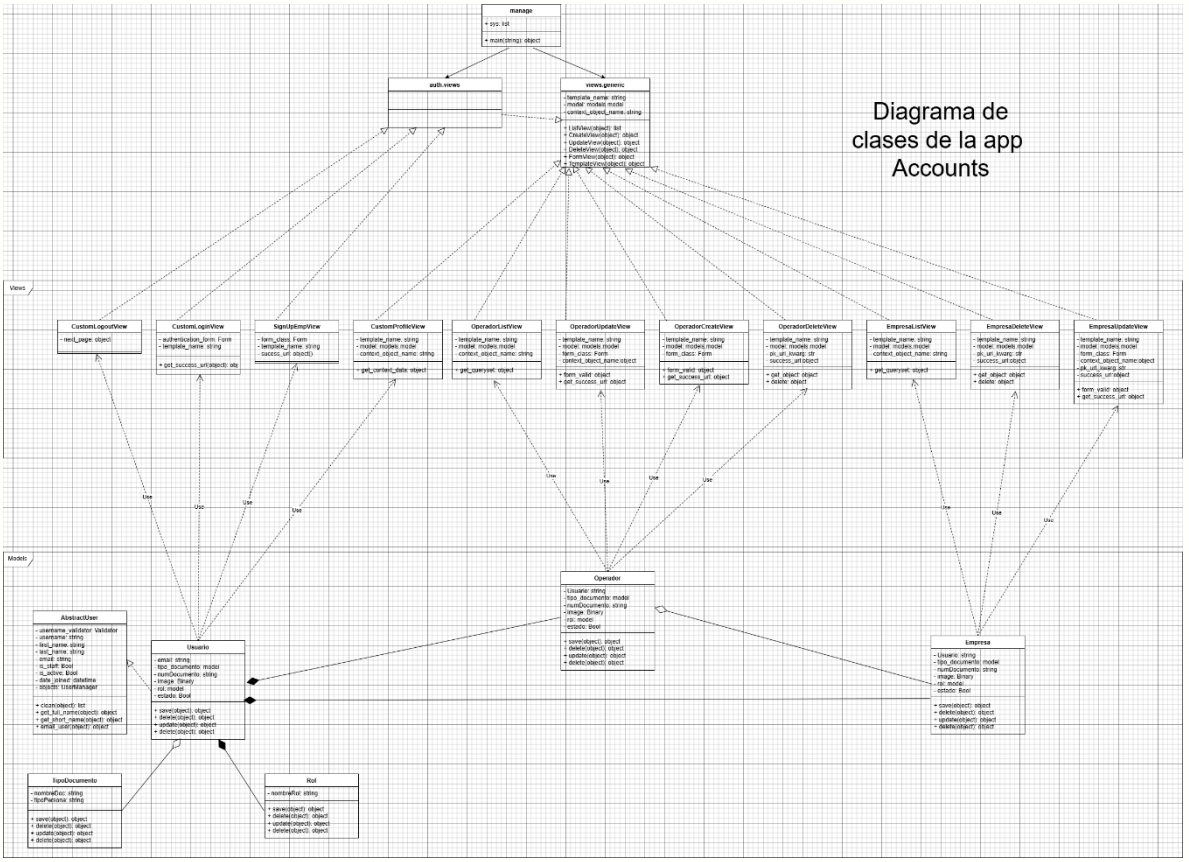


Ilustración 8: Diagrama de clases de la app Acounts. Elaboración Propia

El diagrama de clases de la aplicación App muestra la organización interna de los módulos que implementan el patrón MVT, integrando las vistas, modelos y controladores lógicos del sistema. En la capa de vistas, se gestionan los modelos del dominio; como TipoSensor, CaracteristicasCilindro, CredencialesServidor, TicketSoporte, Mensaje y GasRestante; cuyo flujo de información es administrado mediante el ORM de Django.

En la capa de modelos, destacan las clases Sensor, ResultsGasRestante, ComposicionGas, Alarma y Notificaciones, que bajo un patrón complementario almacenan los datos provenientes de los sensores IoT y los resultados procesados por las tareas asíncronas ejecutadas con Celery, fortaleciendo la escalabilidad del sistema y reduciendo su deuda técnica. Estas vistas se enlazan con sus respectivas plantillas para la renderización dinámica del dashboard, garantizando coherencia entre la capa de datos y la presentación.

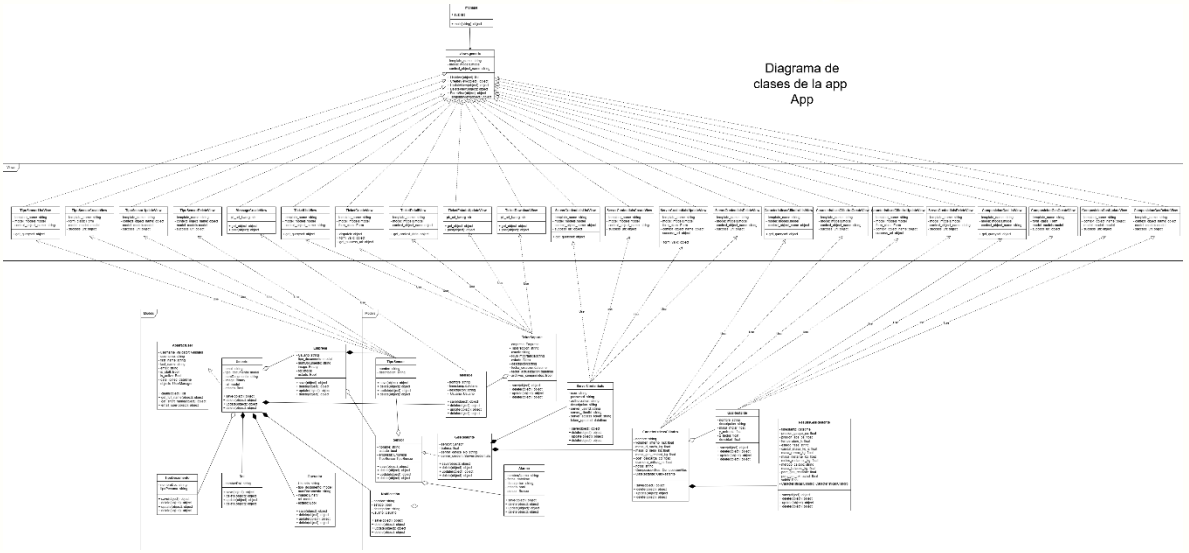


Ilustración 9: Diagrama de clases de la app App. Elaboración Propia

Por otra parte, el diagrama de secuencia representa el flujo de creación de un sensor de presión en el sistema Andes Board, donde intervienen múltiples capas que garantizan bajo acoplamiento y la aplicación del patrón Strategy conforme a los principios SOLID.

El proceso inicia cuando el administrador accede a la vista SensorCreateView, la cual, a través del servicio create_sensor(), delega la operación a SensorStrategy, que determina dinámicamente el tipo de sensor a registrar. Posteriormente, el repositorio ejecuta la persistencia de datos mediante el método create_from_form(), y el adaptador comunica estas acciones con los modelos Sensor y GasRestante, encargados de crear o actualizar los registros correspondientes.

Este flujo demuestra una arquitectura modular en la que las vistas coordinan la interacción del usuario, los servicios encapsulan la lógica de negocio y las estrategias permiten extender el sistema sin modificar su estructura base, asegurando mantenibilidad y escalabilidad en la gestión de sensores IoT.

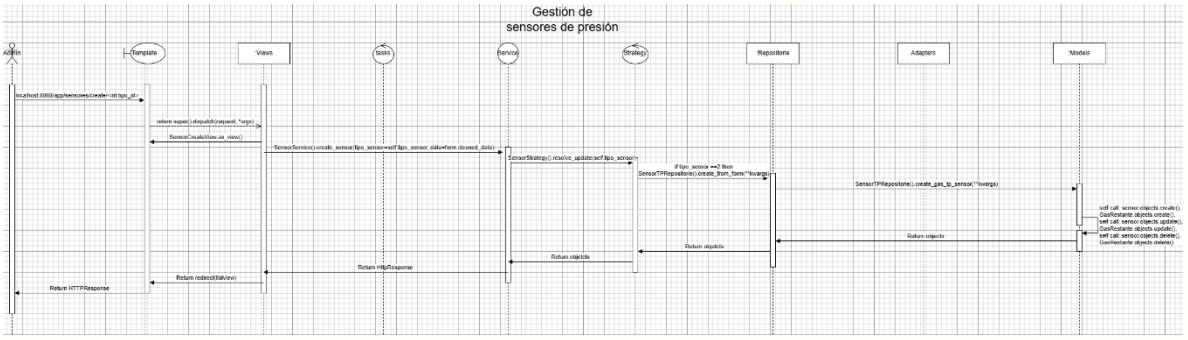


Ilustración 10: Diagrama de secuencias de la creación y gestión de sensores. Elaboración Propia

Finalmente, el diagrama de secuencia de la validación de token, estado y cálculos matemáticos representa el flujo de operaciones asociado a la gestión dinámica de sensores de presión en el sistema Andes Board, donde intervienen múltiples capas para garantizar una ejecución modular y desacoplada.

El proceso inicia cuando el sistema invoca los servicios encargados de actualizar el token de acceso y sincronizar el estado del sensor, asegurando la comunicación continua con la API remota del fabricante. A través del módulo Service, las peticiones se procesan mediante el patrón Strategy, que selecciona el comportamiento adecuado según el tipo de sensor y sus credenciales activas.

Posteriormente, el repositorio y el adaptador gestionan la persistencia y actualización de los registros Sensor y GasRestante, reflejando tanto la información operativa como los resultados de los ecuaciones de Bernoulli ejecutados por las tareas asíncronas de Celery. Estos cálculos determinan parámetros como el flujo másico, la masa removida y la masa restante, manteniendo el historial de mediciones.

El flujo finaliza con la respuesta HTTP hacia la vista correspondiente, completando el ciclo de actualización y asegurando la coherencia entre la información del sensor físico, la base de datos y el dashboard. Este proceso evidencia una arquitectura basada en estrategias, que mantiene el sistema extensible y permite la evolución de los algoritmos y tipos de sensores sin modificar la estructura principal del código.

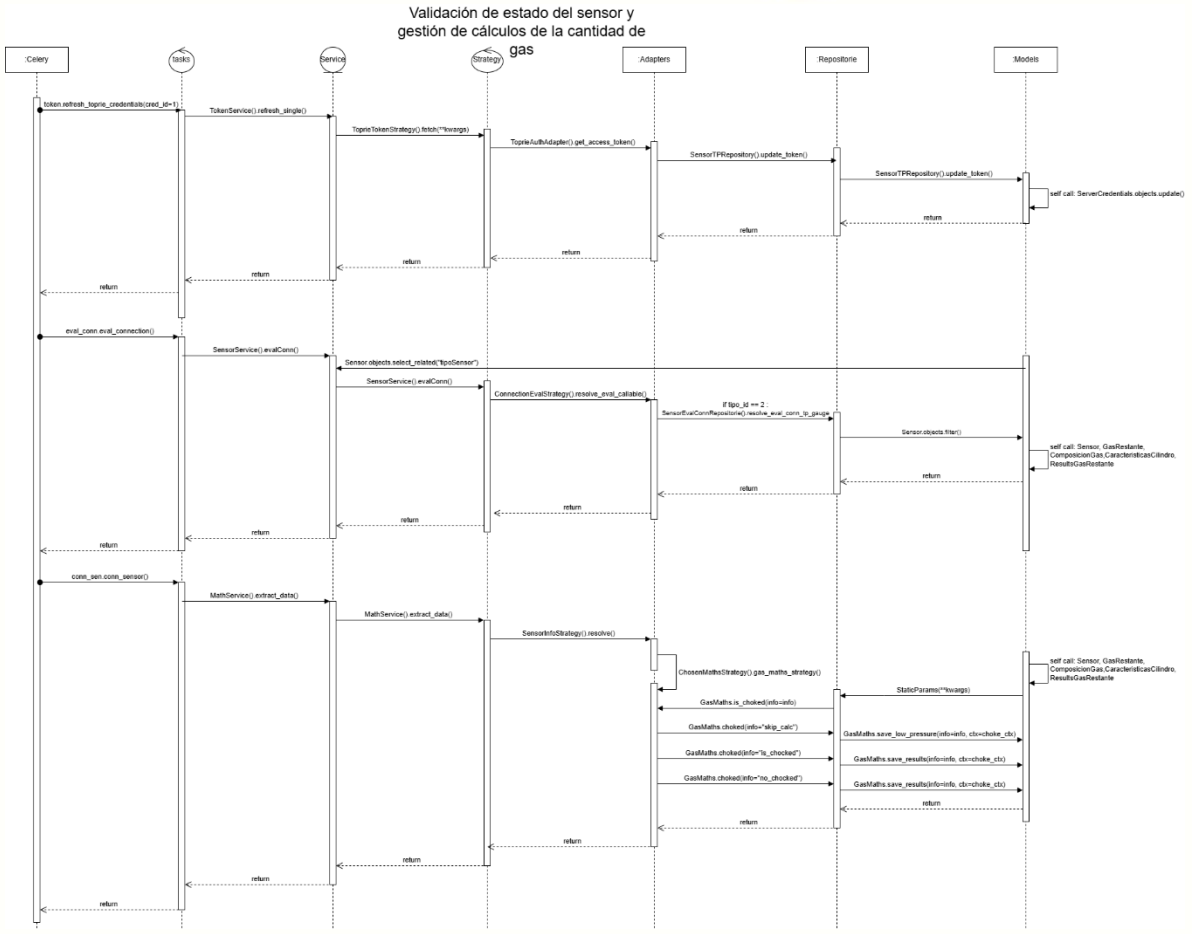


Ilustración 11: Diagrama de secuencias de validación y cálculos matemáticos. Elaboración Propia

4.3 Interfaces y comunicación

4.3.1 Diseño de interfaces de usuario

El diseño de interfaces en Andes Board se fundamenta en tres principios esenciales: consistencia visual, reutilización de componentes y simplicidad funcional. Su objetivo es ofrecer una experiencia de usuario coherente y accesible para los tres tipos de actores del sistema; administrador, empresa y operador; garantizando una interacción fluida con las funciones de monitoreo, gestión y análisis de los sensores de gas GLP.

Las interfaces se desarrollaron utilizando Django Templates junto con TailwindCSS

y Flowbite, frameworks que permitieron definir un conjunto unificado de estilos y componentes reutilizables en todo el sistema. Esta decisión asegura uniformidad visual, reducción de código duplicado y una mejor mantenibilidad del front-end. Además, se integraron principios de diseño responsivo, permitiendo una visualización óptima tanto en equipos de escritorio como en dispositivos móviles, sin afectar la legibilidad de los datos ni la funcionalidad de los menús.

Menús y estructura de navegación

El menú principal constituye el eje central de la navegación del sistema. Fue diseñado como un componente único reutilizable que se adapta dinámicamente al tipo de usuario autenticado mediante validaciones por rol, controladas desde el backend de Django.

De esta forma, los tres perfiles (administrador, empresa y operador) comparten la misma estructura base de menú, pero visualizan únicamente las opciones habilitadas para su rol, evitando duplicar plantillas o rutas.

Los elementos del menú se distribuyen en secciones que agrupan las funciones clave del sistema:

- Empresas: visualización y gestión de entidades registradas.
- Sensores: creación, actualización y seguimiento de dispositivos IoT.
- Dashboard: monitoreo gráfico del comportamiento del gas y estado de los sensores.
- PQRS: gestión de solicitudes y mensajes del usuario.
- Reportes y alertas: visualización de métricas históricas y notificaciones automáticas.

El diseño mantiene una consistencia visual y cromática, empleando íconos monocromáticos adaptables al modo claro y oscuro.

Los menús laterales y encabezados siguen una misma jerarquía visual, lo que mejora la orientación del usuario y facilita la navegación entre módulos.

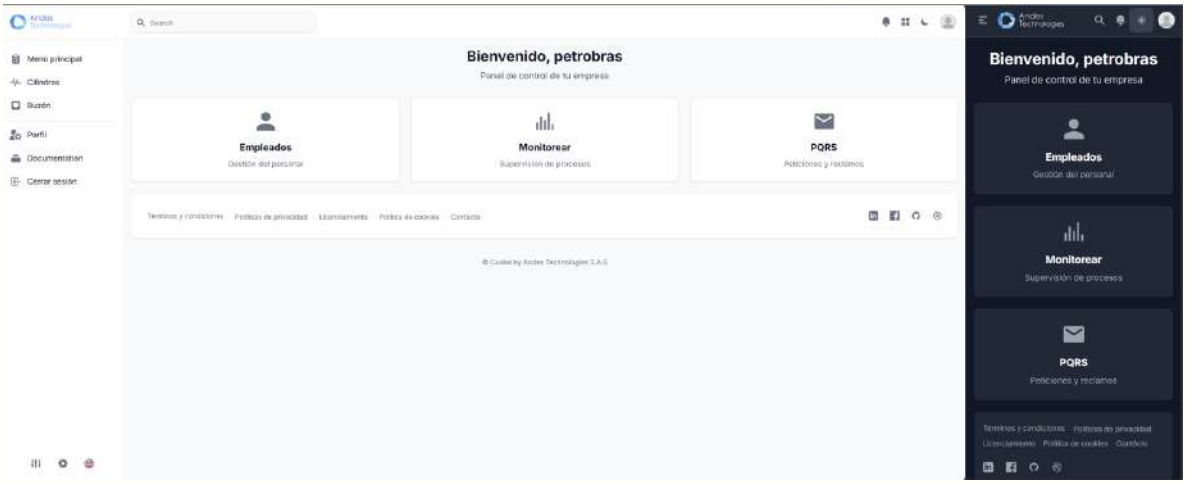


Ilustración 12: Interfaz de menú principal de empresa

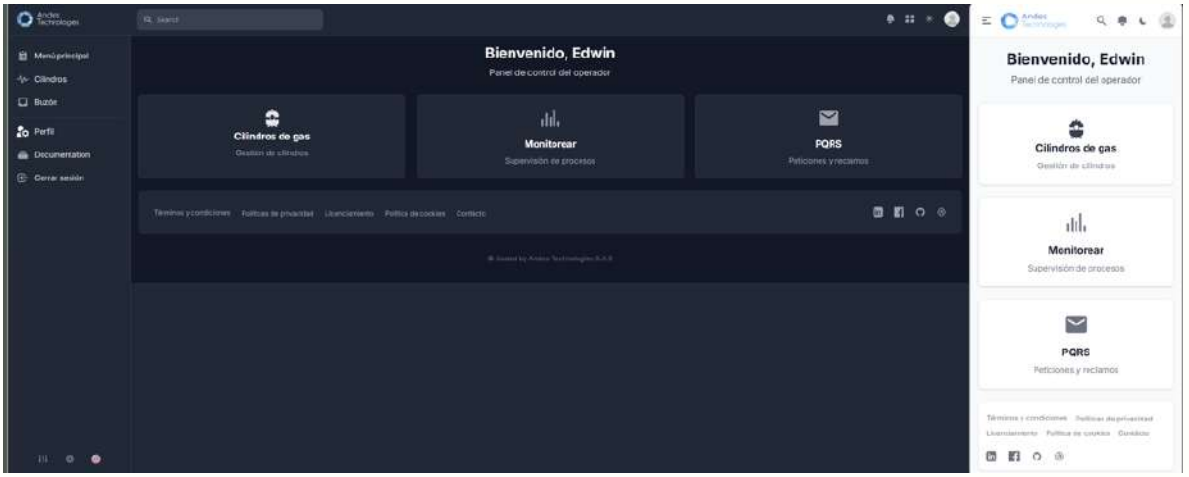


Ilustración 13: Interfaz de menú principal de operador

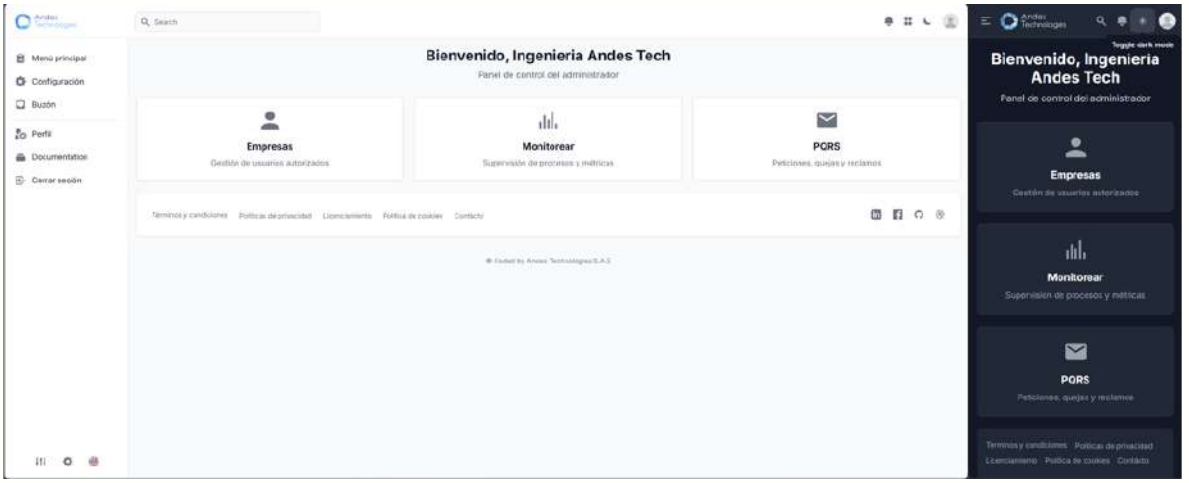


Ilustración 14: Interfaz de menú principal de administrador

Formularios y validaciones

Los formularios representan el punto de interacción directa entre el usuario y el sistema. En Andes Board, estos formularios se implementaron utilizando las clases ModelForm de Django, permitiendo vincular los modelos de base de datos con el front-end de forma segura y validada. Las vistas más relevantes son el registro de cilindro, el inicio de sesión, y los formularios de gestión de cilindros y sensores que cambian su estructura según el tipo de sensor que seleccionó.

Cada formulario incluye:

- Validaciones en tiempo real (HTML5 y backend Django).
- Mensajes de error o confirmación visuales en los campos.
- Componentes reutilizables con estilos uniformes definidos en Tailwind.

El formulario de login, por su parte, permite el acceso a las vistas administrativas según los privilegios del usuario, redirigiendo al panel correspondiente.

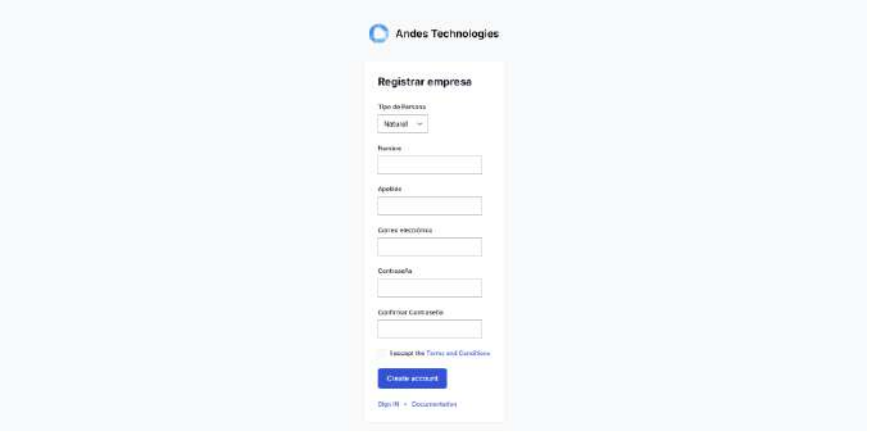


Ilustración 15: Interfaz de creación de empresas

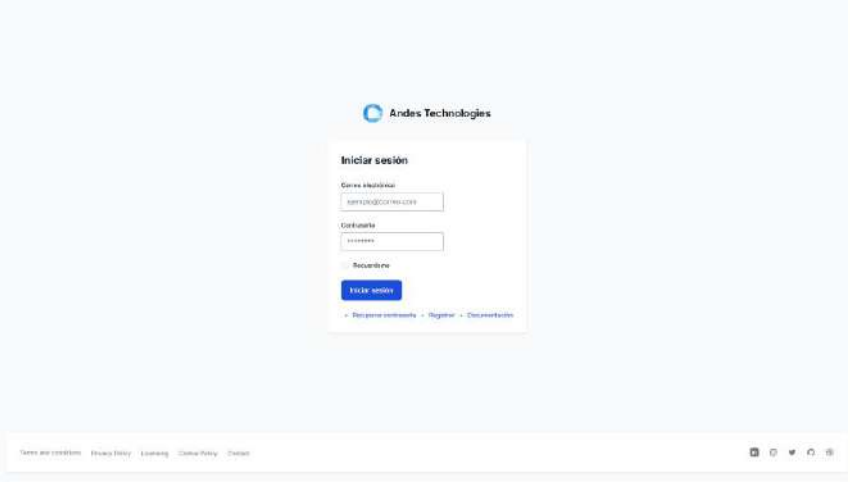


Ilustración 16: Interfaz de inicio de sesión

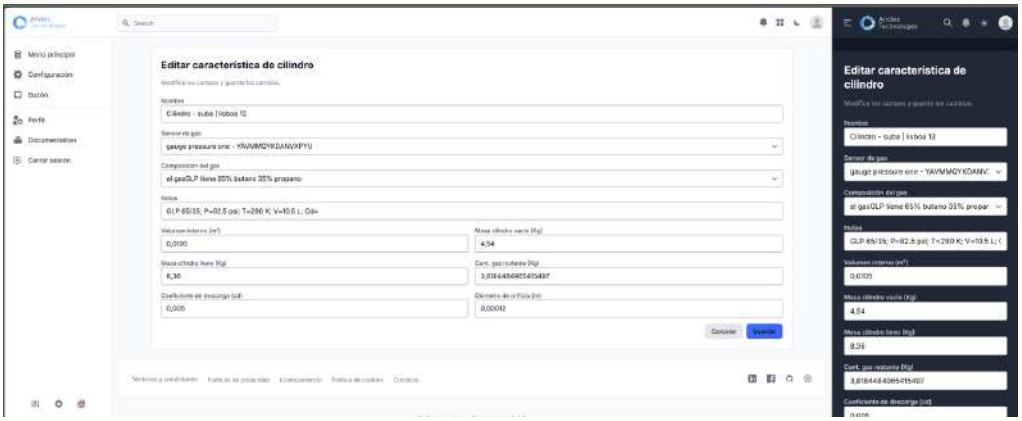


Ilustración 17: Interfaz de gestión de cilindro

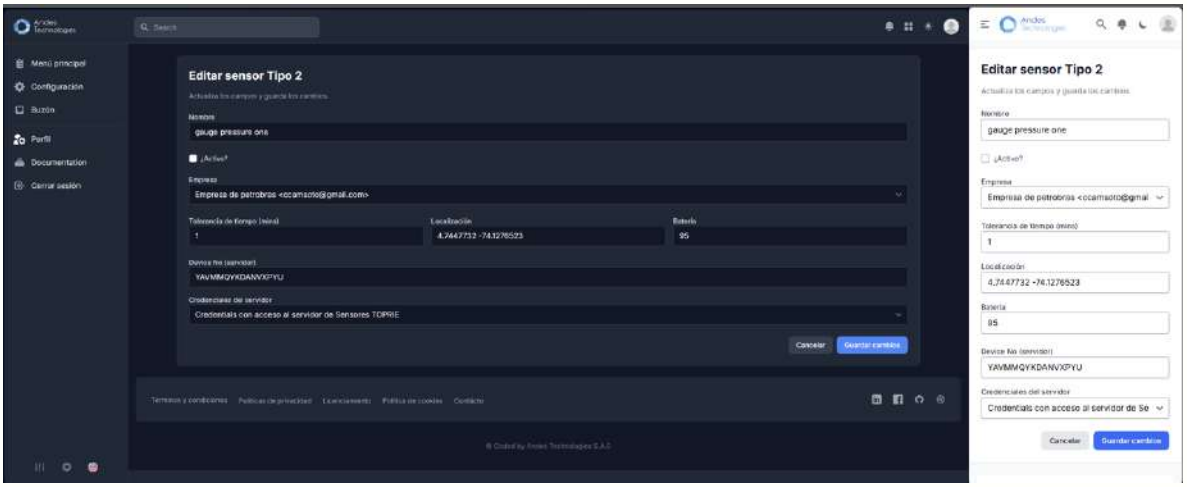


Ilustración 18: Interfaz de gestión del sensor de presión Toprie

Detalles del cilindro y dashboard

El dashboard constituye el núcleo visual de la aplicación, mostrando en tiempo real los datos procesados de los sensores. Fue diseñado con un enfoque de lectura rápida y jerarquización visual, priorizando la interpretación inmediata de los valores más relevantes:

- Capacidad actual de gas (%).
- Presión interna del cilindro.
- Temperatura del entorno.
- Estado de batería y señal del sensor.

Para la representación gráfica se empleó ApexCharts, lo que permite mostrar curvas de tendencia, indicadores porcentuales y comparaciones históricas con actualizaciones automáticas. Los datos son enviados desde la vista Django al DOM como objetos JSON y luego renderizados dinámicamente por JavaScript, garantizando sincronización con los registros más recientes del sistema.

Cada formulario incluye:

- Validaciones en tiempo real (HTML5 y backend Django).
- Mensajes de error o confirmación visuales en los campos.
- Componentes reutilizables con estilos uniformes definidos en Tailwind.

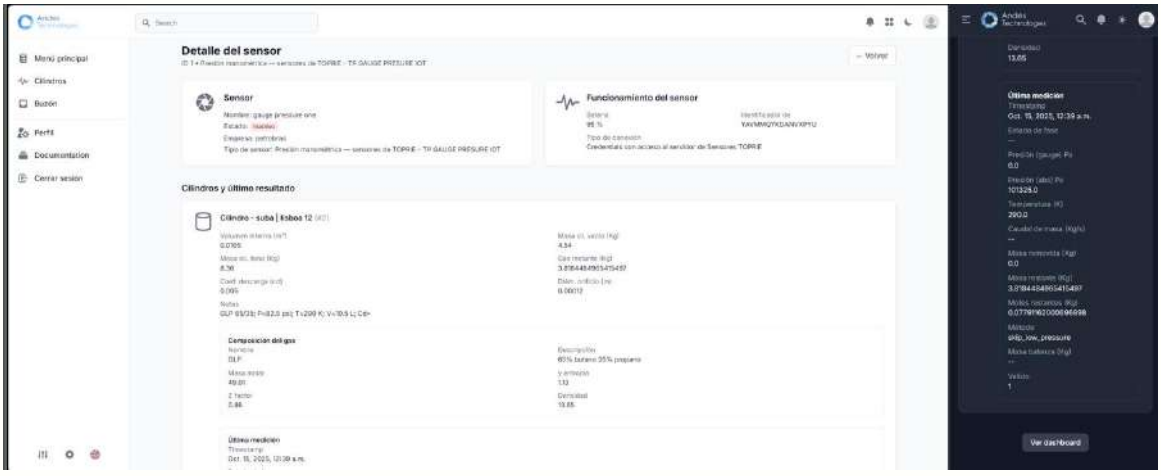


Ilustración 19: Interfaz de detalles del sensor y cilindro

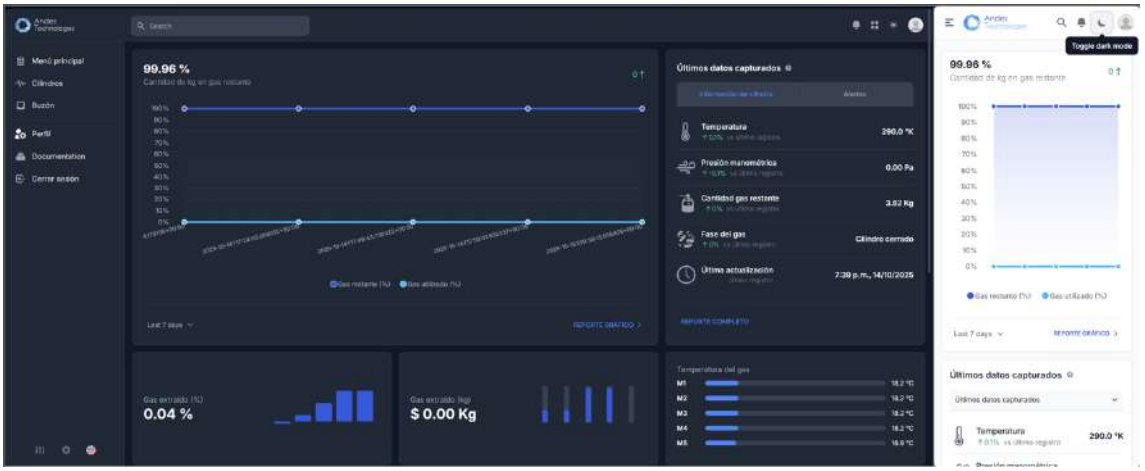


Ilustración 20: Interfaz superior del dashboard

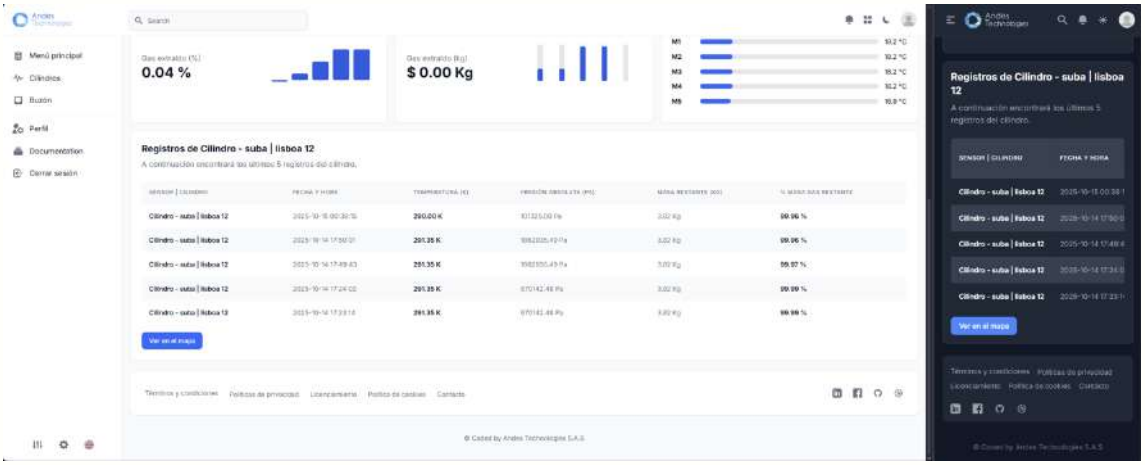


Ilustración 21: Interfaz inferior del dashboard

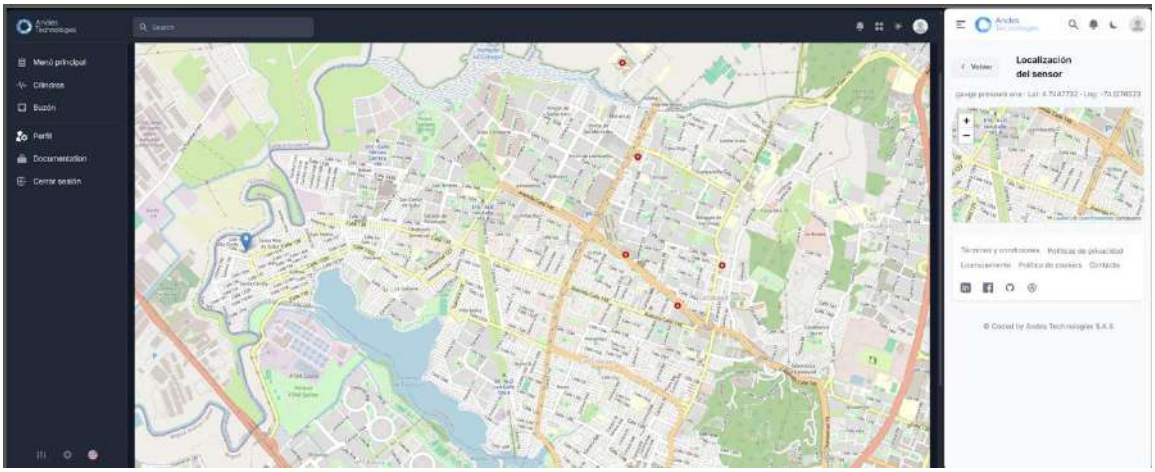


Ilustración 22: Interfaz de rastreo del sensor | cilindro

Listas y tablas de datos

Las tablas de registros presentan la información estructurada de toda la información que es mayormente administrada por el administrador. La tabla común para todos los usuarios es el sistema de tickets. En cuanto al diseño se implementaron con soporte para alternar colores por fila, formato responsivo y tipografía uniforme, facilitando la lectura de los datos técnicos (temperatura, presión, capacidad, hora). Y la gestión de estos.

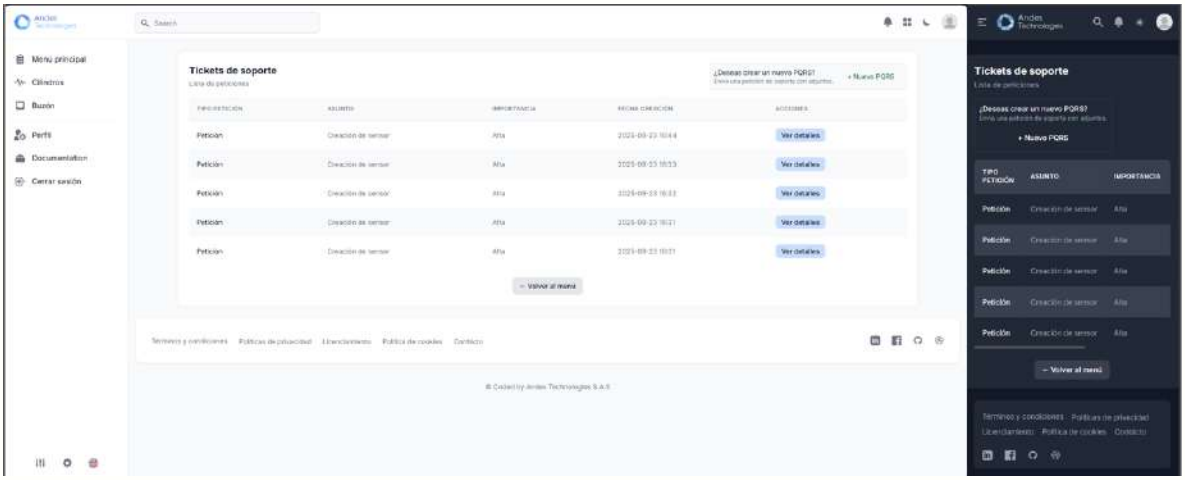


Ilustración 23: Interfaz de gestión de tickets

4.3.2 Diseño de interfaces entre componentes

El diagrama de arquitectura, mostrado en la Ilustración 6, ilustra de manera integral la comunicación entre los principales componentes del sistema Andes Board, tanto a nivel interno como externo. Este diagrama complementa los diagramas de clases y secuencia presentados en apartados anteriores, enfocándose en la interacción entre los servicios distribuidos que conforman el entorno de ejecución.

Los sensores de presión envían datos en tiempo real a través del protocolo MQTT, que garantiza una transmisión ligera y confiable hacia la API remota, la cual gestiona la autenticación y el control de acceso mediante el estándar OAuth.

Una vez recibidos los datos, estos son procesados por la aplicación principal desplegada en Azure App Container, donde los distintos servicios se comunican de forma orquestada a través de contenedores Docker. En este entorno, Redis actúa como intermediario de mensajería para la gestión de tareas en cola, Celery ejecuta los procesos asíncronos de cálculo y actualización de información, y PostgreSQL almacena de forma persistente los registros y resultados procesados.

El sistema integra además módulos de generación de reportes y notificaciones, los cuales producen alertas automáticas que son presentadas en el dashboard web, accesible mediante el protocolo HTTP tanto desde dispositivos móviles como de escritorio.

En conjunto, la arquitectura descrita establece un flujo de comunicación seguro, modular y escalable, que garantiza la continuidad operativa y la integridad de los datos, asegurando que Andes Board pueda mantener la sincronización entre sensores, servicios en la nube y usuarios finales de manera eficiente y confiable.

Capitulo V: Desarrollo de la solución

5.1 Implementación de la arquitectura

5.1.1 Proceso de codificación y desarrollo

El desarrollo del sistema Andes Board se llevó a cabo de manera iterativa, aplicando la metodología ágil Scrum combinada con el ciclo de vida de desarrollo de software (SDLC) en cada sprint. Para cada iteración, se planificaban las tareas a través del backlog del producto, priorizando las historias de usuario que representaban los requerimientos más relevantes del sistema.

Durante la fase de planificación, se definía el objetivo y los criterios de aceptación de cada historia de usuario, asegurando la modularidad del proyecto y disminuir su complejidad a gran escala. En la etapa de análisis, se elaboraban diagramas de flujo y bocetos en papel que representaban el recorrido de la información desde la interfaz hasta la base de datos y viceversa, permitiendo dividir el problema en módulos más pequeños y comprensibles antes de su codificación.

Durante la fase de desarrollo, se implementaron las funcionalidades de manera incremental. Para los módulos simples se recurrió a documentación técnica y recursos educativos en línea, mientras que para las funcionalidades críticas y con un alto grado de complejidad; como la gestión de cilindros, sensores y la implementación de los ecuaciones de Bernoulli en el proceso de limpieza de datos, se emplearon modelos de lenguaje (LLM), que permitieron acelerar el desarrollo y garantizar la precisión en la lógica aplicada. Asimismo, para el diseño de las vistas y la experiencia de usuario (UI/UX), se apoyó el proceso creativo en la generación de componentes visuales y estilos mediante LLM, aplicando TailwindCSS para mantener una coherencia visual y responsiva. Y finalmente, para la depuración de errores complejos se utilizó ChatGPT 4o en modo razonador.

La fase de pruebas se centró en los criterios de aceptación definidos en las historias de usuario, verificando que cada funcionalidad cumpliera con los resultados esperados. Para la validación de los cálculos se realizaron comparaciones con datos empíricos obtenidos de mediciones reales, garantizando la precisión de los resultados presentados en el dashboard.

5.1.2 Herramientas de lenguaje utilizados

Para el desarrollo e implementación del sistema Andes Board se empleó un conjunto de herramientas y lenguajes que puede ver cómo se conectan e interactúan entre si con las herramientas de *Ilustración 4: Diagrama de herramientas*. Cada tecnología fue seleccionada de acuerdo con su función dentro de la arquitectura del sistema, optimizando el procesamiento de datos IoT, la gestión de tareas asíncronas y la visualización de información en tiempo real. Y a continuación se presentan las principales herramientas utilizadas, clasificadas según su propósito dentro del entorno de desarrollo, backend, frontend y despliegue en producción.

Herramientas	Función principal en el sistema Andes Board
Python 3.12.4	Lenguaje principal de programación para la lógica del sistema, cálculos y tareas asíncronas.
Django 4.2.15	Framework backend basado en el patrón MVT, utilizado para el desarrollo del dashboard, API interna y gestión de usuarios.
Celery 5.4	Biblioteca para la ejecución de tareas asíncronas, utilizada para procesar ecuaciones de Bernoulli y notificaciones automáticas.
Redis 5.0.8	Sistema de mensajería (*message broker*) que coordina la comunicación entre Django y Celery para el manejo de colas de tareas.
PostgreSQL 15	Motor de base de datos relacional encargado del almacenamiento estructurado de información proveniente de los sensores y usuarios.
TailwindCSS	Framework de diseño CSS que proporciona una interfaz coherente, responsiva y visualmente uniforme en todo el sistema.
Flowbite	Librería de componentes basada en TailwindCSS para la construcción del dashboard interactivo.
ApexCharts	Librería para la representación gráfica de datos de presión, temperatura y capacidad del gas en el dashboard.
Docker & Docker Compose	Plataforma de contenedorización empleada para orquestar los servicios de Django, Redis, Celery, Beat y PostgreSQL en entornos aislados.
Azure App Container	Servicio de despliegue en la nube que aloja la aplicación y gestiona la escalabilidad y disponibilidad del sistema en producción.
Git & GitHub	Sistema de control de versiones y repositorio remoto utilizado para la gestión del código fuente y la colaboración.
Gunicorn	Servidor web utilizado como proxy inverso para la exposición segura del dashboard y la API mediante el protocolo HTTPS.
SMTP (Simple Mail Transfer Protocol)	Protocolo utilizado por Django para el envío automatizado de alertas y notificaciones a los usuarios.

Tabla 5: Tabla de herramientas

5.2 Integración de componentes

5.2.1 Estrategias de integración

La integración de componentes en Andes Board se desarrolló inicialmente con la configuración de Django y su ORM para la administración de la base de datos PostgreSQL, asegurando la correcta persistencia de la información y el uso de migraciones automáticas. Posteriormente, se integraron los servicios de Redis y Celery, responsables de la mensajería interna y de la ejecución de tareas asíncronas para el procesamiento de los datos provenientes de los sensores IoT.

La orquestación de todos los servicios se realizó mediante Docker Compose, garantizando la portabilidad y el aislamiento de cada contenedor. Cada componente (aplicación Django, PostgreSQL, Redis, Celery y Celery Beat) fue configurando como un servicio independiente con sus propias dependencias, interconectado a través de una red interna definida por Docker. Para mantener la seguridad y flexibilidad de la configuración, se implementó un archivo .env que almacena las variables de entorno necesarias (credenciales, puertos, rutas de conexión y parámetros de Celery), desacoplando la configuración del código fuente.

El archivo docker-compose.yml que define la arquitectura de contenedores se presenta en el Anexo A-1, mientras que el Dockerfile multietapa de la aplicación, utilizado para compilar el entorno Python 3.11.5-slime, se incluye en el Anexo A-2. A continuación se ilustra la integración de los distintos servicios y su configuración y orquestación.

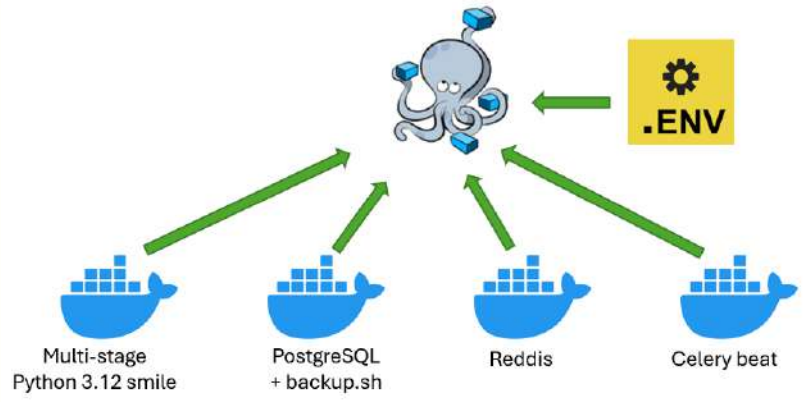


Ilustración 24: servicios y componentes de docker-compose.yml: Elaboración propia

5.2.2 Pruebas de integración

Las pruebas de integración realizadas en Andes Board tuvieron como objetivo validar la comunicación entre los distintos componentes del sistema, garantizando la correcta interacción entre los servicios Django, Celery, Redis y PostgreSQL, así como el flujo completo de datos desde la API del sensor hasta el dashboard web.

Inicialmente, se realizaron pruebas de conectividad y procesamiento entre los contenedores orquestados en Docker Compose, comprobando que las tareas asíncronas de Celery se ejecutaran correctamente al recibir mensajes desde Redis, y que los resultados se persistieran en la base de datos PostgreSQL.

Para evaluar la comunicación con los sensores remotos, se utilizó Postman como herramienta de simulación, enviando peticiones HTTP a la API del servidor de Toprie y verificando la respuesta del sistema. Estas pruebas permitieron validar la autenticación mediante OAuth, la recepción de los datos en formato JSON, y su correcta transformación y almacenamiento en las tablas correspondientes.

Durante el proceso de cálculo y procesamiento de datos, se implementaron técnicas de programación defensiva y sentencias assert para detectar inconsistencias o errores en tiempo de ejecución. Por ejemplo, se emplearon validaciones para asegurar que las presiones registradas fueran positivas y que las unidades físicas mantuvieran coherencia con el modelo termodinámico. El fragmento (Anexo A-3) muestra un ejemplo simplificado de estas pruebas dentro del módulo de cálculo.

Estas validaciones garantizan la confiabilidad del proceso de cálculo y evitan que datos erróneos comprometan el funcionamiento del pipeline. Si bien no se aplicaron metodologías formales de prueba de caja blanca, gris o negra, el enfoque adoptado permitió validar de forma práctica la integración de los módulos y asegurar la estabilidad del sistema en su entorno productivo.

5.3 Documentación técnica

5.3.1 Documentación del código

El código fuente del sistema Andes Board fue documentado siguiendo las convenciones estándar de Python PEP 257, mediante docstrings y comentarios descriptivos que explican la funcionalidad de cada clase, método y función. Cada módulo dentro de la aplicación contiene nombres descriptivos y docstrings que detallan los parámetros, retornos y propósito de los métodos, manteniendo la coherencia semántica y el cumplimiento de los principios SOLID aplicados durante el desarrollo. El anexo A-4, muestra un fragmento simplificado de la documentación.

En la raíz del proyecto se mantiene una carpeta /docs que centraliza los diagramas de arquitectura y base de datos, junto con documentación técnica complementaria. Dentro de esta carpeta se incluyen los archivos .drawio, que pueden abrirse con la herramienta diagrams.net, y el archivo .mwb, que corresponde al modelo relacional de la base de datos y se edita con MySQL Workbench.

Asimismo, se dispone de un archivo README.md con las instrucciones de instalación, ejecución del sistema, tanto en entorno local como dentro de contenedores Docker, lo que facilita la replicabilidad del entorno de desarrollo.

Si bien no se utilizaron herramientas automáticas de generación de documentación como Sphinx o Django Extensions, la organización estructurada del código, junto con los docstrings y recursos incluidos en /docs, proporcionan una base clara y mantenible para la comprensión y evolución del proyecto.

5.3.2 Manuales de usuario y desarrollador

El sistema Andes Board cuenta con tres tipos de usuarios principales: administrador, empresa y operador, cada uno con permisos y responsabilidades específicas dentro del flujo operativo. El siguiente manual describe los pasos principales que debe seguir cada perfil, de acuerdo con el proceso representado en el Diagrama de flujo general (Ilustración 6), facilitando la comprensión del funcionamiento del sistema desde la autenticación hasta la visualización de datos.

Como parte del proceso de documentación del proyecto Andes Board, se elaboraron dos manuales complementarios que cubren tanto el uso funcional del sistema como los aspectos técnicos de ejecución en local y su despliegue en docker.

El manual del usuario está dirigido a los diferentes perfiles que interactúan con la plataforma —administrador, empresa y operador— y describe de manera detallada el proceso de autenticación, gestión de sensores, consulta de datos en el dashboard y revisión de alertas generadas por el sistema. Cada procedimiento se acompaña de capturas de pantalla que facilitan la comprensión de los flujos de uso.

Por su parte, el manual del desarrollador está orientado a los programadores encargados del mantenimiento o extensión del sistema. En él se explica la estructura del proyecto, la configuración del entorno de desarrollo, el uso de contenedores Docker, la integración de CDN, proceso de backup en los servicios de postgresql, servicios como Redis, Celery y PostgreSQL, y las pautas para extender las estrategias, servicios y repositorios del código fuente. Adicionalmente, se incluye un manual de ejecución con las instrucciones necesarias para levantar el entorno en local o en producción, configurando las variables de entorno, dependencias y volúmenes de Docker. Los enlaces de acceso directo a cada uno de estos manuales se encuentran disponibles en el Anexo A-5, donde se detallan las ubicaciones de los documentos y su formato de distribución.

5.3.3 Especificaciones del software

Con el fin de garantizar la replicabilidad, mantenibilidad y despliegue adecuado del sistema Andes Board, se documentaron todas las dependencias utilizadas durante el desarrollo. Estas librerías corresponden a los módulos necesarios para la ejecución del backend, el procesamiento de tareas asíncronas, la comunicación con la base de datos y la generación de reportes. Las versiones listadas fueron las empleadas en el entorno final de pruebas y corresponden al archivo *requirements.txt* del proyecto.

- amqp==5.2.0
- async-timeout==4.0.3
- celery==5.4.0
- certifi==2024.7.4
- charset-normalizer==3.3.2
- click==8.1.7
- click-didyoumean==0.3.1
- click-plugins==1.1.1
- click-repl==0.3.0
- Django==4.2.15
- django-environ==0.11.2
- djangorestframework==3.15.2
- folium==0.17.0
- idna==3.7
- gascompressibility==1.0.0
- pillow==10.4.0
- prompt-toolkit==3.0.47
- psycopg2==2.9.9
- python-dateutil==2.9.0.post0
- python-decouple==3.8
- python-dotenv==1.0.1
- redis==5.0.8
- requests==2.32.3
- six==1.16.0
- urllib3==2.2.2
- wcwidth==0.2.13
- openpyxl==3.1.5

5.3.4 Publicación y licencia del código fuente

Con el fin de garantizar la trazabilidad técnica del proyecto y facilitar la verificación académica del desarrollo realizado, el código fuente del sistema Andes Board se encuentra documentado y organizado en los anexos del presente trabajo. El proyecto se distribuye bajo la licencia **Creative Commons Atribución–Sin Derivados 4.0 Internacional (CC BY-ND 4.0)**, la cual permite su consulta y redistribución con fines académicos o investigativos, siempre que se otorgue el correspondiente crédito al autor. No obstante, esta licencia **no autoriza la modificación, adaptación o creación de obras derivadas**, preservando así la integridad del software desarrollado.

Para complementar la documentación técnica, se incluyen los archivos más relevantes del sistema dentro de los anexos:

- **Anexo A-1:** archivo docker-compose.yml, que define la orquestación de contenedores y servicios del sistema.
- **Anexo A-2:** archivo Dockerfile, utilizado para la construcción de la imagen principal del backend.
- **Anexo A-3:** script de cálculo termodinámico correspondiente al módulo responsable de estimar la masa restante del cilindro de GLP.

De esta forma, el lector dispone de los componentes esenciales del proyecto sin comprometer la protección intelectual del código, garantizando al mismo tiempo su revisión técnica y reproducibilidad en entornos controlados.

Capitulo VI: Evaluación y pruebas

6.1 Pruebas de sistema

6.1.1 Estrategias y tipos de pruebas

Las pruebas del sistema Andes Board se realizaron bajo un enfoque práctico en el cual se validan los logs con el propósito de ser analizados y garantizar la operatividad de todos los módulos y servicios. Por otro lado, dado que el sistema integra distintos componentes, como la API de los sensores, el procesamiento asíncrono mediante Celery, la base de datos PostgreSQL y el sistema web, las pruebas se organizaron en diferentes niveles para cubrir desde el correcto funcionamiento de pequeños módulos o funciones hasta la el funcionamiento global del sistema.

Durante la ejecución, los resultados fueron observados directamente desde las terminales mostrando logs del sistema, en las interfaces, en las consultas SQL y, en gran parte, a través de los registros generados en los logs del sistema. A continuación, se presenta la clasificación general de los tipos de prueba aplicados en el proyecto:

Tipo de prueba	Objetivo	Resultado esperado
Pruebas unitarias	Validar el correcto funcionamiento de funciones individuales y módulos aislados del sistema.	Cada función al ser ejecutada retorna los logs para ser analizados 1 por 1.
Pruebas de integración	Comprobar la comunicación y sincronización entre los servicios de Docker (Django, Redis, PostgreSQL).	Los contenedores se comunican; y los logs registran el proceso de orquestación
Pruebas de aceptación	Confirmar que las historias de usuario satisfacen los criterios definidos y las expectativas del usuario final.	Las funcionalidades son aceptadas al cumplir los flujos esperados definidos en las historias de usuario.

Tabla 6: Tabla de tipos de pruebas

6.1.2 Resultados de las pruebas

Los resultados de las pruebas realizadas sobre el sistema *Andes Board* demuestran la estabilidad entre los diferentes módulos que lo componen. Cada tipo de prueba bien sea unitaria, integración o aceptación, se ejecutó de forma controlada, documentando su resultado y evidencias en anexos e imágenes dentro de este capítulo.

- **Pruebas unitarias:** Se desarrolló un unit testing empleando la clase TestCase del framework Django. En este entorno, se simularon datos provenientes de la base de datos y del sensor IoT con el fin de evaluar el comportamiento de la clase `GasMaths`, responsable de los ecuaciones de Bernoulli del sistema. La función evaluada fue `gm.choked(info=info, ctx=ctx)`, que calcula la masa removida y el flujo másico bajo condiciones críticas. Para ver los cálculos empleados diríjase al anexo A-7.

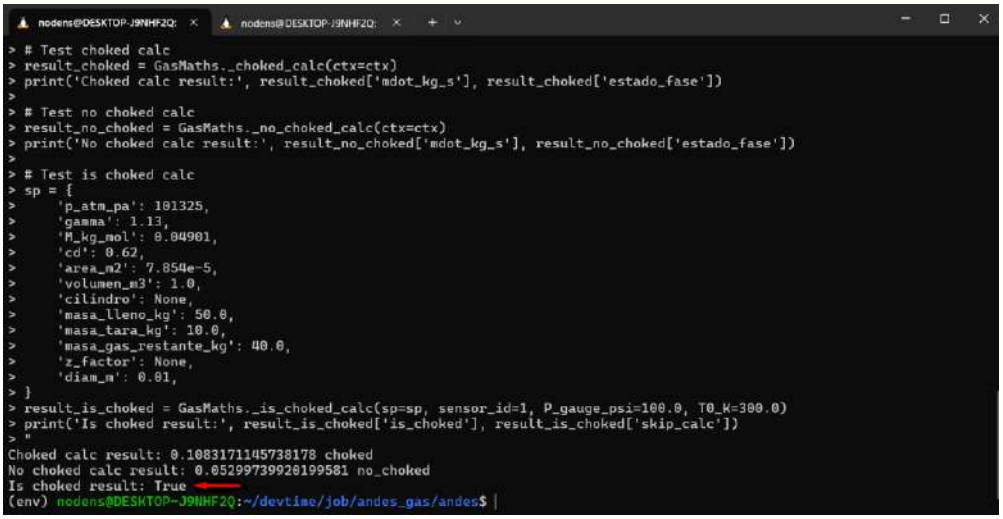


Ilustración 25: Prueba unitaria de cálculos matemáticos. Elaboración propia

El valor obtenido explica que durante un minuto, bajo unas condiciones constantes de presión de salida a 100 psi, fueron extraídos 0.1083 kg/min de masa de gas. Al ser la presión de salida un valor constante y alto, por defecto es un flujo crítico con una salía estrangulada. Ya que si el gas estuviera descendiendo rápidamente estaría siendo un flujo subcrítico no estrangulado. Lo que denotan los logs es que las funciones denotan resultados coherentes en cuanto al comportamiento de los cálculos y realistas en cuanto a los datos.

- **Pruebas de integración:** Se centraron en la verificación del orquestador Docker Compose, encargado de sincronizar los cuatro contenedores principales del sistema: Django, PostgreSQL, Redis, y Celery. (Ver Ilustración 25) El objetivo fue comprobar que la configuración de las variables de entorno (.env) y los enlaces de red garantizaran el despliegue correcto del entorno, permitiendo la comunicación fluida entre los servicios y el funcionamiento estable del sistema en producción.

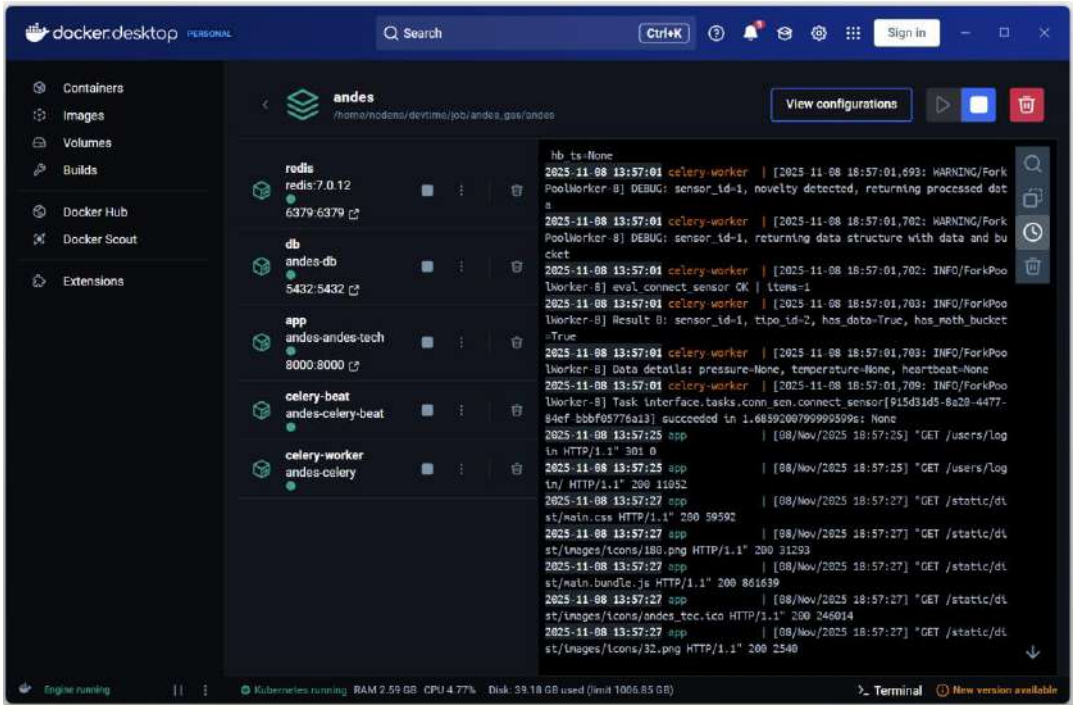


Ilustración 26: Validación de integración de los diferentes sistemas. Elaboración propia

En la imagen se evidencia cómo la **red de tipo puente** permite la interconexión de los cuatro contenedores, donde Celery Beat genera un contenedor adicional para ejecutar las tareas programadas en intervalos definidos.

Durante la ejecución, los registros del sistema mostraron una advertencia relacionada con los permisos de las tareas, lo que permitió i**dentificar una mejora de seguridad**: evitar el uso de superusuarios en las tareas automatizadas y crear subusuarios dedicados en PostgreSQL, ya que tanto Redis como Celery poseen acceso directo a la base de datos y podrían, en caso de configuración insegura, comprometer la integridad de la información.

- **Pruebas de aceptación:** Se centraron en la verificación y cumplimiento de los criterios de aceptación junto al modelo y notación de procesos de negocios (BPMN) que facilita la comprensión del funcionamiento del sistema desde la autenticación hasta la visualización de datos. (Ver Ilustración 5)

    - **RF-01:** Gestión de empresas (CRUD por administrador) Para el inicio de sesión (ver Ilustración 16), tras autenticarse el administrador accede al menú principal (ver Ilustración 14), donde puede ingresar al módulo **"Empresas"** para visualizar la lista, editar los estados y eliminar registros existentes. Esta funcionalidad valida el control centralizado de usuarios empresariales por parte del administrador.

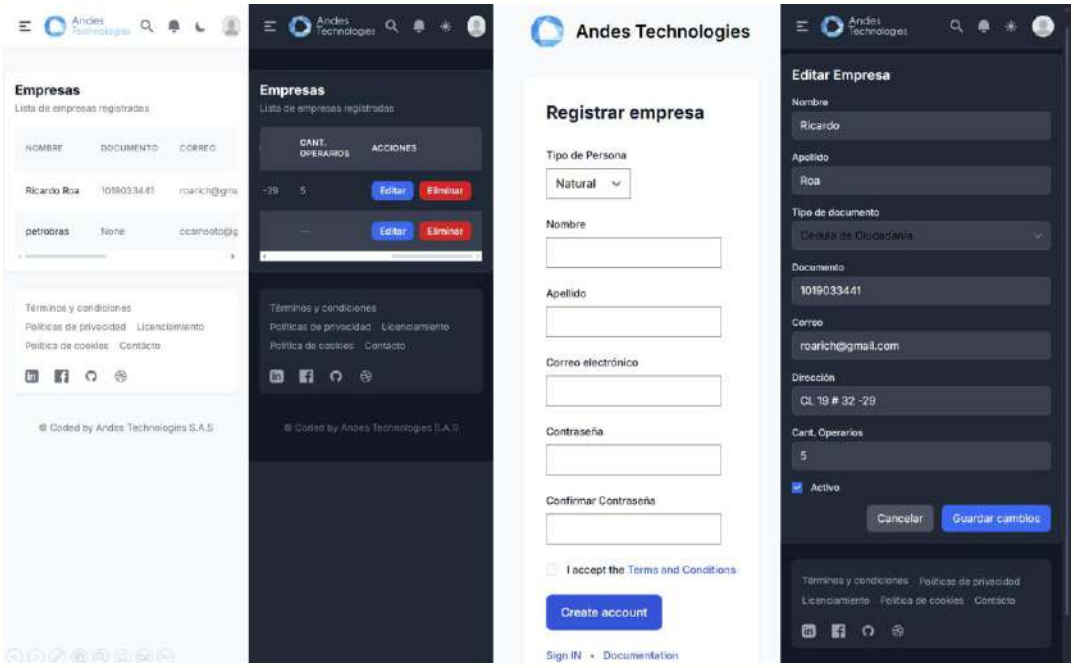


Ilustración 27: Vistas de administración de empleados. Elaboración propia

En la anterior imagen se muestran las diferentes interfaces que permiten la gestión de las empresas.

- **RF-02:** Gestión de operadores (CRUD por empresa) Una vez activada la cuenta de la empresa, la empresa puede acceder al módulo "Operadores" (ver Ilustración 12), donde registra y administra los usuarios encargados de operar los sensores. Se validó que el sistema permita crear, editar y eliminar operadores autenticados correctamente (ver Ilustración 16).

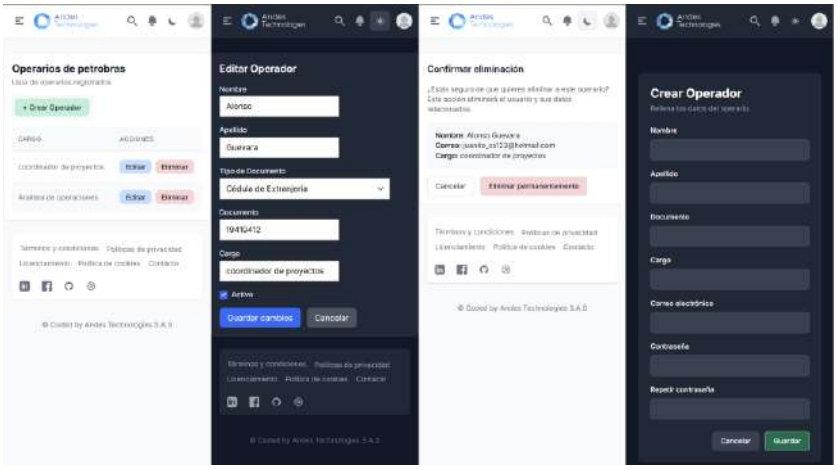


Ilustración 28: Vistas de administración de operarios: Elaboración propia

- **RF-03:** Gestión de cilindros de gas: Desde el menú principal (ver Ilustración 13), donde al seleccionar **"Cilindros"**, los operadores pueden registrar gestionar las propiedades de los cilindros según la empresa asignada. Se comprobó que las operaciones CRUD se realicen exitosamente y se reflejen en la base de datos.

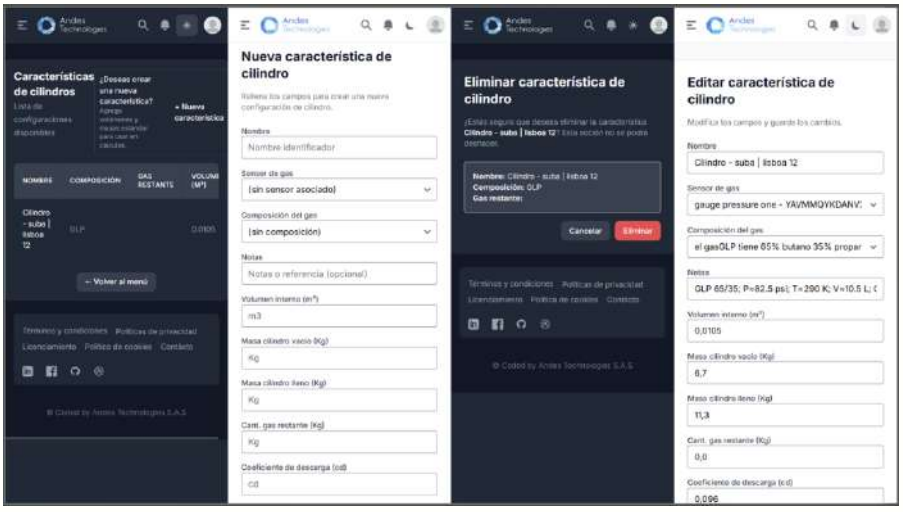


Ilustración 29: Vistas de administración de cilindros de gas. Elaboración propia

- **RF-04:** Gestión y asignación de sensores: El administrador accede al menú

principal (ver Ilustración 14), donde puede ingresar al módulo “Monitorear” para observar el menú de configuración (ver anexo A-7) y seleccionar la opción “sensores”, el administrador puede asignar el sensor a una empresa. Esta funcionalidad valida el control centralizado de usuarios empresariales por parte del administrador.

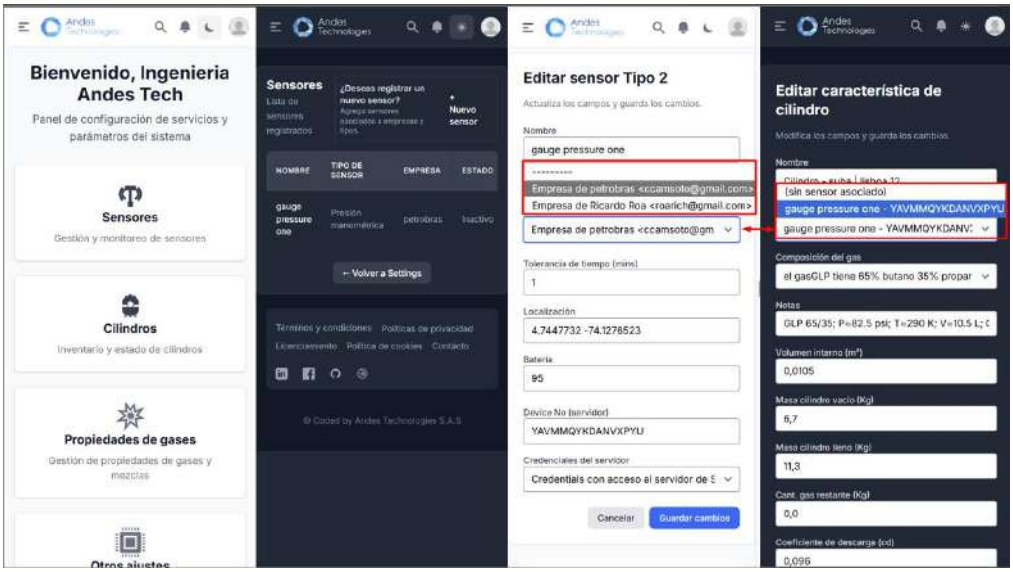


Ilustración 30: Vistas para asignar sensores a cilindro de gas: Elaboración propia

- **RF-05:** Visualización de posición geográfico: La empresa o operador puede visualizar la ubicación de cada sensor directamente en el mapa del dashboard. Se verificó que las coordenadas recibidas desde el sensor IoT se muestren correctamente sobre el mapa en tiempo real. (ver Ilustración 22).
- **RF-06:** Monitoreo de variables del cilindro: El dashboard muestra la presión, temperatura, gas restante y otras variables de cada cilindro. Se comprobó que los valores actualizados coincidan con los registros enviados por la API del fabricante, validando su consistencia. (ver Ilustración 20, Ilustración 21 y anexo A-5 | Video de comparación de resultados).
- **RF-07:** Generación de alertas automáticas: Se verificó que, cuando los valores de presión o batería alcanzan niveles críticos o el sensor se desactiva, el sistema envía notificaciones automáticas por correo electrónico a los usuarios correspondientes. Las alertas fueron registradas en los logs y notificadas al

instante (Ver anexo A-8).

- **RF-08:** Gestión de PQRS: Las empresas y operadores pueden crear, consultar y cerrar solicitudes de soporte (PQRS). Se comprobó que las solicitudes cambien de estado correctamente y que el sistema notifique la resolución al usuario solicitante. (Ver Ilustración 23).
- **RF-09:** Visualización de reportes y dashboard: El sistema presenta indicadores, gráficos e históricos de datos dentro del módulo “Dashboard” y en el área de reportes descargables (.xlsx). Se validó que las consultas se procesen sin errores y que la información se muestre actualizada. (Ver anexo A-9).

6.2 Evaluación de rendimiento

6.2.1 Análisis de rendimiento

Con el fin de evaluar el comportamiento del sistema *Andes Board* en condiciones normales de ejecución, se realizó una observación empírica del rendimiento utilizando el **Administrador de tareas de Windows**, donde se monitorearon los consumos de CPU, memoria, disco y red durante el funcionamiento del sistema en entorno local.

Ilustración 30: Vista de recursos utilizazdos por el sistema de docker: Elaboración propia

En la Ilustración 26 se aprecia que los picos de consumo de CPU corresponden principalmente a procesos externos como Visual Studio Code, Docker Desktop, Microsoft Word y Opera Browser, los cuales se encontraban ejecutándose simultáneamente durante la prueba. Estos procesos no representan la carga real del sistema, por lo que se puede inferir que Andes Board mantiene un consumo moderado de recursos y no requiere un hardware de alto rendimiento para su ejecución estable.

Asimismo, el uso de memoria se mantuvo en niveles inferiores al 90 %, y el consumo de red fue prácticamente nulo, dado que las pruebas se realizaron en un entorno local con comunicación interna entre contenedores Docker. El proceso identificado como andes – App – Docker Desktop mostró un comportamiento constante y estable, confirmando la eficiencia del entorno de contenedorización y la correcta sincronización entre los servicios Django, Redis, Celery y PostgreSQL.

Dado que el proyecto no alcanzó un nivel de detalle que permitiera aplicar pruebas automatizadas de carga o estrés, se realizaron **pruebas cronometradas manuales**, cuyos resultados se presentan a continuación:

Actividad	Tiempo promedio
Descargar y configurar el sistema	6 min 21 s
Iniciar el sistema por primera vez	1 min 13 s
Iniciar el sistema con caché	11 s
Navegar entre diferentes vistas del sistema	< 2 s
Generar reporte *.xlsx*	< 4 s

Tabla 7: Validación del cumplimiento de los requisitos

Los resultados reflejan una alta capacidad de respuesta y bajo tiempo de carga en las operaciones más frecuentes del sistema, evidenciando que la arquitectura basada en Docker Compose contribuye significativamente al rendimiento general del proyecto.

6.2.2 optimización

Durante el desarrollo del sistema Andes Board se implementaron diversas prácticas de optimización orientadas a mejorar el rendimiento general y la eficiencia del procesamiento de datos.

En el backend, se incorporó el uso de tareas asíncronas mediante Celery, lo que permitió delegar las operaciones de cálculo y procesamiento de datos a procesos independientes, evitando bloqueos en el servidor Django y garantizando una respuesta fluida en las solicitudes de los usuarios. De manera complementaria, se configuró Redis como broker de mensajería de alta velocidad, optimizando la comunicación entre los componentes y reduciendo la latencia en la ejecución de tareas distribuidas. En el nivel de persistencia, se implementó la indexación de tablas en PostgreSQL, con el fin de acelerar las consultas realizadas por las vistas y reportes del sistema, especialmente en operaciones de filtrado y agregación de datos.

En la capa de presentación, se aplicaron técnicas de optimización de recursos estáticos utilizando CDN con Webpack y TailwindCSS, que permiten reducir el tamaño de los archivos CSS y JavaScript mediante compilación y minificación, mejorando los tiempos de carga del dashboard web.

Finalmente, la aplicación de los principios SOLID y el patrón DRY (Don’t Repeat Yourself) contribuyeron a una arquitectura más eficiente, con un código modular, reutilizable y fácil de mantener, reduciendo la duplicidad de funciones y mejorando la escalabilidad del sistema.

6.3 Validación y verificación

6.3.1 Validación contra los requisitos

La validación y verificación del sistema *Andes Board* se realizó a partir de los diez requisitos funcionales definidos durante la fase de análisis, contrastando cada uno con las funcionalidades efectivamente implementadas en la aplicación. Este proceso permitió asegurar que todas las operaciones se ejecutan conforme a los criterios de aceptación.

Código	Descripción	Criterio de aceptación	Cumplimiento
RF-01	Gestión de empresas (CRUD por administrador).	El administrador puede crear, editar, listar y eliminar empresas.	Cumplido
RF-02	Gestión de operadores (CRUD por empresa).	La empresa puede registrar y administrar sus operadores autenticados.	Cumplido
RF-03	Gestión de cilindros de gas.	El operador puede registrar, editar y consultar cilindros asociados a sensores.	Cumplido
RF-04	Gestión y asignación de sensores.	El administrador puede registrar y vincular sensores a empresas y cilindros.	Cumplido
RF-05	Visualización de posición geográfica.	El operador puede visualizar la ubicación del sensor en el mapa del dashboard.	Cumplido
RF-06	Monitoreo de variables del cilindro.	El sistema muestra presión, temperatura, batería y señal en tiempo real.	Cumplido
RF-07	Generación de alertas automáticas.	Se envía una notificación por correo ante niveles críticos de presión.	Cumplido
RF-08	Gestión de PQRS.	Los usuarios pueden registrar, consultar y responder solicitudes de soporte.	Cumplido
RF-9	Visualización de reportes y dashboard.	El sistema presenta indicadores, gráficos y registros históricos.	Cumplido

Tabla 8: Validación del cumplimiento de los requisitos

La validación de los diez requisitos confirma que el sistema cumple plenamente con las especificaciones funcionales establecidas. Cada componente fue verificado mediante pruebas manuales y observación directa del flujo de datos, lo que permitió comprobar la consistencia del procesamiento, la correcta comunicación entre módulos y la visualización efectiva de resultados.

6.3.2 Verificación de la solución

La verificación del sistema Andes Board permitió confirmar que los módulos principales —captura de datos, procesamiento de información y visualización en el dashboard— funcionan de manera integrada y coherente con los objetivos definidos durante el diseño.

Las pruebas realizadas, tanto unitarias como de integración y aceptación, demostraron que el sistema responde correctamente ante distintos escenarios de uso, manteniendo la estabilidad de los servicios y la consistencia de los datos.
El análisis de rendimiento evidenció que el consumo de recursos del sistema es bajo y que la comunicación entre los contenedores Django, Redis, Celery y PostgreSQL se mantiene fluida, incluso bajo tareas concurrentes o procesos en segundo plano.

Con base en los resultados obtenidos, se puede afirmar que la solución desarrollada cumple con los requisitos funcionales, técnicos y de rendimiento establecidos. Además, la arquitectura modular implementada facilita su mantenimiento y escalabilidad futura, asegurando que el sistema pueda adaptarse a nuevos sensores o volúmenes de datos sin comprometer su desempeño.

Capitulo VII: Conclusiones y recomendaciones

7.1 Conclusiones

7.1.1 Resumen de hallazagos

- Los resultados obtenidos permiten concluir que sí es posible determinar la cantidad de gas GLP en estado bifásico mediante la combinación de sensores de presión y temperatura, junto con la aplicación de modelos de flujo estrangulado basados en la ecuación de Bernoulli (ver Anexo A-6). Asimismo, la integración de estos datos en un entorno orquestado y aislado mediante contenedores Docker demostró ser funcional y eficiente, permitiendo el monitoreo remoto, la visualización en múltiples dispositivos y la generación de alertas automáticas en tiempo real.
- En cuanto al rendimiento del sistema, se comprobó su estabilidad dentro del entorno Docker, manteniendo un uso moderado de recursos del procesador, un consumo constante cercano a los 20 MB de memoria RAM y picos de actividad de red únicamente durante la descarga de recursos estáticos o imágenes (ver Ilustración 31). Los resultados detallados de las pruebas de rendimiento se presentan en la Tabla 7.
- El sistema superó satisfactoriamente las pruebas unitarias (ver Ilustración 26), las pruebas de integración (ver Ilustración 27) y las pruebas de aceptación, validadas a partir de los criterios definidos en las diez historias de usuario (ver Tabla 8). Estos resultados confirman la correcta interacción entre los componentes principales; API, tareas asíncronas, base de datos y dashboard web y garantizan la coherencia del flujo completo del sistema.
- La documentación técnica y funcional cubre todos los módulos del sistema, incluyendo las 47 vistas desarrolladas para los distintos tipos de usuario (ver Anexo A-5), así como un video explicativo sobre la asignación de variables termodinámicas del cilindro. Además, se elaboraron dos manuales para desarrolladores, uno orientado al levantamiento del sistema y otro enfocado en las pautas para la ampliación y escalabilidad del software.

- En cuanto al seguimiento metodológico bajo Scrum, se observó que varias tareas del product backlog presentaron retrasos en su ejecución, siendo la más crítica la historia de usuario 3 (ver Anexo A-15), la cual experimentó un atraso superior a un mes debido a desafíos técnicos que se abordan en la sección 7.2.1 Mejoras sugeridas.
- Uno de los principales riesgos detectados es la dependencia del servidor remoto del fabricante, que podría afectar la latencia o la disponibilidad del servicio. Como medida preventiva, se plantearon estrategias como el uso de caché local y validaciones dentro del pipeline de datos, con el fin de garantizar la estabilidad operativa incluso ante fallas externas.
- Finalmente, las Ilustraciones 5, 7, 8 y 9 evidencian la conexión e interacción entre los diferentes componentes del software, mostrando cómo estos cooperan para cumplir con los requisitos funcionales y las historias de usuario establecidas.

7.1.2 Cumplimiento de los objetivos

El desarrollo e implementación del sistema Andes Board permitió cumplir con el objetivo general propuesto, demostrando la viabilidad de construir una solución tecnológica capaz de monitorear de forma remota cilindros de gas GLP mediante sensores IoT de presión y temperatura.

El sistema logra calcular de manera precisa el volumen y flujo másico del gas en estado bifásico, valiéndose de ecuaciones de flujo másico implementados en la capa de negocio, y posteriormente integra estos datos en una plataforma web, que permite su visualización de datos y la emisión de alertas automatizadas ante condiciones críticas de presión o batería.

En relación con los objetivos específicos, se alcanzaron los siguientes resultados:

- Definición de requisitos funcionales y no funcionales: A través de historias de usuario, se definieron y priorizaron nueve requisitos funcionales que guiaron el proceso de

desarrollo. Estos fueron posteriormente validados mediante pruebas de aceptación y verificación empírica, garantizando que el sistema cumpliera con las necesidades de monitoreo, gestión de alarmas y visualización de datos en el dashboard web.

- Diseño e implementación de la aplicación web: Se desarrolló una arquitectura monolítica bajo el patrón MVT utilizando Django, lo que permitió mantener la coherencia entre la lógica de negocio, las vistas y los modelos de datos.
  El sistema implementa los principios SOLID y el patrón Strategy para gestionar de manera flexible los diferentes tipos de sensores y cálculos asociados. En cuanto al diseño de interfaz, se emplentó un CDN con webpack,TailwindCSS y Flowbite, que entrega los estilos de forma más veloz, garantizando consistencia visual, usabilidad y compatibilidad en distintos dispositivos. Además, la integración de Celery y Redis permitió manejar tareas asíncronas, como la ingesta y procesamiento de datos del sensor, manteniendo un rendimiento estable sin bloqueos en el servidor.

- Diseño y configuración de la arquitectura de despliegue: El sistema fue empaquetado y desplegado mediante contenedores Docker, orquestados con Docker Compose, permitiendo la integración de los servicios Django, PostgreSQL, Redis, Celery y Celery Beat.
  Si bien no fue posible implementar la conexión directa con Azure IoT Hub debido a las limitaciones del sensor, la integración se logró satisfactoriamente a través de la API del fabricante TOPRIE, asegurando la recepción y procesamiento automatizado de los datos de presión y temperatura.
  Este despliegue demostró ser estable, escalable y portable, cumpliendo con los parámetros de eficiencia y alta disponibilidad definidos en el alcance del proyecto.

A pesar de las limitaciones observadas en la conexión directa con servicios IoT y en la validación del flujo no estrangulado, la solución desarrollada constituye una base sólida y operativa para futuras versiones del sistema y su escalamiento hacia entornos industriales o comerciales más amplios.

7.2 Recomendaciones

7.2.1 Mejoras sugeridas

Durante el desarrollo e implementación del sistema Andes Board se identificaron diversas oportunidades de mejora que permitirán incrementar su estabilidad, alcance y seguridad en futuras versiones del proyecto.

- En primer lugar, se propone **ampliar la capacidad del dashboard para mostrar información histórica más allá de los últimos cinco registros**. Esto permitiría generar promedios y tendencias de consumo por día, semana, mes o año, facilitando el análisis del comportamiento del gas y la toma de decisiones basadas en datos históricos. Esta mejora requeriría la optimización de las consultas a la base de datos y el uso de técnicas de agregación en PostgreSQL o Pandas para evitar sobrecargar el servidor.
- En cuanto a la parte física del sistema, se recomienda utilizar sensores IoT con mayor compatibilidad con servicios en la nube, ya que el sensor de Toprie tiene serias limitaciones de conexión, de modo que la conexión sea directa y no dependa de una API intermediaria como la del fabricante Toprie. Esto permitiría reducir la latencia, aumentar la confiabilidad en la recepción de datos y aprovechar las herramientas nativas de telemetría y seguridad de Azure.
- Desde el punto de vista de las ecuaciones de Bernoulli, se identificó la necesidad de completar la validación del modelo de flujo no estrangulado, el cual no pudo ser verificado empíricamente debido a las limitaciones técnicas del sensor y la imposibilidad de controlar las condiciones de presión diferencial necesarias. Incluir esta validación en futuras versiones permitiría ampliar el rango de aplicación del sistema y mejorar la precisión de las estimaciones en diferentes escenarios de operación.
- En materia de seguridad, se recomienda restringir los permisos de acceso a la base de datos, ya que actualmente los procesos de Celery y Redis operan con privilegios de superusuario. Para mitigar riesgos potenciales, deben implementarse roles y subusuarios

con permisos mínimos necesarios, aplicando políticas de privilegio restringido y autenticación por entorno.

- Finalmente, se sugiere refactorizar el pipeline de datos para incorporar técnicas de validación automática y manejo de excepciones más robusto. Esto permitirá detectar inconsistencias en los datos transmitidos por los sensores, reforzando la confiabilidad general del sistema.

En conjunto, estas mejoras contribuirán a fortalecer la escalabilidad, seguridad y precisión del sistema Andes Board, consolidando su potencial como una herramienta avanzada de monitoreo y análisis para el consumo de gas GLP en entornos residenciales e industriales.

7.2.2 Futuras líneas de investigación o desarrollo

El sistema Andes Board constituye una base sólida para el desarrollo de soluciones IoT aplicadas al monitoreo de gases licuados, sin embargo, su potencial puede expandirse significativamente a través de nuevas líneas de investigación y desarrollo tecnológico.

Una primera línea de trabajo consiste en incorporar modelos de predicción de consumo y comportamiento del gas GLP, empleando técnicas de aprendizaje automático (Machine Learning) para anticipar la duración estimada del cilindro, detectar patrones de consumo anómalos y optimizar la logística de distribución. Esto permitiría evolucionar el sistema de un modelo reactivo a uno predictivo.

Asimismo, se propone la migración gradual hacia una arquitectura basada en microservicios, en la cual cada componente (procesamiento, notificaciones, API, dashboard, etc.) opere de manera independiente. Este enfoque facilitaría la escalabilidad horizontal, el mantenimiento modular y la integración de nuevos servicios,

especialmente en entornos de alta concurrencia.

En el ámbito de comunicaciones, una línea prometedora es la integración de sensores compatibles con tecnologías de red de baja potencia (LPWAN) como LoRaWAN o NB-IoT, que permitirían ampliar el alcance del monitoreo en zonas rurales o de difícil acceso, reduciendo costos operativos y consumo energético.

Finalmente, desde el punto de vista académico, futuras investigaciones podrían enfocarse en validar experimentalmente ecuaciones de flujo másico alternativos al flujo estrangulado, o incluso explorar ecuaciones de gases reales con mayor precisión para mezclas específicas de propano y butano, mejorando así la exactitud de las estimaciones en diferentes condiciones de temperatura y presión.

En conjunto, estas líneas de desarrollo consolidan a Andes Board como una plataforma adaptable y escalable, con capacidad de evolucionar hacia un ecosistema integral de monitoreo inteligente de GLP que combine IoT, analítica avanzada y despliegue en la nube.

Capitulo VIII: Referencias

- LibreTexts. (s. f.). *8.05: Phase Diagrams*. Chemistry LibreTexts. Recuperado el 16 de septiembre de 2025, de https://chem.libretexts.org/Courses/University_of_Toronto/UTSC%3A_First-Year_Chemistry_Textbook_(Fall_2025)/08%3A_Liquids_and_Solids_and_Phase_Changes/8.05%3A_Phase_Diagrams
- Joonaki, E., et al. (2025). Thermodynamic properties of hydrogen containing systems: Ideal gas critical flow factor and calculation by C_{i}*. Lancashire Institutional Repository. Recuperado de https://knowledge.lancashire.ac.uk/id/eprint/54180/9/54180%20Joonaki%20et%20al.%20VOR.pdf
- Kološ, I. (2021). Numerical Analysis of Flow Around a Cylinder in Critical and Subcritical Regimes. Sustainability, 13(4). https://doi.org/10.3390/su13041837
- (Autor desconocido) (s.f.). Ideal Gas Processes and Work Formulas (Lecture Notes). Université M'hamed Bouguerra de Boumerdés. Recuperado de https://www.studocu.com/row/document/universite-mhamed-bouguerra-de-boumerdes/multivariable-control-systems/thermodynamics-bygyggiiggigi/34975577
- (Autor desconocido) (2017). Principles and Applications of Thermal Analysis (ed. Paul Gabbott). Recuperado de https://gateformme.files.wordpress.com/2017/04/principles-and-applications-of-thermal-analysis.pdf
- Palacios, (2011). Quantitative dispersion analysis of leakages of flammable and/or … leading to a biphasic flow. Tesis doctoral. Recuperado de https://www.tdx.cat/bitstream/handle/10803/316776/TAMS1de1.pdf?isAllowed=y&sequence=1
- (Autor desconocido) (s.f.). Two-phase flow: Applications of Gas-Liquid, Gas-Solid, Solid-Liquid flows in engineering. Recuperado de https://che.rvrjcce.ac.in/Autonomous-R16.pdf

- (Autor desconocido) (s.f.). Thermodynamics 01 November 2020. Recuperado de https://www.coursehero.com/file/90076964/Thermodynamicspdf/
- (Autor desconocido) (s.f.). The cataphoresis of suspended particles. Part I. Royal Society Publishing. https://royalsocietypublishing.org/doi/10.1098/rspa.1931.0133
- (Autor desconocido) (s.f.). BSCMSC | Physics | Mathematical Analysis. Recuperado de https://www.scribd.com/document/615993634/BSCMSC
- (Autor desconocido) (s.f.). An adaptive 3D virtual learning environment for training software developers in scrum (tema mezcla, pero contiene elementos de modelado de flujos). https://arxiv.org/abs/2111.05327
- "CCBV.co.uk". (s.f.). Coding Courses by CCBV. Recuperado de https://ccbv.co.uk
- Velasco, M. V. (2021). Revisión sistemática de la metodología Scrum para el desarrollo de software. Revista Científica Dominio de las Ciencias, 7(4), 434-447. Recuperado de https://dialnet.unirioja.es/descarga/articulo/8384028.pdf
- SÁENZ BLANCO, F. (2018). Conformación de equipos ágiles para la …. Recuperado de https://www.scielo.org.co/scielo.php?pid=S1692-85632018000200039&script=sci_arttext
- Armijos, L. M. (2024). Estudio de la adopción de metodologías ágiles XP y Scrum. Revista Espacios, 45(4). Recuperado de https://www.revistaespacios.com/a24v45n04/24450406.html
- Sellés, J. M. (2009). Métodos ágiles para el desarrollo de software. Recuperado de https://upcommons.upc.edu/bitstreams/11e14375-e421-42b4-918a-f3af75dhedownload
- Cadavid, A. N. (2013). Revisión de metodologías ágiles para el desarrollo de software. Revista Redalyc. Recuperado de https://www.redalyc.org/pdf/4962/496250736004.pdf
- Pinciroli, F., Barros Justo, J. L., Zeligueta, L., & Pma, M. (2017). Systematic Mapping Protocol – Coverage of Aspect-Oriented Methodologies for the Early Phases of the Software Development Life Cycle. arXiv. https://arxiv.org/abs/1702.02653

Anexos

Número de anexo	Descripción: archivo docker-compose.yaml	Código fuente

A-1	services:
	  andes-tech:
	  container_name: app
	  restart: always
	  build:
	  context: .
	  dockerfile: Dockerfile
	  command: sh -c "python manage.py migrate --no-input && python manage.py collectstatic --no-input && python manage.py runserver 0.0.0.0:8000"
	  ports:
	  - "8000:8000"
	  depends_on:
	  - db
	  - redis
	
	  db:
	  container_name: db
	  build:
	  context: .
	  dockerfile: ./docker/Dockerfile.postgres
	  volumes:
	  - postgres_data:/var/lib/postgresql/data
	  - ./docker/db:/docker-entrypoint-initdb.d
	  environment:
	  POSTGRES_DB: ${DB_NAME}
	  POSTGRES_USER: ${DB_USER}
	  POSTGRES_PASSWORD: ${DB_PASSWORD}
	  ports:
	  - "5432:5432"
	
	  redis:
	  image: redis:7.0.12
	  container_name: redis
	  restart: always
	  ports:	

	- "6379:6379" celery: container_name: celery-worker build: context: . restart: always environment: - DJANGO_SETTINGS_MODULE=andes.settings command: celery -A andes.celery worker --loglevel=info volumes: - .:/app depends_on: - db - redis celery-beat: container_name: celery-beat build: context: . environment: - DJANGO_SETTINGS_MODULE=andes.settings command: celery -A andes.celery beat --loglevel=info volumes: - .:/app depends_on: - db - redis volumes: postgres_data:

Número de anexo	Descripción: archivo dockerfile	Código fuente
A-2	# — Builder Stage ——— FROM python:3.11.5-slim AS builder ENV PYTHONDONTWRITEBYTECODE=1 \ PYTHONUNBUFFERED=1	

```
WORKDIR /app

# 1) Instala dependencias de sistema para Python y Node
RUN apt-get update && \
  apt-get install -y --no-install-recommends \
   build-essential libgdbm-dev libpq-dev gcc \
   curl ca-certificates gnupg && \
  rm -rf /var/lib/apt/lists/*

# 2) Instalar Node.js 18
RUN mkdir -p /etc/apt/keyrings && \
  curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-
repo.gpg.key \
   | gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg && \
  echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg]
https://deb.nodesource.com/node_18.x nodistro main" \
   > /etc/apt/sources.list.d/nodesource.list && \
  apt-get update && \
  apt-get install -y --no-install-recommends nodejs && \
  rm -rf /var/lib/apt/lists/*

# 3) Copiar y construir dependencias Python
COPY requirements.txt .
RUN pip install --upgrade pip && \
  pip wheel --no-cache-dir --wheel-dir /app/wheels -r requirements.txt

# 4) Copiar fuente y dependencias Node
COPY . .

# 5) Instalar dependencias Node y generar assets
RUN npm ci --legacy-peer-deps && \
  npm run build

# 6) Instalar Python desde las ruedas
RUN pip install --no-cache-dir --no-index --find-links=/app/wheels \
  -r requirements.txt --prefix=/install

# 7) Preparar directorios estáticos y migraciones
ENV PATH=/install/bin:$PATH \
  PYTHONPATH=/install/lib/python3.11/site-packages
```

	# — Runtime Stage —
	FROM python:3.11.5-slim
	
	ENV PYTHONDONTWRITEBYTECODE=1 \
	  PYTHONUNBUFFERED=1 \
	  DJANGO_SETTINGS_MODULE=andes.settings
	
	WORKDIR /app
	
	# 1) Instalar runtime deps
	RUN apt-get update && \
	  apt-get install -y --no-install-recommends libpq-dev && \
	  rm -rf /var/lib/apt/lists/*
	
	# 2) Copiar paquetes Python instalados
	COPY --from=builder /install /install
	ENV PATH=/install/bin:$PATH \
	  PYTHONPATH=/install/lib/python3.11/site-packages:/app
	
	# 3) Copiar la fuente completa (incluye static/dist)
	COPY --from=builder /app /app
	
	# 4) Crear STATIC_ROOT
	RUN mkdir -p /app/staticfiles
	
	EXPOSE 8000
	
	# 5) Comando runtime: migrar, collectstatic y levantar dev server
	CMD ["sh", "-c", "python manage.py migrate --no-input && python manage.py collectstatic --no-input && exec python manage.py runserver 0.0.0.0:8000"]

Número de anexo	Descripción: Fragmento de código del proceso de cálculos: archivo dockerfile	Código fuente

A-3	def calcular_masa_gas(presion, temperatura, volumen):
	  assert presion > 0, "La presión debe ser un valor positivo"
	  assert temperatura > 0, "La temperatura debe ser un valor positiva"
	  try:
	  masa = (presion * volumen) / (8.314 * temperatura)	

	return round(masa, 3) except Exception as e: print(f"Error en cálculo de masa: {e}") return None

Número de anexo	Descripción: Fragmento de código de la clase GasMath y sus docstrings	Código fuente

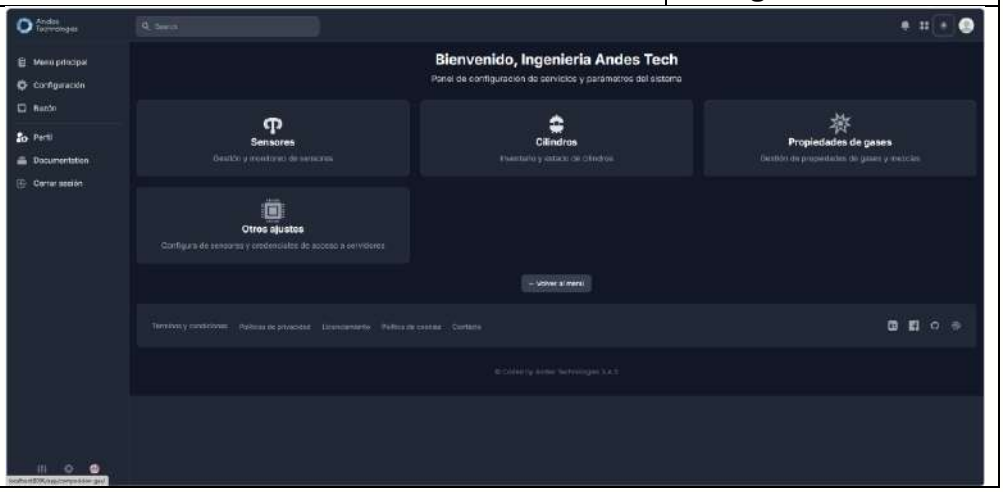
A-4	# ---------- Cálculo CHOKED ----------
	def choked(self, *, info: Dict[str, Any], ctx: Dict[str, Any]) -> Dict[str, Any]:
	"""
	mdot choked (isentrópico + Cd):
	mdot = Cd * A * p0 * sqrt(gamma/(Rspec*T0)) * ((gamma+1)/2)^(-(gamma+1)/(2*(gamma-1)))
	"""
	area = ctx.get("area_m2")
	cd = ctx["cd"]
	p0 = ctx["p0_abs_pa"]
	T0 = norm_temperature_to_K(ctx.get("T0_K"))
	gamma = ctx["gamma"]
	R_spec = ctx["R_spec"]
	
	if not area:
	raise ValueError("No se tiene área de orificio. Define diametro_orificio_m o area_orificio_m2.")
	
	factor = ((gamma + 1.0) / 2.0) ** (-(gamma + 1.0) / (2.0 * (gamma - 1.0)))
	mdot = cd * area * p0 * ((gamma / (R_spec * T0)) ** 0.5) * factor # kg/s
	
	masa_remov_kg = mdot * 60.0
	
	# masa inicial m0
	m0 = ctx.get("masa_gas_restante_kg")
	if m0 is None:
	if ctx["masa_lleno_kg"] is not None and ctx["masa_tara_kg"] is not None:
	m0 = ctx["masa_lleno_kg"] - ctx["masa_tara_kg"]
	masa_restante_kg = max((m0 or 0.0) - masa_remov_kg, 0.0) if m0 is not None else None	

	# porcentaje
	total_ini = None
	if ctx.get("masa_lleno_kg") is not None and ctx.get("masa_tara_kg") is not None:
	total_ini = ctx["masa_lleno_kg"] - ctx["masa_tara_kg"]
	porc = (masa_restante_kg / total_ini * 100.0) if (masa_restante_kg is not None and total_ini and total_ini > 0) else None
	
	return {
	**ctx,
	"T0_K": T0,
	"mdot_kg_s": mdot,
	"masa_removida_kg": masa_remov_kg,
	"masa_restante_kg": masa_restante_kg,
	"porc_masa_gas_restant": porc,
	"estado_fase": "choked",
	"metodo": "choked",
	}

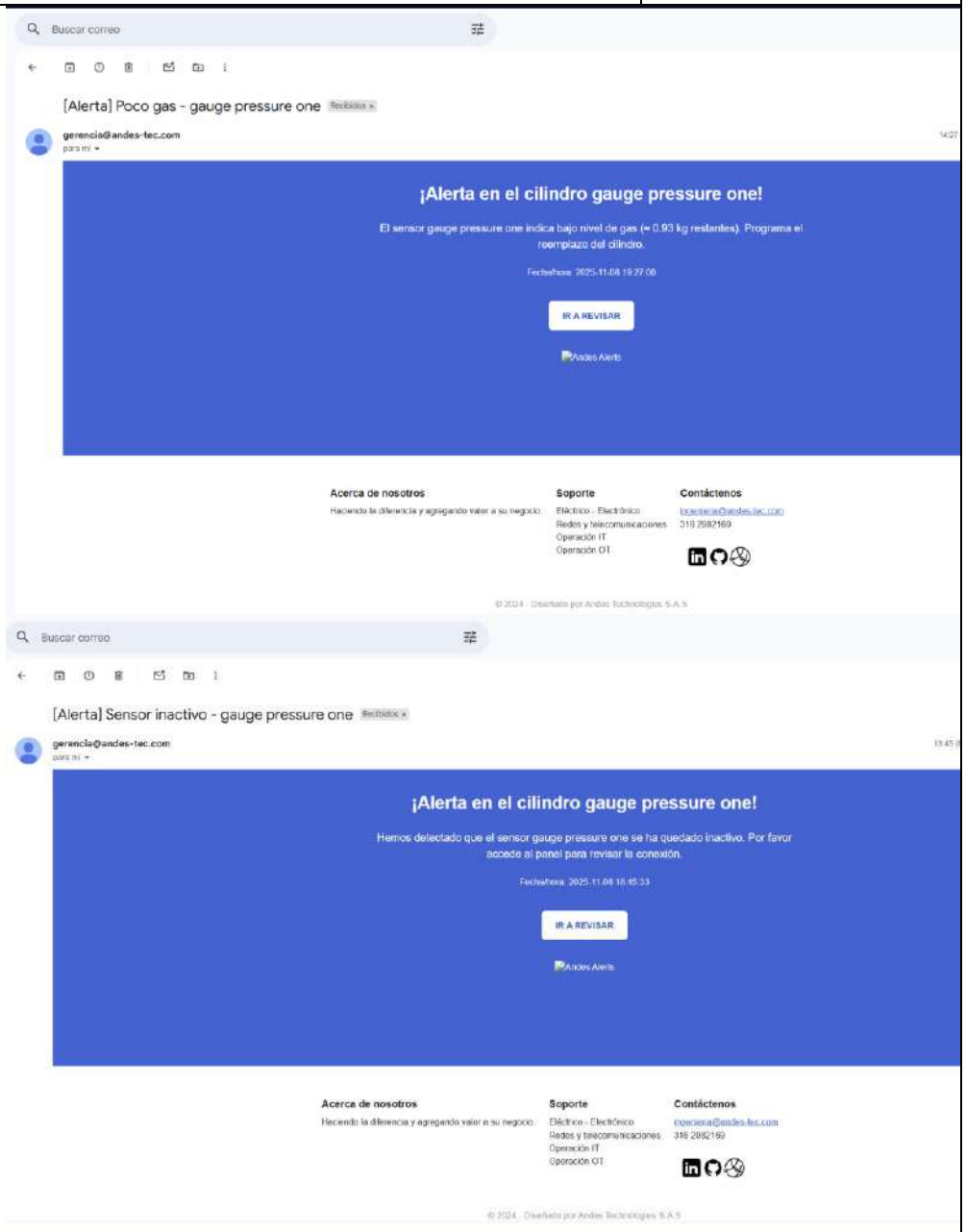
Número de anexo	Descripción: Manuales de usuario y desarrollo	Documentación
A-5	Manual de ejecución: django-dash.pdf Manual de desarrollador: manual de desarrollador.pdf Manual del usuario: manual del usuario.pdf Video de comparación de resultados: https://youtu.be/VpMUKNpnN0Y	

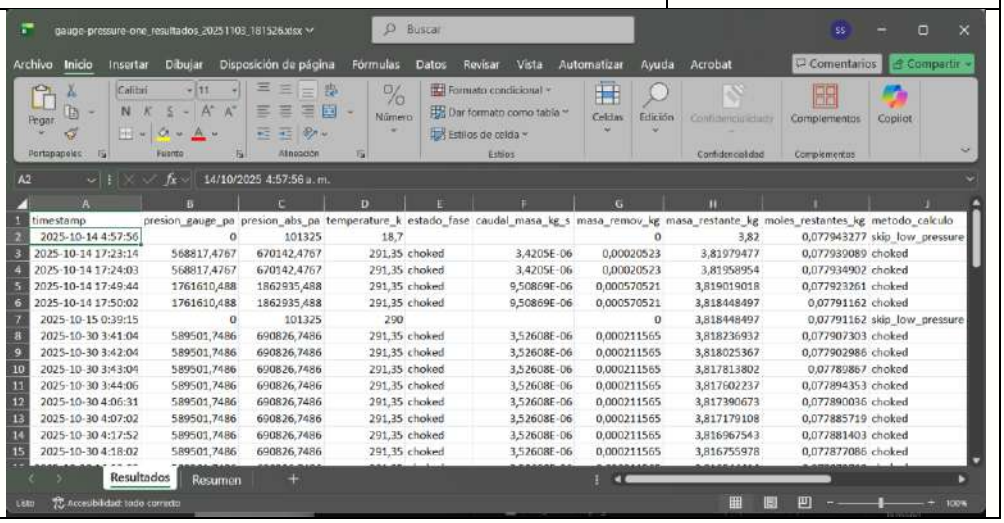
Número de anexo	Descripción: Cálculo de flujo	Imágen de cálculos
A-6	Cálculos de masa molar: densidad y masa molar glp.pdf	

	CALCULOS MATEMÁTICOS — Andes Technologies Calcula la masa de gas que sale cada segundo, considerando la presión manométrica, la temperatura, el tamaño efectivo del orificio y la eficiencia de la válvula: **Variables** $\dot{m} = C_d A p_0 \sqrt{\frac{\gamma}{R_{spec} T_0}} \left(\frac{\gamma+1}{2}\right)^{-\frac{\gamma+1}{2(\gamma-1)}}$ $\dot{m}$ – masa extraída por minuto; C_d – eficiencia hidráulica del paso; A – área del orificio $\pi * r^2$; P_0 – presión absoluta $P_{gauge} + P_{atm}$ γ = Comportamiento termodinámico $\frac{C_p}{C_v}$; R_{spec} = cte. específica del gas; T_0 = Temperatura absoluta; M = Masa molar de la mezcla **Datos** $\dot{m} = x\frac{Kg}{seg}$ $C_d = 0.96$ $A = 0.00113$ m $P_0 = 670142$ pa $\gamma = 1.13$ $R_{spec} = 169.6 \frac{J}{mol*Kg}$ $T_0 = 290$ °K $M = 0.04901 \frac{Kg}{mol}$ $\dot{m} = (1.09 \times 10^{-6}) \cdot 670{,}142 \cdot \underbrace{\left[\sqrt{\frac{1.13}{169.6 \cdot 290}} \left(\frac{1.13+1}{2}\right)^{-\frac{1.13+1}{2(1.13-1)}}\right]}_{\approx 0.00286} \approx 0.125 \text{ kg/min}$ A esa presión y temperatura, el cilindro libera ~0.125 kg de gas por minuto. Masa extraída en un intervalo de tiempo Δt $m_{extraída} = \dot{m} \cdot \Delta t$ $m_{extraída} \approx 0.125 \times 4 = 0.50$ kg

Número de anexo	Descripción: Menú de configuración del administrador.	Imágen de menú de configuración
A-7		

Número de anexo	Descripción: Correos electrónicos emitidos de alarma.	Imagen correo electrónico
A-8		



Número de anexo	Descripción: Reporte .xlsx generado por el sistema.	Imágen del reporte
A-9		

timestamp	presion_gauge_pa	presion_abs_pa	temperature_k	estado_fase	caudal_masa_kg_s	masa_remov_kg	masa_restante_kg	moles_restantes_kg	metodo_calculo
2025-10-14 4:57:56	0	101325	18,7			0	3,82	0,077943277	skip_low_pressure
2025-10-14 17:23:14	568817,4767	670142,4767	291,35	choked	3,4205E-06	0,00020523	3,81979477	0,077939089	choked
2025-10-14 17:24:03	568817,4767	670142,4767	291,35	choked	3,4205E-06	0,00020523	3,81958954	0,077934902	choked
2025-10-14 17:49:44	1761610,488	1862935,488	291,35	choked	9,50869E-06	0,000570521	3,819019018	0,077923261	choked
2025-10-14 17:50:02	1761610,488	1862935,488	291,35	choked	9,50869E-06	0,000570521	3,818448497	0,07791162	choked
2025-10-15 0:39:15	0	101325	290			0	3,818448497	0,07791162	skip_low_pressure
2025-10-30 3:41:04	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,818236932	0,077907303	choked
2025-10-30 3:42:04	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,818025367	0,077902986	choked
2025-10-30 3:43:04	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,817813802	0,07789867	choked
2025-10-30 3:44:06	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,817602237	0,077894353	choked
2025-10-30 4:06:31	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,817390673	0,077890036	choked
2025-10-30 4:07:02	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,817179108	0,077885719	choked
2025-10-30 4:17:52	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,816967543	0,077881403	choked
2025-10-30 4:18:02	589501,7486	690826,7486	291,35	choked	3,52608E-06	0,000211565	3,816755978	0,077877086	choked

Número de anexo	Descripción: Tablero de Trello.	Documentación de metodología
A-10	Acceso de Trello: https://trello.com/invite/b/68f1c3fa2e7728102b07b6cd/ATTIbfffe83c37a84f1aff2d17dd243444c6ABF2D9BD/mi-tablero-de-trello	

Número de anexo	Descripción: Opciones de sensores sin la capacidad de enviar datos al sistema.	Documentación de metodología

A-11	

Andes Technologies

Cotizaciones y opciones Mantenimiento preventivo y nivel de gas

Fecha: 09 octubre 2024

Versión: 0.1

Opciones de sensor para medir el nivel de gas

Estos son los proveedores que sí respondieron

Sensores ultrasónicos

Empresa: USA

Precio: Pago único por $ 274.000 COP

Página web externa que monitorea los sensores.
App para Android y IOS.
No ofrecen API con la que podamos extraer los datos del sensor.

Guía técnica: https://acortar.link/dy89lV

Empresa: China

Precio: Pago único por $ 126.800 COP

Página web externa que monitorea los sensores.
App para Android y IOS.
Ofrecen API con la que podamos extraer los datos del sensor.

- Su plataforma y API están en mantenimiento.

Guía técnica: https://acortar.link/aVvAg3

Sensores superpuestos sobre el medidor de nivel Rochester

El sensor de Hiteks

Empresa: italiana – Proveedor: Mediciones y servicios asociados
Precio: 1'104.564 COP | 240 EUR
Página web externa que monitorea los sensores.
App para Android y IOS.
Ofrecen API con la que podamos extraer los datos del sensor.

Guía técnica: https://acortar.link/EVdoEG

A partir de aquí, encontrará otros sensores que miden el nivel de gas, los proveedores no me han respondido.

Número de anexo	Descripción: Tablero de Trello.	Documentación de metodología
A-12	Acceso de Trello: https://trello.com/invite/b/68f1c3fa2e7728102b07b6cd/ATTIbfffe83c37a84f1aff2d17dd243444c6ABF2D9BD/mi-tablero-de-trello	

Número de anexo	Descripción: Historia de usuario 1	Documentación de metodología
A-13	ADMINISTRADOR: UH 12860 – GESTIÓN DE EMPRESAS Yo como administrador del área/proceso de gestión de usuarios quiero gestionar los usuarios de tipo "empresa" (crear, ver, actualizar, eliminar). Para mantener un control de acceso seguro y eficiente para las empresas que utilizan el sistema. CRITERIOS DE ACEPTACIÓN • El sistema debe mostrar una vista con la lista completa de todas las empresas registradas. • Debo poder crear nuevas empresas, modificarlas y eliminarlas. • La creación de una empresa debe registrarse exitosamente en la base de datos. PASOS: ¿COMO LO QUIERO? • A partir de un menú principal, quiero seleccionar la opción “empresas” • Se genera una ventana emergente que contiene una lista con todas las empresas, allí necesito un botón que me permita crear una empresa. • Se generará un formulario que me solicite los siguientes campos: Nombre Tipo de documento Numero de documento Estado Cantidad operarios Dirección Mail Contraseña • Al guardar el formulario, el sistema me redirige a la lista de empresas. • Cada empresa en la lista debe tener opciones para "Actualizar" y "Eliminar". SUPUESTOS • El administrador ha iniciado sesión previamente en el sistema. • Un administrador existe únicamente en la empresa "Andes Technologies".	

	ELEMENTOS DE ACCIÓN • Desarrollar el sistema de autenticación y redireccionamiento de usuarios a sus módulos correspondientes, incluyendo la opción "Empresas" en el menú principal. • Implementar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la gestión de empresas, con conexión a la base de datos. • Desarrollar un sistema de registro (logs) para rastrear las acciones realizadas en la gestión de empresas.

Número de anexo	Descripción: Historia de usuario 2	Documentación de metodología
A-14	**EMPRESA: UH 12860 –GESTIÓN DE OPERADORES** Yo como empresa del proceso de gestión de usuarios necesito gestionar los usuarios de tipo "operador" (crear, ver, actualizar, eliminar) para controlar el acceso y la gestión de la información específica de la empresa por parte de sus operadores **CRITERIOS DE ACEPTACIÓN** • El sistema debe mostrar una vista con la lista completa de los operadores de la empresa. • Debo poder crear nuevos operadores, modificarlos y eliminarlos. • La creación de un operador debe registrarse exitosamente en la base de datos. **PASOS: ¿COMO LO QUIERO?** • A partir de un menú principal, quiero seleccionar la opción “empleados” • Se genera una ventana emergente que contiene una lista con todos los empleados, allí necesito un botón que me permita crear o actualizar un operador. • Se generará un formulario que me solicite o muestre los siguientes campos: Nombre Tipo de documento Numero de documento Estado Mail	

	Contraseña • Al hacer clic en el botón guardar, debe redirigirme a la lista de operadores. • La lista debe tener la opción de borrar **SUPUESTOS** • El usuario de la empresa ha iniciado sesión previamente en el sistema. • Cada empresa es responsable de gestionar sus propios operadores. **ELEMENTOS DE ACCIÓN** • Desarrollar el sistema de autenticación y redireccionamiento de usuarios a sus módulos correspondientes, incluyendo la opción "Operadores" en el menú principal. • Implementar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para la gestión de operadores, con conexión a la base de datos. • Desarrollar un sistema de registro (logs) para rastrear las acciones realizadas en la gestión de operadores.

Número de anexo	Descripción: Historia de usuario 3	Documentación de metodología
A-15	**ADMINISTRADOR: UH 12860 –GESTIÓN DE SENSORES** Yo como administrador del área/proceso de gestión de sensores quiero gestionar los sensores TP2401 y realizar pruebas de conexión entre ellos y el sistema para asegurar la operatividad y el correcto funcionamiento de los sensores para la recolección de datos. **CRITERIOS DE ACEPTACIÓN** • El sistema debe mostrar una vista con la lista completa de todos los sensores registrados. • Debo poder crear nuevos sensores, modificarlos y eliminarlos. • El sistema debe permitirme validar la conexión de un sensor y mostrar su estado. **PASOS: ¿COMO LO QUIERO?** • A partir de un menú principal, quiero seleccionar la opción “sensores”	

	• Se genera una ventana emergente que contiene una lista con todos los sensores, allí necesito un botón que me permita crear o actualizar un sensor. • Se generará un formulario que me solicite o muestre los siguientes campos: Nombre imei Numero serial • Cuando la conexión haya sido establecida, se llenarán los siguientes campos: Estado Localización Bateria Señal Ultima actualización Id de empresa. • Al hacer clic en el botón guardar, debe redirigirme a la lista de operadores. • La lista debe tener la opción que permita validar la conexión entre los sensores y el sistema **SUPUESTOS** • El administrador ha iniciado sesión previamente en el sistema. • El sensor debe estar encendido y tener conexión a internet. • El sensor debe poder conectarse a un servidor propio (vía MQTT o API local). **ELEMENTOS DE ACCIÓN** • Desarrollar el sistema de autenticación y redireccionamiento de usuarios a sus módulos, incluyendo la opción "Sensores" en el menú principal. • Implementar las operaciones CRUD para la gestión de sensores, con conexión a la base de datos. • Desarrollar un sistema de logs para el seguimiento de acciones en la gestión de sensores. • Implementar la funcionalidad para realizar tests de conexión con los sensores.

Número de anexo	Descripción: Historia de usuario 4	Documentación de metodología
A-16	OPERADOR: UH 12860 – ASIGNACIÓN DE SENSORES DE USO DOMÉSTICO Yo como operador del área/proceso de gestión de sensores quiero realizar la configuración inicial de los sensores para su uso en cilindros residenciales/domésticos para permitir al sistema recibir y procesar correctamente la información del sensor para calcular la cantidad de gas en el cilindro. CRITERIOS DE ACEPTACIÓN • El sistema debe mostrar una vista con la lista de todos los sensores registrados que pertenecen a mi empresa. • Debo poder asociar un sensor a un único cilindro de uso doméstico. • El sistema debe proporcionar un formulario para la configuración inicial del sensor. PASOS: ¿COMO LO QUIERO? • A partir de un menú principal, quiero seleccionar la opción “capacidad actual” • Se generará un formulario que me solicite los siguientes campos que aparecen en la etiqueta del cilindro de gas o Peso del cilindro vacío – etiquieta (kg) o Peso máximo de cilindro – etiquieta (kg) o Peso actual del cilindro – báscula (kg) o Máxima presión de salida de gas – etiqueta (psi) • Al hacer clic en el botón guardar, debe redirigirme al dashboard principal. • Cuando la conexión haya sido establecida, se llenarán los siguientes campos: Fecha | hora presión temperatura Actual porcentaje de gas Cantidad actual de litros Densidad Otros… • La lista debe tener la opción que permita validar la conexión entre los sensores y el sistema	

	SUPUESTOS • El operador ha iniciado sesión previamente en el sistema. • El sensor debe estar encendido y tener conexión a internet. • El sensor debe tener una conexión a un servidor propio (vía MQTT o API local). • El sensor debe estar conectado al cilindro doméstico mediante una válvula. • La configuración inicial correcta del sensor es crucial para la precisión de los cálculos. . El sistema debe recibir satisfactoriamente la información del sensor.

Número de anexo	Descripción: Historia de usuario 5	Documentación de metodología
A-17	**OPERADOR: UH 12860 – RASTREAR LA POSICIÓN DEL SENSOR** Yo como operador del área/proceso de gestión de monitoreo necesito visualizar en un mapa la ubicación del cilindro o tanque geoestacionario seleccionado para conocer la ubicación en tiempo real del cilindro y del sensor. **CRITERIOS DE ACEPTACIÓN** • El sistema debe mostrar un mapa con la ubicación precisa del sensor. **PASOS: ¿COMO LO QUIERO?** • A partir de un menú principal, quiero seleccionar la opción “monitorear” • Se generará una lista que contiene los sensores de la sede en específico. • Debe seleccionar el cilindro que desea monitorear. • Cargará el dashboard que mostrará los datos ya calculados del sensor. • En la interfaz del dashboard, se hará visible un botón que indique “rastrear” • Cargará la vista de un mapa que muestra la ubicación del cilindro | sensor. **SUPUESTOS** • El operador ha iniciado sesión previamente en el sistema.	

	• El sensor debe estar encendido y tener conexión a internet. • El sensor debe tener una conexión a un servidor propio (vía MQTT o API local). • El sensor debe estar conectado al tanque geoestacionario o cilindro de gas mediante una válvula. • El pipeline de datos debe estar funcionando y activo. • El sistema recibe la información del sensor de forma satisfactoria. • El botón de posición solo se activa cuando un sensor está seleccionado. **ELEMENTOS DE ACCIÓN** • Desarrollar un sistema de autenticación y redireccionamiento, incluyendo la opción "Monitorear" en el menú principal. • Implementar un módulo de comunicación que soporte API (para entornos locales) y MQTT (para entornos de producción). • Desarrollar un sistema de logs para el seguimiento de acciones. • Integrar un componente de mapeo que visualice la geolocalización de los sensores.

Número de anexo	Descripción: Historia de usuario 6	Documentación de metodología
A-18	**USUARIO GENERAL: UH 12860 – VISUALIZAR DASHBOARD** Yo como usuario general del área/proceso de gestión de monitoreo necesito visualizar en un dashboard datos como la cantidad de gas en el cilindro seleccionado para cumplir el objetivo principal del monitoreo, que es calcular y mostrar las variables clave de los cilindros o tanques geoestacionarios. **CRITERIOS DE ACEPTACIÓN** ○ El sistema debe mostrar un dashboard con la información calculada enviada por el sensor. ○ La información en el dashboard debe corresponder a un único sensor y cilindro o tanque a la vez. **PASOS: ¿COMO LO QUIERO?** • Desde el menú principal, selecciono la opción "Monitorear". • Se abre una ventana emergente donde elijo entre "Cilindros Domésticos" o "Tanques Geoestacionarios".	

	• Se muestra una lista de sensores del tipo seleccionado, pertenecientes a la sede actual. • Selecciono el cilindro o tanque que deseo monitorear. • El dashboard cargará y mostrará los datos ya calculados del sensor seleccionado. **SUPUESTOS** • El usuario ha iniciado sesión previamente en el sistema. • El sensor debe estar encendido y tener conexión a internet. • El sensor debe tener una conexión a un servidor propio (vía MQTT o API local). • El sensor debe estar conectado al tanque geoestacionario o cilindro de gas mediante una válvula. • El pipeline de datos debe estar funcionando y activo. • El sistema recibe y transforma satisfactoriamente la información del sensor. **ELEMENTOS DE ACCIÓN** • Desarrollar un sistema de autenticación y redireccionamiento, incluyendo la opción "Monitorear" en el menú principal. • Implementar un módulo de comunicación que soporte API (para entornos locales) y MQTT (para entornos de producción). • Desarrollar un sistema de logs para el seguimiento de acciones. • Desarrollar el pipeline que procesa la información de los sensores mediante ecuaciones de Bernoulli. • Diseñar e implementar la interfaz del dashboard para la visualización de datos.

Número de anexo	Descripción: Historia de usuario 7	Documentación de metodología
A-19	**OPERADOR: UH 12860 – GENERAR REPORTE** Yo como operador del área/proceso de gestión de monitoreo necesito generar un reporte de un cilindro o tanque geoestacionario específico para realizar análisis históricos de los datos obtenidos del sensor y calculados por el pipeline. **CRITERIOS DE ACEPTACIÓN**	

	• El sistema debe permitir descargar un reporte en formato .xlsx que contenga los datos de un único sensor. • El reporte debe incluir los datos calculados por el pipeline. **PASOS: ¿COMO LO QUIERO?** • A partir de un menú principal, quiero seleccionar la opción “monitorear” • Se genera una ventana emergente que contiene 2 opciones, “cilindros domésticos” y “tanques geoestacionarios”, donde debe seleccionarse la opción que desea monitorear. • Se generará una lista que contiene los sensores de la sede de ese tipo en específico. • Debe seleccionar el cilindro que desea monitorear. • Cargará el dashboard que mostrará los datos ya calculados del sensor. • En la interfaz del dashboard, habrá un botón que indique “Reporte completo” • Se iniciará la descarga de un archivo .xlsx **SUPUESTOS** • El operador ha iniciado sesión previamente en el sistema. • El sensor debe estar encendido y tener conexión a internet. • El sensor debe tener una conexión a un servidor propio (vía MQTT o API local). • El sensor debe estar conectado al tanque geoestacionario o cilindro de gas mediante una válvula. • El pipeline de datos debe estar funcionando y activo. • El sistema recibe la información del sensor de forma satisfactoria. **ELEMENTOS DE ACCIÓN** • Desarrollar un sistema de autenticación y redireccionamiento, incluyendo la opción "Monitorear" en el menú principal. • Implementar un módulo de comunicación que soporte API (para entornos locales) y MQTT (para entornos de producción). • Desarrollar un sistema de logs para el seguimiento de acciones. • Desarrollar la funcionalidad de exportación de datos a formato .xlsx con los datos del pipeline.

Número de anexo	Descripción: Historia de usuario 8	Documentación de metodología
A-20	EMPRESA: UH 12860 – SOLICITUD DE SOPORTE TÉCNICO Yo como empresa del área/proceso de soporte técnico necesito un sistema de gestión de tiquetes para registrar peticiones, quejas, reclamos y sugerencias (PQRS) para reportar posibles fallas, enviar recomendaciones y solicitar soporte técnico relacionado con el sistema. CRITERIOS DE ACEPTACIÓN • El sistema debe permitirme registrar nuevas PQRS. • Debo poder ver una lista de todas las PQRS que he realizado. • Las PQRS con estado "Cerrada" y con más de 30 días de antigüedad no deben aparecer en mi vista. PASOS: ¿COMO LO QUIERO? • A partir de un menú principal, quiero seleccionar la opción “Soporte” • Se genera una ventana emergente que contiene una lista con todas mis anteriores peticiones, para solicitar soporte técnico, debe hacer clic en “crear” • Se generará un formulario que me solicite o muestre los siguientes campos: Tipo de petición Asunto Nivel de importancia Estado Texto Archivos adjuntos Id de empleado. • Al hacer clic en el botón guardar, debe redirigirme a la lista de peticiones. SUPUESTOS • El usuario debió haber iniciado sesión previamente. ELEMENTOS DE ACCIÓN • Desarrollar un sistema de autenticación y redireccionamiento, incluyendo la opción "Soporte" en el menú principal. • Implementar las operaciones CRUD para la gestión de solicitudes de soporte con conexión a la base de datos.	

	• Desarrollar un sistema de logs para el seguimiento de las acciones realizadas en el módulo de soporte.

Número de anexo	Descripción: Historia de usuario 9	Documentación de metodología
A-21	ADMINISTRADOR: UH 12860 – SOPORTE TÉCNICO Yo como administrador del área/proceso de soporte técnico quiero hacer seguimiento al estado de las solicitudes de soporte (PQRS) y actualizar su progreso para informar a las empresas sobre el avance y la resolución de sus solicitudes de soporte técnico. CRITERIOS DE ACEPTACIÓN • El sistema debe mostrar una vista de las PQRS que aún no han sido resueltas. • El sistema debe mostrar una vista de todas las PQRS que han sido resueltas. • Las PQRS con estado "Finalizada" y con más de 30 días de antigüedad no deben aparecer en las vistas de seguimiento. PASOS: ¿COMO LO QUIERO? • A partir de un menú principal, quiero seleccionar la opción “Soporte” • Se genera una ventana emergente que contiene una lista con todas las solicitudes de soporte técnico realizadas. • Al seleccionar una solicitud, se cargarán los datos que haya insertado el usuario. Tipo de petición Asunto Nivel de importancia Estado Texto Archivos adjuntos Id de empleado. • El administrador podrá cambiar entre los siguientes estados: ○ En proceso. ○ Finalizada. SUPUESTOS	

	Listado de supuestos que se asumen como ciertos/reales para poder lograr el requerimiento. • El usuario debió haber iniciado sesión previamente. ELEMENTOS DE ACCIÓN Aquellas acciones técnicas que deben ser ejecutadas para solucionar la necesidad • Desarrollar un sistema de autenticación y redireccionamiento, incluyendo la opción "Soporte" en el menú principal. • Implementar las operaciones CRUD para la gestión de solicitudes de soporte con conexión a la base de datos.

Número de anexo	Descripción: Historia de usuario 10	Documentación de metodología
A-22	EMPRESA: UH 12860 – SOPORTE TÉCNICO Yo como empresa del área/proceso de gestión de monitoreo necesito recibir notificaciones automáticas de alarmas a través de correo electrónico para estar informado sobre cilindros con menos del 20% en su capacidad de glp o fuera de los parámetros normales en los cilindros/tanques y poder tomar acciones correctivas a tiempo. CRITERIOS DE ACEPTACIÓN • El sistema debe generar una alarma y una notificación cuando el porcentaje de gas de un cilindro descienda por debajo del 20%. • El sistema debe generar una alarma y una notificación cuando la presión de un tanque geoestacionario exceda los límites de seguridad que tiene en su etiqueta. • Debo poder ver una lista de todas las alarmas activas y su nivel de severidad. • Las alarmas resueltas deben poder marcarse como cerradas y archivarse. PASOS: ¿COMO LO QUIERO? • A partir de un menú principal, quiero seleccionar la opción “monitorear” • Se genera una ventana emergente que contiene 2 opciones, “cilindros domésticos” y “tanques geoestacionarios”, donde debe seleccionarse la opción que desea monitorear.	

	• Se generará una lista que contiene los sensores de la sede de ese tipo en específico. • Debe seleccionar el cilindro que desea monitorear. • Cargará el dashboard que mostrará los datos ya calculados del sensor. • En la interfaz del dashboard, habrá un apartado que indique “alarmas” • Se mostrará una lista con todas las alarmas que tiene activas. • Las alarmas que ya hayan sido procesadas puede seleccionar “eliminar” **SUPUESTOS** • Los parámetros para el disparo de cada tipo de alarma están previamente configurados en el sistema. • El sistema de notificación SMTP para emails está configurado y funcionando con las credenciales de andes technologies. • El sensor envía datos de forma continua y confiable. • El pipeline de datos está funcionando y procesando la información del sensor para detectar las condiciones de alarma. **ELEMENTOS DE ACCIÓN** • Desarrollar la lógica de negocio para la detección de condiciones de alarma basadas en los datos de los sensores. • Implementar un módulo de notificaciones para enviar alertas (ej. email, push notifications). • Desarrollar la interfaz de usuario para la visualización y gestión de alarmas.

Número de anexo	Descripción: Sensor HELM H200SE	Documentación de hardware

A-23

Andes Technologies

SENSORES DE PRESIÓN

Hay 2 sensores de presión con el certificado anti explosiones CNEX más alto

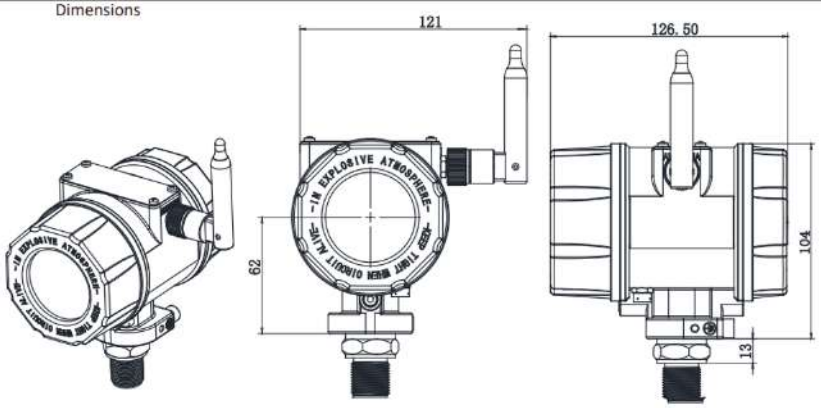
- **HM200 A – 17G – 1 - S3 - F0 - C1 - D** (Bateria industrial, 3.7V/38Ah - 2 años si enviamos datos cada 5 mins.)
- **HM200 B – 17G – 1 - S3 - F0 - C1 - D** (Bateria recargable, 3.7V/20Ah 1 año si enviamos datos cada 5 mins.)

Sensor	Precio
HM200 A – 17G – 1 - S3 - F0 - C1 - D	218.12 USD
HM200 B – 17G – 1 - S3 - F0 - C1 - D	245.12 USD

Nota: No tienen un distribuidor official en Colombia. Y recomiendan contratar una empresa de logistica para importer el sensor

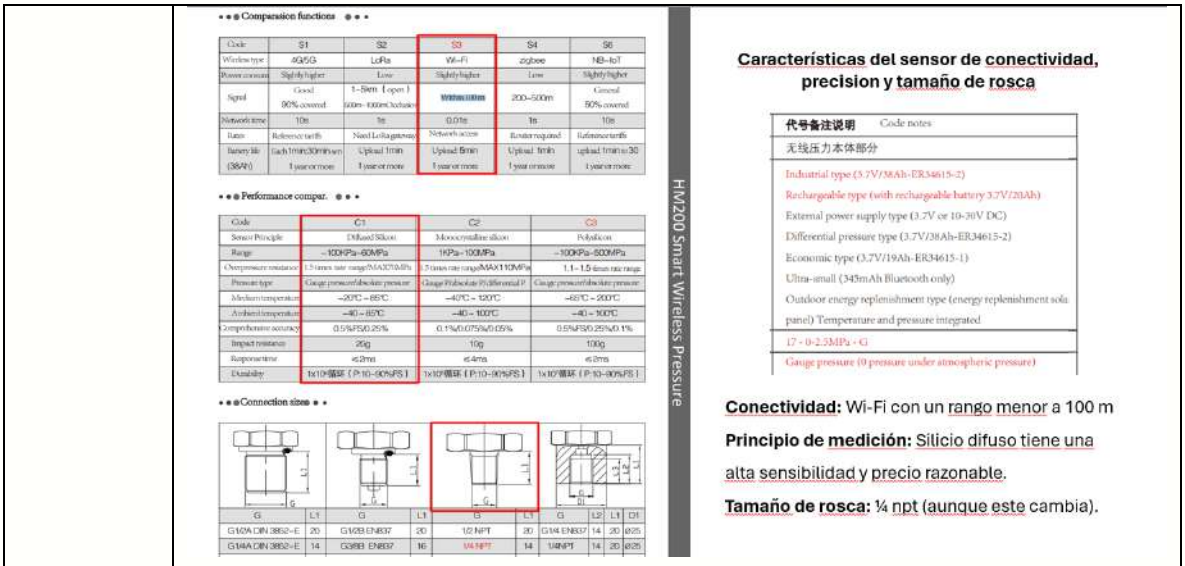
南京宏沐科技有限公司
Nanjing Hong mu Technology Co., Ltd

Dimensions



Unos pocos cm más grande que el sensor de presión amarillo que tenemos

Video: https://detail.1688.com/offer/828703187725.html?spm=a2615.2177701.shopprom.4.13ef2185cwW0MM



• • ● Comparsion functions ● • •

Code	S1	S2	S3	S4	S6
Wireless type	4G/5G	LoRa	Wi-Fi	zigbee	NB-IoT
Power consum	Slightly higher	Low	Slightly higher	Low	Slightly higher
Signal	Good 90% covered	1~5km (open) 500m~800m(obstacle)	[illegible]	200~500m	General 80% covered
Network time	10s	1s	0.01s	1s	10s
Rates	Reference tariffs	Need LoRa gateway	Network access	Router required	Reference tariffs
Battery life (3.6Ah)	Each 1min/30min/1h/s/n 1 year or more	Upload 1min 1 year or more	Upload 5min 1 year or more	Upload 1min 1 year or more	upload 1min to 30 1 year or more

• • ● Performance compar. ● • •

Code	C1	C2	C3
Sensor Principle	Diffused Silicon	Monocrystalline silicon	Polysilicon
Range	~100KPa~60MPa	1KPa~100MPa	~100KPa~500MPa
Overpressure resistance	1.5 times rate range(MAX210MPa)	1.5 times rate range(MAX110MPa)	1.1~1.5 times rate range
Pressure type	Gauge pressure/Absolute pressure	Gauge P/Absolute P/Differential P	Gauge pressure/Absolute pressure
Medium temperature	-20℃~85℃	-40℃~120℃	-65℃~200℃
Ambient temperature	-40~85℃	-40~100℃	-40~100℃
Comprehensive accuracy	0.5%FS/0.25%	0.1%/0.075%/0.05%	0.5%FS/0.25%/0.1%
Impact resistance	20g	10g	100g
Response time	≤2ms	≤4ms	≤8ms
Durability	1x10⁷循环（P:10~90%FS）	1x10⁷循环（P:10~90%FS）	1x10⁷循环（P:10~90%FS）

• • ● Connection sizes ● • •

G	L1	G	L1	G	L1	G	L2	L1	D1
G1/2A DIN 3852-E	20	G1/2B EN837	20	1/2 NPT	20	G1/4 EN837	14	20	Ø25
G1/4A DIN 3852-E	14	G3/8B EN837	16	1/4 NPT	14	1/4NPT	14	20	Ø25

HM200 Smart Wireless Pressure

Características del sensor de conectividad, precision y tamaño de rosca

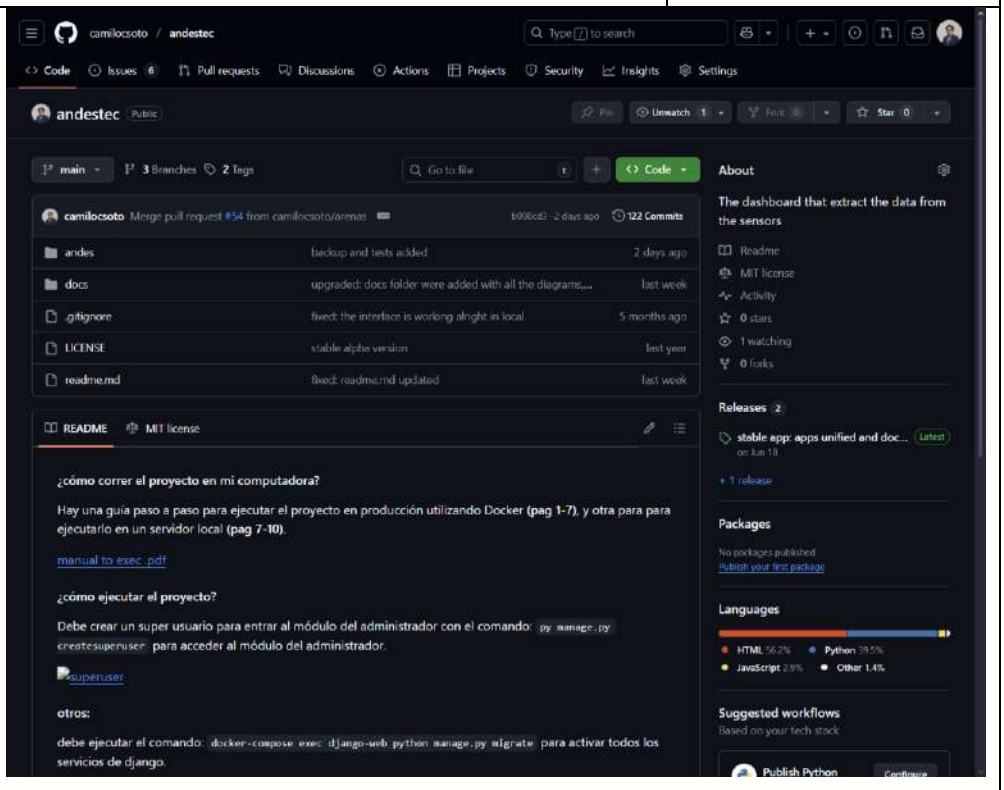
代号备注说明 Code notes
无线压力本体部分
Industrial type (3.7V/38Ah-ER34615-2)
Rechargeable type (with rechargeable battery 3.7V/20Ah)
External power supply type (3.7V or 10-30V DC)
Differential pressure type (3.7V/38Ah-ER34615-2)
Economic type (3.7V/19Ah-ER34615-1)
Ultra-small (345mAh Bluetooth only)
Outdoor energy replenishment type (energy replenishment solar panel) Temperature and pressure integrated
17 - 0-2.5MPa - G
Gauge pressure (0 pressure under atmospheric pressure)

Conectividad: Wi-Fi con un rango menor a 100 m

Principio de medición: Silicio difuso tiene una alta sensibilidad y precio razonable.

Tamaño de rosca: ¼ npt (aunque este cambia).

Número de anexo	Descripción: Git y GitHub del sistema	Documentación de hardware
A-24		



camilocsoto / andestec

Type / to search

Code | Issues 6 | Pull requests | Discussions | Actions | Projects | Security | Insights | Settings

andestec Public

Unwatch 1 | Fork 0 | Star 0

main | 3 Branches | 2 Tags | Go to file t | Code

camilocsoto Merge pull request #54 from camilocsoto/arenas 1098ed3 · 2 days ago · 122 Commits

andes	backup and tests added	2 days ago
docs	upgraded: docs folder were added with all the diagrams,...	last week
.gitignore	fixed: the interface is working alright in local	5 months ago
LICENSE	stable alpha version	last year
readme.md	fixed: readme.md updated	last week

README | MIT license

¿cómo correr el proyecto en mi computadora?

Hay una guia paso a paso para ejecutar el proyecto en producción utilizando Docker **(pag 1-7)**, y otra para para ejecutarlo en un servidor local **(pag 7-10)**.

manual to exec .pdf

¿cómo ejecutar el proyecto?

Debe crear un super usuario para entrar al módulo del administrador con el comando: `py manage.py createsuperuser` para acceder al módulo del administrador.

superuser

otros:

debe ejecutar el comando: `docker-compose exec django-web python manage.py migrate` para activar todos los servicios de django.

About

The dashboard that extract the data from the sensors

Readme
MIT license
Activity
0 stars
1 watching
0 forks

Releases 2

stable app: apps unified and doc... Latest
on Jun 18

+ 1 release

Packages

No packages published
Publish your first package

Languages

HTML 56.2% | Python 39.5% | JavaScript 2.8% | Other 1.4%

Suggested workflows

Based on your tech stack

Publish Python | Configure