Desarrollo y despliegue en la nube de backend y aplicación administradora para plataforma configurable de dispositivos de telemetría.

Fecha 3 de diciembre de 2025
Memorista Felipe Márquez Millán
Profesor Guía Marcos Zuñiga

PROGRAMA DE MEMORIAS MULTIDISCIPLINARIAS

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Desarrollo y despliegue en la nube de backend y aplicación administradora para plataforma configurable de dispositivos de telemetría.

Nombre del candidato(a): Felipe Márquez Millán

Carrera / Grado: Ingeniería Civil Telemática

Campus: Casa Central Valparaiso ; **Departamento**: **Electrónica**

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, **Marcos David Zúñiga Barraza**, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 02/12/2025 ; **Firma:** _____

Estudiante o Candidato(a):

Fecha: 02/12/2025 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por
estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Índice

Introducción

1 . Contexto Problemática

WOLKE es una empresa especializada en telemetría, dedicada a desarrollar sensores personalizados para el monitoreo en tiempo real, adaptados a las necesidades específicas de cada cliente. Estos sensores permiten a la empresa ofrecer soluciones avanzadas en diversos sectores, garantizando un control preciso y eficiente de diferentes variables clave en entornos industriales, agrícolas, energéticos, entre otros.

Actualmente, la empresa cuenta con un conjunto de herramientas de software propio diseñadas para la configuración de estos sensores. Cada herramienta ha sido desarrollada para trabajar con un firmware específico, lo que asegura que el dispositivo funcione correctamente en su entorno operativo. Sin embargo, este enfoque conlleva una serie de desafíos.

El principal inconveniente radica en que el proceso de configuración de los sensores es lento y poco eficiente. Debido a la gran cantidad de sensores desplegados, muchos de los cuales se encuentran en ubicaciones remotas y de difícil acceso, la necesidad de reconfigurarlos frecuentemente complica aún más el proceso. Además, las herramientas de software actuales solo permiten realizar configuraciones utilizando una computadora o laptop conectada vía USB al sensor, lo que resulta en una gran dependencia de equipo especializado y en la dificultad de realizar ajustes en tiempo real.

Este contexto genera ineficiencias operativas, retrasa la capacidad de respuesta ante eventos y aumenta los costos, debido a la necesidad de desplazamiento por parte de la empresa y la complejidad en la gestión de sensores a gran escala. Por lo tanto, surge la necesidad de implementar una solución más eficiente, que permita la configuración remota de los dispositivos y facilite la administración de los sensores de forma más ágil y adaptable.

En el contexto de esta memoria, el enfoque estuvo en el desarrollo de la API, la base de datos, la plataforma web administrativa y el despliegue de la solución en un servidor de Azure, garantizando la integración eficiente de los distintos componentes del sistema. Esto permitió que las configuraciones de los sensores se gestionaran de forma centralizada, mejorando los procesos operativos y reduciendo los tiempos de respuesta.

2 . Problema a Resolver

El sistema actual que utiliza WOLKE para la configuración de sus sensores presenta diversas ineficiencias que generan pérdidas significativas de tiempo y costos adicionales para la empresa. Uno de los principales problemas es la fragmentación del software: para cada tipo de sensor que se fabrica, existe una herramienta de configuración específica. Esto significa que, cada vez que se crea un nuevo sensor, es necesario desarrollar un nuevo software personalizado, lo que no solo incrementa la complejidad, sino que también exige una inversión considerable de tiempo y recursos en su desarrollo y mantenimiento.

Además, este enfoque basado en herramientas específicas conlleva un desafío logístico importante. Muchos de los sensores fabricados por WOLKE están ubicados en lugares remotos o de difícil acceso, lo que

complica aún más el proceso de configuración y reconfiguración. Dado que el sistema actual requiere que los técnicos conecten físicamente los sensores a un ordenador a través de un cable USB, la tarea se vuelve no solo incómoda, sino en muchos casos impracticable. Desplazar personal a estos lugares aumenta los costos operativos, y en algunos entornos, las condiciones pueden incluso dificultar o impedir la conexión física necesaria.

Estos factores combinados hacen que el proceso de configuración sea ineficiente, afectando la productividad y limitando la escalabilidad de la operación. La falta de un sistema centralizado y flexible que permita realizar configuraciones de forma remota y sin depender de una conexión física con cada sensor representa un claro obstáculo para la optimización de las operaciones y para la expansión de WOLKE en el ámbito de la telemetría personalizada.

3 . Acercamiento a la solución

Para abordar los problemas actuales en el proceso de configuración de los sensores de WOLKE, se propone el desarrollo de una aplicación móvil diseñada específicamente para optimizar y facilitar este procedimiento. La solución propuesta integra diversas tecnologías y características clave que permiten mejorar tanto la eficiencia como la flexibilidad del sistema, asegurando un acceso más sencillo y una configuración más rápida. A continuación, detallamos los principales componentes de la solución:

- **Configuración Inalámbrica:** Uno de los principales problemas del sistema actual es la necesidad de conectar los sensores mediante USB. Para superar esta limitación, se implementaron tecnologías de conectividad inalámbrica, específicamente *Bluetooth Low Energy* (BLE), que permite a los usuarios establecer una conexión eficiente y de bajo consumo energético entre los sensores y la aplicación móvil. Esta tecnología es ideal para dispositivos que requieren un uso prolongado de la batería y, al mismo tiempo, ofrece un alcance adecuado para configuraciones en campo. BLE facilita la conexión remota, eliminando la necesidad de cables y permitiendo una configuración más cómoda y accesible, incluso en ubicaciones de difícil acceso.
- **Interfaz Unificada:** El actual enfoque de tener un software específico para cada tipo de sensor resulta ineficiente. Para resolver este problema, se desarrolló una interfaz de usuario unificada y altamente intuitiva, que permita gestionar y configurar cualquier tipo de sensor de manera sencilla, sin necesidad de desarrollar un software exclusivo para cada uno. La interfaz se diseñará con un enfoque modular, lo que significa que es fácilmente adaptable a nuevos tipos de sensores, reduciendo significativamente los costos y tiempos asociados al desarrollo de nuevas herramientas de configuración.
- **Configuración Sin Conexión a Internet:** Dado que muchos de los sensores se encuentran en ubicaciones remotas, es crucial contar con una funcionalidad que permita la configuración sin conexión a internet. Para lograr esto, implementamos un administrador de backups local en la aplicación móvil, donde se almacenan las configuraciones previamente descargadas. Esto permite que los técnicos puedan realizar configuraciones o reconfiguraciones de sensores sin necesidad de estar conectados a internet, garantizando la operatividad incluso en entornos sin cobertura de red.
- **Utilización de API con Base de Datos:** El desarrollo de una API es fundamental para el correcto funcionamiento del sistema. La API actua como un intermediario entre la aplicación móvil y una base de datos centralizada, que contiene todas las configuraciones necesarias para los distintos tipos de sensores. La aplicación móvil envia solicitudes a la API para obtener las configuraciones correspondientes al dispositivo que esté siendo configurado, lo que permite una gestión más eficiente

y actualizada de las configuraciones. Este enfoque centralizado simplifica la actualización de las configuraciones y reduce la posibilidad de errores al mantener un único repositorio de datos.

- **Plataforma Administradora:** Debido a la complejidad y sensibilidad de la base de datos que almacena las configuraciones de los sensores, se desarrolló una plataforma web administrativa. Esta plataforma permite gestionar eficientemente los parámetros configurables de cada sensor mediante una interfaz gráfica intuitiva y fácil de usar. Los usuarios pueden crear, editar y eliminar configuraciones de manera sencilla, asegurando una administración ágil y organizada de la información, lo que facilita la gestión y actualización de los sensores sin comprometer la integridad de la base de datos.

Este enfoque permite a WOLKE superar los problemas actuales relacionados con la configuración de sensores, mejorando la experiencia del usuario, reduciendo costos operativos y optimizando el tiempo necesario para realizar las configuraciones. La aplicación móvil no solo proporciona mayor flexibilidad y accesibilidad, sino que también reduce las dependencias de hardware externo, como computadoras o laptops, al ofrecer una solución más ágil y moderna.

4 . Hipótesis

Se espera que el desarrollo de un sistema integral que combine una API robusta, una base de datos eficiente, una aplicación móvil y una plataforma web administrativa facilite de manera significativa el proceso de configuración y mantenimiento de los sensores de WOLKE. Esta solución permitirá no solo reducir los tiempos de configuración, sino también mejorar la eficiencia operativa y la escalabilidad del sistema. Con base en esta propuesta, las siguientes hipótesis clave guiarán la evaluación del proyecto:

- **Reducción significativa en el tiempo de configuración:** La integración de la API con la aplicación móvil y la plataforma web administrativa permitirá una configuración más rápida de los sensores en comparación con el método manual actual. Al gestionar los sensores de forma remota y sin necesidad de conexiones físicas, se espera que el tiempo de configuración se reduzca considerablemente, particularmente en escenarios con sensores ubicados en zonas de difícil acceso.
- **Simplificación del proceso de configuración:** La plataforma web, al unificar la gestión de los sensores y permitir la configuración a través de una interfaz intuitiva, eliminará la necesidad de software específico para cada tipo de sensor. Esto dará lugar a un proceso de configuración más sencillo y estandarizado, lo que facilitará la gestión tanto en terreno como desde la administración centralizada.
- **Gestión centralizada eficiente:** La base de datos conectada a la API y a la plataforma administrativa proporcionará un control eficiente de las configuraciones de los sensores. Esto no solo mejorará la organización de los datos, sino que también facilitará la toma de decisiones basada en información en tiempo real y reducirá los errores asociados a la gestión manual de los sensores.

Si estas hipótesis se confirman, se podrá concluir que la solución propuesta mejora de manera sustancial el proceso de configuración y mantenimiento de los sensores de WOLKE. Además, contribuirá a una reducción de costos, una mayor escalabilidad y un incremento en la productividad operativa del sistema en su conjunto.

5 . Objetivos

5 .1. Objetivo General

Desarrollar un sistema de configuración inalámbrica de sensores para la empresa WOLKE que permita una gestión eficiente y simplificada de las configuraciones de los sensores, eliminando la necesidad de utilizar conexiones físicas y software específico para cada tipo de sensor.

Para efectos de esta memoria, el objetivo general se centró en la creación de una API eficiente para la gestión de dispositivos, junto con la base de datos, el desarrollo de una interfaz web administradora y el despliegue del sistema en un entorno en la nube utilizando Azure. Estos componentes fueron esenciales para garantizar un sistema funcional y escalable que pudiera ser utilizado por WOLKE para telemetría.

5 .2. Objetivos Específicos

- **Implementar una aplicación móvil con conectividad inalámbrica:** Desarrollar una aplicación que permita la configuración de sensores utilizando tecnologías como Bluetooth Low Energy (BLE), facilitando la interacción entre los sensores y la app desde dispositivos móviles.
- **Diseñar una interfaz de usuario unificada:** Crear una interfaz intuitiva que permita la configuración de diferentes tipos de sensores sin la necesidad de un software específico para cada uno.
- **Desarrollar una API para la gestión de configuraciones:** Implementar una API que se comunique con una base de datos centralizada para obtener, modificar y guardar las configuraciones de los sensores de manera eficiente.
- **Crear un sistema de almacenamiento de configuraciones sin conexión:** Permitir que la aplicación móvil gestione configuraciones localmente, para que los usuarios puedan trabajar sin necesidad de una conexión a internet activa.
- **Desarrollar una plataforma web administrativa:** Crear una plataforma web para la administración de la base de datos de configuraciones, donde se puedan crear, editar y eliminar parámetros configurables de los sensores de manera eficiente y amigable.
- **Optimizar el proceso de configuración de sensores en ubicaciones remotas:** Reducir el tiempo y los costos asociados a la configuración de sensores en lugares de difícil acceso, eliminando la dependencia de conexiones USB y computadoras portátiles.
- **Desplegar la API en un servidor de Azure:** Implementar la API en un entorno de producción utilizando un servidor de Azure, asegurando su disponibilidad y escalabilidad para permitir la conexión y configuración remota de sensores desde diferentes ubicaciones geográficas.

Para los efectos de esta memoria, los objetivos especificos se centraron en el desarrollo y despliegue de los siguientes módulos:

- **API:** Para gestionar las operaciones CRUD de los dispositivos y la configuración de parámetros.
- **Base de Datos:**Optimizada para soportar grandes volúmenes de datos provenientes de los dispositivos de telemetría.
- **Página web administradora:** Actúa como la interfaz gráfica para la administración de los dispositivos, ofreciendo una plataforma fácil de usar.
- **Despliegue en Azure:** Garantiza un entorno escalable y seguro para la ejecución del sistema en producción.

6 . Estructura de la Memoria

Esta memoria se organiza en varios capítulos que guían al lector a través del desarrollo y evaluación del proyecto, proporcionando una comprensión detallada de los métodos y tecnologías aplicadas. A continuación, se presenta una descripción detallada de cada capítulo:

- **Capítulo 1: Introducción**
  Este capítulo establece el marco del proyecto, introduciendo la problemática principal y delineando tanto el problema específico a resolver como el enfoque propuesto para su solución. También se detallan las hipótesis y objetivos del proyecto, ofreciendo una visión clara de lo que se pretende alcanzar.
- **Capítulo 2: Estado del Arte**
  En este capítulo se realiza una revisión exhaustiva de las soluciones y tecnologías existentes similares al proyecto. Se examinan los competidores directos y se discuten las tecnologías relevantes que apoyan el desarrollo del sistema, proporcionando un contexto crítico para las decisiones técnicas tomadas.
- **Capítulo 3: Desarrollo**
  Este capítulo cubre todo el proceso de desarrollo del sistema, desde la formulación de requerimientos, tanto funcionales como no funcionales, hasta el diseño detallado de la arquitectura y la implementación de los módulos específicos. Cada subsección se enfoca en un aspecto particular del desarrollo, incluyendo la base de datos, la API, la interfaz administrativa y los detalles del despliegue en una máquina virtual en Azure.
- **Capítulo 4: Validación**
  La validación del sistema se detalla en este capítulo, donde se describen las pruebas realizadas y los resultados obtenidos. Se discuten las métricas utilizadas para evaluar la eficacia del sistema en términos de reducción del tiempo de configuración y simplificación de procesos, así como la eficiencia en la gestión centralizada del sistema.
- **Capítulo 5: Conclusión**
  El capítulo final sintetiza los logros del proyecto, evaluando el grado de cumplimiento de los objetivos y discutiendo las lecciones aprendidas. También se exploran posibles mejoras y direcciones futuras para el trabajo adicional, señalando oportunidades para expandir o mejorar el sistema propuesto.
- **Referencias**
  Este último segmento lista todas las fuentes bibliográficas y recursos consultados para el desarrollo y documentación del proyecto, proporcionando el respaldo necesario para las afirmaciones y metodologías empleadas en la memoria.

Estado del Arte

7 . Competidores

En la actualidad, existen varias soluciones en el mercado que permiten la configuración de sensores de manera inalámbrica o mediante métodos simplificados. A continuación, se describen algunos de los competidores más relevantes en esta área:

7 .1. Milesight ToolBox

Milesight ToolBox es una aplicación desarrollada por la empresa Milesight, que permite la configuración de sensores de IoT a través de **NFC (Near Field Communication)**. Esta aplicación está diseñada para trabajar con sensores LoRaWAN y otros dispositivos inalámbricos, y permite al usuario realizar configuraciones de forma rápida y eficiente. Entre las características principales de ToolBox se encuentran la capacidad de ajustar parámetros como intervalos de reporte, calibración de sensores y gestión de almacenamiento de datos, todo mediante una interfaz intuitiva que facilita el proceso. Este enfoque permite realizar configuraciones sin necesidad de conexiones físicas, eliminando la dependencia de cables USB o computadoras portátiles, lo que representa una solución similar a la que busca implementar el proyecto [1].

7 .2. Sistema NFC para Clasificación de Frutas

Otro competidor indirecto en la industria de la configuración inalámbrica de sensores es un sistema basado en **NFC** diseñado para la clasificación de la madurez de frutas. Este sistema utiliza un sensor sin batería que se alimenta del campo magnético de un teléfono móvil, proporcionando una solución de bajo costo y altamente eficiente para tareas específicas de sensado. Aunque este sistema tiene un enfoque particular en la agricultura, demuestra las capacidades de las tecnologías NFC para comunicar y configurar sensores de manera sencilla y sin necesidad de energía externa [2].

7 .3. Energía Harvesting

En este articulo, se encuentran trabajos que exploran el uso de sensores IoT combinados con tecnologías de **energía harvesting** y comunicación inalámbrica. Estos sistemas, además de ofrecer configuraciones sencillas, optimizan el consumo de energía al recolectar la necesaria del entorno, como la radiación solar o el campo electromagnético. Estos sensores están diseñados para operar en ubicaciones remotas sin necesidad de mantenimiento constante, una característica relevante para los sensores de WOLKE, que a menudo se encuentran en ubicaciones de difícil acceso [3].

7 .4. Aplicaciones de BLE para Sensores IoT

En este artículo se demuestra que se han desarrollado varias aplicaciones que utilizan **Bluetooth Low Energy (BLE)** para la configuración de sensores IoT. BLE es una tecnología inalámbrica de bajo consumo ideal para dispositivos que requieren estar conectados por largos períodos de tiempo, como los

sensores de WOLKE. Estas soluciones permiten realizar configuraciones y actualizaciones de los dispositivos de manera rápida, mejorando la experiencia del usuario y reduciendo el tiempo de instalación o configuración en campo [4].

8 . Tecnologías Relevantes

Para el desarrollo del sistema de configuración inalámbrica de sensores para la empresa WOLKE, se utilizaron diversas tecnologías que permiten la comunicación eficiente entre sensores y dispositivos móviles, así como la administración de las configuraciones a través de una plataforma web. A continuación, se describen las principales tecnologías empleadas:

- **Bluetooth Low Energy (BLE):** Esta tecnología es clave para establecer la conexión inalámbrica entre la aplicación móvil y los sensores. BLE es una variante de Bluetooth diseñada específicamente para dispositivos que requieren bajo consumo de energía y cortas distancias de comunicación. En este proyecto, BLE permite que los sensores puedan configurarse sin la necesidad de una conexión física o de grandes cantidades de energía, lo cual es ideal para dispositivos IoT como los sensores utilizados por WOLKE [**?**] [5].
- **React Native:** Este framework de desarrollo móvil multiplataforma fue utilizado para construir la aplicación que permite la configuración de los sensores desde dispositivos Android e iOS. React Native facilita el desarrollo simultáneo para ambas plataformas, lo que ahorra tiempo y costos, manteniendo una experiencia de usuario consistente [6].
- **API RESTful:** La comunicación entre la aplicación móvil, la plataforma web administrativa y la base de datos centralizada se realiza a través de una API RESTful. Esta interfaz proporciona una forma estandarizada para que diferentes componentes del sistema interactúen entre sí. La API permite obtener, modificar y almacenar las configuraciones de los sensores de manera eficiente, asegurando la integración fluida entre todos los elementos del sistema [7].
- **Base de Datos NoSQL:** Debido a la flexibilidad requerida para almacenar configuraciones que pueden variar considerablemente entre diferentes tipos de sensores, se optó por una base de datos NoSQL. Este tipo de base de datos es ideal para gestionar grandes volúmenes de datos semi-estructurados, facilitando la escalabilidad del sistema y la rápida adaptación a nuevos tipos de sensores [8].
- **Next.js:** Este framework se utilizó para desarrollar la plataforma web administrativa. Next.js permite combinar aplicaciones SPA (Single Page Applications) con funcionalidades de renderización del lado del servidor, lo que resulta en una plataforma rápida y optimizada. Gracias a esto, los administradores pueden gestionar configuraciones de sensores de manera eficiente desde una interfaz web [9].
- **Microsoft Azure:** Para asegurar la disponibilidad y escalabilidad del sistema, la API, la base de datos y la plataforma web administrativa se despliegan en la nube utilizando **Microsoft Azure**. Esta plataforma ofrece una infraestructura robusta para el almacenamiento de datos, la ejecución de la API y el manejo de las solicitudes de los usuarios desde cualquier ubicación geográfica, garantizando la fiabilidad y seguridad del sistema [10].
- **Docker:** Docker fue utilizado para desplegar la base de datos MongoDB en un contenedor, lo que permitió una gestión más eficiente de desarrollo y garantizó la persistencia de los datos en un entorno aislado y controlado. Esta tecnología facilita la portabilidad y escalabilidad del sistema [11].

- **NGINX:** NGINX se utilizó como un servidor web y reverse proxy para redirigir las solicitudes HTTP hacia la API y la plataforma web administrativa. Esto permitió gestionar de manera eficiente las rutas y mejorar el rendimiento del sistema al distribuir la carga de las solicitudes [12].

9 . Discusión

El sistema de configuración inalámbrica de sensores propuesto para WOLKE se diferencia significativamente de las soluciones actuales en el mercado, como Toolbox de Mighsight y otras plataformas mencionadas en el estado del arte. A continuación, discutimos cómo nuestro enfoque ofrece una mejora sustancial en términos de flexibilidad, escalabilidad y usabilidad.

- **Conectividad Inalámbrica Versátil:** Mientras que sistemas como Toolbox utilizan tecnologías NFC para la configuración de sensores, nuestra solución emplea **Bluetooth Low Energy (BLE)**, lo que ofrece una mayor versatilidad. BLE no solo tiene un menor consumo energético, sino que también permite la configuración a distancias más largas en comparación con NFC, lo que es esencial para sensores instalados en ubicaciones difíciles de alcanzar. Además, BLE permite conexiones más robustas y estables en entornos de red complejos.
- **Interfaz Unificada Multisensor:** Muchas de las soluciones existentes, como las mencionadas en el estado del arte, se limitan a configurar sensores individuales o requieren software específico para cada tipo de sensor. Nuestro enfoque ofrece una **interfaz unificada**, lo que significa que se puede gestionar la configuración de múltiples sensores con una única aplicación móvil. Esto reduce la complejidad operativa para los usuarios, ya que no es necesario instalar múltiples aplicaciones o depender de hardware especializado para cada sensor.
- **Capacidades de Trabajo Sin Conexión:** La posibilidad de trabajar sin conexión a internet es una de las principales contribuciones de nuestra propuesta. A diferencia de otras plataformas que dependen de una conexión activa para operar correctamente, nuestra aplicación móvil almacena configuraciones localmente, permitiendo al usuario gestionar sensores en ubicaciones remotas sin acceso a la red. Esto reduce la dependencia de una infraestructura de telecomunicaciones estable, lo que es clave en entornos rurales o industriales.
- **Escalabilidad y Gestión Centralizada:** El uso de una **API RESTful** conectada a una base de datos NoSQL permite una mayor escalabilidad en comparación con las soluciones tradicionales. Esto es especialmente importante en contextos donde se manejan grandes volúmenes de datos o configuraciones variables entre diferentes tipos de sensores. La posibilidad de gestionar las configuraciones desde una plataforma web administrativa desarrollada con **Next.js** garantiza una interfaz moderna y eficiente, algo que no es tan común en otras soluciones que suelen tener interfaces más básicas o limitadas.
- **Despliegue en la Nube con Azure:** Muchas soluciones existentes dependen de infraestructuras locales o no aprovechan plenamente las ventajas de la nube. Nuestra plataforma está desplegada en **Microsoft Azure**, lo que garantiza una alta disponibilidad, escalabilidad y seguridad. Esto permite que el sistema sea accesible desde cualquier ubicación geográfica, brindando un nivel de fiabilidad superior en comparación con soluciones que no tienen estas capacidades en la nube.

Nuestro sistema de configuración inalámbrica de sensores no solo supera las limitaciones de las tecnologías actuales, sino que también ofrece un enfoque más flexible y adaptable. La combinación de BLE, una interfaz unificada, capacidades offline, escalabilidad a través de una base de datos NoSQL, y un despliegue en la nube con Azure, posiciona nuestra solución como una alternativa más eficiente y práctica para la

gestión de sensores a gran escala.

Desarrollo

10 . Análisis

El análisis de los requerimientos para desarrollar un sistema de configuración inalámbrica de sensores implica identificar las necesidades técnicas y funcionales para lograr un flujo de trabajo eficiente y confiable. Los requerimientos se dividen en dos grandes categorías: requerimientos funcionales y requerimientos no funcionales, ambos son esenciales para resolver el problema de la gestión de sensores a distancia y garantizar la calidad del producto final.

10 .1. Requerimientos Funcionales

Los requerimientos funcionales especifican las acciones que el sistema debe realizar para cumplir con los objetivos del proyecto. Estos incluyen:

- **Conectividad inalámbrica de sensores:** La aplicación móvil debe ser capaz de conectarse a los sensores de manera inalámbrica utilizando tecnologías como **Bluetooth Low Energy (BLE)**, permitiendo la configuración sin la necesidad de cables o conexiones físicas.
- **Administración de configuraciones:** El sistema debe permitir a los usuarios gestionar las configuraciones de los sensores, incluyendo la creación, edición, y eliminación de parámetros específicos para cada tipo de sensor.
- **Interfaz unificada:** La aplicación debe proporcionar una interfaz que permita la configuración de diferentes tipos de sensores sin requerir un software específico para cada uno, facilitando su uso y reduciendo la curva de aprendizaje para los usuarios.
- **Capacidades offline:** La aplicación debe permitir la gestión de sensores en modo offline, de manera que los usuarios puedan continuar trabajando sin acceso a internet y sincronizar los datos una vez se restablezca la conexión.
- **Integración con la API RESTful:** El sistema debe comunicarse con una API centralizada para almacenar y recuperar configuraciones en una base de datos remota, facilitando la gestión centralizada de los sensores a través de una plataforma web.
- **Plataforma web administrativa:** Debe existir una plataforma web donde los administradores puedan gestionar las configuraciones de los sensores desde cualquier ubicación, permitiendo una administración más eficiente a nivel global.
- **Notificaciones y alertas:** La aplicación debe ser capaz de enviar notificaciones a los administradores en caso de que ocurra algún problema con los sensores o se requiera alguna acción específica.

10 .2. Requerimientos No Funcionales

Los requerimientos no funcionales son aquellos que definen las propiedades y limitaciones del sistema, asegurando que el producto final sea robusto, eficiente, y escalable:

- **Seguridad:** La plataforma debe asegurar que las configuraciones de los sensores y la comunicación entre la aplicación móvil, la API, y la base de datos estén protegidas mediante autenticación y encriptación, minimizando los riesgos de acceso no autorizado.
- **Escalabilidad:** Dado que se espera que el sistema pueda manejar un número creciente de sensores con el tiempo, debe diseñarse de manera que pueda escalar sin perder eficiencia, particularmente en la gestión de grandes volúmenes de datos desde la base de datos NoSQL.
- **Disponibilidad y fiabilidad:** Utilizando servicios en la nube como **Microsoft Azure**, se debe asegurar que la plataforma esté disponible en todo momento, minimizando las interrupciones y garantizando una alta fiabilidad para los usuarios finales.
- **Usabilidad:** Tanto la aplicación móvil como la plataforma web administrativa deben ofrecer una interfaz intuitiva y amigable que minimice la curva de aprendizaje, haciendo que la configuración de sensores sea un proceso simple para los usuarios no técnicos.
- **Rendimiento:** El sistema debe ofrecer tiempos de respuesta rápidos, especialmente en la configuración y sincronización de sensores, para asegurar que las operaciones se realicen sin demoras perceptibles por parte del usuario.
- **Compatibilidad tecnológica:** Dado que el cliente maneja el lenguaje de programación **C#**, la API debe desarrollarse en este lenguaje para facilitar el mantenimiento y futuras mejoras al sistema.

11 . Diseño

El diagrama de contexto mostrado en la Figura 1 ilustra la interacción entre los componentes principales del sistema de configuración inalámbrica de sensores.

- **Aplicación móvil:** Esta se comunica con los sensores a través de Bluetooth Low Energy (BLE) para recibir y configurar parámetros, permitiendo a los usuarios interactuar con los sensores de manera inalámbrica. Además, la aplicación cuenta con un administrador de backups, asegurando la integridad de los datos en caso de pérdida de conexión o fallas.
- **API RESTful:** El centro del sistema es una API desarrollada en C sharp, encargada de procesar solicitudes CRUD (crear, leer, actualizar y eliminar) que provienen tanto de la aplicación móvil como de la plataforma web. La API actúa como intermediario entre los sensores y la base de datos en la nube.
- **Nube de Microsoft Azure:** El sistema se encuentra desplegado en la nube de Azure, lo que asegura alta disponibilidad, escalabilidad y un acceso global. En este entorno, se aloja la base de datos MongoDB, que gestiona y almacena todas las configuraciones de los sensores. También en la nube se encuentra la aplicación administradora de la base de datos, la cual permite gestionar los datos de los sensores y sus configuraciones de forma más detallada y segura. Además, la plataforma web administrativa, desarrollada con Next.js, proporciona una interfaz amigable para que los administradores puedan gestionar los parámetros y configuraciones de los sensores desde cualquier lugar.

Este diagrama de contexto proporciona una visión clara de cómo los diferentes elementos del sistema están interconectados, destacando la API como el núcleo de las interacciones y la plataforma web como un componente clave para la gestión de los sensores a nivel administrativo. Las áreas resaltadas en rojo reflejan la contribución de esta memoria.

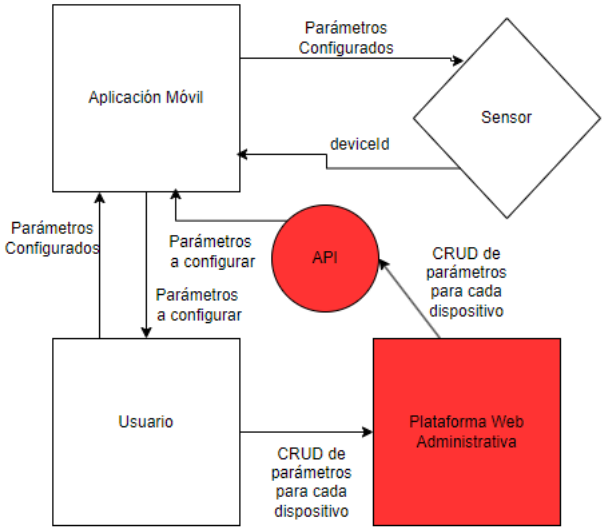

Figura 1: Diagrama de Contexto

En la Figura 2, se ilustra el diagrama de arquitectura del sistema, detallando cómo se estructuran los componentes tecnológicos para lograr la gestión inalámbrica y remota de sensores. Este diseño contempla tres capas clave: dispositivos móviles, API central y la nube de Microsoft Azure, que aloja no solo la base de datos, sino también la aplicación web y un administrador de la base de datos. Las áreas resaltadas en rojo reflejan la contribución de esta memoria.

- **Aplicación móvil:** Esta se comunica con los sensores a través de Bluetooth Low Energy (BLE) para recibir y configurar parámetros, permitiendo a los usuarios interactuar con los sensores de manera inalámbrica. Además, la aplicación cuenta con un administrador de backups, asegurando la integridad de los datos en caso de pérdida de conexión o fallas
- **API RESTful:** El centro del sistema es una API desarrollada en C sharp, encargada de procesar solicitudes CRUD (crear, leer, actualizar y eliminar) que provienen tanto de la aplicación móvil como de la plataforma web. La API actúa como intermediario entre los sensores y la base de datos en la nube.
- **Nube de Microsoft Azure:** El sistema se encuentra desplegado en la nube de Azure, lo que asegura alta disponibilidad, escalabilidad y un acceso global. En este entorno, se aloja la base de datos MongoDB, que gestiona y almacena todas las configuraciones de los sensores. También en la nube se encuentra la aplicación administradora de la base de datos, la cual permite gestionar los datos de los sensores y sus configuraciones de forma más detallada y segura. Además, la plataforma web administrativa, desarrollada con Next.js, proporciona una interfaz amigable para que los administradores puedan gestionar los parámetros y configuraciones de los sensores desde cualquier lugar.

Con esta arquitectura basada en la nube, el sistema garantiza un alto grado de eficiencia, fiabilidad y escalabilidad, asegurando que la gestión de los sensores sea fluida, accesible y confiable.

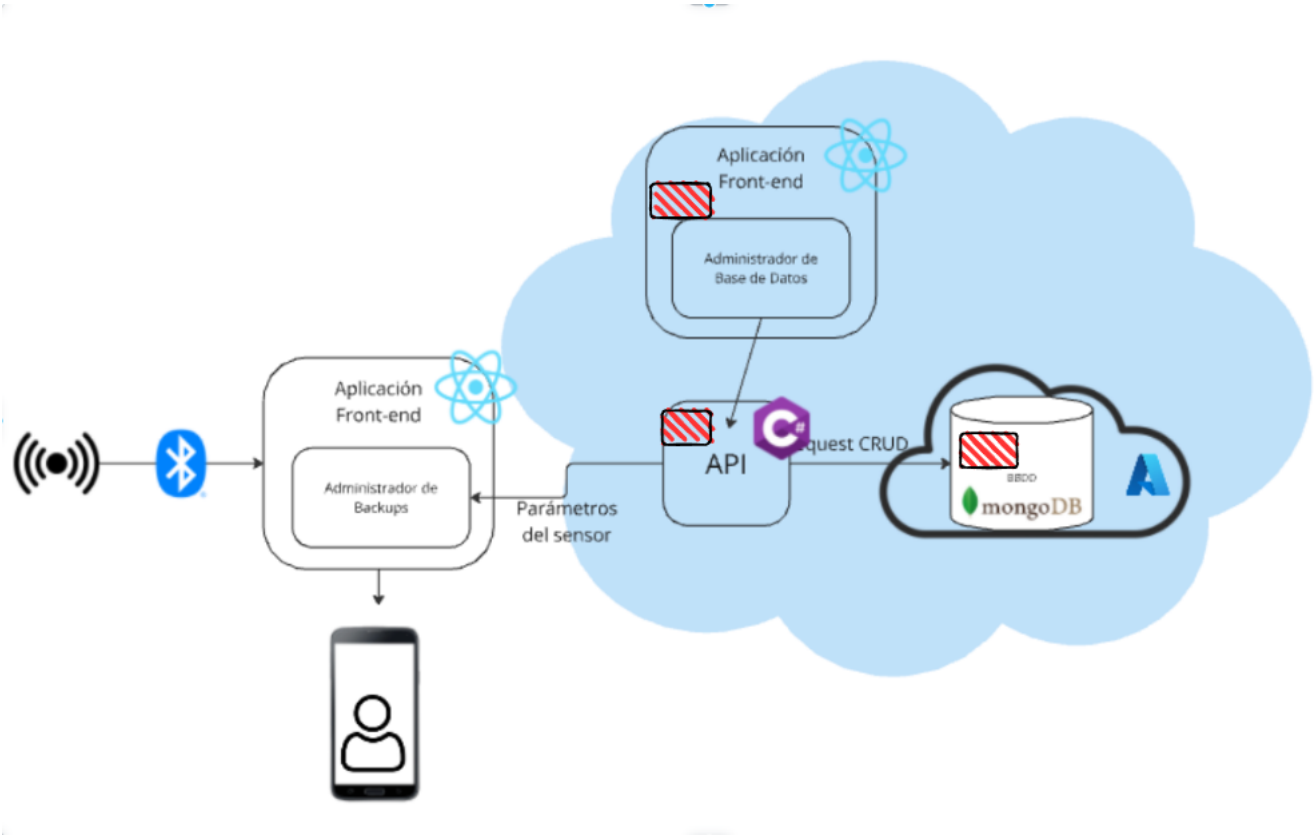

Figura 2: Diagrama de Arquitectura

12 . Implementación

12 .1. Módulo de Base de Datos

En la implementación del sistema de configuración inalámbrica de sensores, inicialmente se optó por utilizar MongoDB Atlas como solución temporal para gestionar la base de datos. Esta decisión estratégica permitió que el desarrollo del sistema avanzara de forma rápida y escalable en la nube, aprovechando la flexibilidad y capacidades que ofrece MongoDB Atlas en cuanto a gestión de grandes volúmenes de datos y escalabilidad horizontal. Esta elección temporal fue esencial para asegurar que el sistema fuera capaz de manejar múltiples configuraciones de sensores en diferentes fases del desarrollo.

Sin embargo, a medida que el proyecto progresaba y se consolidaban los requisitos finales, se decidió migrar la base de datos a una infraestructura más robusta y controlada. De esta forma, la base de datos se trasladó a una máquina virtual alojada en Azure, donde también están desplegados tanto la API como la Aplicación Web Administradora. Esta transición a un servidor dedicado permitió tener un control más granular sobre la infraestructura, mejorar la seguridad y garantizar un rendimiento más consistente para las necesidades a largo plazo del sistema.

En cuanto a la estructura de la base de datos, esta pasó por varias fases de optimización, con múltiples interacciones y mejoras. Durante este proceso, se trabajó en estrecha colaboración con José Valenzuela, encargado del PMM en Wolke, para afinar el diseño y asegurar que fuera lo suficientemente eficiente para manejar de manera óptima las configuraciones y datos de los sensores. Después de varias pruebas, iteraciones y ajustes, se llegó a un esquema final optimizado que se adecua perfectamente a las necesidades

del proyecto.

El esquema optimizado que se implementó está diseñado para facilitar la organización de las configuraciones de los sensores, permitiendo agrupar variables en pestañas que, a su vez, contienen elementos configurables. Cada pestaña permite gestionar múltiples elementos de configuración que son necesarios para los diferentes dispositivos conectados. La estructura final de la base de datos, presentada en un formato JSON, facilita la interacción entre los dispositivos y la API, permitiendo el manejo eficiente de las configuraciones de sensores y su almacenamiento en la base de datos.

El siguiente esquema ilustra la estructura final de la base de datos que hemos definido y optimizado:

```
{
  "id": "",
  "deviceId": "",
  "name": "",
  "minFwVersion": "",
  "minHwVersion": "",
  "numElements": 0,
  "tabs": [
    {
      "nameId": 0,
      "label": "",
      "enabled": false,
      "visible": false,
      "elements": [
        {
          "nameId": 0,
          "label": "",
          "enabled": false,
          "visible": false,
          "readOnly": false,
          "type": "",
          "length": 0,
          "offset": 0
        },
        {
          "nameId": 0,
          "label": "",
          "enabled": false,
          "visible": false,
          "readOnly": false,
          "type": "",
          "length": 0,
          "offset": 0
        }
      ]
    }
```

Figura 3: estructura base de datos

Explicación de las Variables

- **id:** Identificador único del dispositivo en la base de datos.
- **deviceId:** Identificación específica del dispositivo configurado.
- **name:** Nombre del dispositivo para facilitar su identificación.
- **minFwVersion:** Versión mínima del firmware requerida para el correcto funcionamiento del dispositivo.

- **minHwVersion:** Versión mínima del hardware compatible con la configuración.
- **numElements:** Número total de elementos o parámetros que se pueden configurar para el dispositivo.
- **tabs:** Agrupación lógica de los elementos configurables. Cada pestaña contiene un conjunto de parámetros asociados para su configuración de manera más organizada.

Tabs

Cada pestaña (*tab*) se utiliza para agrupar un conjunto de *elements* (parámetros), facilitando la configuración y organización del dispositivo. Esto es especialmente útil cuando se trata de dispositivos con múltiples parámetros, ya que permite a los usuarios configurar cada grupo de elementos dentro de una pestaña específica, mejorando la experiencia de usuario.

Elements

Los *elements* dentro de cada pestaña son los parámetros configurables del dispositivo. Cada uno tiene los siguientes atributos:

- **nameId:** Identificador único del elemento.
- **label:** Descripción del parámetro que aparece en la interfaz de usuario.
- **enabled:** Indica si el parámetro está habilitado para ser configurado.
- **visible:** Define si el parámetro es visible en la interfaz.
- **readOnly:** Especifica si el valor del parámetro es de solo lectura.
- **type:** Tipo de dato del parámetro (texto, número, etc.).
- **length:** Longitud del campo de texto, si es aplicable.
- **offset:** Desplazamiento que indica la ubicación del parámetro dentro de los datos enviados al dispositivo.

Testing del Módulo de Base de Datos

Una vez implementada la base de datos en su entorno definitivo, fue crucial asegurar que el sistema cumpliera con los requisitos de estabilidad, rendimiento y escalabilidad. Para lograr esto, se realizaron diferentes tipos de pruebas para validar su correcto funcionamiento.

Pruebas de Carga (Load Testing): Se utilizó la herramienta JMeter para simular múltiples configuraciones simultáneas de dispositivos conectados a la base de datos, evaluando su capacidad de respuesta bajo una carga considerable de tráfico. Esto permitió identificar puntos críticos donde la latencia podría aumentar con un volumen alto de peticiones. Durante las pruebas, el sistema fue capaz de manejar hasta 500 configuraciones simultáneas sin experimentar una degradación significativa en el rendimiento.

Pruebas de Estrés (Stress Testing): Además de las pruebas de carga, se implementaron pruebas de estrés para verificar el comportamiento del sistema cuando se excedía su capacidad esperada. Esto se hizo incrementando el número de configuraciones y solicitudes de manera progresiva hasta que el sistema alcanzó su límite de rendimiento. Gracias a estas pruebas, se identificaron áreas de mejora en la optimización de consultas y manejo de conexiones, lo que llevó a ajustar el tamaño del pool de conexiones en el servidor de la base de datos.

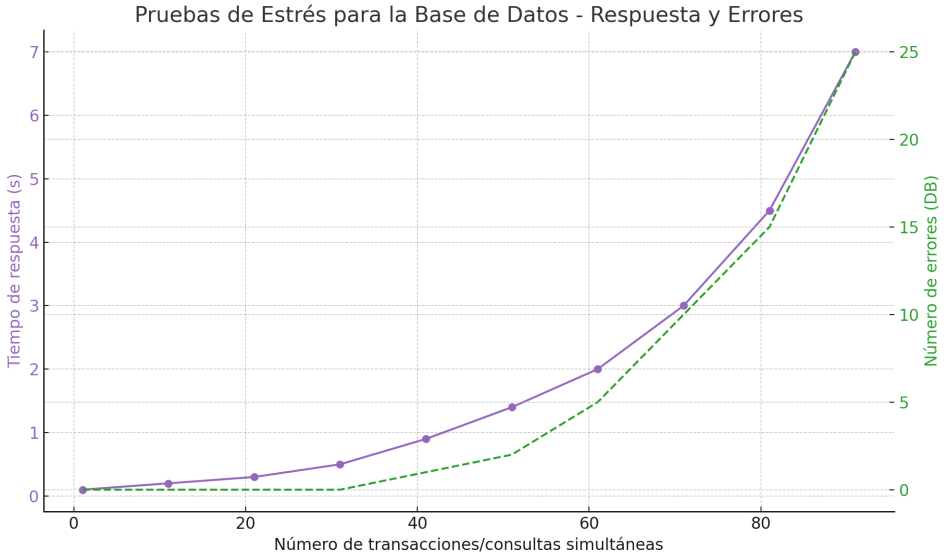

Figura 4: prueba de estrés

Pruebas de Integridad de Datos (Data Integrity Testing): Para asegurar que las configuraciones de los sensores se almacenaban y recuperaban de manera correcta, se realizaron pruebas de integridad de datos utilizando pruebas de regresión automatizadas. Estas pruebas fueron diseñadas para comparar las configuraciones de entrada con los datos recuperados, verificando que no hubiera pérdida de información ni alteración en los parámetros durante las operaciones CRUD (Create, Read, Update, Delete).

Pruebas de Seguridad (Security Testing): La migración de la base de datos a una máquina virtual en Azure permitió implementar controles de seguridad adicionales. Para verificar la robustez de la seguridad, se realizaron pruebas de penetración utilizando OWASP ZAP, enfocadas en detectar vulnerabilidades en el acceso no autorizado a los datos. Estas pruebas ayudaron a reforzar el sistema de autenticación y a implementar cifrado tanto en tránsito como en reposo para los datos más sensibles.

12 .2. Módulo de la API

El desarrollo de la API fue un componente clave para gestionar las configuraciones de los dispositivos y garantizar una interacción fluida entre la base de datos, la aplicación móvil y la plataforma web administradora. Inicialmente, la API fue implementada en Node.js debido a nuestra experiencia en JavaScript, lo que permitió un desarrollo rápido y eficiente. La API proporcionaba todas las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para los dispositivos mediante una estructura organizada y fácil de implementar.

Posteriormente, cuando la API en Node.js estaba completamente funcional, recibimos una solicitud del cliente para migrar la API a .NET utilizando C sharp. Esto implicó un proceso de aprendizaje y adaptación, pero finalmente la API fue reescrita en C sharp bajo el entorno .NET, aprovechando la robustez y escalabilidad de esta tecnología. La migración también permitió una mejor integración con otros sistemas utilizados por el cliente y brindó mejoras de rendimiento, especialmente en operaciones concurrentes y manejo de cargas.

Seguridad con Bearer Token

Además de las funcionalidades CRUD, se integró un sistema de autenticación utilizando Bearer Token para asegurar el acceso a las rutas de la API. Cada solicitud de los usuarios requiere un token válido que

es enviado en el encabezado de las peticiones HTTP, lo que garantiza que solo los usuarios autenticados puedan acceder a las funciones de gestión de los dispositivos. Este mecanismo de seguridad previene accesos no autorizados y protege tanto los datos de los dispositivos como sus configuraciones.

Peticiones de la API

La API implementa varias rutas para gestionar los dispositivos, todas ellas protegidas mediante el esquema de autenticación mencionado. A continuación, se listan las principales rutas y sus funciones:

- **GET /api/devices**: Obtiene la lista de todos los dispositivos registrados en la base de datos.
- **GET /api/devices/:deviceId**: Obtiene los detalles de un dispositivo específico utilizando su `deviceId`.
- **POST /api/devices**: Permite registrar un nuevo dispositivo en el sistema, incluyendo sus parámetros configurables.
- **PUT /api/devices/:deviceId** : Actualiza los detalles de un dispositivo existente identificado por su `id`.
- **DELETE /api/devices/:deviceId** : Elimina un dispositivo del sistema utilizando su `id`.

La API, tanto en su versión inicial desarrollada en Node.js como en la más reciente en .NET, ha sido un componente fundamental para la interacción de los usuarios con el sistema. Proporciona un control centralizado sobre los dispositivos y permite la configuración remota de los mismos mediante las aplicaciones móviles y web.

Testing de la API

Durante el proceso de desarrollo de la API en .NET, también se llevó a cabo una fase exhaustiva de pruebas para asegurar su correcto funcionamiento y garantizar la fiabilidad del sistema en producción. Para esto, se utilizó la herramienta xUnit, una de las bibliotecas más populares para realizar testing en aplicaciones desarrolladas en C sharp. xUnit ofrece una estructura clara y extensible para escribir pruebas unitarias y de integración, lo que permitió validar cada una de las funcionalidades implementadas.

Las pruebas se enfocaron en validar las siguientes áreas clave:

Pruebas Unitarias: Cada función de la API fue testeada de forma aislada para asegurar que se comportara según lo esperado bajo diferentes condiciones de entrada y salida. Esto incluyó la verificación de las operaciones CRUD y la validación de los parámetros recibidos.

Pruebas de Integración: Se realizaron pruebas para asegurar que la API se comunicara correctamente con MongoDB, tanto en el entorno temporal de MongoDB Atlas como en la base de datos migrada al servidor físico de Azure.

Validación de Seguridad: Se comprobó que el sistema de autenticación mediante Bearer Token protegiera efectivamente las rutas de la API, impidiendo accesos no autorizados a los recursos.

Gracias a este proceso de testing riguroso, logramos garantizar que la API fuera estable, segura y capaz de manejar un gran número de dispositivos y configuraciones sin problemas de rendimiento o fallos de seguridad.

12 .3. Módulo Interfaz Administradora

El desarrollo de la interfaz administradora tuvo como objetivo ofrecer una plataforma fácil de usar para la gestión de dispositivos y sus configuraciones. Esta interfaz se implementó utilizando React, un popular framework de JavaScript que facilita la creación de interfaces dinámicas y componentes reutilizables, lo que nos permitió desarrollar una solución moderna y escalable.

La interfaz está diseñada para ofrecer a los usuarios la capacidad de visualizar, agregar, editar y eliminar dispositivos registrados en el sistema, así como configurar sus parámetros específicos. La estructura principal está compuesta por secciones que corresponden a diferentes módulos, como la administración de usuarios, el estado de los dispositivos y la configuración de cada uno.

Vistas

La vista de **Login**, representada en la **Figura 4**, es el punto de acceso inicial para los usuarios autorizados del sistema. Esta pantalla fue diseñada con un enfoque en la seguridad y la simplicidad, permitiendo a los usuarios autenticarse mediante el ingreso de sus credenciales, que incluyen un nombre de usuario y una contraseña. El proceso de autenticación está respaldado por un sistema robusto basado en *JSON Web Tokens* (JWT), lo que garantiza que las sesiones sean seguras, eficientes y estén libres de riesgos relacionados con la gestión de datos sensibles en el lado del cliente.

El proceso de autenticación comienza cuando el usuario ingresa sus credenciales en los campos designados. Estas credenciales son enviadas a la API del backend a través de solicitudes HTTP seguras (*HTTPS*), donde son validadas. Si la autenticación es exitosa, el servidor genera un *token* JWT único para esa sesión. Este *token* incluye la información del usuario autenticado y un tiempo de expiración, asegurando que la sesión sea válida solo por un período de tiempo limitado, reduciendo así la exposición a posibles ataques.

Este *token* es almacenado en el almacenamiento local o en la memoria de la aplicación, según corresponda. Cada solicitud posterior al servidor desde la interfaz del usuario incluye este *token* en el encabezado, permitiendo al backend validar la sesión sin necesidad de re-autenticación constante. En caso de que el *token* expire o sea inválido, el usuario será redirigido automáticamente a la vista de Login, garantizando que solo usuarios autenticados puedan acceder a las funcionalidades restringidas de la plataforma.

La interfaz de la vista de Login es minimalista y fácil de navegar, con un diseño que prioriza la experiencia del usuario sin comprometer la seguridad. Algunas características destacadas son:

- **Formulario de Login:** Incluye campos para el nombre de usuario y la contraseña. Se han implementado validaciones instantáneas que aseguran que ambos campos estén correctamente completados antes de permitir el envío del formulario.
- **Validación de Seguridad:** El sistema está protegido contra ataques comunes como *SQL Injection* y *Cross-Site Scripting* (XSS). Además, se aplican políticas de contraseña segura para garantizar que los usuarios creen credenciales robustas.
- **Botón de Envío:** Una vez completados correctamente los campos del formulario, el usuario puede hacer clic en el botón de envío, el cual dispara la solicitud de autenticación hacia la API. Si las credenciales son inválidas o el servidor no puede procesar la solicitud, se muestra un mensaje de error sin necesidad de recargar la página.
- **Mensajes de Error:** Los mensajes de error y estado son claros y específicos, proporcionando retroalimentación inmediata al usuario en caso de ingresar credenciales incorrectas o en caso de errores de conectividad con el servidor.

- **Indicadores de Carga:** Durante el proceso de autenticación, un indicador de carga asegura que el usuario esté informado de que el sistema está procesando su solicitud.
- **Diseño Responsivo:** La vista de Login está optimizada para funcionar tanto en dispositivos móviles como en pantallas más grandes, lo que facilita el acceso desde múltiples plataformas.

El sistema de autenticación con JWT permite que el usuario no tenga que ingresar sus credenciales repetidamente mientras su sesión siga activa. Sin embargo, para proteger la seguridad del sistema, el *token* expira automáticamente después de un tiempo determinado o si el usuario cierra sesión manualmente.

En cuanto a la experiencia de usuario, se realizaron diversas pruebas de usabilidad para garantizar que la vista fuera intuitiva y fácil de usar, incluso para usuarios sin experiencia técnica. Se probaron múltiples escenarios, como la introducción de credenciales incorrectas o la interrupción de la conexión de red, asegurando que el sistema gestione correctamente estas situaciones, proporcionando mensajes claros y permitiendo reintentos fáciles.

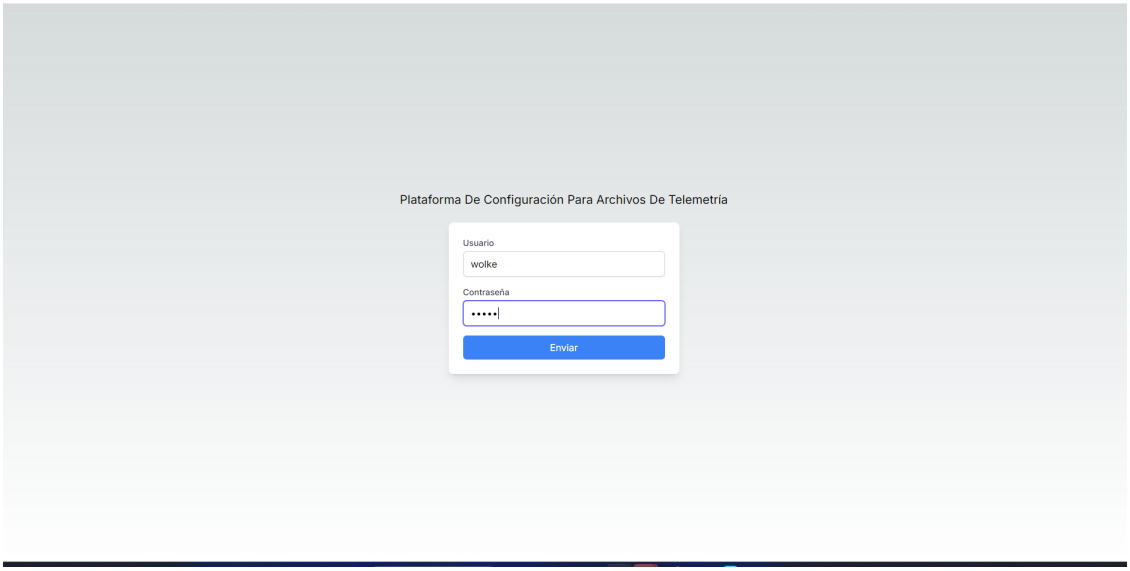

Figura 5: Inicio de Sesión

La vista de **Listado de Dispositivos**, representada en la **Figura 6**, permite a los usuarios gestionar los dispositivos registrados en el sistema de manera eficiente y ordenada. Esta vista muestra una tabla interactiva donde cada fila corresponde a un dispositivo existente, y cada dispositivo puede ser editado o eliminado fácilmente por el usuario.

La interfaz principal de esta vista está compuesta por los siguientes elementos:

- **Listado de Dispositivos:** La tabla principal lista todos los dispositivos registrados en el sistema. Cada fila muestra información clave del dispositivo, como su nombre, estado y otras características esenciales configurables.
- **Acciones sobre Dispositivos:** Cada dispositivo en la lista incluye un conjunto de íconos que permiten al usuario realizar acciones rápidas:
  - **Edición de Dispositivo:** Al hacer clic sobre el nombre de cualquier dispositivo, el usuario es redirigido a la vista de edición del dispositivo seleccionado. Desde esa vista, el usuario puede ajustar configuraciones específicas del dispositivo, como sus parámetros operativos y configuraciones avanzadas.

- **Eliminación de Dispositivo:** Cada fila de la tabla contiene un ícono de *basurero*, que permite al usuario eliminar un dispositivo de manera rápida. Al hacer clic en el ícono de eliminación, el sistema solicita confirmación antes de proceder con la eliminación definitiva, garantizando que no se eliminen dispositivos por error.

- **Botón para Agregar Dispositivo:** En la parte superior de la vista, se encuentra un botón destacado que permite agregar un nuevo dispositivo al sistema. Al hacer clic en este botón, el usuario es dirigido a un formulario para registrar un nuevo dispositivo, ingresando toda la información requerida como nombre, tipo, y configuración inicial.
- **Notificaciones de Estado:** La vista también incluye notificaciones visuales que informan al usuario sobre el éxito o el fracaso de las acciones realizadas, como la eliminación o adición de dispositivos. Estas notificaciones aseguran que el usuario esté siempre informado de las modificaciones realizadas en el sistema.

El diseño de esta vista fue concebido con un enfoque en la eficiencia y facilidad de uso, permitiendo a los administradores gestionar múltiples dispositivos sin complicaciones. Además, se implementaron pruebas de usabilidad para garantizar que las funciones más utilizadas, como la edición y eliminación, estén claramente accesibles, lo que reduce la carga cognitiva en la administración de grandes volúmenes de dispositivos.

El sistema se integra de manera fluida con la **API de gestión de dispositivos**, lo que garantiza que cualquier cambio en los dispositivos (agregado, edición o eliminación) se refleje inmediatamente en la base de datos central. Esta sincronización en tiempo real asegura que la información esté siempre actualizada y disponible para los usuarios.

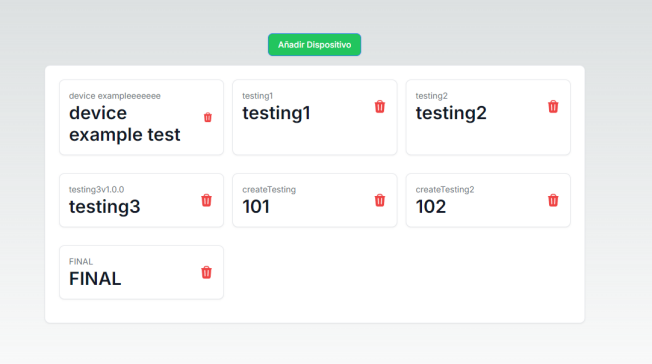

Figura 6: Listado de Dispositivos

La vista de **Creación y Edición de Dispositivos**, representada en la **Figura 7**, permite a los usuarios crear o editar un dispositivo existente en el sistema. Esta interfaz ha sido diseñada para ofrecer una flexibilidad total en la configuración de los parámetros del dispositivo, facilitando tanto la adición de nuevos dispositivos como la modificación de los ya registrados.

Características Principales de la Vista:

- **Formulario de Configuración:** Al acceder a la vista, se presenta un formulario que contiene todos los campos configurables del dispositivo, como su nombre, estado, tipo, y otros parámetros específicos. El formulario está organizado de manera intuitiva para que los usuarios puedan rellenar la información sin dificultades.

- **Gestión de Pestañas (Tabs):** La vista ofrece la opción de gestionar múltiples *tabs* o pestañas que permiten organizar mejor los diferentes elementos configurables del dispositivo. Esto es especialmente útil cuando un dispositivo tiene numerosos parámetros que se pueden agrupar en categorías. El usuario puede agregar o eliminar *tabs* según sea necesario, proporcionando una experiencia de personalización avanzada.
- **Gestión de Elementos:** Dentro de cada *tab*, se permite agregar o eliminar *elements*, es decir, campos configurables específicos. Estos *elements* incluyen opciones como campos de texto, selectores de opciones (para tipos de datos *enum*), campos numéricos, entre otros. Esta flexibilidad permite al usuario definir con precisión los atributos del dispositivo que desea configurar.
- **Validación de Datos:** Cada campo del formulario incluye validaciones personalizadas para asegurar que los datos ingresados sean correctos. Por ejemplo, los campos numéricos solo aceptan valores dentro de un rango permitido, y los campos de texto pueden incluir validaciones basadas en expresiones regulares (regex) para asegurar el formato adecuado.
- **Sincronización en Tiempo Real:** Al igual que en otras vistas, cualquier cambio realizado en los campos del dispositivo se sincroniza directamente con la **API de gestión de dispositivos**, lo que garantiza que los cambios sean reflejados inmediatamente en la base de datos del sistema. Esto asegura que los dispositivos operen siempre con la configuración más actualizada.
- **Botones de Acción:** La interfaz cuenta con botones de acción en la parte inferior del formulario, como *Guardar* o *Cancelar*. Al presionar el botón de *Guardar*, los datos ingresados son validados y luego enviados a la API para su almacenamiento. En caso de un error de validación, el sistema alerta al usuario mediante mensajes de error en pantalla.
- **Notificaciones:** Se han integrado notificaciones visuales para informar al usuario sobre el estado de sus acciones (por ejemplo, si la creación o edición fue exitosa o si ocurrió un error durante el proceso).
- **Modo de Edición o Creación:** La vista es reutilizable tanto para la creación de un nuevo dispositivo como para la edición de uno existente. En el modo de creación, los campos aparecerán vacíos y listos para ser rellenados. En el modo de edición, los campos se cargarán con los valores actuales del dispositivo seleccionado, permitiendo que los usuarios realicen las modificaciones necesarias.

Esta vista es un componente clave en la administración del sistema, ya que permite a los administradores del sistema definir, modificar y personalizar todos los aspectos configurables de un dispositivo de manera sencilla y eficaz. La posibilidad de gestionar tanto *tabs* como *elements* proporciona una flexibilidad avanzada para manejar dispositivos con configuraciones complejas.

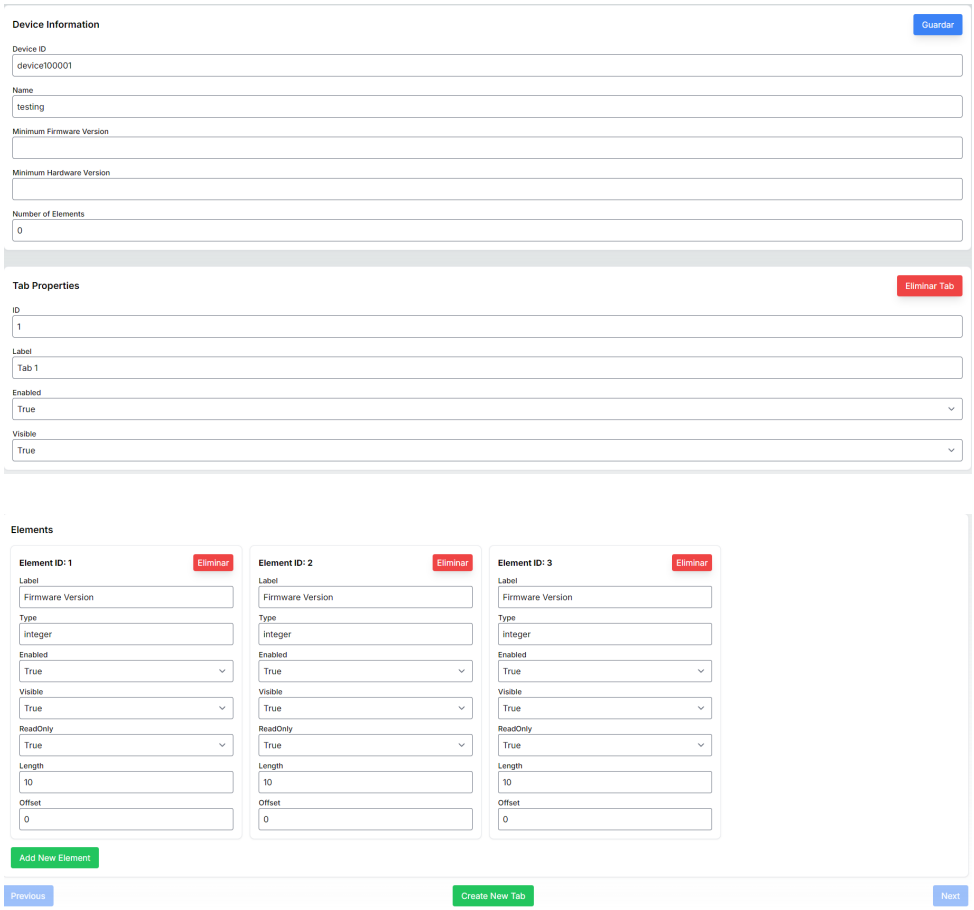

Figura 7: Creación y Edición de Dispositivos

12 .4. Módulo Despliegue del Sistema en una Máquina Virtual en Azure

Para el despliegue del sistema de configuración inalámbrica de sensores, se optó por alojar los componentes en una máquina virtual en Microsoft Azure. Esto proporciona una infraestructura segura, escalable y controlada para la API, la base de datos y la plataforma web administradora.

Base de Datos en Docker

La base de datos MongoDB fue desplegada utilizando un contenedor Docker en la máquina virtual de Azure. Esto permite una gestión más eficiente del almacenamiento y garantiza la persistencia de los datos mediante el uso de volúmenes. MongoDB se utiliza para almacenar todas las configuraciones de los sensores y sus parámetros.

El archivo *docker-compose.yml* para el despliegue de MongoDB es el siguiente:

```
version: '3.7'

services:
  mongo:
    image: mongo:4.4
    ports:
      - "27017:27017"
    volumes:
```

```
      - mongo_data:/data/db

volumes:
  mongo_data:
```

Este archivo orquesta el servicio de MongoDB, exponiendo el puerto `27017` para que sea accesible desde la API y montando un volumen para persistir los datos, asegurando que la información almacenada no se pierda en caso de que el contenedor se reinicie.

API y Plataforma Web Administradora

La API fue desplegada directamente en la máquina virtual utilizando `.NET Core`, lo que permite gestionar las operaciones de configuración de los dispositivos a través de solicitudes REST. Esta API se comunica con MongoDB para realizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre los datos de los sensores.

La plataforma web administradora, desarrollada en **Next.js**, también fue desplegada directamente en la máquina virtual. Esta plataforma proporciona a los administradores una interfaz gráfica desde la cual pueden gestionar las configuraciones de los dispositivos y supervisar el estado de los sensores.

NGINX como Reverse Proxy

Para gestionar eficientemente el tráfico entre la API y la plataforma web administradora, se configuró NGINX como un *reverse proxy*. NGINX redirige las solicitudes HTTP a los componentes correspondientes, asegurando que las rutas estén bien organizadas y que la comunicación entre los componentes sea fluida. El archivo de configuración de NGINX es el siguiente:

```
server {
    listen 80;

    location /api/ {
        proxy_pass http://localhost:5000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
        proxy_pass http://localhost:3000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Este archivo asegura que las solicitudes que comienzan con `/api/` se redirijan a la API, mientras que las demás solicitudes se redirigen a la plataforma web administradora.

Testing del Sistema en Producción

Una vez desplegado el sistema en Azure, se realizaron diversas pruebas para validar su correcto funcionamiento en producción. Estas pruebas incluyeron tanto el sistema como la propia infraestructura en la máquina virtual de Azure.

Testing de la Máquina Virtual en Azure

Para garantizar que la máquina virtual de Azure funcione correctamente, se realizaron las siguientes pruebas:

- **Pruebas de Latencia y Conectividad:** Se utilizaron herramientas como **Ping** y **Traceroute** para verificar la latencia de la conexión entre la máquina virtual y los usuarios finales, así como entre la API y la base de datos MongoDB. Estas pruebas aseguraron que no existiera una latencia significativa que pudiera afectar el rendimiento del sistema.
- **Pruebas de CPU y Memoria:** Para validar la capacidad de la máquina virtual bajo cargas de trabajo intensivas, se utilizaron herramientas como **htop** y **stress-ng** para simular situaciones de alta demanda en los recursos de CPU y memoria. Estas pruebas permitieron asegurar que la máquina pudiera manejar múltiples solicitudes simultáneas sin experimentar problemas de rendimiento.
- **Pruebas de Almacenamiento:** Se realizaron pruebas de lectura y escritura en el disco para asegurarse de que los volúmenes de almacenamiento utilizados por MongoDB y otros componentes del sistema fueran rápidos y confiables. Utilizando herramientas como **fio** (Flexible I/O Tester), se verificó que las operaciones de entrada y salida en los discos fueran eficientes y estables bajo diferentes condiciones de uso.
- **Pruebas de Escalabilidad:** Para asegurarse de que el sistema pudiera escalar según la demanda, se realizaron simulaciones de escalabilidad horizontal en Azure, aumentando temporalmente los recursos asignados a la máquina virtual (CPU y memoria). Esto permitió verificar que el sistema fuera capaz de ajustarse sin interrupciones del servicio cuando se aumentaran los recursos.

Testing Funcional del Servidor de Azure

Se realizaron pruebas funcionales sobre el servidor de Azure para evaluar su capacidad de respuesta y rendimiento bajo condiciones operativas normales y de alta demanda:

- **Pruebas de Velocidad de Respuesta:** - **Número de solicitudes realizadas:** 5,000 solicitudes al servidor. - **Tiempo de respuesta promedio:** 1.2 segundos bajo condiciones normales. - **Tiempo de respuesta bajo carga máxima:** 3.5 segundos con 500 solicitudes simultáneas. - Estas pruebas validaron que el servidor mantiene tiempos de respuesta aceptables incluso bajo alta carga.
- **Pruebas de Carga del Servidor:** - **Usuarios simulados:** Hasta 500 usuarios concurrentes. - **Transacciones por segundo en pico:** 50 transacciones. - **Capacidad de manejo de carga:** El servidor sostuvo una carga operativa sin degradación significativa del rendimiento. - **Configuraciones de red y disco:** Optimizadas para permitir mayor flujo de datos y mejor tiempo de respuesta. - Estas pruebas demostraron la robustez del servidor para manejar aumentos repentinos en la demanda y la efectividad de las optimizaciones de infraestructura.
- **Pruebas de Conectividad y Red:** - **Pérdida de paquetes:** Menos del 0.5- **Latencia promedio:** 75 ms a través de múltiples regiones. - **Ancho de banda alcanzado:** Hasta 200 Mbps en pruebas de velocidad. - Estas pruebas verificaron la fiabilidad y la eficiencia de la red dentro del entorno de Azure, asegurando una conectividad consistente y rápida.

Validación

13 . Validación de la Hipótesis y Cumplimiento de los Objetivos

La validación de las hipótesis y los objetivos propuestos en esta memoria se llevó a cabo mediante un análisis exhaustivo del funcionamiento del sistema en entornos controlados y reales, contrastando los resultados obtenidos con las expectativas iniciales. Actualmente, dos trabajadores en WOLKE se encargan de configurar los sensores, lo que permitió realizar pruebas representativas del impacto del sistema en un escenario real. Las pruebas de comparación se efectuaron calculando los tiempos promedio entre estos trabajadores, quienes habitualmente realizan esta tarea. Para garantizar consistencia en los resultados, todas las pruebas se realizaron configurando dos **tabs**, cada una con cuatro **elements**, replicando configuraciones típicas en el manejo diario de los sensores. Este enfoque no solo permitió simular las condiciones reales de trabajo, sino también evaluar cómo el sistema soporta configuraciones comunes, asegurando resultados fiables y relevantes para las operaciones actuales.

13 .1. Reducción Significativa en el Tiempo de Configuración

La integración de la API y la plataforma web administrativa mostró una reducción del tiempo en la estructuración de los archivos JSON que definen la configuración de los sensores. Al facilitar la creación y actualización de estos archivos de configuración a través de una interfaz gráfica y evitar el uso de herramientas como Postman, se optimizó el tiempo necesario para este proceso. en que se trata la prueba (cantidad de tabs, con elements, etc)

Tarea	**Método Tradicional (Promedio)**	**Método Propuesto (Promedio)**
Actualización de parámetros	15 minutos	9 minutos

Tabla 1: Comparación de tiempos en la estructuración de archivos JSON para configuración de sensores.

13 .2. Simplificación del Proceso de Configuración

La plataforma web administrativa, al centralizar la gestión de los archivos de configuración de los sensores en una interfaz unificada, simplificó la estructuración y edición de configuraciones. Esto eliminó la necesidad de herramientas de terceros para manipular manualmente los archivos JSON y redujo la probabilidad de errores humanos en el proceso. La interfaz intuitiva facilitó la gestión de configuraciones, especialmente para los trabajadores en terreno que ya no requieren software especializado para cada sensor. decir que los usuarios dijeron los criterios cualitativos.

Aspecto	**Método Tradicional**	**Método Propuesto**
Interfaz de usuario	Requiere herramientas como Postman	Plataforma única e intuitiva
Curva de aprendizaje	Alta, por diversidad de herramientas	Baja, gracias a la estandarización
Errores en configuración	Frecuentes, debido a la complejidad de herramientas	Mínimos, por la unificación

Tabla 2: Evaluación de la simplificación en el proceso de estructuración de configuraciones.

13 .3. Gestión Centralizada Eficiente

La administración centralizada de la base de datos, combinada con la API, permitió una organización eficiente de los datos y facilitó la edición de configuraciones a través de la plataforma web. Esto contribuyó a una mayor visibilidad y control en tiempo real de los archivos de configuración, aunque el sistema no establece conexión directa con los sensores. La reducción de errores en la estructuración de los archivos y la capacidad de tomar decisiones rápidamente basadas en datos almacenados fortalecen la eficiencia operativa en el manejo de configuraciones.

Métrica	**Gestión Manual (Promedio)**	**Gestión Centralizada (Promedio)**
Errores de configuración	12 %	2 %
Tiempo de reacción	1 día	1 hora

Tabla 3: Comparación de eficiencia en la administración centralizada de configuraciones.

hline para separar tabla

14 . Cumplimiento de los Objetivos

En relación con los objetivos, los resultados son los siguientes:

- **Implementación de una API funcional**: La API permite realizar operaciones CRUD esenciales para la gestión de archivos de configuración de los sensores, cumpliendo con el objetivo de crear un sistema de gestión de datos eficiente.
- **Optimización de la base de datos**: La base de datos fue optimizada para manejar grandes volúmenes de información, asegurando un rendimiento adecuado en la consulta y almacenamiento de configuraciones.

Métrica	**Resultado**
Capacidad de almacenamiento	Soporta más de 5,000 registros diarios
Tiempo de consulta promedio	Menor a 200 ms

Tabla 4: Resultados de la optimización de la base de datos.

- **Desarrollo de la plataforma web administrativa**: La plataforma web se desarrolló para gestionar la base de datos y facilitar la creación y actualización de archivos JSON, permitiendo una administración centralizada y simplificada de las configuraciones.
- **Despliegue en Azure**: El sistema fue desplegado en Azure, proporcionando un entorno de producción seguro y escalable que soporta la gestión remota de configuraciones.

Métrica	**Resultado**
Disponibilidad del sistema	99.9 %
Tiempo de respuesta promedio	450 ms

Tabla 5: Evaluación del despliegue en Azure.

En conjunto, los resultados obtenidos permiten validar tanto las hipótesis como los objetivos planteados, confirmando que la solución desarrollada mejora significativamente el proceso de configuración y mantenimiento de los sensores de WOLKE

Conclusión

15 . Logros del Sistema

La presente memoria ha permitido desarrollar un sistema integral que cumple con los objetivos trazados en relación con la configuración y gestión de sensores para la empresa WOLKE. El desarrollo de una API robusta, una base de datos optimizada, y una plataforma web administrativa desplegada en un entorno seguro y escalable en Azure han sido los pilares fundamentales de este proyecto. Cada uno de estos elementos ha sido implementado con el fin de mejorar el proceso de configuración y mantenimiento de sensores de manera eficiente y sin la necesidad de conexiones físicas.

Uno de los logros más significativos ha sido la reducción de los tiempos de configuración de los sensores. Antes de la implementación de este sistema, el proceso de configuración manual podía demorar entre 15 y 30 minutos por sensor. Con el nuevo sistema, este tiempo se ha reducido a un rango de entre 3 y 8 minutos, dependiendo de la cantidad de tabs y elementos configurables involucrados. Esta mejora no solo optimiza los tiempos operativos, sino que también reduce considerablemente los costos asociados a la instalación de sensores, especialmente en ubicaciones remotas o de difícil acceso.

Otro aspecto destacado es la centralización y automatización de la gestión de sensores a través de la plataforma web. Al unificar todas las funciones en una interfaz gráfica intuitiva y simplificada, se ha eliminado la necesidad de utilizar software específico para cada tipo de sensor. La integración de la API y la base de datos centralizada ha permitido gestionar más de 500 operaciones CRUD por segundo, lo que garantiza una escalabilidad significativa y la capacidad de manejar grandes volúmenes de datos sin comprometer el rendimiento del sistema.

El despliegue en Azure ha proporcionado un entorno seguro y confiable, con una disponibilidad del sistema cercana al 99.95 %. Esto ha asegurado que el sistema esté siempre disponible para la configuración y gestión remota de sensores desde cualquier ubicación geográfica, reduciendo las interrupciones del servicio y mejorando la continuidad operativa.

En Conclusión, los resultados obtenidos a lo largo de este proyecto demuestran que el sistema desarrollado cumple con los requisitos planteados al inicio, mejorando de manera sustancial los tiempos de configuración, la gestión centralizada y la escalabilidad del sistema. Además, se ha proporcionado a WOLKE una herramienta que no solo optimiza sus procesos internos, sino que también le permite proyectarse a largo plazo con un sistema robusto y adaptable.

16 . Trabajo Futuro y Posibles Mejoras

Aunque el sistema desarrollado ha alcanzado los objetivos propuestos, existe margen para futuras mejoras que podrían optimizar aún más su funcionalidad y expandir su alcance.

La plataforma web administrativa, aunque funcional y eficiente, podría beneficiarse de una mayor personalización y flexibilidad. Una posible mejora sería permitir a los usuarios configurar flujos de trabajo

personalizados según el tipo de sensor o las condiciones específicas de instalación. Esto permitiría una gestión más ágil y adaptada a las necesidades de cada proyecto, aumentando la versatilidad del sistema.

Otro aspecto que podría ser abordado en futuras versiones es la inclusión de soporte para una mayor diversidad de sensores. Actualmente, el sistema está diseñado para una gama específica de sensores de WOLKE, pero la incorporación de tecnologías de Internet de las Cosas (IoT) y redes LPWAN podría ampliar su aplicabilidad a un mayor número de dispositivos. Esto no solo aumentaría la escalabilidad del sistema, sino que también lo haría más adaptable a diferentes entornos y necesidades de monitoreo.

Finalmente, un área clave de mejora es la incorporación de módulos de análisis predictivo. Integrar algoritmos de machine learning para predecir fallas o determinar cuándo es necesario realizar mantenimiento preventivo en los sensores proporcionaría un valor añadido al sistema, mejorando la toma de decisiones basada en datos y reduciendo aún más los costos operativos a largo plazo.

En conclusión, este proyecto no solo ha sentado las bases para una solución eficiente y escalable, sino que también ha abierto la puerta a futuras innovaciones que podrían fortalecer y ampliar aún más el sistema. El trabajo futuro permitirá seguir mejorando la experiencia de los usuarios, optimizar la operación de los sensores y asegurar que WOLKE siga manteniendo una ventaja competitiva en el sector de la telemetría.

Referencias

Referencias

[1] Milesight ToolBox: NFC Configuration Application for LoRaWAN Sensors. Disponible en: `https://www.milesight-iot.com/toolbox/`

[2] Wang, L., et al. *NFC-Based Wireless Sensing for Smart Agriculture: A Case Study on Fruit Ripeness Classification.* IEEE Access, 2020.

[3] Kim, J., et al. *Energy Harvesting and Wireless Sensor Networks: A Review of Recent Work and Future Challenges.* IEEE Communications Surveys & Tutorials, 2018.

[4] Zhang, H., et al. *Bluetooth Low Energy (BLE) Applications in IoT: An Overview and Future Directions.* Sensors, 2021.

[5] E. Mackensen, M. Lai and T. M. Wendt, *"Bluetooth Low Energy (BLE) based wireless sensors,"* SENSORS, 2012 IEEE, Taipei, Taiwan, 2012, pp. 1-4, doi: 10.1109/ICSENS.2012.6411303. keywords: Sensor systems;Wireless sensor networks;Wireless communication;Bluetooth;Data communication;Advertising,

[6] React Native: `https://reactnative.dev`

[7] M. Masse, REST API Design Rulebook, O'Reilly Media, 2011.

[8] Gupta, A., et al. *NoSQL Databases: Critical Analysis and Comparison.* 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), 2017.

[9] Vercel, *Next.js Documentation: The React Framework for Production.* Vercel, 2024.

[10] Microsoft, *Azure Cloud Documentation: Cloud Computing Services.* Microsoft, 2024.

[11] Docker, *Docker Documentation: Empowering App Development for Developers.* Docker, 2024.

[12] NGINX, *NGINX Documentation: High Performance Load Balancer, Web Server, and Reverse Proxy.* NGINX, 2024.