Módulo web de administración basado en la Tecnología de Gemelo Digital para la plataforma Smart campus UIS.

Jean Carlo Rodríguez Pico, Juan Camilo Robayo Giraldo, Juan Pablo Ávila Quitian

Trabajo de Grado para Optar el Título de Ingeniero de Sistemas

Director
Gabriel Rodrigo Pedraza Ferreira
Doctor en Ingeniería de Software

Codirector
Henry Andrés Jiménez Herrera
Magister en Ingeniería de Sistemas e Informática

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingeniería De Sistemas E Informática
Bucaramanga
2026

Dedicatoria

A Dios por brindarnos la fortaleza y paciencia necesaria para afrontar los momentos de dificultad y tensión durante el pregrado, por ser una guía constante en este recorrido lleno de exigencias y satisfacciones durante nuestra formación académica.

A nuestras familias, por su apoyo incondicional durante estos años de estudio, por acompañarnos en los momentos de alegría, sostenernos en las dificultades y celebrar con nosotros cada logro alcanzado. Su amor, comprensión, confianza y aliento constante fueron el impulso necesario para superar cada desafío. Gracias por formarnos en valores y enseñarnos la importancia de la constancia, el esfuerzo, la disciplina, el trabajo en equipo y la perseverancia.

A nuestros docentes, por compartir su conocimiento, orientación y compromiso con nuestra formación profesional; a nuestros compañeros de carrera por el apoyo, colaboración y aprendizajes compartidos y cada una de las personas de la sede del Socorro y la sede principal en Bucaramanga de la Universidad Industrial de Santander que contribuyeron en nuestro crecimiento personal y profesional. Su acompañamiento y enseñanza fueron fundamentales para la culminación de esta etapa.

A nuestro profesor y director de Trabajo de Grado, por su valiosa orientación, disposición y compromiso con el desarrollo del proyecto. Sus aportes y conocimientos nos permitieron afrontar los desafíos del proceso y fortalecer nuestras competencias.

Agradecimientos

Agradecemos a Dios por brindarnos la fortaleza, sabiduría y perseverancia necesarias para culminar esta etapa formativa, por acompañarnos en este camino lleno de aprendizajes retos, y crecimiento personal y profesional.

A nuestras familias por su apoyo incondicional, confianza y comprensión durante todo el proceso de formación académica. Su acompañamiento diario, generosidad, aliento motivador y confianza, nos impulsó durante estos años de exigencias académicas y aprendizajes.

Agradecemos a la Universidad Industrial de Santander por abrirnos las puertas permitiendo cumplir sueños personales y profesionales. Por los espacios académicos, recursos y formación necesaria para nuestro desarrollo como ingenieros de sistemas.

Finalmente, agradecemos a todas las personas que contribuyeron en el desarrollo de este Trabajo de Investigación y a la culminación satisfactoria de esta etapa académica.

Tabla de Contenido

Pág.

Lista de Tablas

PÁG.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Glosario

API: Application Programming Interface

BACKEND: Parte lógica del sistema.

BD: Base de Datos.

CRUD: Create, Read, Update, Delete.

DTO: Data Transfer Object.

ENDPOINT: Punto de acceso a un servicio.

FRONTEND: Interfaz de usuario.

HTTP: Hypertext Transfer Protocol.

IoT: Internet of Things.

JPA: Java Persistence API.

JSON: JavaScript Object Notation.

JWT: JSON Web Token.

MQTT: Message Queuing Telemetry Transport.

ORM: Object Relational Mapping.

REST: Representational State Transfer.

SPA: Single Page Application.

UI: User Interface.

UX: User Experience.

Resumen

Título: Módulo web de administración basado en la Tecnología de Gemelo Digital para la plataforma Smart Campus UIS*

Autor: Juan Pablo Ávila Quitian, Juan Camilo Robayo Giraldo, Jean Carlo Rodríguez Pico**

Palabras Clave: IoT, Smart Campus, Aplicación Web, Gemelo Digital

Descripción:

El Internet de las Cosas (IoT) se ha consolidado como una tecnología fundamental para la transformación digital, permitiendo la interconexión de dispositivos físicos para la captura y transmisión de información. La iniciativa Smart Campus UIS, desarrollada por investigadores de la Universidad Industrial de Santander, busca integrar aplicaciones y dispositivos IoT en el entorno universitario. Sin embargo, la plataforma presentaba limitaciones relacionadas con la administración de dispositivos, la trazabilidad y el monitoreo de los recursos conectados. En este trabajo se desarrolló un módulo web de administración basado en la tecnología de Gemelo Digital para la plataforma Smart Campus UIS. La solución implementada permite representar virtualmente los dispositivos IoT mediante gemelos digitales sincronizados con su contraparte física, facilitando la gestión centralizada, la visualización de telemetría y la supervisión del estado. La metodología se estructuró en cinco fases, como resultado, se construyó un prototipo funcional para la administración de dispositivos IoT y sus representaciones digitales dentro de la plataforma. El componente de administración implementado habilita la posibilidad de tomar decisiones en la plataforma Smart Campus UIS utilizando el estado construido por el sistema y representado mediante el gemelo digital. La validación se realizó mediante escenarios de prueba que permitieron verificar la gestión de dispositivos, la sincronización bidireccional de la información y el registro de eventos. Los resultados evidenciaron el correcto funcionamiento del sistema, fortaleciendo la trazabilidad de las operaciones y proporcionando una base tecnológica para la evolución de la plataforma Smart Campus UIS.

---

*Trabajo de grado

**Facultad de Ingenierías Físico mecánicas. Escuela de Ingeniería de Sistemas. Director: Gabriel Rodrigo Pedraza Ferreira PhD. Ciencias de la Computación

Abstract

Title: Web-Based Administration Module Using Digital Twin Technology for the Smart Campus UIS Platform.

Author: Juan Pablo Ávila Quitian, Juan Camilo Robayo Giraldo, Jean Carlo Rodríguez Pico.

Key Words: IoT, Smart Campus, Web Application, Digital Twin.

Description:

The Internet of Things (IoT) has become a fundamental technology for digital transformation, enabling the interconnection of physical devices for data acquisition and transmission. The Smart Campus UIS initiative, developed by researchers at Universidad Industrial de Santander, seeks to integrate IoT applications and devices within the university environment. However, the platform presented limitations related to device management, traceability, and monitoring of connected resources.

This work developed a web-based administration module based on Digital Twin technology for the Smart Campus UIS platform. The implemented solution provides virtual representations of IoT devices through digital twins synchronized with their physical counterparts, enabling centralized management, telemetry visualization, and device status monitoring.

The methodology was structured into five phases. As a result, a functional prototype was developed for managing IoT devices and their digital representations within the platform. The implemented administration component enables decision-making in the Smart Campus UIS platform by using the system state represented through the digital twin. Validation was carried out through test scenarios that verified device management, bidirectional synchronization, and event logging. The results demonstrated the correct operation of the system, strengthening operational traceability and providing a technological foundation for the evolution of the Smart Campus UIS platform.

---

*Bachelor Thesis

**Faculty of Physical-Mechanical Engineering. School of Systems and Computer Engineering.

Advisor: Gabriel Rodrigo Pedraza Ferreira, PhD in Computer Science.

Introducción

El concepto de Smart Campus surge como una extensión de las ciudades inteligentes adaptada al entorno universitario. Estas iniciativas buscan integrar dispositivos IoT dentro de las instituciones educativas con el propósito de optimizar el consumo energético, mejorar la seguridad, administrar los recursos de manera eficiente y fortalecer los procesos académicos mediante el uso de tecnologías emergentes.

Investigadores de la Universidad Industrial de Santander (UIS) desarrollaron la iniciativa Smart Campus UIS como una plataforma orientada a integrar aplicaciones y dispositivos IoT dentro del campus universitario. Esta iniciativa se encuentra en proceso de consolidación y busca convertirse en una infraestructura tecnológica capaz de centralizar información, apoyar la toma de decisiones y facilitar el desarrollo de soluciones para la gestión institucional. Sin embargo, durante su evolución se identificaron limitaciones relacionadas con la administración centralizada de dispositivos y aplicaciones, así como con la representación integrada del estado de los dispositivos conectados.

De manera paralela, el paradigma del Gemelo Digital (Digital Twin) ha surgido como una estrategia tecnológica para representar virtualmente un sistema físico mediante una conexión bidireccional que permite reflejar su estado en tiempo real. En el contexto del IoT, este enfoque facilita la sincronización entre los dispositivos físicos y sus representaciones digitales, proporcionando una visión unificada de su estado para apoyar los procesos de monitoreo y administración.

El desarrollo del módulo web de administración basado en gemelos digitales para el Smart Campus UIS representa una oportunidad para integrar estos avances tecnológicos en el entorno

universitario. A través de este módulo se pretende ofrecer una interfaz que centralice la gestión de dispositivos y aplicaciones, incorporando un modelo de sincronización bidireccional que refleje las condiciones del campus. De esta manera, se busca superar las limitaciones actuales y consolidar un entorno confiable, escalable y flexible que fortalezca el proceso de digitalización institucional.

Este proyecto se enmarca en la convergencia de tendencias tecnológicas globales como el Internet de las Cosas, los Smart Campus y los Gemelos Digitales, y responde a la necesidad de la Universidad Industrial de Santander de contar con herramientas que optimicen la administración y gestión del campus universitario. Su implementación contribuirá al fortalecimiento del desarrollo tecnológico institucional y a la generación de conocimiento en un campo emergente, posicionando a la universidad como referente en la aplicación de gemelos digitales en entornos académicos inteligentes.

1. Objetivos

1.1 Objetivo General

Desarrollar un módulo de administración para la plataforma Smart Campus UIS que permita la gestión de la plataforma a través del enfoque de Gemelo Digital.

1.2 Objetivos Específicos

- Determinar el conjunto de requerimientos para la gestión de la plataforma Smart Campus UIS, que involucre la gestión de aplicaciones de sus dispositivos asociados.
- Diseñar la arquitectura del módulo de administración basado en Gemelo Digital, definiendo los componentes del software, flujo de datos y mecanismos de sincronización entre dispositivos físicos y sus representaciones virtuales.
- Implementar un prototipo funcional del módulo web de administración, que integre gestión de dispositivos y aplicaciones IoT con sus gemelos digitales, asegurando coherencia y trazabilidad en tiempo real.
- Validar el módulo de administración propuesto mediante escenarios de configuración de aplicaciones y dispositivos en la plataforma Smart Campus UIS, verificando su coherencia, trazabilidad y funcionamiento.

2. Estado del arte

El Internet de las Cosas (IoT) se ha consolidado como una de las tecnologías fundamentales para la construcción de entornos inteligentes, permitiendo la interconexión de dispositivos físicos mediante redes digitales para la adquisición, procesamiento y análisis de datos en tiempo real. Según Santhi, Rajendra y Vijayalakshmi, el IoT se fundamenta en la integración de sensores, sistemas embebidos, redes de comunicación y plataformas de procesamiento que permiten monitorear y controlar objetos físicos de forma remota. Los autores destacan que una arquitectura IoT está compuesta por dispositivos inteligentes, capas de comunicación, servicios de procesamiento y aplicaciones de usuario, permitiendo la automatización de procesos y la toma de decisiones basada en datos (Santhi, Rajendra & Vijayalakshmi, 2006).

La aplicación de estas tecnologías ha dado origen al concepto de Smart Campus, el cual busca transformar los entornos universitarios mediante la integración de dispositivos conectados, plataformas de monitoreo y sistemas de análisis de información. Diversas investigaciones reportan aplicaciones orientadas al monitoreo energético, control de acceso, gestión de espacios académicos, supervisión ambiental y optimización de recursos institucionales. Sin embargo, varios estudios identifican desafíos relacionados con la interoperabilidad entre sistemas heterogéneos, la administración centralizada de dispositivos IoT, la escalabilidad de las plataformas y la consistencia de la información generada por múltiples fuentes de datos.

Con el propósito de abordar estas limitaciones, diferentes investigaciones han incorporado el paradigma de Gemelo Digital (Digital Twin), el cual permite construir una representación virtual de un activo físico y mantener una sincronización continua entre ambos entornos. De acuerdo con Tao et al. (2019), los gemelos digitales facilitan el monitoreo en tiempo real, la simulación de

escenarios, la validación de configuraciones y la toma de decisiones basada en información actualizada. En entornos IoT, este enfoque contribuye a mejorar la trazabilidad de los dispositivos, fortalecer la coherencia de los datos mediante mecanismos de sincronización bidireccional y optimizar los procesos de gestión y supervisión de infraestructuras inteligentes.

Estos son los elementos fundamentales que conforman una arquitectura basada en Internet de las Cosas, destacando la interacción entre dispositivos físicos, sensores, redes de comunicación, plataformas de procesamiento y aplicaciones de usuario. Esta integración permite la captura, transmisión y análisis de datos provenientes del entorno físico, facilitando la automatización de procesos y la toma de decisiones basada en información en tiempo real. Asimismo, evidencia cómo los diferentes componentes tecnológicos trabajan de manera coordinada para construir ecosistemas inteligentes, constituyendo la base sobre la cual se desarrollan iniciativas de Smart Campus y soluciones soportadas por tecnologías de Gemelo Digital.

Actualmente, plataformas como Azure Digital Twins, AWS IoT Device Management y Google Cloud IoT ofrecen mecanismos avanzados para la administración de dispositivos mediante representaciones virtuales, monitoreo remoto y sincronización de información en tiempo real. Estas soluciones han demostrado ser efectivas en entornos industriales y empresariales, facilitando la gestión de infraestructuras complejas y grandes volúmenes de dispositivos conectados. Sin embargo, se trata de plataformas propietarias cuyos servicios dependen de modelos de licenciamiento y recursos de pago, lo que puede limitar su adopción en entornos académicos y de investigación. Adicionalmente, estas soluciones están diseñadas para escenarios generales y no contemplan de manera específica las necesidades de administración centralizada de dispositivos, aplicaciones y servicios presentes en un ecosistema Smart Campus universitario.

En este contexto, surge la necesidad de desarrollar soluciones especializadas que integren la gestión centralizada de dispositivos IoT con el paradigma de Gemelo Digital, adaptadas a los requerimientos particulares de la plataforma Smart Campus UIS. Este enfoque permite fortalecer la escalabilidad del sistema, mejorar la trazabilidad de los eventos generados por los dispositivos y garantizar una mayor coherencia de la información mediante mecanismos de sincronización bidireccional entre los dispositivos físicos y sus representaciones digitales.

3. Marco de Referencia

En este capítulo se presentan los fundamentos teóricos y tecnológicos que sustentan el desarrollo del módulo de administración basado en tecnología de Gemelo Digital para la plataforma Smart Campus UIS.

3.1 Internet de las cosas (IoT)

El Internet de las Cosas (IoT) se define como un paradigma tecnológico que permite la interconexión de objetos físicos mediante sensores, actuadores y tecnologías de comunicación, con el fin de recopilar, procesar e intercambiar información del entorno en tiempo real. Este enfoque integra redes de sensores inalámbricos, sistemas embebidos y plataformas de computación en la nube, permitiendo que los dispositivos interactúen entre sí y con aplicaciones digitales. De esta manera, el IoT posibilita la creación de entornos inteligentes capaces de monitorear variables físicas, analizar datos y ejecutar acciones automáticas orientadas a la optimización de procesos y la toma de decisiones basada en información contextual (Gubbi et al., 2013).

La arquitectura del IoT está compuesta de varias capas funcionales que permiten la comunicación entre el entorno físico y las aplicaciones. Estas son: la capa de percepción, encargada de la adquisición de datos mediante sensores y actuadores; la capa de comunicación, que transmite

la información mediante protocolos como MQTT, HTTP o WebSockets; la capa de procesamiento, donde se realiza el almacenamiento y análisis de datos; y por último la capa de aplicación, que proporciona interfaces para la visualización y administración del sistema. Esta estructura permite el desarrollo de plataformas escalables facilitando la gestión centralizada de dispositivos y la integración de múltiples fuentes de información en tiempo real.

El IoT ha sido ampliamente aplicado en diferentes dominios como ciudades inteligentes, automatización industrial, monitoreo ambiental y campus inteligentes. En este contexto, esta tecnología permite integrar sensores, plataformas de monitoreo y aplicaciones web para la gestión eficiente de infraestructura, seguridad y consumo energético. Adicionalmente, constituye la base tecnológica para el desarrollo del Smart Campus UIS, permitiendo la interconexión de dispositivos, la recolección de datos sincronizados y la administración centralizada de recursos mediante plataformas digitales orientadas a mejorar la operación del campus universitario.

3.2 Smart Campus

El concepto de Smart Campus surge como una adaptación del modelo de Smart City en el entorno universitario, incorporando tecnologías IoT para mejorar la eficiencia operativa, la sostenibilidad y la gestión institucional. Un campus inteligente integra sensores, plataformas digitales y aplicaciones que permiten monitorear variables ambientales, consumo energético, seguridad, movilidad y uso de infraestructura.

Las soluciones Smart Campus permiten centralizar la información proveniente de múltiples dispositivos y facilitar la toma de decisiones basada en datos. Asimismo, estas plataformas promueven la automatización de procesos y la optimización del uso de recursos.

3.3 Gemelo Digital

El Gemelo Digital (Digital Twin) es una representación virtual de un objeto o sistema físico diseñada para reflejar con precisión su comportamiento en el mundo real. Esta representación abarca el ciclo de vida completo del activo, se actualiza mediante dato sincronizados y emplea simulación, aprendizaje automático y razonamiento para apoyar la toma de decisiones. Este enfoque permite conectar el entorno físico con su contraparte digital, facilitando el monitoreo continuo, el análisis del rendimiento y la optimización del sistema a lo largo del tiempo (IBM, 2020).

Los gemelos digitales son construidos a partir de la integración de sensores, plataformas IoT, modelos de datos y servicios de procesamiento que representan digitalmente el estado, las propiedades y el comportamiento de los dispositivos físicos. Esta sincronización entre el mundo físico y virtual realiza simulaciones, validar configuraciones y analizar escenarios sin afectar el entorno real. Asimismo, la disponibilidad de datos sincronizados identifica anomalías, predecir fallos y optimizar la operación de los sistemas mediante el uso de modelos analíticos y técnicas de inteligencia artificial.

En el contexto de plataformas IoT, el uso de gemelos digitales permite mantener una comunicación bidireccional entre los dispositivos físicos y su representación virtual, garantizando la coherencia de los datos y la trazabilidad de las operaciones.

3.4 Aplicaciones Web

Las aplicaciones web constituyen la capa de interacción entre usuarios y sistemas IoT. Estas aplicaciones permiten visualizar dispositivos, monitorear estados, modificar configuraciones y administrar recursos desde una interfaz centralizada.

El desarrollo de aplicaciones web modernas se basa en arquitecturas cliente-servidor, donde el frontend gestiona la interfaz de usuario y el backend proporciona servicios de negocio mediante APIs. Este enfoque permite construir plataformas escalables, desacopladas y mantenibles.

3.5 Servicios REST

Los servicios REST constituyen un estilo arquitectónico para el desarrollo de APIs que permitiendo la comunicación entre aplicaciones mediante el protocolo HTTP. Este enfoque utiliza operaciones estándar como GET, POST, PUT y DELETE para gestionar recursos.

En plataformas IoT, los servicios REST permiten registrar dispositivos, consultar estados, modificar configuraciones y administrar aplicaciones. La arquitectura REST facilita la interoperabilidad entre sistemas y permite desacoplar el frontend del backend.

3.6 Comunicación bidireccional

Las plataformas IoT requieren comunicación sincronizada entre dispositivos y sistemas de administración. Se emplean mecanismos de comunicación bidireccional que permiten enviar y recibir datos de manera continua.

La comunicación bidireccional permite que los cambios realizados desde la interfaz web se reflejen en los dispositivos físicos y que las actualizaciones del entorno físico se visualicen en sincronizado con la plataforma.

3.7 Broker

Un broker de mensajería es un componente intermedio que permite la comunicación entre productores y consumidores de datos. En arquitecturas IoT, el broker gestiona la distribución de mensajes entre dispositivos y aplicaciones. Este componente desacopla los dispositivos del sistema central y facilita la comunicación asíncrona.

3.8 Protocolo MQTT

MQTT es un protocolo de mensajería ligero basado en el modelo publish/subscribe, ampliamente utilizado en sistemas IoT. Este protocolo permite que los dispositivos publiquen datos en tópicos específicos y que las aplicaciones se suscriban a dichos tópicos para recibir información.

MQTT es adecuado para entornos con dispositivos distribuidos debido a su bajo consumo de ancho de banda y su capacidad de comunicación en tiempo real. Además, permite la comunicación bidireccional entre dispositivos y sistemas de administración.

3.9 Dashboard

Los dashboards son interfaces visuales que representan información de manera gráfica e interactiva, facilitando el monitoreo y la toma de decisiones en tiempo real. En plataformas IoT, los dashboards permiten visualizar el estado de los dispositivos, métricas operativas, alertas y configuraciones desde una interfaz centralizada. Estas herramientas integran gráficos, indicadores y paneles de control que mejoran la interpretación de los datos generados por sensores y aplicaciones.

4. Marco Metodológico

El presente proyecto de grado se desarrolló bajo una metodología de investigación aplicada con componente tecnológico, orientado al diseño, construcción y validación de una solución informática para la plataforma Smart Campus UIS. El trabajo de grado se desarrolló teniendo en cuenta que el propósito principal del proyecto no se limitó a la descripción teórica de un problema, sino que estuvo dirigido a la creación de un módulo web de administración capaz de responder a una necesidad real del entorno institucional, relacionada con la gestión centralizada de aplicaciones

y dispositivos IoT mediante el enfoque de Gemelo Digital, posteriormente se explica cada fase y las actividades que se realizaron en cada una.

Figura 1.
Diagrama de las respectivas fases del proyecto.

Nota: Diagrama explicando el orden metodológico para la ejecución del proyecto.

4.1 Fase de Ambientación Tecnológica

En esta primera fase, se realiza una investigación preliminar, apropiación conceptual y técnica de las metodologías necesarias para el desarrollo del proyecto.

Actividades:

- Revisión de la literatura sobre IoT, Smart Campus y Gemelos Digitales.
- Configuración del entorno de desarrollo y exploración de tecnologías Frontend y Backend.
- Análisis de los protocolos de comunicación IoT.

4.2 Fase de Planeación y Requerimientos

En esta fase se identifican los requerimientos funcionales y no funcionales del módulo, considerando las necesidades de la gestión de dispositivos, aplicaciones y Gemelos Digitales para la plataforma Smart Campus UIS.

Actividades:

- Identificación del problema de la plataforma y análisis de las actividades del sistema.
- Definición de requerimientos funcionales y no funcionales e identificación de las restricciones técnicas.

4.3 Fase de Diseño

Se definió la arquitectura del módulo de administración basado en Gemelo Digital, estableciendo componentes del sistema, flujo de datos y mecanismo de comunicación entre dispositivo y plataforma.

Actividades:

- Diseño de la arquitectura del sistema.
- Definición de componentes.
- Diseño de la base de datos.
- Elaboración de diagramas

4.4 Fase de Implementación

En esta fase se desarrolló el prototipo funcional del módulo web de administración integrando la gestión de dispositivos, aplicaciones IoT y Gemelo Digital. Adicionalmente se implementó los servicios de backend, interfaz del frontend y mecanismos de comunicación en tiempo real.

Actividades:

- Desarrollo de dashboard.
- Desarrollo de servicios REST.
- Implementación de comunicación bidireccional.

- Integración del frontend y backend.
- Desarrollo de funcionalidades de la gestión de dispositivos.

4.5 Fase de Validación

En esta última fase se verificó el funcionamiento del módulo de administración mediante una serie de pruebas para evaluar la gestión de dispositivos, sincronización bidireccional y coherencia entre Gemelo Digital y dispositivo.

Actividades:

- Prueba de registro de dispositivos y de gestión de aplicaciones.
- Validación de gemelo digital y la sincronización bidireccional.
- Validación del dashboard de administración.
- Verificación de trazabilidad de eventos.
- Elaboración del informe final.

5. Fase de ambientación tecnológica.

En este capítulo se presentan las actividades desarrolladas durante la fase de ambientación tecnológica del proyecto, correspondiente a la primera etapa de la metodología propuesta. El propósito de esta fase fue comprender el contexto tecnológico de la plataforma Smart Campus UIS mediante la revisión de literatura especializada, el análisis de los desarrollos existentes, la caracterización del entorno tecnológico y el estudio de los principales protocolos de comunicación utilizados en soluciones IoT.

Como resultado de esta fase fue posible establecer el estado inicial de la plataforma, identificar sus principales limitaciones y definir las bases conceptuales y tecnológicas que orientaron el diseño e implementación del módulo web de administración basado en tecnología de

Gemelo Digital. Las actividades desarrolladas permitieron comprender la interacción entre los diferentes componentes del sistema y justificar las decisiones arquitectónicas adoptadas durante el desarrollo del proyecto.

5.1 Revisión de literatura y antecedentes tecnológicos

Como parte de la fase de ambientación tecnológica se realizó una revisión de literatura relacionada con las principales tecnologías involucradas en el desarrollo del proyecto, incluyendo Internet de las Cosas (IoT), Smart Campus y Gemelos Digitales. Esta actividad tuvo como propósito comprender el estado actual de estas tecnologías, identificar arquitecturas de referencia y reconocer estrategias utilizadas para la integración y administración de dispositivos inteligentes.

La revisión bibliográfica permitió identificar que las plataformas Smart Campus requieren mecanismos que faciliten la administración centralizada de dispositivos, la integración de diferentes fuentes de información y la representación digital de los activos físicos para apoyar los procesos de monitoreo y toma de decisiones. Los fundamentos conceptuales y el análisis detallado de la literatura consultada se presentan en el Capítulo 2 correspondiente al Estado del Arte.

De manera complementaria, durante esta fase se revisaron los desarrollos previos realizados dentro de la iniciativa Smart Campus UIS con el propósito de comprender la evolución de la plataforma y reconocer los componentes existentes antes del inicio del presente trabajo. Este análisis permitió identificar las capacidades ya implementadas y establecer las necesidades funcionales que posteriormente fueron abordadas mediante el desarrollo del módulo de administración basado en Gemelo Digital.

5.2 Estado inicial de la plataforma Smart Campus UIS

La iniciativa Smart Campus UIS constituye un proyecto de investigación orientado al desarrollo de soluciones tecnológicas para la gestión inteligente del entorno universitario mediante

la integración de dispositivos IoT, aplicaciones web y servicios de software. Durante la fase de ambientación tecnológica se realizó un análisis del estado inicial de la plataforma Smart Campus UIS con el propósito de identificar los componentes existentes, comprender su funcionamiento y establecer el alcance del módulo de administración propuesto.

Al inicio del proyecto, la plataforma disponía de una estructura básica conformada por una base de datos, una aplicación backend y una interfaz web desarrolladas como parte de trabajos previos de la iniciativa Smart Campus UIS. Estos componentes representaban una primera aproximación a la administración de dispositivos IoT y constituían la base sobre la cual se planteó el desarrollo del presente trabajo.

La interfaz existente permitía realizar operaciones básicas de registro y eliminación de dispositivos; sin embargo, estas funcionalidades no se encontraban integradas con el funcionamiento real de los dispositivos IoT, por lo que no era posible establecer comunicación con ellos, conocer su estado operativo, visualizar información de telemetría o administrar su comportamiento desde la plataforma. De igual manera, la arquitectura existente no incorporaba un modelo de Gemelo Digital que permitiera representar virtualmente los dispositivos ni mantener sincronización entre el entorno físico y su representación digital.

Como parte del análisis realizado durante esta fase también se identificó que la plataforma carecía de mecanismos que permitieran integrar dispositivos IoT mediante protocolos de comunicación especializados, limitando las posibilidades de validación y monitoreo del sistema. En consecuencia, fue necesario diseñar e incorporar nuevos componentes de software orientados a la comunicación, sincronización y administración de dispositivos, permitiendo transformar la propuesta inicial en un prototipo funcional capaz de integrar el entorno físico con su representación digital.

Tabla 1.

Estado inicial de los componentes de Smart Campus UIS

Componentes existentes antes del proyecto	Estado
Integración de dispositivos IoT	Disponible
Visualización de información	Disponible
Servicios de comunicación	No disponible
Persistencia de datos	No disponible
Gestión centralizada de dispositivos	No disponible
Gestión centralizada de aplicaciones	No disponible
Representación mediante Gemelo Digital	No disponible
Sincronización bidireccional	No disponible

5.3 Caracterización del entorno tecnológico

Como parte de la fase de ambientación tecnológica se realizó una caracterización del entorno de desarrollo asociado a la plataforma Smart Campus UIS, con el propósito de identificar las herramientas, componentes de software y tecnologías disponibles antes del inicio del presente trabajo. Esta actividad permitió comprender la estructura tecnológica existente y establecer los elementos que podían ser reutilizados o fortalecidos durante el desarrollo del módulo de administración.

El análisis evidenció que la plataforma disponía de una arquitectura basada en una aplicación web, un componente backend y un sistema de persistencia de información. Sin embargo, dichos componentes presentaban un nivel de integración limitado y no contaban con

mecanismos que permitieran la comunicación con dispositivos IoT, la administración centralizada de recursos o la sincronización entre el entorno físico y una representación digital.

Durante esta etapa también se revisaron las tecnologías utilizadas en los diferentes componentes de la plataforma, identificando el lenguaje de programación empleado en el backend, las herramientas utilizadas para el desarrollo de la interfaz web, el sistema gestor de bases de datos y los mecanismos de comunicación disponibles. Esta caracterización permitió comprender las capacidades existentes y facilitó la definición de una estrategia de integración que garantizara la compatibilidad con los desarrollos previos de la iniciativa Smart Campus UIS.

Como resultado de este análisis se definió el entorno tecnológico sobre el cual se desarrolló el módulo de administración, procurando mantener compatibilidad con la infraestructura existente e incorporando nuevos componentes que permitieran soportar la administración de dispositivos IoT, la representación mediante Gemelos Digitales y la sincronización de información entre los diferentes elementos del sistema.

Tabla 2.
Tecnologías identificadas para el desarrollo.

Categoría	Tecnología identificada	Función
Backend	Java	Desarrollo de la lógica de negocio
Framework Backend	Spring Boot	Servicios de aplicación
Frontend	JavaScript	Desarrollo de la interfaz web
Framework Frontend	React	Construcción de la interfaz
Persistencia	MySQL	Almacenamiento de información

Comunicación	MQTT	Intercambio de mensajes IoT
Comunicación Web	WebSocket	Actualización en tiempo real
Simulación	Node-RED	Simulación e integración de dispositivos IoT

La caracterización realizada permitió establecer que la infraestructura existente ofrecía una base adecuada para el desarrollo del módulo de administración. No obstante, fue necesario incorporar componentes adicionales orientados a la integración de dispositivos IoT, la sincronización mediante Gemelo Digital y la validación funcional del sistema, manteniendo compatibilidad con la arquitectura previamente definida dentro de la iniciativa Smart Campus UIS.

5.4 Análisis de protocolos de comunicación IoT

Como parte de la fase de ambientación tecnológica se realizó un análisis de los principales protocolos de comunicación utilizados en aplicaciones basadas en Internet de las Cosas (IoT). Esta actividad tuvo como propósito identificar las características, ventajas y limitaciones de las diferentes alternativas disponibles, con el fin de seleccionar el protocolo más adecuado para soportar la comunicación entre los dispositivos IoT y el módulo de administración desarrollado para la plataforma Smart Campus UIS.

El análisis se centró en protocolos ampliamente utilizados en soluciones IoT, considerando aspectos como el modelo de comunicación, consumo de recursos, confiabilidad, facilidad de integración y aplicabilidad en entornos de monitoreo en tiempo real. La Tabla 3 presenta un resumen de los protocolos analizados durante esta fase.

Tabla 3.
Protocolos de comunicación analizados.

Protocolo	Modelo de comunicación	Principales características	Aplicabilidad

MQTT	Publicación/Suscripción	Bajo consumo de ancho de banda, comunicación asíncrona y alta eficiencia en dispositivos con recursos limitados.	Sistemas de monitoreo y dispositivos IoT.
AMQP	Mensajería orientada a colas	Alta confiabilidad, confirmación de entrega y enrutamiento avanzado de mensajes.	Sistemas empresariales y aplicaciones distribuidas.
HTTP	Cliente-Servidor	Amplia compatibilidad y facilidad de integración con aplicaciones web.	Servicios web y aplicaciones tradicionales.
CoAP	Cliente-Servidor	Protocolo ligero diseñado para dispositivos con restricciones de memoria y procesamiento.	Redes de sensores y dispositivos embebidos.
LoRaWAN	Comunicación inalámbrica de largo alcance	Bajo consumo energético y cobertura extendida para redes IoT.	Redes de sensores distribuidos y aplicaciones de ciudad inteligente.

A partir del análisis realizado se identificó que MQTT constituye una alternativa adecuada para el desarrollo del módulo de administración debido a su modelo de publicación y suscripción,

su bajo consumo de recursos y su capacidad para soportar la comunicación asíncrona entre múltiples dispositivos. Estas características facilitan el intercambio de información en tiempo real y favorecen la sincronización entre los dispositivos IoT y sus representaciones digitales, aspectos fundamentales para la implementación del paradigma de Gemelo Digital dentro de la plataforma Smart Campus UIS.

La selección de este protocolo permitió establecer una base de comunicación flexible y escalable para el desarrollo del proyecto, garantizando la interoperabilidad entre los diferentes componentes del sistema y facilitando la integración de nuevos dispositivos durante futuras etapas de evolución de la plataforma.

5.5 Análisis del paradigma de Gemelo Digital

Como parte de la fase de ambientación tecnológica se analizó la aplicabilidad del paradigma de Gemelo Digital dentro del contexto de la plataforma Smart Campus UIS, con el propósito de evaluar su capacidad para responder a las necesidades identificadas durante el análisis del estado inicial de la plataforma.

La revisión realizada permitió identificar que uno de los principales retos consistía en disponer de un mecanismo que representara el estado de los dispositivos IoT dentro de la plataforma, facilitando su administración y permitiendo mantener sincronizada la información proveniente del entorno físico. En este sentido, el paradigma de Gemelo Digital se identificó como una alternativa adecuada al proporcionar una representación virtual de los dispositivos, capaz de reflejar su estado operativo y almacenar la información asociada a su funcionamiento.

A partir del análisis efectuado se estableció que la incorporación de un Gemelo Digital permitiría integrar en una única representación la información de cada dispositivo IoT, facilitando la consulta de su estado, la visualización de variables de telemetría y la administración de sus

configuraciones desde la plataforma Smart Campus UIS. Asimismo, este enfoque proporciona una base para mantener sincronizada la información entre el entorno físico y su representación digital, favoreciendo el monitoreo continuo y la administración centralizada de los dispositivos.

Con base en estos resultados, se definió el paradigma de Gemelo Digital como el eje central del módulo de administración desarrollado en el presente trabajo, constituyéndose en el componente encargado de representar virtualmente los dispositivos IoT y apoyar la gestión de la plataforma mediante información actualizada sobre el estado de los recursos conectados. Esta decisión orientó las actividades desarrolladas durante las fases de diseño, implementación y validación del sistema.

6. Análisis y definición de requerimientos del módulo

El cumplimiento del primer objetivo se realizó una revisión de la plataforma Smart Campus UIS, comprendiendo la lógica de integración de dispositivos y servicios backend y aplicaciones de administración. Se identificaron las limitaciones relacionadas con la ausencia de un módulo para la administración de dispositivos y aplicaciones y la inexistencia de mecanismos formales de sincronización entre entorno físico y un entorno digital.

Como parte fundamental del desarrollo del proyecto se identificaron los requerimientos del sistema estableciendo un alcance preciso y las funcionalidades esperadas. A continuación, se presenta una explicación de los requerimientos funcionales y no funcionales, definición de actores del sistema y la identificación de casos de uso.

6.1 Definición de actores

Los actores del sistema son entidades que interactúan con el sistema para ejecutar los diferentes casos de uso. Para el módulo de administración basado en tecnología de Gemelo Digital se identifican los siguientes actores:

- Administrador: Responsable de la gestión y configuración del sistema. Este actor puede registrar, editar y eliminar dispositivos, asignar propiedades, gestionar configuraciones y supervisar el comportamiento de los gemelos digitales. Además, administra la estructura general de la plataforma Smart Campus UIS.
- Viewer: Usuario que accede al sistema para consultar la información de los dispositivos y su gemelo digital. Puede visualizar gráficas, monitorear variables, consultar el historial de telemetría y supervisar el estado de los dispositivos conectados.
- Dispositivo IoT: Entidad externa que interactúa con el sistema enviando datos de telemetría provenientes del entorno físico. Estos datos son utilizados para actualizar el gemelo digital y mantener la sincronización entre el dispositivo físico y su representación virtual.

6.2 Diagrama de Casos de Uso

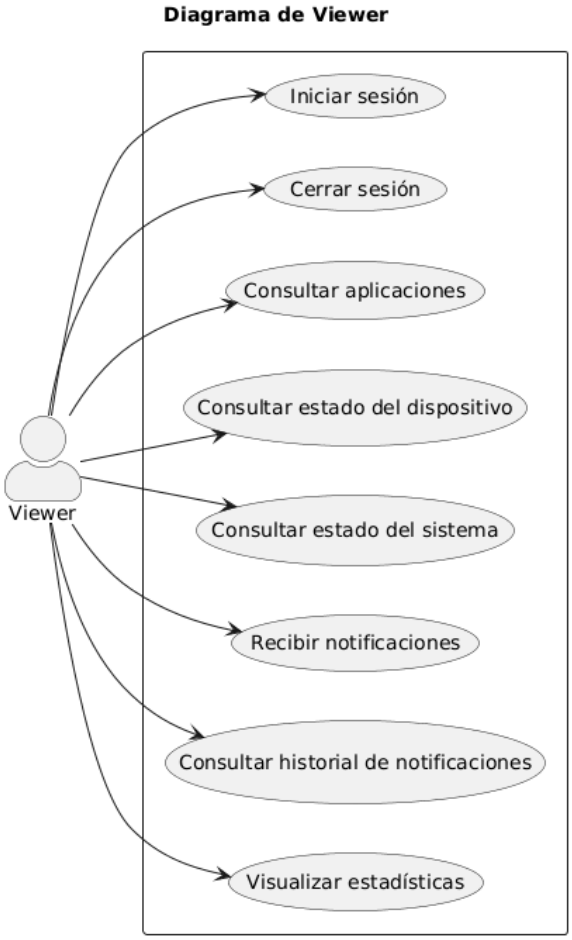
El diagrama de casos de uso permite representar de manera general las funcionalidades que ofrece el sistema y la forma en que los diferentes actores interactúan con este. A través de este diagrama se identifican las principales operaciones que el módulo web de administración debe soportar, así como las relaciones entre los actores y dichas funcionalidades.

Para el módulo de administración se identificaron dos actores humanos, correspondientes al Administrador y al Viewer, además de un actor externo representado por el Dispositivo IoT. El Administrador dispone de funcionalidades orientadas a la configuración y gestión de la plataforma, mientras que el Viewer tiene acceso únicamente a operaciones de consulta y monitoreo. Por su

parte, el Dispositivo IoT interactúa con el sistema mediante el intercambio de información de telemetría, permitiendo mantener actualizada la representación digital de los dispositivos dentro de la plataforma.

La Figura 2 presenta los casos de uso correspondientes al actor Viewer, quien dispone de funcionalidades orientadas a la consulta y supervisión de la información generada por los dispositivos IoT y sus respectivos gemelos digitales. Estas operaciones permiten visualizar el estado de los dispositivos, consultar la información de telemetría y realizar el seguimiento del comportamiento general del sistema sin modificar su configuración.

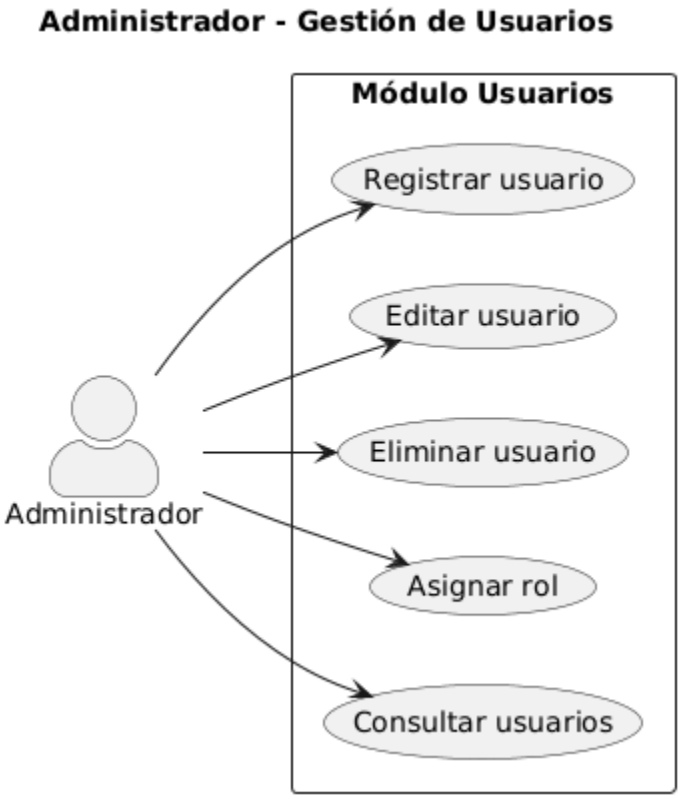
Figura 2.
Casos de uso del Viewer.



Nota: Este diagrama representa las funcionalidades disponibles para el actor Viewer dentro del sistema Smart Campus. Fuente: elaboración propia.

La Figura 3 presenta los casos de uso asociados al actor Administrador relacionados con la gestión de usuarios. Estas funcionalidades permiten administrar la información de los usuarios registrados en la plataforma, así como controlar los permisos necesarios para el acceso y operación del módulo de administración.

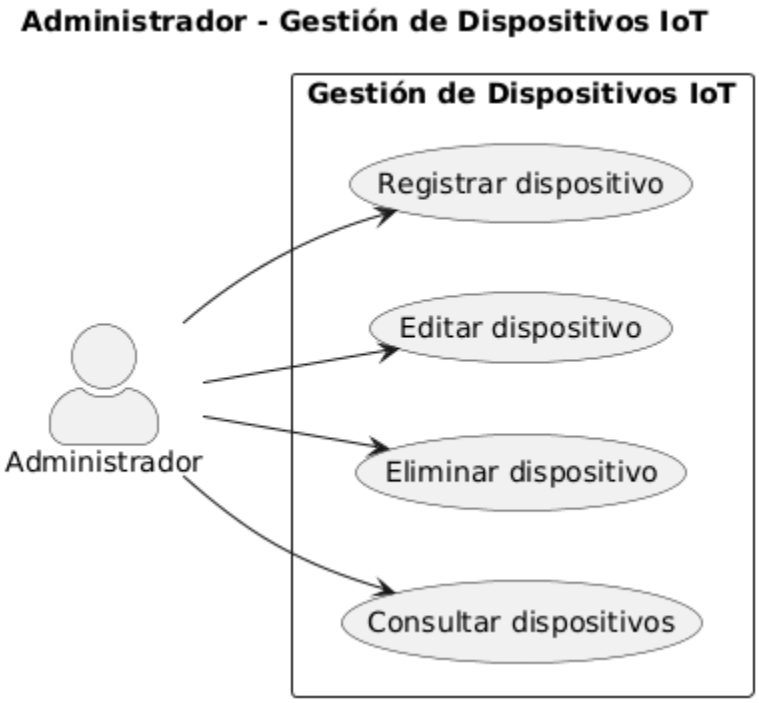
Figura 3.
Casos de uso del Administrador para la gestión de usuarios.



Nota: Este diagrama representa las funcionalidades disponibles para el actor Administrador relacionadas con la administración de usuarios dentro del sistema Smart Campus.

Las funcionalidades relacionadas con la administración de dispositivos IoT se presentan en la Figura 4. En este conjunto de casos de uso se incluyen las operaciones necesarias para registrar, modificar, eliminar y administrar los dispositivos conectados a la plataforma, constituyendo uno de los componentes principales del módulo desarrollado.

Figura 4.
Casos de uso del Administrador para la gestión de dispositivos IoT.

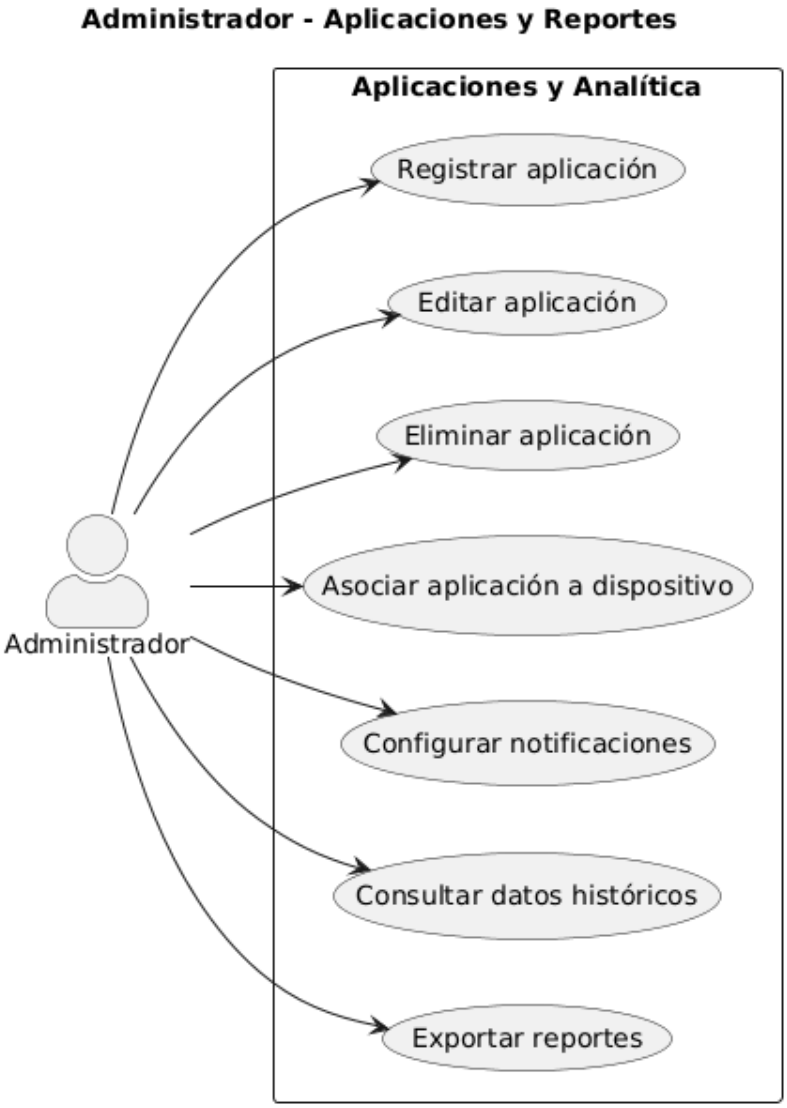


Nota: Este diagrama representa las funcionalidades disponibles para el actor Administrador relacionadas con la gestión de dispositivos IoT dentro de la plataforma Smart Campus.

Finalmente, la Figura 5 presenta los casos de uso asociados a la gestión de aplicaciones, configuración de notificaciones y consulta de información histórica. Estas funcionalidades

complementan la administración integral de la plataforma, permitiendo gestionar los diferentes componentes que soportan el funcionamiento del sistema.

Figura 5.
Casos de uso del Administrador para la gestión de aplicaciones.



Nota: Este diagrama presenta las funcionalidades del actor Administrador relacionadas con la gestión de aplicaciones, configuración de notificaciones y administración de información histórica. Fuente: elaboración propia.

En conjunto, los diagramas de casos de uso describen el comportamiento funcional del módulo de administración y permiten identificar las responsabilidades de cada actor dentro del sistema. Estos diagramas constituyeron la base para la definición de los requerimientos funcionales y orientaron posteriormente las fases de diseño, implementación y validación del módulo desarrollado.

6.3 Requerimientos funcionales

La definición de los requerimientos funcionales se realizó a partir de los resultados obtenidos durante la fase de ambientación tecnológica y del análisis del estado inicial de la plataforma Smart Campus UIS. En esta etapa se identificaron las funcionalidades existentes, las limitaciones del sistema y las necesidades asociadas a la administración de dispositivos IoT, aplicaciones y gemelos digitales.

Como parte del proceso de análisis se revisó la arquitectura de la plataforma, los componentes previamente desarrollados y los flujos de interacción esperados entre los usuarios y el sistema. Asimismo, se tomaron como referencia los diagramas de casos de uso elaborados en la sección anterior, los cuales permitieron identificar las operaciones que debía soportar el módulo de administración para cumplir con los objetivos del proyecto.

A partir de este análisis se establecieron los requerimientos funcionales presentados a continuación, describiendo para cada uno su propósito, prioridad, precondiciones, postcondiciones y flujo de eventos. Estos requerimientos constituyeron la base para el diseño, implementación y validación del módulo web de administración basado en tecnología de Gemelo Digital.

Tabla 4.
Requerimiento funcional 1: Gestión de dispositivos.

ID	REQ-1	Gestión de Dispositivos	
Prioridad	Alta	Complejidad	Alta

Descripción: La aplicación debe administrar los Dispositivos asociados, permitiendo su creación, consulta, edición, eliminación y clonación a partir de configuraciones existentes dentro de la plataforma Smart Campus UIS.

Precondiciones:

- El usuario debe estar autenticado en la plataforma.
- La base de datos de dispositivos debe estar operativa.

Postcondiciones:

- El dispositivo queda registrado, actualizado o eliminado según la acción realizada.
- Se reflejan los cambios en la base de datos y en la configuración.
- Se genera confirmación de la operación al usuario

Flujo normal de eventos:

1. El usuario selecciona la opción “Gestión de Dispositivos”.
2. Visualiza la lista de dispositivos existentes y su estado.
3. Selecciona crear, editar, eliminar o copiar un dispositivo.
4. Configura las propiedades del dispositivo (nombre, tipo, parámetros de conexión, etc.).
5. Guarda los cambios.
6. El sistema confirma la operación y actualiza la base de datos

Flujo alternativo de eventos:

- Error de conexión al dispositivo:
    1. El sistema detecta que no puede establecer comunicación con el dispositivo.
    2. Se muestra un mensaje de error.
    3. La operación se cancela hasta restablecer la conexión.
- Duplicación fallida:
    1. El usuario intenta copiar un dispositivo existente.
    2. El sistema detecta que el dispositivo origen tiene configuraciones inválidas.
    3. Se muestra un mensaje de error y se solicita corrección antes de duplicar.

Tabla 5.
Requerimiento funcional 2: Gestión de usuarios.

ID	REQ-2	Gestión de usuarios.	
Prioridad	Baja	Complejidad	Media
Descripción: La aplicación debe administrar los usuarios de la plataforma Smart Campus UIS, permitiendo al administrador consultar, crear y eliminar cuentas de usuario.			
Precondiciones: • El administrador debe estar autenticado en la plataforma. • La base de datos de usuarios debe estar operativa.			
Postcondiciones: • El usuario queda registrado o eliminado según la acción realizada. • Se reflejan los cambios en la base de datos. • Se genera confirmación de la operación al administrador			
Flujo normal de eventos: 1. El administrador selecciona la opción “Gestión de Usuarios”. 2. Visualiza la lista de usuarios existentes. 3. Selecciona crear o eliminar un usuario. 4. Ingresa los datos del nuevo usuario (nombre, correo, rol, etc.) o confirma la eliminación. 5. Guarda los cambios. 6. El sistema confirma la operación y actualiza la base de datos			
Flujo alternativo de eventos: • Error de duplicación: 1. El administrador intenta crear un usuario con un correo ya existente. 2. El sistema muestra una alerta indicando duplicidad. 3. El administrador debe ingresar un correo diferente. • Error de conexión a la base de datos: 1. El sistema detecta que no puede acceder a la base de datos de usuarios. 2. Se muestra un mensaje de error. 3. La operación se cancela hasta restablecer la conexión.			

Tabla 6.
Requerimiento funcional 3: Sincronización bidireccional.

ID	REQ-3	Sincronización Bidireccional	
Prioridad	Muy Alta	Complejidad	Alta

Descripción: La aplicación debe implementar un mecanismo de sincronización bidireccional entre dispositivos IoT y sus gemelos digitales, utilizando protocolos de mensajería como MQTT para asegurar que cualquier cambio se refleje de inmediato en ambos lados.

Precondiciones:

- El broker MQTT debe estar activo.
- El dispositivo debe estar conectado y operativo.
- El gemelo digital debe estar previamente registrado en la plataforma.

Postcondiciones:

- El estado del dispositivo y su gemelo digital queda sincronizado.
- La información se almacena en la base de datos para trazabilidad.
- Se genera confirmación de la operación

Flujo normal de eventos:

1. El dispositivo publica telemetría vía MQTT.
2. El backend recibe el mensaje.
3. El sistema procesa la información recibida (parseo de JSON).
4. Se actualiza el estado del gemelo digital en la plataforma.
5. La información es almacenada en la base de datos.
6. El sistema confirma la sincronización exitosa

Flujo alternativo de eventos:

- Dispositivo inexistente:
    1. El sistema recibe telemetría de un dispositivo no registrado.
    2. Se registra un error en el log del sistema.
    3. Se notifica al administrador para revisión

Tabla 7.
Requerimiento funcional 4: Visualización de gemelo digital.

ID	REQ-4	Visualización de Gemelo Digital	
Prioridad	Alta	Complejidad	Media

Descripción: La aplicación debe permitir al administrador consultar las propiedades de los dispositivos registrados mediante su gemelo digital, mostrando datos sincronizados y ofreciendo información histórica en caso de fallos de conexión.

Precondiciones:

- El dispositivo debe estar previamente registrado en la plataforma.
- El gemelo digital debe estar configurado y asociado al dispositivo.
- La base de datos debe estar operativa.

Postcondiciones:

- El gemelo digital se actualiza sincronizadamente con la información del dispositivo.
- Los datos consultados quedan disponibles para visualización y análisis.
- Se garantiza la trazabilidad de la información en la base de datos.

Flujo normal de eventos:

1. El administrador selecciona la opción “Visualización de Gemelo Digital”.
2. El sistema recupera los datos del gemelo digital asociado al dispositivo.
3. Se muestran las propiedades del dispositivo (estado, batería, telemetría) en simultaneo.
4. El administrador puede consultar la información en la interfaz.

Flujo alternativo de eventos:

- Fallo de conexión:
    1. El sistema detecta que no puede recuperar datos.
    2. Se muestran los datos históricos almacenados en la base de datos.
    3. Se notifica al administrador sobre la pérdida de conexión

Tabla 8.
Requerimiento funcional 6: Registro histórico y trazabilidad.

ID	REQ-6	Registro Histórico y Trazabilidad	
Prioridad	Alta	Complejidad	Alta

Descripción: La aplicación debe registrar de manera automática todas las acciones y cambios de configuración realizados en los dispositivos, permitiendo al administrador consultar el historial completo para auditoría y trazabilidad.

Precondiciones:

- El sistema debe estar en ejecución.
- La base de datos debe estar operativa.
- El administrador debe estar autenticado para consultar el historial.

Postcondiciones:

- El registro histórico se actualiza con cada acción realizada.
- Los datos quedan disponibles para consulta y auditoría.
- Se garantiza la trazabilidad de las operaciones en la plataforma.

Flujo normal de eventos:

1. Cada acción realizada en la plataforma se guarda automáticamente en la base de datos.
2. El administrador accede a la opción “Historial de acciones”.
3. El sistema recupera los registros almacenados.
4. El administrador consulta el historial de estados y cambios de configuración.

Flujo alternativo de eventos:

- Fallo en la base de datos:
    1. El sistema detecta que no puede almacenar la acción en la base de datos.
    2. Se activa el mecanismo de respaldo (backup).
    3. Los datos se guardan en el sistema de respaldo hasta que la base de datos principal se restablezca.

Tabla 9.
Requerimiento funcional 7: Motor de reglas básico.

ID	REQ-7	Motor de Reglas Básico	
Prioridad	Media	Complejidad	Alta

Descripción: La aplicación debe proporcionar un motor de reglas que permita al administrador definir condiciones y acciones automáticas sobre los dispositivos, de manera que se ejecuten sin intervención manual cuando se cumplan los criterios establecidos.

Precondiciones:

- El dispositivo debe estar registrado en la plataforma.
- El administrador debe estar autenticado para definir reglas.
- La base de datos de reglas debe estar operativa.

Postcondiciones:

- La regla definida queda almacenada en la base de datos.
- El sistema aplica automáticamente la regla cuando se cumple la condición.
- Se genera confirmación de la ejecución de la regla

Flujo normal de eventos:

1. El administrador accede a la opción “Motor de Reglas”.
2. Define una nueva regla indicando condición (ej. batería < 10%) y acción (ej. cambiar estado a crítico).
3. El sistema valida la sintaxis y compatibilidad de la regla.
4. La regla se almacena en la base de datos.
5. Cuando se cumple la condición, el sistema ejecuta la acción automáticamente.
6. Se registra la ejecución en el historial para trazabilidad.

Flujo alternativo de eventos:

- Regla inválida → mostrar error.

Tabla 10.
Requerimiento funcional 8: Autenticación de usuarios.

ID	REQ-8	Autenticación de usuarios.	
Prioridad	Alta	Complejidad	Media

Descripción: La aplicación debe permitir a un usuario registrarse en la plataforma, iniciar sesión, recuperar su cuenta en caso de que olvide sus credenciales.

Precondiciones:

- El usuario debe tener acceso a la aplicación.
- El sistema debe estar conectado a la base de datos de usuarios.
- Debe existir un mecanismo de validación de credenciales.

Postcondiciones:

- El usuario queda autenticado en la plataforma.
- Se actualizan los datos del perfil en la base de datos si fueron modificados

Flujo normal de eventos:

1. El usuario accede a la pantalla de inicio de sesión.
2. Ingresa sus credenciales (usuario/contraseña).
3. El sistema valida las credenciales contra la base de datos.
4. Si son correctas, el usuario accede a la plataforma.
5. El usuario puede modificar su perfil desde la sección correspondiente

Flujo alternativo de eventos:

- Registro de nuevo usuario:
    1. El usuario selecciona “Registrarse”.
    2. Ingresa sus datos personales y credenciales.
    3. El sistema valida que no exista un usuario con el mismo correo.
    4. Se crea la cuenta y se confirma al usuario.
- Recuperación de credenciales:
    1. El usuario selecciona “Olvidé mi contraseña”.
    2. Ingresa su correo electrónico registrado.
    3. El sistema envía un enlace de recuperación.
    4. El usuario establece una nueva contraseña.

Tabla 11.
Requerimiento funcional 9: Gestión de aplicaciones.

ID	REQ-9	Gestión de aplicaciones IoT	
Prioridad	Media	Complejidad	Alta

Descripción: La aplicación debe administrar las aplicaciones desplegadas en la plataforma IoT, permitiendo su creación, consulta, edición, eliminación.
Precondiciones: • El administrador debe estar autenticado en la plataforma. • Debe existir al menos un dispositivo disponible para asignación. • La base de datos de aplicaciones debe estar operativa.
Postcondiciones: • La aplicación queda registrada, actualizada o eliminada según la acción realizada. • Se reflejan los cambios en la base de datos. • Se genera confirmación de la operación al administrador.
Flujo normal de eventos: 1. El administrador selecciona la opción “Gestión de aplicaciones”. 2. Visualiza la lista de aplicaciones existentes. 3. Selecciona crear, editar o eliminar una aplicación. 4. Configura los parámetros de la aplicación (nombre, descripción, etc.). 5. Guarda los cambios. 6. El sistema confirma la operación y actualiza la base de datos
Flujo alternativo de eventos: • Error de compatibilidad → mostrar alerta.

6.4 Requerimientos no funcionales

Además de las funcionalidades definidas para el módulo de administración, fue necesario establecer un conjunto de requerimientos no funcionales que describen las características de calidad que debe cumplir el sistema durante su operación. Estos requerimientos fueron identificados a partir del análisis del entorno tecnológico, las necesidades de los usuarios y las condiciones de funcionamiento de la plataforma Smart Campus UIS.

Los requerimientos no funcionales establecen criterios relacionados con aspectos como escalabilidad, usabilidad, desempeño y adaptabilidad de la interfaz, los cuales orientaron las

decisiones de diseño e implementación del sistema. A continuación, se presentan los requerimientos no funcionales definidos para el desarrollo del proyecto.

Tabla 12.
Requerimiento no funcional 1: Escalabilidad.

Identificación del Requerimiento	RNF-01
Nombre	Escalabilidad
Descripción	La aplicación debe permitir la integración de nuevos dispositivos IoT y aplicaciones sin afectar el rendimiento general de la plataforma.
Prioridad	Alta

Tabla 13.
Requerimiento no funcional 2: Diseño responsivo.

Identificación del Requerimiento	RNF-02
Nombre	Diseño Responsivo
Descripción	La aplicación debe tener un diseño responsivo de forma que la información se vea de manera agradable y legible en diversos tamaños de pantalla.
Prioridad	Alta

Tabla 14.
Requerimiento no funcional 3: Facilidad de uso.

Identificación del Requerimiento	RNF-03
Nombre	Facilidad de uso

Descripción	La aplicación debe disponer de una interfaz de usuario con menús organizados, botones claramente identificados y mensajes informativos que orienten al usuario durante la ejecución de las diferentes funcionalidades, favoreciendo una experiencia de uso clara y consistente.
Prioridad	Alta

Tabla 15.
Requerimiento no funcional 4: Alto rendimiento.

Identificación del Requerimiento	RNF-04
Nombre	Alto Rendimiento
Descripción	La aplicación debe ser rápida y no bloquearse durante el flujo normal de navegación del usuario, garantizando tiempos de respuesta óptimos.
Prioridad	Alta

7. Diseño de la arquitectura basado en Gemelo Digital

Definimos la arquitectura del módulo**,** estableciendo la estructura del sistema, los componentes de software, los modelos de datos y los mecanismos de comunicación necesarios para garantizar la integración entre los dispositivos IoT y sus gemelos digitales.

Se orientó a una arquitectura cliente-servidor desacoplada, permitiendo separar la lógica de presentación, la lógica de negocio y la gestión de datos. Este enfoque facilita la escalabilidad, el mantenimiento del sistema y la integración del módulo.

7.1 Arquitectura del sistema

La arquitectura propuesta está compuesta por cuatro componentes principales: el frontend, el backend, la base de datos y la capa de integración con dispositivos IoT. Estos elementos interactúan entre sí para permitir la gestión centralizada de dispositivos, la representación mediante gemelos digitales y la sincronización bidireccional de la información.

El frontend corresponde a la interfaz web desarrollada para la administración del sistema, desde donde los usuarios pueden visualizar dispositivos, consultar el estado de los gemelos digitales y gestionar configuraciones. Este componente se comunica con el backend mediante servicios REST.

El backend se encarga de la lógica, procesamiento de datos y gestión de la comunicación entre los dispositivos IoT y la plataforma. A través de este componente se implementan los servicios de administración, sincronización y trazabilidad.

La base de datos almacena la información relacionada con dispositivos, aplicaciones, gemelos digitales, estados y registros históricos, garantizando la persistencia de los datos y la coherencia del sistema.

Finalmente, la capa de integración IoT permite la comunicación con los dispositivos mediante protocolos de mensajería, facilitando el envío y recepción de datos en tiempo real.

7.2 Frontend

El frontend se desarrolló como una aplicación web que permite la interacción entre el usuario y el sistema. Este componente ofrece funcionalidades para la visualización de dispositivos, consulta del gemelo digital, gestión de configuraciones y monitoreo en tiempo real.

La interfaz se diseñó bajo un enfoque responsivo, permitiendo su uso en diferentes dispositivos y garantizando una experiencia de usuario intuitiva.

7.3 Backend

El backend implementa la lógica del sistema mediante servicios REST que permiten gestionar dispositivos, usuarios y gemelos digitales. Este componente actúa como intermediario entre el frontend y la base de datos, además de encargarse de procesar la información proveniente de los dispositivos IoT.

Dentro del backend se implementaron mecanismos para el registro de eventos, control de estados y validación de datos, garantizando la coherencia de la información en la plataforma.

7.4 Base de datos

La base de datos fue diseñada para almacenar la información del sistema de forma estructurada. En ella se registran los dispositivos, sus configuraciones, los gemelos digitales asociados, los estados de operación y el historial de eventos.

Este diseño permite mantener la trazabilidad de las acciones realizadas en la plataforma, facilitando la auditoría y el análisis del comportamiento de los dispositivos.

7.5 Módulo de Gemelo Digital

El módulo de gemelo digital constituye el núcleo del sistema, ya que permite representar digitalmente los dispositivos físicos mediante gemelos digitales. Cada gemelo digital almacena el estado, las propiedades y la configuración del dispositivo asociado.

Este componente permite reflejar la información del entorno físico y, a su vez, enviar configuraciones desde la plataforma hacia los dispositivos, garantizando una sincronización bidireccional.

7.6 Flujos de datos

El sistema implementa dos flujos principales de información:

Flujo de monitoreo: En este flujo, los dispositivos IoT envían datos de telemetría hacia la plataforma mediante protocolos de comunicación. El backend recibe esta información, la procesa y actualiza el estado del gemelo digital, permitiendo su visualización en la interfaz web.

Flujo de control: En este caso, el usuario realiza modificaciones desde la interfaz de administración, las cuales son enviadas al backend. El sistema actualiza el gemelo digital y transmite los cambios al dispositivo físico, garantizando la coherencia entre ambos entornos.

7.7 Mecanismo de sincronización bidireccional

Para garantizar la coherencia entre los dispositivos físicos y sus gemelos digitales, se implementó un mecanismo de sincronización bidireccional basado en el intercambio de mensajes en tiempo real.

Este mecanismo permite que cualquier cambio realizado en el dispositivo físico se refleje automáticamente en el gemelo digital, y que las configuraciones definidas en la plataforma se propaguen hacia el dispositivo.

La sincronización se apoya en el uso de protocolos de mensajería y servicios de backend que gestionan el flujo de información, asegurando consistencia, disponibilidad y actualización continua de los datos.

7.8 Modelo de datos del sistema

El modelo de datos del sistema fue diseñado para representar la relación entre los dispositivos físicos y sus gemelos digitales, permitiendo la gestión de estados, configuraciones y persistencia de la información.

A nivel de implementación, el backend define entidades que corresponden directamente a tablas en la base de datos, gestionadas mediante el uso de JPA/Hibernate. Las principales entidades identificadas son:

- Dispositivo: representa el dispositivo dentro del sistema. Contiene atributos como identificador, nombre, tipo, estado y referencia a la aplicación a la que pertenece.
- DigitalTwin (Gemelo Digital): entidad que modela el gemelo digital asociado a cada dispositivo. Almacena el estado actual del dispositivo, así como propiedades configurables que permiten su control desde la plataforma.
- Registro de eventos: entidad encargada de almacenar los cambios de estado y eventos generados por los dispositivos y sus gemelos digitales, permitiendo mantener trazabilidad del sistema.

  La relación entre estas entidades se establece principalmente de la siguiente forma:

  - Un dispositivo tiene asociado un único gemelo digital.

- Los registros de eventos se asocian tanto a dispositivos como a gemelos digitales.

Este modelo permite mantener una representación consistente del sistema, facilitando la sincronización entre el estado físico y el estado virtual gestionado en la plataforma.

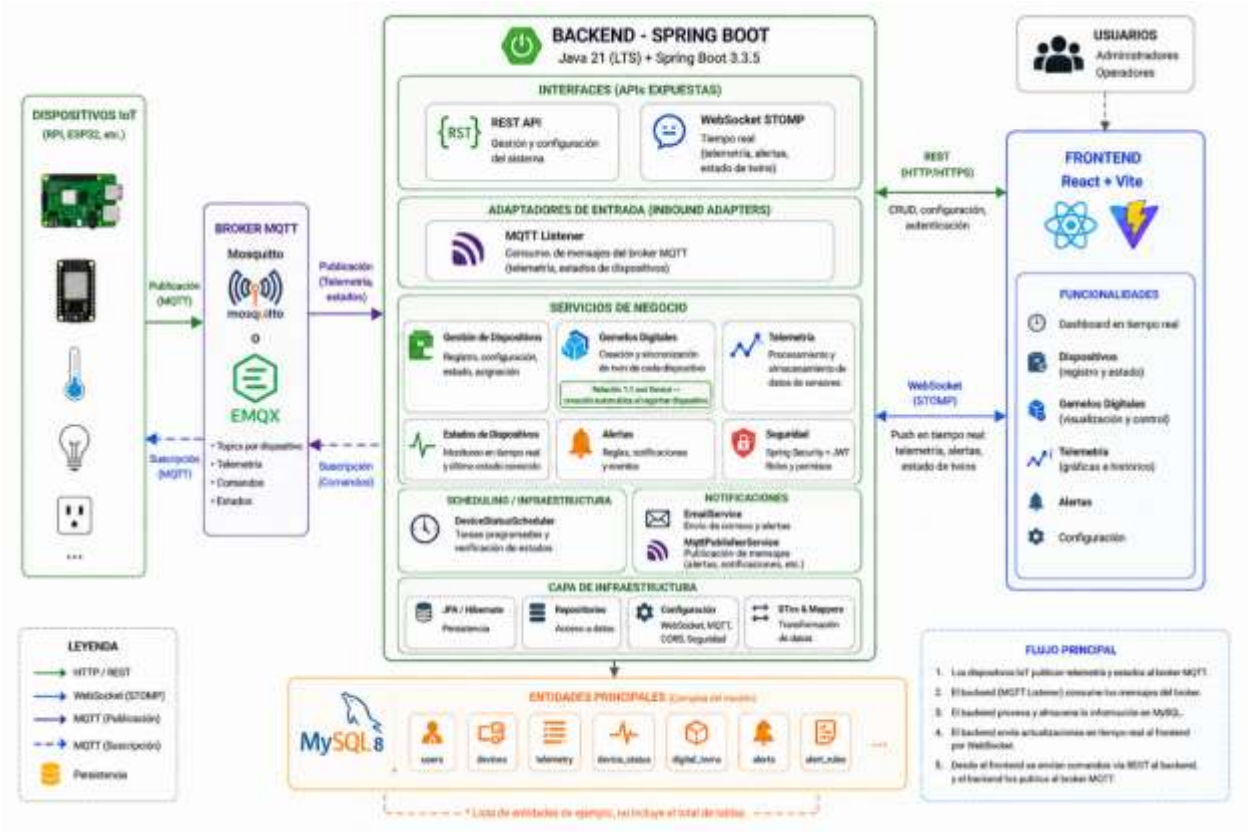
8. Implementación del módulo web de administración

Este módulo constituye el núcleo operativo del sistema, dado que integra la gestión de dispositivos y aplicaciones IoT con sus correspondientes gemelos digitales (digital twins), garantizando coherencia entre el estado físico y su representación virtual, así como trazabilidad sincronizada de todos los eventos del ecosistema.

El desarrollo del prototipo responde directamente al objetivo específico propuesto: implementar un módulo web funcional que articule la administración de dispositivos IoT con el paradigma del gemelo digital, asegurando sincronización bidireccional y registro histórico de eventos. Para ello, se adoptó una arquitectura por capas sobre un stack tecnológico moderno — Spring Boot 3.3.5 en el backend el cual se desplegó en Railway y React en el frontend desplegado en la plataforma Vercel.

El módulo web de administración fue concebido bajo una arquitectura cliente-servidor desacoplada, en la que el frontend y el backend se comunican exclusivamente a través de una API REST, con soporte adicional de WebSocket para la transmisión de eventos en tiempo real. Este diseño favorece la escalabilidad horizontal del sistema, facilita el mantenimiento independiente de cada capa y permite la integración futura con otros módulos de la plataforma Smart Campus UIS.

Figura 6.
Arquitectura para la gestión de dispositivos y gemelos digitales.



La figura presenta la arquitectura general del sistema desarrollado, donde los dispositivos IoT (simulados mediante Node-RED) se comunican a través del protocolo MQTT con el backend implementado en Spring Boot y desplegado en Railway. Este backend gestiona la lógica del sistema mediante una API REST, un listener MQTT y comunicación con WebSocket STOMP, además de integrar la gestión de gemelos digitales. Finalmente, el frontend desarrollado con React y Vite, desplegado en Vercel, permite la visualización y administración sincronizada de los dispositivos y sus estados.

8.1 Arquitectura por Capas del Backend

El backend sigue el patrón de arquitectura en capas ampliamente adoptado en aplicaciones basadas en Spring Boot. La estructura se organiza de la siguiente manera:

- Capa de Controladores (Controller): Expone los endpoints REST y gestiona las peticiones HTTP entrantes, delegando la lógica al nivel de servicio.
- Capa de Servicios (Service): Contiene la lógica de negocio del sistema, incluyendo la creación automática del gemelo digital al registrar un dispositivo y la gestión del motor de sincronización.
- Capa de Repositorios (Repository): Abstrae el acceso a la base de datos mediante Spring Data JPA, permitiendo operaciones CRUD sin necesidad de escribir consultas SQL explícitas.
- Capa de Modelos (Model): Define las entidades del dominio —Device, DigitalTwin, Application— y sus relaciones, mapeadas directamente a tablas en MySQL 8.
- Capa de Configuración (Config): Gestiona la configuración de seguridad (Spring Security), WebSocket y otros aspectos transversales del sistema.

El flujo general de una petición sigue la dirección: Controller → Service → Repository → Base de Datos, garantizando separación de responsabilidades y alta cohesión en cada componente.

Tabla 16.
Tecnología del Backend.

Tecnología	Versión	Uso

Java	21	Lenguaje principal
Spring Boot	3.3.5	Framework principal
Spring Security	3.3.5	Autenticación y autorización
Spring Integration MQTT	3.3.5	Comunicación con broker MQTT
Spring WebSocket (STOMP)	3.3.5	Sincronizado frontend-backend
Spring Data JPA	3.3.5	Persistencia de datos
MySQL	8.x	Base de datos relacional
JJWT	0.11.5	Generación y validación de tokens JWT
Eclipse Paho MQTT	1.2.5	Cliente MQTT
Lombok	latest	Reducción de código boilerplate
SpringDoc OpenAPI	2.6.0	Documentación Swagger

Tabla 17.
Tecnología del Frontend.

Tecnología	Versión	Uso
React	18.x	Framework UI
Vite	5.x	Bundler y servidor de desarrollo
Axios	latest	Peticiones HTTP a la API
@stomp/stompjs	latest	Cliente WebSocket STOMP
sockjs-client	latest	Fallback WebSocket

jsPDF	latest	Generación de reportes PDF
jspdf-autotable	latest	Tablas en PDF
Chart.js	latest	Gráficas de telemetría
js-yaml	latest	Importación de dispositivos desde YAML

Tabla 18.
Infraestructura.

Servicio	**Uso**
Railway	Despliegue del backend Spring Boot + base de datos MySQL
Vercel	Despliegue del frontend React
HiveMQ Cloud	Broker MQTT (SSL, puerto 8883)
Node-RED	Relay MQTT: recibe en smartcampus/ack y publica en smartcampus/confirm

La Figura 7 presenta la estructura de carpetas del proyecto backend, organizada según el patrón de arquitectura en capas descrito anteriormente.

Figura 7.
Estructura del proyecto.

Nota: La figura muestra la organización de directorios del backend, incluyendo los paquetes controller, service, repository, model y config, correspondientes a las capas descritas en esta sección.

8.2 Relación entre Dispositivo y Gemelo Digital

Una de las decisiones de diseño más relevantes del sistema es la relación uno a uno (1:1) entre la entidad Device y la entidad DigitalTwin. Esta relación asegura que todo dispositivo registrado en el sistema posea, desde el momento de su creación, una representación virtual íntegramente inicializada. El gemelo digital se instancia automáticamente al persistir el dispositivo, con los siguientes valores predeterminados:

- status: OFFLINE (estado inicial hasta recibir telemetría activa).
- telemetryJson: {} (objeto JSON vacío, a ser poblado por el motor de sincronización).
- lastUpdate: timestamp del momento de creación (marca temporal de la última sincronización).

Esta estrategia garantiza integridad referencial desde el primer momento y elimina la posibilidad de que un dispositivo exista sin su contraparte virtual, lo que es esencial para el paradigma del gemelo digital.

Figura 8.
Arquitectura para la gestión de dispositivos y gemelos digitales.

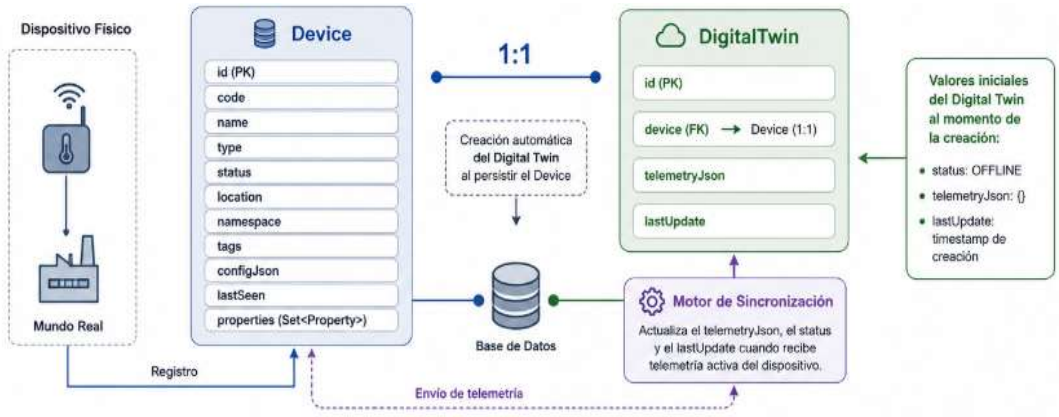


Tabla 19.
Propiedades del Digital Twin.

Campo	Tipo	Descripción
id	Long	Identificador único
device	Device	Dispositivo físico asociado (relación 1-1)
telemetryJson	String	JSON con los últimos valores de cada propiedad
lastUpdate	LocalDateTime	Última actualización del twin

Tabla 20.
Propiedades del Dispositivo.

Campo	Tipo	Descripción
id	Long	Identificador único
code	String	Código único del dispositivo (ej: RPI-001)
name	String	Nombre descriptivo
type	String	Tipo: SENSOR, ACTUATOR, CAMERA, CONTROLLER
status	String	Estado: ONLINE, OFFLINE, ERROR, WARNING, LOW_BATTERY, MAINTENANCE
location	String	Ubicación física
namespace	String	Agrupación lógica (ej: edificio-a/piso-2)
tags	String	Etiquetas separadas por coma
configJson	String	JSON con configuración flexible del dispositivo
lastSeen	LocalDateTime	Última vez que envió telemetría
properties	Set<Property>	Propiedades asociadas

8.3 Stack Tecnológico

La selección del stack tecnológico fue determinada por criterios de madurez, soporte comunitario, compatibilidad con estándares REST y capacidad para gestionar comunicaciones en tiempo real. En la Tabla 1 se resume el conjunto de tecnologías empleadas.

Tabla 21.
Stack tecnológico del módulo web de administración Smart Campus UIS.

Categoría	Tecnología	Justificación
Backend	Java 21 (LTS)	Versión de soporte extendido con mejoras de rendimiento y sintaxis moderna.
Backend	Spring Boot 3.3.5	Framework maduro para microservicios REST; configuración automática y ecosistema amplio.
Backend	Spring Data JPA	ORM que simplifica la persistencia y reduce código boilerplate.
Backend	Spring Security	Gestión de autenticación y autorización de la API REST.
Backend	WebSocket	Protocolo para comunicación bidireccional con el frontend.
Frontend	React + JavaScript	Biblioteca declarativa para interfaces reactivas; desplegada en Vercel.
Base de datos	MySQL 8.x	Motor relacional robusto y compatible con Spring Data JPA.
Construcción	Maven 3.9+	Gestión de dependencias y ciclo de vida del proyecto backend.

Despliegue	Vercel	Plataforma de despliegue continuo para el frontend con integración GitHub.
Utilitarios	Lombok / Postman	Reducción de código repetitivo y pruebas de endpoints REST, respectivamente.

8.4 Módulos Implementados

El prototipo funcional está compuesto por cuatro módulos principales que trabajan de forma coordinada para cumplir el objetivo de gestión integrada de dispositivos IoT y sus gemelos digitales:

8.4.1 Módulo de Gestión de Dispositivos IoT

Este módulo constituye el núcleo de la plataforma y expone un CRUD completo para el registro, consulta, actualización y eliminación de dispositivos físicos. Cada dispositivo se representa mediante atributos que describen su identidad, tipo, ubicación y estado operacional dentro del campus.

Los endpoints REST implementados son:

- GET /devices — Retorna el listado completo de dispositivos registrados.
- GET /devices/{id} — Retorna el detalle de un dispositivo específico.
- POST /devices — Registra un nuevo dispositivo y crea su gemelo digital asociado de forma automática.
- PUT /devices/{id} — Actualiza los atributos de un dispositivo existente.
- DELETE /devices/{id} — Elimina el dispositivo y su gemelo digital vinculado.

La persistencia se gestiona mediante Spring Data JPA sobre MySQL 8, garantizando consistencia transaccional en todas las operaciones. La serialización JSON fue configurada para evitar recursividad infinita derivada de la relación bidireccional Device-DigitalTwin.

8.4.2 Módulo de Gemelos Digitales (Digital Twins)

El módulo de gemelos digitales implementa el paradigma de representación virtual de activos físicos (Grieves y Vickers, 2017). Cada instancia de DigitalTwin mantiene un estado sincronizado con su dispositivo físico correspondiente, almacenando:

- Estado operacional (status): ONLINE, OFFLINE o ERROR.
- Telemetría en formato JSON (telemetryJson): estructura flexible que permite almacenar cualquier conjunto de métricas producidas por el dispositivo.
- Marca temporal de última sincronización (lastUpdate): permite calcular latencia y detectar dispositivos inactivos.

La creación automática del gemelo digital al registrar un dispositivo representa una garantía arquitectónica de que el sistema nunca presentará inconsistencias entre el inventario físico y su representación virtual. Este mecanismo fue implementado en la capa de servicios, asegurando que ambas entidades se persistan dentro de la misma transacción.

8.4.3 Motor de Sincronización y Trazabilidad

El motor de sincronización es el componente responsable de mantener coherencia entre el estado del dispositivo físico y su gemelo digital. Cuando se recibe telemetría —ya sea mediante peticiones REST o eventos WebSocket— el motor actualiza el campo telemetryJson del gemelo correspondiente y registra el evento en el historial del sistema.

El registro histórico de eventos provee trazabilidad completa de las operaciones realizadas sobre cada dispositivo: creaciones, actualizaciones, cambios de estado y telemetría recibida. Esta bitácora es accesible desde la interfaz web de administración, permitiendo auditar el comportamiento del ecosistema IoT en cualquier ventana temporal.

8.5 Interfaz Web de Administración

La interfaz web fue desarrollada con React y desplegada en la plataforma Vercel, accesible en la URL: https://smartcampus-admin-module.vercel.app/. El diseño de la interfaz prioriza la usabilidad del administrador del campus, organizando las funcionalidades en vistas diferenciadas para cada dominio del sistema.

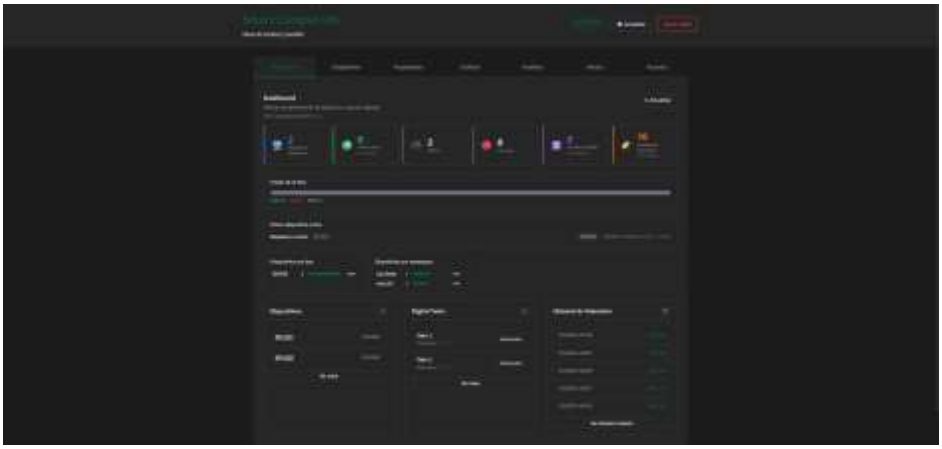
Las vistas principales del módulo son:

- Dashboard: Panel central que presenta indicadores de estado del ecosistema IoT —número de dispositivos activos, offline y en error—, así como actividad reciente de telemetría.
- Gestión de Dispositivos: Listado tabular con capacidades de registro, edición y eliminación de dispositivos, acompañado del estado de su gemelo digital.
- Detalle de Gemelo Digital: Vista dedicada que muestra el estado actual, la telemetría almacenada en JSON y el historial de sincronizaciones de cada dispositivo.
- Gestión de Aplicaciones IoT: Administración de las aplicaciones registradas en el sistema y sus asociaciones con dispositivos físicos.
- Registro Histórico: Bitácora de eventos del sistema con filtros por dispositivo, tipo de evento y rango temporal.

La comunicación entre el frontend y el backend se realiza exclusivamente a través de la API REST, con solicitudes HTTP asíncronas. Para la actualización sincronizada del estado de los

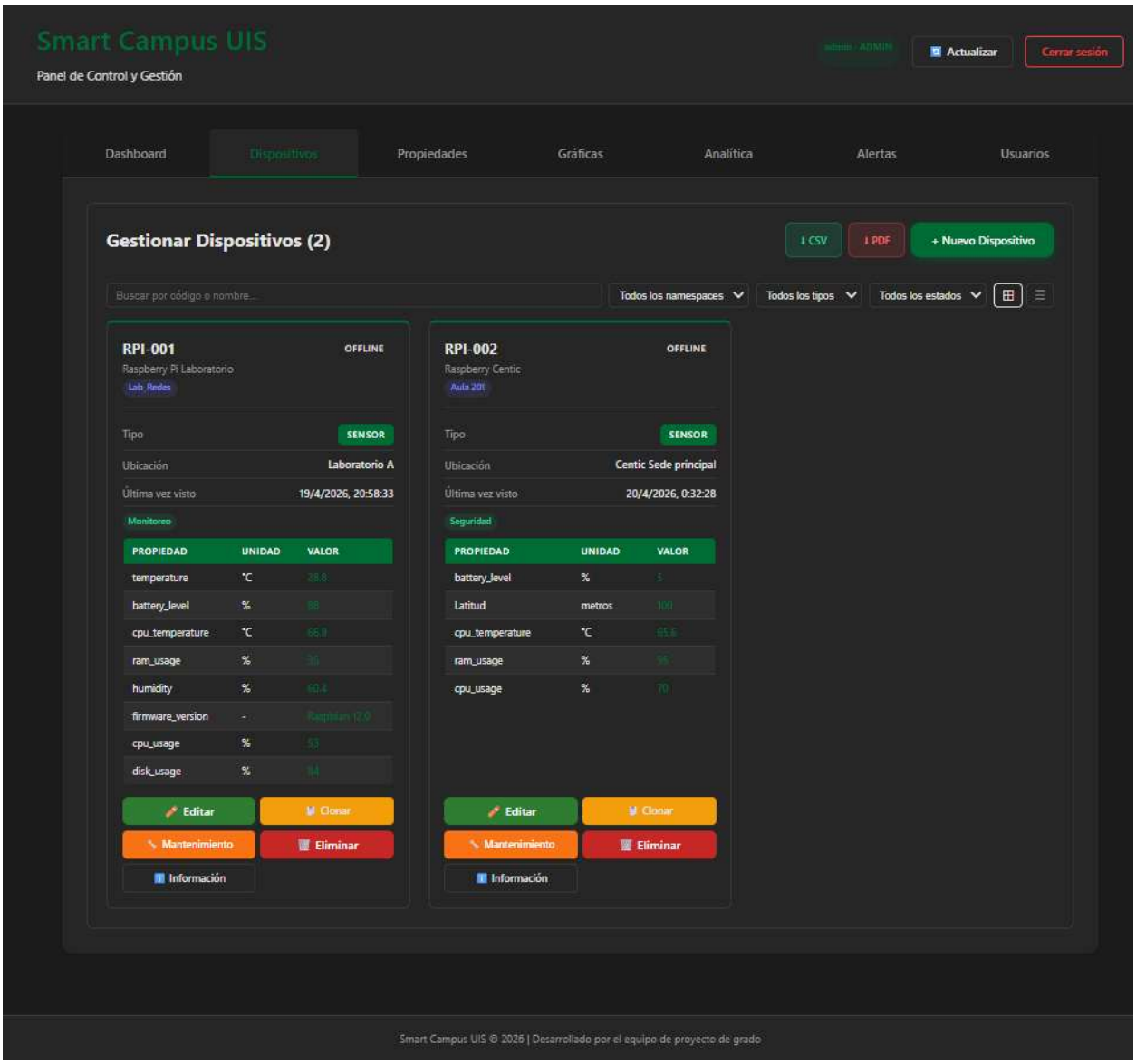
dispositivos, se implementó una conexión WebSocket que permite reflejar cambios en la interfaz sin necesidad de recargar la página.

Figura 9.
Dashboard principal de la plataforma Smart Campus UIS.



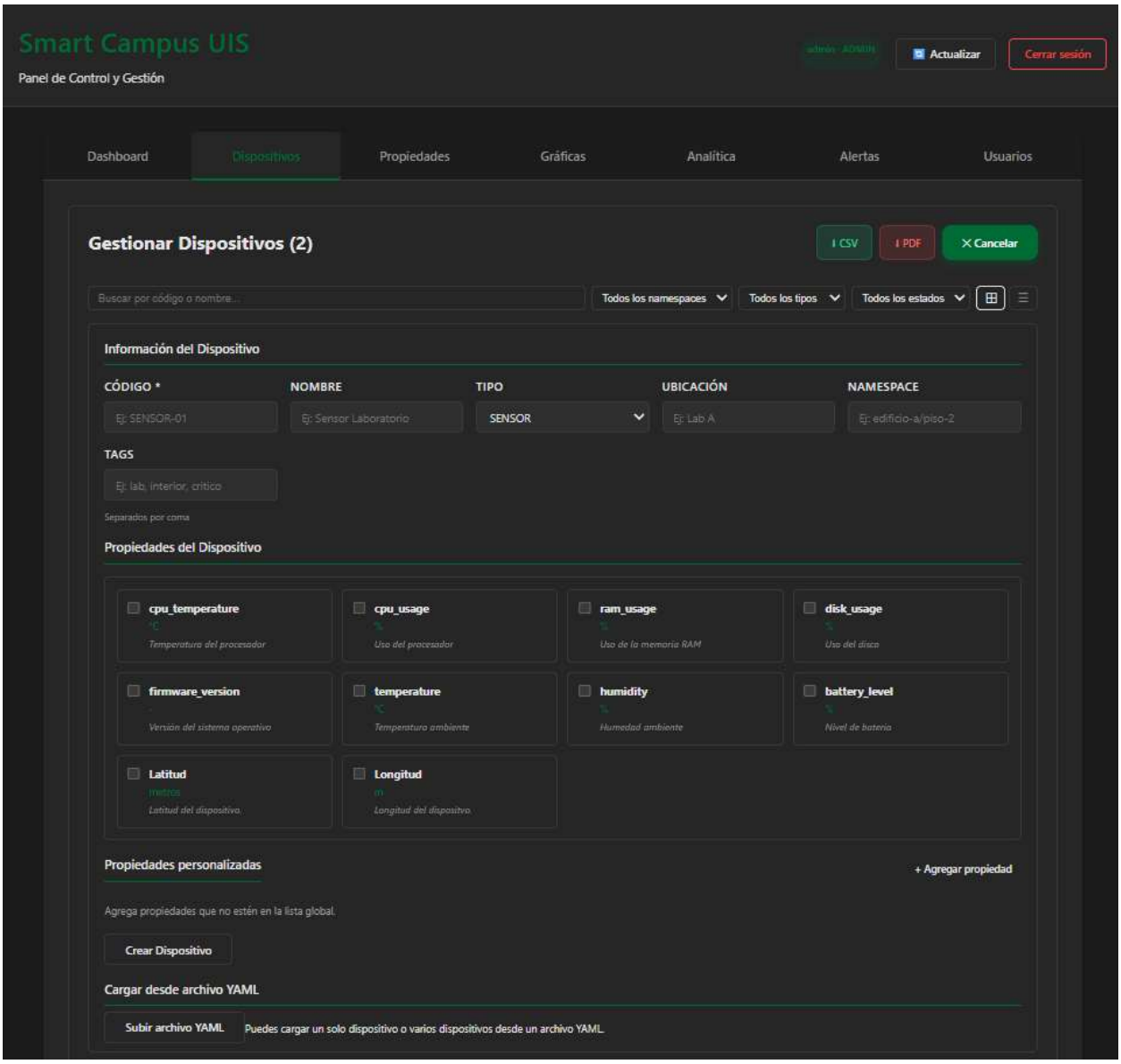
Nota: *Dashboard principal del módulo web de administración Smart Campus UIS. La vista central presenta indicadores de estado del ecosistema IoT y actividad reciente de telemetría.*

Figura 10.
Listado de dispositivos registrados.



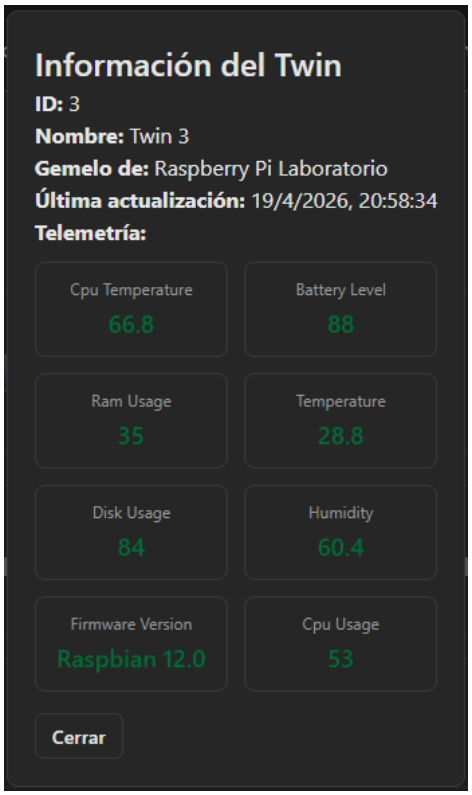
Nota: *Vista de gestión de dispositivos IoT. Cada fila corresponde a un dispositivo registrado con su identificador, tipo, ubicación y estado actual del gemelo digital asociado.*

Figura 11.
Formulario de registro de nuevo dispositivo.



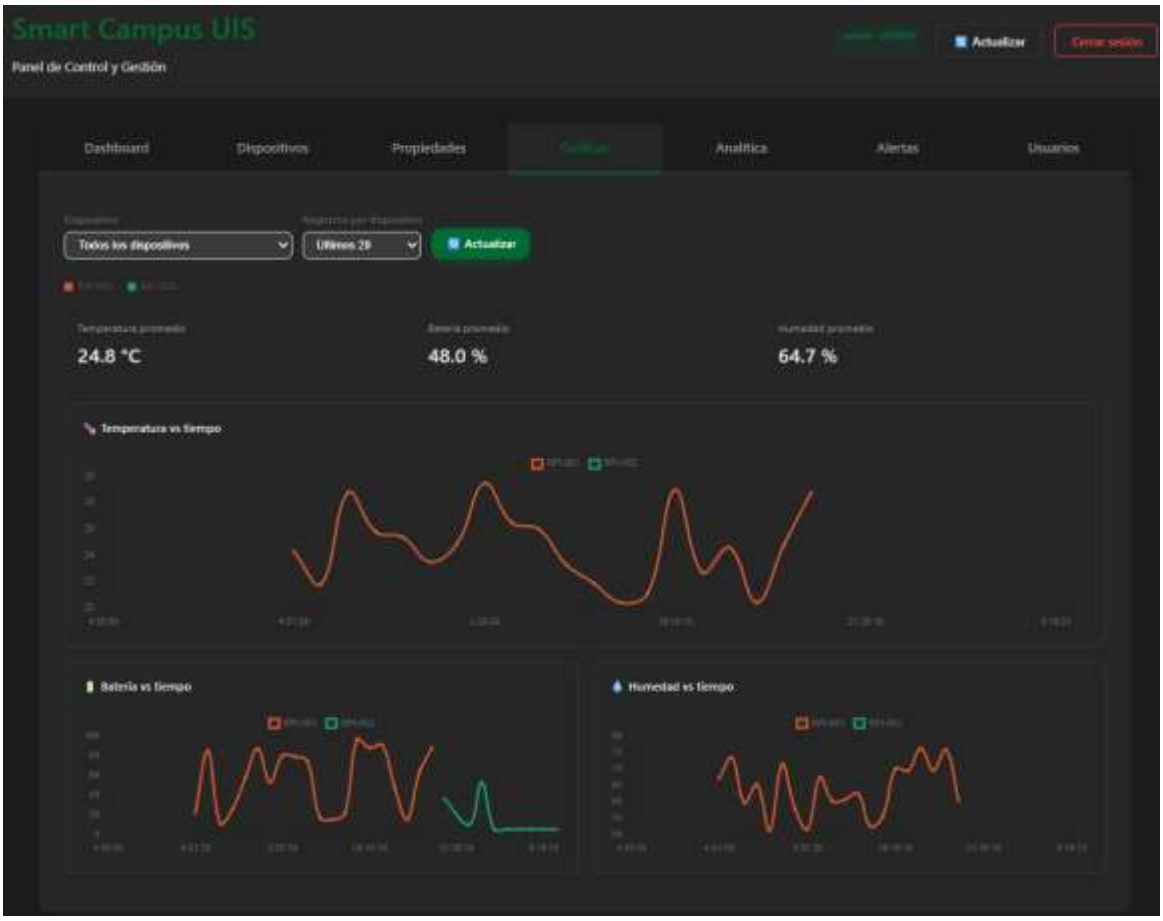
Nota: *Formulario de registro de dispositivo IoT. Al guardar, el sistema crea automáticamente el gemelo digital asociado con estado inicial OFFLINE.*

Figura 12.
Vista del gemelo digital de un dispositivo.



Nota: *Panel de detalle del gemelo digital. Se visualizan el estado operacional, la telemetría almacenada en formato JSON y la marca temporal de la última sincronización.*

Figura 13.
Panel de visualización de telemetría y analítica del sistema.



Nota: *En esta vista se presentan métricas de los dispositivos IoT, incluyendo temperatura, nivel de batería y humedad, así como su comportamiento histórico mediante gráficos temporales. Fuente: elaboración propia.*

9. Validación del módulo de administración

La validación del sistema se llevó a cabo sobre el entorno de producción mediante un conjunto estructurado de pruebas funcionales y unitarias orientadas a verificar el correcto comportamiento de cada componente del módulo, la coherencia de su interacción y la capacidad del sistema bajo las condiciones reales del entorno donde se encuentra desplegado. Adicionalmente, se diseñaron flujos instrumentados en Node-RED que simularon el

comportamiento de un dispositivo IoT real (Raspberry Pi), permitiendo evaluar el ciclo completo de comunicación bidireccional entre el dispositivo físico y su gemelo digital, desde la publicación de telemetría hasta la confirmación final de la sincronización en la base de datos.

El proceso de validación se estructuró en tres etapas: una primera etapa de pruebas funcionales orientadas a verificar el comportamiento correcto de cada funcionalidad expuesta por el módulo; una segunda etapa de pruebas unitarias y mediciones de tiempos de respuesta, dirigida a evaluar la capacidad del sistema considerando las características del entorno de despliegue tanto del backend como del frontend; y una tercera etapa de validación del ciclo completo de comunicación bidireccional mediante un flujo orquestado en Node-RED.

9.1 Pruebas Funcionales del Sistema

Para validar el funcionamiento del módulo se diseñó y ejecutó un conjunto de pruebas funcionales organizadas en siete categorías, abarcando desde el registro de dispositivos hasta la verificación de los servicios desplegados. El propósito de estas pruebas fue confirmar que cada componente del sistema cumple con su responsabilidad definida y que la interacción entre los diferentes módulos se ejecuta de forma coherente.

las pruebas se ejecutaron directamente sobre el entorno de producción, con el backend operando en su plataforma de despliegue y el frontend accesible desde su URL pública. Cada caso de prueba se definió previamente con una entrada conocida, un resultado esperado y un criterio de éxito. Las pruebas se ejecutaron de forma manual utilizando Postman para los endpoints REST, el cliente MQTT integrado en Node-RED para validar el flujo de telemetría, y el navegador web para verificar la respuesta del frontend ante las actualizaciones.

Las categorías de pruebas definidas fueron las siguientes:

- Registro de dispositivos y creación automática del gemelo digital**:** se validó que al registrar un nuevo dispositivo mediante el endpoint POST /devices, el sistema genere automáticamente su correspondiente gemelo digital con los valores iniciales definidos (status: OFFLINE, telemetryJson: {}, lastUpdate con timestamp de creación). Se probaron los cuatro tipos de dispositivos soportados (SENSOR, ACTUATOR, CAMERA, CONTROLLER) así como los casos de error por duplicación de código y campos obligatorios faltantes.
- Actualización de telemetría en el Digital Twin: se validó la correcta recepción y procesamiento de mensajes MQTT publicados en el tópico smartcampus/telemetry, verificando que el backend actualiza el campo telemetryJson del gemelo digital y refresca su marca temporal de última sincronización. Se probaron mensajes con diferentes tipos de telemetría (temperatura, humedad, uso de CPU) y mensajes con múltiples campos simultáneos.
- Endpoints REST: se ejecutaron pruebas sobre todos los endpoints expuestos por la API del backend, verificando las operaciones CRUD completas para dispositivos y la consulta de gemelos digitales. Se incluyeron pruebas de casos negativos como consultas con identificadores inexistentes para validar el manejo correcto de errores HTTP.
- Persistencia e integridad en MySQL: se verificó que las operaciones realizadas desde la API se reflejen correctamente en la base de datos, manteniendo la integridad referencial 1:1 entre las entidades Device y DigitalTwin, así como la eliminación en cascada cuando se elimina un dispositivo.

- Comunicación WebSocket frontend-backend: se validó el establecimiento de la conexión STOMP entre el frontend y el backend, la recepción de notificaciones en tiempo real ante cambios en los gemelos digitales y la reconexión automática del cliente ante pérdidas temporales de conexión.
- Disponibilidad de los servicios desplegados: se realizaron verificaciones periódicas de disponibilidad mediante peticiones HTTP al endpoint de health check del backend y carga directa del frontend desde el navegador, registrando los resultados durante el periodo de pruebas.

En total se ejecutaron 30 pruebas funcionales, todas con resultado exitoso, cubriendo los flujos críticos del sistema. Los resultados consolidados se presentan en la siguiente tabla:

Tabla 22.
Resumen de pruebas funcionales ejecutadas.

Categoría	Pruebas ejecutadas	Pruebas exitosas	Tasa de éxito
Registro de dispositivo y creación automática del gemelo digital	6	6	100%

	Actualización de telemetría en el Digital Twin vía MQTT	5	5	100%
	Endpoints REST (GET, POST, PUT, DELETE)	8	8	100%
	Persistencia e integridad en MySQL	4	4	100%
	Comunicación WebSocket frontend-backend	3	3	100%
	Disponibilidad del backend	2	2	100%
	Disponibilidad del frontend	2	2	100%
	Total	30	30	100%

Los resultados obtenidos confirman que todos los componentes del módulo operan correctamente de forma individual y en conjunto, validando la coherencia entre el estado físico de los dispositivos y su representación virtual en el sistema.

9.2 Pruebas Unitarias y Capacidad del Sistema

Una vez verificada la correcta funcionalidad del módulo, se realizó una segunda etapa de validación orientada a evaluar el desempeño y la capacidad real del sistema bajo las restricciones del entorno donde se encuentra desplegado. Esta evaluación combinó pruebas unitarias sobre los componentes principales del backend con mediciones reales de los tiempos de respuesta de la API, del ciclo de sincronización física-virtual y del comportamiento del sistema ante cargas concurrentes progresivas.

Metodología de las pruebas unitarias: se diseñaron pruebas unitarias sobre los servicios principales del backend con el objetivo de verificar de forma aislada el comportamiento de la lógica de negocio. Las pruebas se enfocaron en tres áreas críticas: la creación automática del gemelo digital al registrar un dispositivo, la actualización del estado del twin al procesar telemetría entrante, y la validación de los datos antes de su persistencia en la base de datos. Cada prueba se diseñó con datos de entrada controlados y resultados esperados predefinidos, permitiendo identificar de forma precisa cualquier desviación en el comportamiento del componente evaluado.

Metodología de las mediciones de tiempo: los tiempos de respuesta de la API REST se midieron utilizando Postman, ejecutando diez iteraciones por cada endpoint contra el backend desplegado en producción. Para cada iteración se registró el tiempo total desde la emisión de la petición hasta la recepción completa de la respuesta, y posteriormente se calculó el promedio aritmético. Las mediciones se realizaron en estado caliente del servicio, es decir, tras haber emitido

peticiones previas para evitar el efecto del arranque en frío que se produce cuando el servicio permanece inactivo durante varios minutos.

Para evaluar el ciclo de sincronización física-virtual se utilizó Node-RED como herramienta de orquestación, midiendo el tiempo transcurrido entre la publicación de un mensaje de telemetría en smartcampus/telemetry y la confirmación final del ciclo en smartcampus/confirm. Adicionalmente, se midió el tiempo de propagación de eventos al frontend a través de WebSocket.

Tabla 23.
Resultados de las pruebas unitarias.

Componente evaluado	Pruebas ejecutadas	Pruebas exitosas	Tasa de éxito
Creación automática del gemelo digital	5	5	100%
Actualización del estado del Digital Twin	4	4	100%
Validación de datos antes de persistencia	4	4	100%
Procesamiento de mensajes MQTT entrantes	3	3	100%
Manejo de excepciones y casos de error	4	4	100%
Total	20	20	100%

Tabla 24.
Tiempos promedio de respuesta de la API REST.

Operación	Endpoint	Iteraciones	Tiempo promedio (ms)
Listado de dispositivos	GET /devices	10	312
Consulta de dispositivo por ID	GET /devices/{id}	10	224
Registro de dispositivo	POST /devices	10	548
Actualización de dispositivo	PUT /devices/{id}	10	387
Eliminación de dispositivo	DELETE /devices/{id}	10	341
Consulta del gemelo digital	GET /twins/{id}	10	246
Listado de gemelos digitales	GET /twins	10	329
Promedio general	—	**70**	**341**

Los tiempos obtenidos muestran que el sistema responde con un promedio general de 341 milisegundos para operaciones CRUD en estado caliente. Las operaciones de consulta (GET) presentan los tiempos más bajos al no requerir modificaciones en la base de datos, mientras que la operación de registro (POST) presenta el tiempo más elevado debido a la doble operación de

persistencia que implica: la creación del dispositivo y la de su gemelo digital asociado dentro de la misma transacción. Es importante señalar que en condiciones de arranque en frío (tras periodos de inactividad superiores a 10 minutos) los tiempos de la primera petición pueden incrementarse hasta 8–15 segundos, comportamiento esperado en plataformas de despliegue con suspensión automática de contenedores inactivos.

Tabla 25.
Tiempos del ciclo de sincronización física-virtual.

Aspecto evaluado	Tiempo promedio
Ciclo MQTT completo (publicación → ACK → confirmación)	742 ms
Procesamiento del mensaje en el backend	268 ms
Actualización del Digital Twin en base de datos	194 ms
Propagación de cambios al frontend vía WebSocket	312 ms
Tiempo total desde la publicación hasta la visualización en interfaz	Inferior a 1.5 segundos

El ciclo completo de sincronización del gemelo digital se completa en menos de un segundo y medio bajo condiciones normales de operación, cumpliendo con los requerimientos de monitoreo en tiempo real establecidos para la plataforma Smart Campus UIS.

Pruebas de carga concurrente y límites observados: se realizaron pruebas incrementales para identificar el punto a partir del cual el sistema comienza a experimentar degradación. Se incrementó progresivamente el número de dispositivos registrados, las conexiones WebSocket

activas y el volumen de mensajes MQTT por segundo, registrando el comportamiento del backend en cada nivel. Los resultados permitieron identificar los umbrales operativos del sistema en su configuración actual de despliegue.

Tabla 26.
Capacidad observada y umbrales del sistema.

Aspecto evaluado	Resultado observado
Dispositivos registrados sin degradación notable	25 dispositivos
Mensajes MQTT procesados por segundo de forma estable	10 mensajes/segundo
Conexiones WebSocket concurrentes estables	20 conexiones
Uso promedio de memoria del backend bajo carga	420 MB
Uso promedio de CPU del backend bajo carga	45% (picos de 78%)
Tasa de error en peticiones REST bajo carga	0.4%
Total de peticiones procesadas durante las pruebas	380 peticiones
Disponibilidad del backend durante el periodo de pruebas	97.8%

Disponibilidad del frontend durante el periodo de pruebas	99.9%

Análisis de los umbrales detectados: durante las pruebas se identificó que el sistema mantiene un comportamiento estable hasta los valores reportados en la tabla anterior. A partir de 30 dispositivos enviando telemetría simultáneamente, los tiempos de respuesta de la API REST comienzan a incrementarse de forma notable, superando el segundo de latencia. Cuando el volumen de mensajes MQTT supera los 15 mensajes por segundo de forma sostenida, se observan pérdidas ocasionales de mensajes en el broker, y al sobrepasar las 25 conexiones WebSocket simultáneas, el consumo de memoria del backend se aproxima al límite del contenedor, lo que ocasionalmente provoca reinicios automáticos del servicio.

Estos umbrales son consistentes con las características del plan de despliegue utilizado durante el desarrollo del proyecto, en el cual el backend opera con recursos limitados de memoria y CPU. Para un despliegue productivo definitivo dentro de la infraestructura del Smart Campus UIS, sería recomendable escalar verticalmente los recursos del backend, ampliar la capacidad del broker MQTT y considerar la implementación de mecanismos de balanceo de carga, ajustes que permitirían soportar volúmenes significativamente superiores de dispositivos sin modificaciones en el código del módulo.

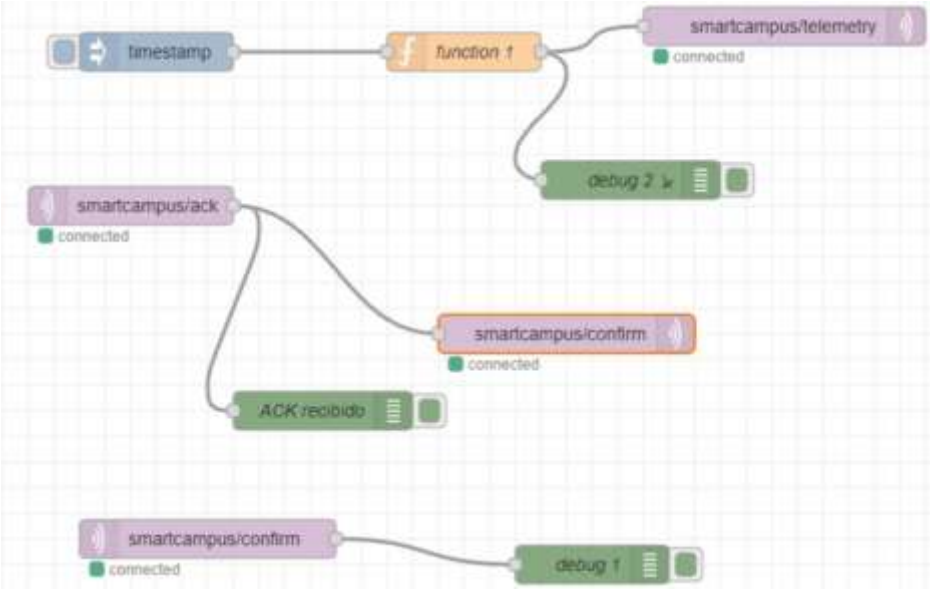
La disponibilidad observada del 97.8% en el backend se explica por la presencia de arranques en frío y reinicios automáticos del contenedor tras periodos prolongados de inactividad, comportamiento característico del plan de despliegue gratuito utilizado. El frontend, al estar

distribuido mediante una red de entrega de contenidos, mantuvo una disponibilidad cercana al 100% durante todo el periodo de validación.

9.3 Validación del Módulo Mediante Node-RED

La validación del módulo de administración se llevó a cabo mediante la implementación de un flujo de prueba en Node-RED, herramienta de programación visual basada en flujos que permite orquestar servicios, APIs y protocolos de comunicación. El flujo diseñado simula el comportamiento de un dispositivo IoT real —concretamente una Raspberry Pi— y verifica el correcto funcionamiento del ciclo completo de telemetría e intercambio de mensajes MQTT con el backend de la plataforma Smart Campus UIS.

Figura 14.
Flujo de validación del ciclo de comunicación MQTT en Node-RED.



El flujo de validación está compuesto por nueve nodos que representan los distintos momentos del ciclo de comunicación bidireccional. A continuación, se describe el rol de cada nodo dentro del escenario de validación:

Nodo 1 – timestamp (Inject): Actúa como disparador del flujo de validación. Al activarse manualmente, simula el evento en el que un dispositivo IoT inicia el envío de datos hacia la plataforma. En un escenario de producción, este nodo sería reemplazado por el dispositivo físico, que publica automáticamente según su ciclo de muestreo configurado.

Nodo 2 – function 1 (Function): Es el generador de datos simulados. Construye el mensaje JSON con los datos del dispositivo, incluyendo variables como temperatura, humedad y uso de CPU, con valores aleatorios que representan lecturas reales de sensores. Este nodo es fundamental para verificar que el backend procesa correctamente distintos valores de telemetría sin errores de parsing o validación.

Nodo 3 – smartcampus/telemetry (MQTT Out): Es el publicador de telemetría. Toma el mensaje JSON generado por el nodo anterior y lo publica en el broker MQTT HiveMQ Cloud mediante el tópico smartcampus/telemetry. El backend Spring Boot, suscrito a este tópico, recibe el mensaje y lo procesa para actualizar el Gemelo Digital correspondiente.

Nodo 4 – debug 2 (Debug): Actúa como monitor de salida del mensaje de telemetría. Muestra en el panel de depuración de Node-RED el contenido JSON enviado al broker, permitiendo verificar que el mensaje fue construido correctamente antes de su transmisión. Este nodo facilita la inspección de los datos en cada iteración del escenario de prueba.

Nodo 5 – smartcampus/ack (MQTT In): Es el receptor del mensaje de confirmación (ACK) generado por el backend. Una vez que el servidor procesa la telemetría recibida, publica un ACK

en el tópico smartcampus/ack. Este nodo, suscrito a dicho tópico, captura la respuesta del servidor como evidencia de que la telemetría fue recibida y procesada exitosamente.

Nodo 6 – ACK recibido (Debug): Muestra en el panel de depuración el mensaje de confirmación recibido del backend, permitiendo verificar el contenido del ACK y confirmar que el servidor respondió correctamente al mensaje de telemetría publicado por el dispositivo simulado.

Nodo 7 – smartcampus/confirm naranja (MQTT Out): Es el publicador de confirmación. Reenvía el ACK recibido al tópico smartcampus/confirm, simulando la respuesta del dispositivo físico que confirma la recepción del ACK del servidor. Esta acción cierra la primera fase del ciclo bidireccional y dispara la actualización final del Gemelo Digital en el backend.

Nodo 8 – smartcampus/confirm morado (MQTT In): Está suscrito al tópico smartcampus/confirm para verificar que el mensaje de confirmación fue publicado correctamente en el broker MQTT, operando como punto de verificación interna del flujo antes de registrar el resultado final de la prueba.

Nodo 9 – debug 1 (Debug): Es el monitor final del flujo. Muestra en el panel de depuración la confirmación final recibida, indicando que el ciclo completo de comunicación bidireccional se ejecutó sin errores y que el Gemelo Digital fue actualizado satisfactoriamente, completando así el escenario de validación.

9.4 Escenarios de Configuración Validados

A través del flujo Node-RED, se ejecutaron múltiples escenarios de validación orientados a verificar la coherencia, trazabilidad y funcionamiento del módulo de administración. Los escenarios cubrieron los siguientes aspectos:

- Registro de un nuevo dispositivo IoT en la plataforma y verificación de la creación automática de su Gemelo Digital con los valores iniciales correctos (status: OFFLINE, telemetryJson: {}).
- Envío de telemetría simulada desde Node-RED al tópico smartcampus/telemetry y verificación de que el backend actualiza el Gemelo Digital con los valores de temperatura, humedad y uso de CPU recibidos.
- Verificación del mensaje ACK publicado por el backend en smartcampus/ack como evidencia de procesamiento exitoso de la telemetría.
- Confirmación del ciclo bidireccional completo mediante la publicación en smartcampus/confirm y la actualización final del Digital Twin en el backend.
- Validación de la trazabilidad de eventos, comprobando que cada etapa del ciclo queda registrada con su timestamp correspondiente en la base de datos MySQL.
- Verificación de la coherencia estructural de los mensajes JSON intercambiados en cada tópico MQTT a lo largo del flujo completo.

10. Conclusiones

El presente trabajo de grado abordó el desarrollo de un módulo web de administración basado en la tecnología de Gemelo Digital para la plataforma Smart Campus UIS, con el propósito de superar las limitaciones identificadas en la gestión centralizada de dispositivos IoT, la

coherencia de los datos y la trazabilidad de las operaciones dentro del entorno universitario. El proyecto logró concretar una solución funcional, accesible y representativa del estado actual de las tecnologías aplicadas a entornos académicos inteligentes, dando respuesta efectiva a cada uno de los objetivos planteados.

A partir del análisis del estado actual de la iniciativa Smart Campus UIS se identificaron los requerimientos funcionales y no funcionales necesarios para la operación del módulo, así como los actores involucrados y sus interacciones dentro del sistema. Esta caracterización inicial permitió establecer una base estructurada que orientó las decisiones de diseño y desarrollo, dando lugar a una arquitectura cliente-servidor desacoplada, organizada por capas y soportada en una infraestructura distribuida que integra el frontend, el backend, la base de datos y la capa de comunicación con dispositivos IoT mediante servicios REST, WebSocket y un canal MQTT seguro. La decisión arquitectónica más relevante consistió en establecer una relación uno a uno entre las entidades Device y DigitalTwin, garantizando la integridad referencial desde el momento de creación de cada dispositivo y eliminando la posibilidad de inconsistencias entre el entorno físico y su representación virtual.

Sobre esta base de diseño se desarrolló un prototipo funcional completamente operativo, implementado con Java 21 y Spring Boot 3.3.5 en el backend, React con Vite en el frontend y MySQL 8 como motor de persistencia, con todos los componentes desplegados en entornos de nube accesibles públicamente. El prototipo cubre la totalidad de los requerimientos funcionales definidos, incluyendo el ciclo CRUD completo de dispositivos, la creación automática de gemelos digitales, la sincronización bidireccional mediante MQTT, el motor de reglas básico, el sistema de alertas y notificaciones por correo, la autenticación con tokens JWT y el registro histórico de eventos. Como producto tangible del trabajo se entrega un módulo accesible desde la URL

https://smartcampus-admin-module.vercel.app/, con el código fuente disponible en repositorio público para su consulta, mantenimiento y extensión por parte de futuros equipos de desarrollo dentro de la universidad.

La validación del módulo se ejecutó a través de tres etapas complementarias que arrojaron resultados cuantitativos verificables. Se realizaron treinta pruebas funcionales sobre los flujos críticos del sistema y veinte pruebas unitarias sobre los servicios principales del backend, alcanzando en ambos casos una tasa de éxito del 100%. Las mediciones de los tiempos de respuesta de la API REST arrojaron un promedio general de 341 milisegundos, y la verificación del ciclo de sincronización física-virtual confirmó que el sistema completa el ciclo en menos de un segundo y medio. Adicionalmente, las pruebas de carga permitieron identificar los umbrales operativos del módulo en su configuración actual de despliegue, observándose un funcionamiento estable hasta veinticinco dispositivos concurrentes y veinte conexiones WebSocket simultáneas, con una disponibilidad superior al 97% durante el periodo de validación. Estos resultados demuestran empíricamente que el módulo cumple con las funcionalidades esperadas y opera con tiempos de respuesta adecuados para escenarios de monitoreo en tiempo real, satisfaciendo los criterios establecidos en los requerimientos no funcionales del sistema.

El trabajo realizado constituye un aporte significativo en varios niveles. A nivel local, el módulo desarrollado cubre una necesidad concreta de la plataforma Smart Campus UIS, dotándola por primera vez de una herramienta de administración centralizada que articula el inventario físico de dispositivos con sus representaciones virtuales, fortaleciendo el proceso de digitalización institucional impulsado por la Universidad Industrial de Santander. A nivel regional, el proyecto se posiciona como un referente de aplicación de tecnologías emergentes como el Gemelo Digital, el Internet de las Cosas y la comunicación MQTT en tiempo real dentro del ámbito universitario

colombiano, en un contexto donde estas iniciativas todavía son escasas. A nivel nacional e internacional, el trabajo contribuye al campo emergente de los Smart Campus universitarios mediante una propuesta abierta, extensible y completamente documentada, alineada con las tendencias globales descritas en la literatura científica reciente sobre la integración de gemelos digitales en entornos académicos inteligentes.

Finalmente, el módulo se entrega bajo una arquitectura que permite su evolución progresiva, contemplando explícitamente las líneas de trabajo futuro identificadas, lo que garantiza que la solución desarrollada no sea un producto cerrado sino una base sólida sobre la cual la universidad puede continuar construyendo capacidades inteligentes adicionales. En síntesis, el desarrollo del módulo web de administración basado en Gemelo Digital alcanza los objetivos planteados, demuestra su viabilidad técnica mediante resultados cuantitativos verificables y aporta a la Universidad Industrial de Santander una herramienta funcional, escalable y representativa del estado del arte en tecnologías de gestión inteligente de entornos universitarios.

11.Trabajo Futuro

El módulo de administración basado en gemelo digital desarrollado en este proyecto constituye la base funcional sobre la cual la plataforma Smart Campus UIS puede crecer, incorporar nuevas capacidades y consolidarse como un ecosistema institucional inteligente. Si bien los objetivos planteados fueron alcanzados y validados en un entorno controlado, el alcance del trabajo realizado abre múltiples líneas de extensión que pueden ser abordadas en investigaciones y desarrollos posteriores. A continuación, se presentan las direcciones de trabajo futuro identificadas, organizadas en cuatro ejes principales.

11.1 Validación con Dispositivos Físicos Reales

La validación del módulo realizada en el presente proyecto se ejecutó mediante dispositivos simulados a través de Node-RED, los cuales permitieron verificar el ciclo completo de comunicación bidireccional y el comportamiento del gemelo digital ante distintos escenarios de telemetría. Como continuación natural del trabajo, se propone ejecutar una fase de validación con dispositivos IoT físicos reales desplegados en distintas zonas del campus universitario, con el propósito de evaluar el comportamiento del sistema en condiciones operativas auténticas.

Esta validación física debería contemplar al menos los siguientes elementos:

Integración con hardware específico: desplegar dispositivos basados en Raspberry Pi, ESP32 y microcontroladores similares, equipados con sensores reales de temperatura, humedad, ocupación, consumo energético y calidad del aire. La conexión de estos dispositivos al broker MQTT permitirá validar el comportamiento del módulo ante variaciones reales del entorno, latencias de red institucional y patrones de telemetría no replicables mediante simulación.

Despliegue piloto en una zona específica del campus: identificar un edificio o piso de la universidad como escenario piloto para la implementación del módulo, permitiendo evaluar su funcionamiento en un contexto representativo antes de extender la solución a la totalidad de la infraestructura. Esta aproximación incremental facilita la detección temprana de incidencias y la recopilación de retroalimentación por parte de los usuarios reales del sistema.

Pruebas de resistencia y disponibilidad prolongada**:** ejecutar pruebas de operación continua durante periodos extendidos (semanas o meses) que permitan evaluar el comportamiento del sistema ante factores como pérdida intermitente de conexión, reinicios de dispositivos, agotamiento de batería en sensores autónomos y variaciones de carga en distintos momentos del día.

Medición de capacidad real: verificar empíricamente los umbrales operativos identificados durante la validación en entorno simulado, ajustando los valores reportados en función del comportamiento observado con dispositivos físicos. Esta medición permitirá dimensionar con precisión los recursos de infraestructura necesarios para un despliegue institucional a gran escala.

11.2 Ampliación del Entorno de Despliegue

Los resultados obtenidos durante la validación evidenciaron umbrales de capacidad asociados al plan de despliegue gratuito utilizado durante el desarrollo del proyecto. Como línea de trabajo futuro se propone migrar el módulo a una infraestructura institucional dedicada que permita superar las limitaciones actuales de memoria, CPU y conexiones concurrentes.

Esta migración debería contemplar el escalado vertical del backend para soportar volúmenes superiores de dispositivos, la implementación de mecanismos de balanceo de carga ante el crecimiento del número de usuarios concurrentes, la ampliación del plan del broker MQTT para sostener throughput superior al actual, y la incorporación de réplicas de la base de datos que garanticen alta disponibilidad. Adicionalmente, se recomienda evaluar la posibilidad de alojar el sistema dentro de la infraestructura tecnológica de la Universidad Industrial de Santander, garantizando soberanía sobre los datos generados por los dispositivos del campus y reduciendo la dependencia de servicios externos.

11.3 Nuevas Integraciones Funcionales

El paradigma del Gemelo Digital y la naturaleza extensible del protocolo MQTT permiten que el módulo desarrollado funcione como una plataforma base sobre la cual se pueden integrar múltiples capacidades adicionales orientadas a enriquecer la gestión del Smart Campus UIS. A continuación, se identifican las integraciones más relevantes propuestas como trabajo futuro.

Analítica avanzada y aprendizaje automático: explotar el registro histórico de telemetría almacenado en la base de datos para entrenar modelos de aprendizaje automático orientados al mantenimiento predictivo, la detección de anomalías en el comportamiento de los dispositivos y la predicción de consumo de recursos. La integración con bibliotecas de ciencia de datos permitiría generar reportes inteligentes que apoyen la toma de decisiones estratégicas dentro de la administración del campus, ampliando el alcance actual del motor de reglas hacia automatizaciones basadas en patrones aprendidos del comportamiento histórico.

Visualización geoespacial del campus: integrar un mapa interactivo del campus universitario que permita ubicar los dispositivos físicamente, visualizar su estado en tiempo real sobre la planta arquitectónica de cada edificio y filtrar la información por zona, edificio o tipo de sensor. Esta funcionalidad complementaría el dashboard actual con una capa espacial que mejoraría la identificación rápida de incidencias en ubicaciones específicas del campus.

Gestión integral de aplicaciones IoT compuestas: ampliar el modelo de datos para soportar la gestión de aplicaciones IoT compuestas por múltiples dispositivos relacionados, permitiendo definir flujos de trabajo entre sensores y actuadores, agrupar dispositivos por escenarios de uso (por ejemplo, gestión energética de un edificio completo) y administrar el ciclo de vida completo de una solución IoT desde una única interfaz.

Interoperabilidad con plataformas externas: desarrollar conectores que permitan integrar el módulo con plataformas comerciales de gestión IoT como AWS IoT Device Management, Azure IoT Hub o Google Cloud IoT, así como con sistemas institucionales existentes en la universidad. Esta interoperabilidad permitiría sincronizar el inventario de dispositivos con sistemas externos cuando sea necesario, sin perder el control centralizado del Smart Campus UIS.

Aplicación móvil para administradores: desarrollar una aplicación móvil complementaria al módulo web que permita a los administradores recibir notificaciones críticas, consultar el estado de los dispositivos y ejecutar acciones básicas desde dispositivos portátiles, facilitando la respuesta ante incidencias sin necesidad de acceso a un equipo de escritorio.

Soporte para múltiples protocolos de comunicación: ampliar el sistema para soportar protocolos adicionales al MQTT actual, como CoAP para dispositivos de muy bajo consumo energético, LoRaWAN para dispositivos distribuidos en zonas extensas del campus sin cobertura WiFi, y OPC UA para integración con sistemas industriales presentes en laboratorios y zonas técnicas de la universidad.

Gemelos digitales jerárquicos y agregados: evolucionar el modelo actual de gemelo digital uno-a-uno hacia un esquema jerárquico que permita representar gemelos digitales de entidades compuestas, como un salón completo (agrupando sus sensores, luminarias y aires acondicionados), un piso o un edificio entero. Esta abstracción habilitaría análisis y operaciones a niveles superiores del campus sin perder el detalle individual de cada dispositivo.

11.4 Seguridad y Gobernanza de Datos

La validación del módulo se enfocó en su comportamiento funcional y de capacidad, sin contemplar pruebas específicas de seguridad informática orientadas a evaluar la resistencia del sistema ante ataques externos. Como línea de trabajo futuro se propone abordar este aspecto mediante las siguientes acciones:

Auditoría de seguridad integral: ejecutar pruebas de penetración sobre el módulo desplegado, identificando vulnerabilidades en la API REST, en la conexión WebSocket y en el manejo de mensajes MQTT. Estas pruebas deberían incluir intentos de inyección de mensajes

maliciosos, ataques de denegación de servicio y validación del manejo de tokens de autenticación bajo condiciones adversas.

Integración con el sistema de identidad institucional: complementar el sistema de autenticación JWT actualmente implementado con la integración al sistema de identidad institucional de la Universidad Industrial de Santander mediante Single Sign-On (SSO), permitiendo a los administradores acceder al módulo con sus credenciales institucionales sin requerir la gestión de cuentas adicionales.

Cifrado de datos en reposo: complementar el cifrado actualmente aplicado en tránsito (HTTPS y TLS sobre MQTT) con cifrado de la información sensible almacenada en la base de datos, particularmente aquella relacionada con credenciales de dispositivos y configuraciones críticas del sistema.

Cumplimiento normativo: alinear el módulo con la normativa colombiana de protección de datos personales (Ley 1581 de 2012) y con las políticas institucionales de gestión de información de la Universidad Industrial de Santander, contemplando aspectos como el consentimiento informado para la captura de datos, los plazos de retención de información y los procedimientos de respuesta ante incidentes de seguridad.

Referencias Bibliográficas

Amazon Web Services. (s. f.). *AWS IoT Device Management*. Recuperado el 25 de agosto de 2025, de https://aws.amazon.com/es/iot-device-management/

Amazon Web Services. (s. f.). *What is IoT?* https://aws.amazon.com/iot/

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3.ª ed.). Addison-Wesley.

Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, *8*, 108952–108971. https://doi.org/10.1109/ACCESS.2020.2998358

Glaessgen, E., & Stargel, D. (2012, abril). *The digital twin paradigm for future NASA and U.S. Air Force vehicles* [Ponencia]. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Honolulu, Hawái. https://doi.org/10.2514/6.2012-1818

Google Cloud. (2024, agosto 9). *Arquitectura de productos de la plataforma de IoT en Google Cloud*. https://cloud.google.com/architecture/connected-devices/iot-platform-product-architecture?hl=es-419

Grieves, M., & Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. En F.-J. Kahlen, S. Flumerfelt, & A. Alves (Eds.), *Transdisciplinary perspectives on complex systems: New findings and approaches* (pp. 85–113). Springer. https://doi.org/10.1007/978-3-319-38756-7_4

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Hernández, D., & Muñoz, J. (2020). Smart campus: Una aproximación a la transformación digital de las universidades. *Revista Iberoamericana de Tecnologías del Aprendizaje*, *15*(4), 208–215.

IBM. (2006). *An architectural blueprint for autonomic computing* (4.ª ed.). IBM Corporation. https://www.ibm.com/downloads/cas/WWEZOWV9

IBM. (2020). *Cheat sheet: What is digital twin?* IBM Think. Recuperado el 19 de septiembre de 2025, de https://www.ibm.com/think/topics/digital-twin

Jiménez, H., Cárcamo, E., & Pedraza, G. (2020). Plataforma software extensible para campus inteligente basada en microservicios. *RISTI: Revista Ibérica de Sistemas e Tecnologias de Informação*, (Extra 38), 270–282.

Maataoui, F. (2024). *Gemelo digital de un dispositivo IoT: Desarrollo y validación del registrador de temperatura y humedad Akodata* [Trabajo de fin de grado, Universitat Politècnica de Catalunya]. UPCommons. https://upcommons.upc.edu/entities/publication/88d6f388-8b38-4fa6-bc22-13a1ce6d33c8

Microsoft. (s. f.). *Azure IoT Hub*. Recuperado el 25 de agosto de 2025, de https://azure.microsoft.com/es-es/products/iot-hub

Min-Allah, N., Alrashed, S., Ali, I., Alnumay, W., & Mashaly, M. (2019). A survey of smart campus: Challenges and opportunities. *IEEE Access*, *7*, 137044–137066. https://doi.org/10.1109/ACCESS.2019.2942313

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann. ISO. (2018). *ISO 9241-11:2018 Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. International Organization for Standardization.

Repsol. (s. f.). *Gemelos digitales: Qué es, tipos y ejemplos*. Recuperado el 19 de septiembre de 2025, de https://www.repsol.com/es/energia-avanzar/innovacion/gemelos-digitales/index.cshtml

Rodríguez Martínez, P. (2025). *La intercomunicación de gemelos digitales: Estado del arte sobre la conexión de entidades de gemelos digitales* [Trabajo de fin de máster, Universidad Politécnica de Madrid]. OA UPM. https://oa.upm.es/85437/1/TFM_PABLO_RODRIGUEZ_MARTINEZ.pdf

Santhi, T., Rajendra, J., & Vijayalakshmi, Y. (2016). A review on the state of the art of Internet of Things. *International Journal of Advanced Research in Computer and Communication Engineering*. https://www.ijarcce.com/upload/2016/july-16/IJARCCE%2038.pdf

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, *15*(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

Villanueva, D., & Cáceres, A. (2021). Smart campus and sustainable development goals: A systematic review. *Sustainability*, *13*(22), 12456. https://doi.org/10.3390/su132212456

Apéndices

Apéndice A**.**

Plan de pruebas funcionales del sistema

Este apéndice detalla los treinta (30) casos de prueba funcionales ejecutados sobre el módulo de administración basado en Gemelo Digital, organizados en las siete categorías presentadas en la sección 9.1, especificando para cada caso el objetivo evaluado, la entrada utilizada y el resultado esperado.

A.1 Registro de dispositivos y creación automática del gemelo digital

#	Caso de prueba	Entrada	Resultado esperado
1	Registro de dispositivo tipo SENSOR	POST /devices con type: SENSOR y campos obligatorios completos	Dispositivo creado; gemelo digital generado automáticamente con status: OFFLINE, telemetryJson: {}
2	Registro de dispositivo tipo ACTUATOR	POST /devices con type: ACTUATOR	Dispositivo y gemelo digital creados correctamente
3	Registro de dispositivo tipo CAMERA	POST /devices con type: CAMERA	Dispositivo y gemelo digital creados correctamente
4	Registro de dispositivo tipo CONTROLLER	POST /devices con type: CONTROLLER	Dispositivo y gemelo digital creados correctamente
5	Registro con código duplicado	POST /devices con un code ya existente	El sistema rechaza la operación y retorna un error de duplicidad
6	Registro con campos obligatorios faltantes	POST /devices sin campo name o type	El sistema retorna un error de validación y no crea el dispositivo

A.2 Actualización de telemetría en el Digital Twin vía MQTT

#	Caso de prueba	Entrada	Resultado esperado
1	Telemetría de temperatura	Mensaje MQTT en smartcampus/telemetry con campo temperature	El campo telemetryJson del gemelo digital se actualiza con el valor recibido
2	Telemetría de humedad	Mensaje MQTT con campo humidity	El campo telemetryJson se actualiza correctamente
3	Telemetría de uso de CPU	Mensaje MQTT con campo cpuUsage	El campo telemetryJson se actualiza correctamente
4	Mensaje con múltiples campos simultáneos	Mensaje MQTT con temperature, humidity y cpuUsage en un solo payload	Todos los campos se reflejan correctamente en telemetryJson
5	Actualización de marca temporal	Envío de cualquier mensaje de telemetría	El campo lastUpdate del gemelo digital se refresca con el timestamp de recepción

A.3 Endpoints REST

#	Caso de prueba	Entrada	Resultado esperado
1	GET /devices	Petición sin parámetros	Retorna el listado completo de dispositivos (200 OK)
2	GET /devices/{id}	ID de dispositivo existente	Retorna el detalle del dispositivo (200 OK)
3	POST /devices	Datos válidos de un nuevo dispositivo	Dispositivo creado (201 Created)
4	PUT /devices/{id}	ID existente y datos actualizados	Dispositivo actualizado (200 OK)
5	DELETE /devices/{id}	ID de dispositivo existente	Dispositivo y gemelo digital asociado eliminados (200/204)
6	GET /twins/{id}	ID de gemelo digital existente	Retorna el estado del gemelo digital (200 OK)
7	GET /twins	Petición sin parámetros	Retorna el listado completo de gemelos digitales (200 OK)
8	GET /devices/{id} con ID inexistente	ID no registrado en el sistema	El sistema retorna un error HTTP 404 controlado

A.4 Persistencia e integridad en MySQL

#	Caso de prueba	Entrada	Resultado esperado
1	Persistencia tras registro	Creación de un dispositivo vía POST	El registro se refleja correctamente en la base de datos MySQL
2	Integridad referencial 1:1	Consulta cruzada Device–DigitalTwin	Cada dispositivo tiene asociado exactamente un gemelo digital
3	Eliminación en cascada	DELETE de un dispositivo con gemelo digital asociado	El gemelo digital se elimina automáticamente junto con el dispositivo
4	Consistencia tras actualización	PUT sobre un dispositivo existente	Los cambios se reflejan de forma consistente en la base de datos

A.5 Comunicación WebSocket frontend-backend

#	Caso de prueba	Entrada	Resultado esperado
1	Establecimiento de conexión STOMP	Apertura del frontend	Conexión WebSocket STOMP establecida correctamente con el backend
2	Notificación en tiempo real	Cambio de estado en un gemelo digital	El frontend recibe la notificación y actualiza la vista sin recargar
3	Reconexión automática	Corte temporal de la conexión de red	El cliente reestablece la conexión WebSocket automáticamente al restablecerse la red

A.6 Disponibilidad del backend

#	Caso de prueba	Entrada	Resultado esperado
1	Verificación de health check	Petición periódica al endpoint de salud del backend	Respuesta 200 OK durante el periodo de pruebas
2	Disponibilidad tras inactividad	Petición tras un periodo de inactividad prolongado	El backend responde, aunque con mayor latencia por arranque en frío

A.7 Disponibilidad del frontend

#	Caso de prueba	Entrada	Resultado esperado
1	Carga directa desde navegador	Acceso a la URL del frontend	La interfaz carga correctamente
2	Disponibilidad continua	Verificaciones periódicas durante el periodo de pruebas	Frontend accesible de forma consistente (cercano al 100%)