Universidad Zaragoza
1542

Trabajo Fin de Grado

Análisis y diseño de un sistema IoT de extracción, procesamiento, visualización y predicción de datos en entornos industriales

IoT System Analysis and Design for Data Extraction, Processing, Visualization and Prediction in Industrial Environments

Autor

Revilla Mata, Eduardo

Director

Cortés Arcos, Tomás

Escuela Universitaria Politécnica La Almunia

Septiembre 2024

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Análisis y diseño de un sistema IoT de extracción, procesamiento, visualización y predicción de datos en entornos industriales

IoT System Analysis and Design for Data Extraction, Processing, Visualization and Prediction in Industrial Environments

424.20.48

Autor: Revilla Mata, Eduardo

Director: Cortés Arcos, Tomás

Fecha: 09 2024

INDICE DE CONTENIDO BREVE

1. RESUMEN ____ **1**

2. ABSTRACT ____ **3**

3. INTRODUCCIÓN ____ **5**

4. ESTADO DEL ARTE ____ **11**

5. DESARROLLO ____ **40**

6. RESULTADOS ____ **101**

7. CONCLUSIONES ____ **106**

8. OBJETIVOS DE DESARROLLO SOSTENIBLE ____ **108**

9. BIBLIOGRAFÍA ____ **109**

INDICE DE CONTENIDO

1. RESUMEN 1

1.1. PALABRAS CLAVE 2

2. ABSTRACT 3

2.1. KEY WORDS 4

3. INTRODUCCIÓN 5

3.1. JUSTIFICACIÓN DEL TRABAJO 5

3.2. OBJETIVOS 6

3.3. ALCANCE 7

3.4. FASES DE TRABAJO 9

4. ESTADO DEL ARTE 11

4.1. ANTECEDENTES 12

4.1.1. Predix de General Electric 12

4.1.2. Mindsphere de Siemens 13

4.2. MARCO TEÓRICO 15

4.2.1. Industria 4.0 15

4.2.2. Internet of Things (IoT) 16

4.2.3. Industrial Internet of Things (IIoT) 18

4.2.3.1. Protocolos de comunicación IIoT 19

4.2.3.1.1. MQTT (Message Queue Telemetry Transport) 19

4.2.3.1.2. Open Protocol Communication Unified Architecture 22

4.2.4. Pirámide CIM 25

4.2.5. Arquitectura de sistemas IoT 27

4.2.6. Arquitectura de soluciones de software 28

4.2.6.1. Descripción general 28

4.2.6.2. Elementos de una web 28

4.2.6.3. Tipos de arquitecturas web 28

4.2.6.3.1. Clasificación por modelo de servicio 29

4.2.6.3.2. Clasificación por modelo de comunicación 30

4.2.7. Bases de datos 31

4.2.7.1. Relacionales 31

4.2.7.1. No relacionales 31
4.2.7.2. PostgreSQL 32
4.2.8. Inteligencia artificial 32
4.2.8.1. Aprendizaje supervisado 32
4.2.8.2. Aprendizaje no supervisado 33
4.2.8.1. Modelo Random Forest 33
4.2.9. Herramientas 34
4.2.9.1. Docker y contenedores 34
4.2.9.2. Python 34
4.2.9.3. Reflex.dev 34
4.2.9.3.1. Arquitectura y diseño 35
4.2.9.3.2. Frontend 35
4.2.9.3.3. Backend 36
4.2.9.3.4. Procesamiento de eventos 36
4.2.9.4. Librería paho-mqtt 37
4.2.9.5. Librería opcua 37
4.2.9.1. Librerías Pandas y Numpy 38
4.2.9.2. Librerías SQLAlchemy y SQLModel 38
4.2.9.3. Librerías Passlib y Bcrypt 38
4.2.9.1. Librería Scikit-learn 39

5. DESARROLLO 40

5.1. ESPECIFICACIONES DEL SISTEMA IOT 40
5.1.1. Requisitos funcionales 40
5.1.2. Requisitos no funcionales 41
5.2. ANÁLISIS Y TECNOLOGÍAS 42
5.2.1. Análisis del problema 42
5.2.2. Análisis del entorno 43
5.2.2.1. Protocolos de comunicación 43
5.2.2.2. Hardware 45
5.2.2.2.1. Simulación de dispositivos 45
5.2.2.2.2. Alojamiento de la plataforma 45
5.2.2.3. Software 45
5.2.2.4. Almacenamiento de datos 46
5.2.2.5. Predicción de datos 46
5.2.1. Especificaciones técnicas 48
5.3. DISEÑO DEL SISTEMA IIOT 49

5.3.1. Arquitectura de infraestructura ____ *49*
5.3.1.1. Raspberry PI ____ 50
5.3.1.2. PC Windows ____ 51
5.3.1.3. Navegador web ____ 52
5.3.2. Arquitectura de plataforma ____ *52*
5.3.2.1. Diseño funcional ____ 52
5.3.2.1.1. Casos de uso ____ 53
5.3.2.2. Diseño de la interfaz ____ 54
5.3.2.2.1. Funcionalidades principales ____ 55
5.3.2.3. Diseño de seguridad ____ 57
5.3.3. Arquitectura del dato ____ *58*
5.3.3.1. Módulo MQTT ____ 58
5.3.3.2. Módulo OPC UA ____ 59
5.3.3.3. Viaje del dato ____ 59
5.3.4. Arquitectura de almacenamiento ____ *60*
5.3.4.1. Modelo de entidad-relación ____ 60
5.3.4.2. Modelo relacional ____ 65
5.3.4.3. Normalización ____ 67
5.3.1. Arquitectura de predicciones ____ *68*
5.4. IMPLEMENTACIÓN DEL SISTEMA IIOT ____ 69
5.4.1. Simulación de entornos industriales ____ *69*
5.4.1.1. Dispositivo MQTT ____ 69
5.4.1.2. Broker MQTT ____ 71
5.4.1.3. Servidor OPC UA ____ 72
5.4.1.3.1. OPCUA_config.json ____ 73
5.4.1.3.1. Telemetry_config.json ____ 73
5.4.2. Plataforma IIoT ____ *74*
5.4.2.1. Base de datos ____ 74
5.4.2.2. Estructura de archivos ____ 74
5.4.2.3. Backend ____ 75
5.4.2.3.1. Módulo de extracción MQTT ____ 75
5.4.2.3.1. Módulo de extracción OPC UA ____ 78
5.4.2.3.2. Módulo de predicciones ____ 80
5.4.2.4. Frontend y estados ____ 81
5.4.2.4.1. Welcome ____ 81
5.4.2.4.1. Authentification ____ 84
5.4.2.4.1. Verify ____ 88

5.4.2.4.2. Home 90
5.4.2.4.3. Dashboard 92
5.4.2.4.4. New Factory 95
5.4.2.4.5. New Source 97

6. RESULTADOS 101

6.1. CAPTURA DE DATOS 101
6.2. PROCESAMIENTO DE DATOS 102
6.3. ALMACENAMIENTO DE DATOS 102
6.4. VISUALIZACIÓN DE DATOS 102
6.5. PREDICCIÓN DE DATOS 103
6.6. GESTIÓN DE USUARIOS Y AUTENTICACIÓN 103
6.7. RENDIMIENTO 104
6.8. SEGURIDAD 104
6.9. USABILIDAD 104
6.10. MANTENIBILIDAD 105
6.11. COMPATIBILIDAD 105

7. CONCLUSIONES 106

8. OBJETIVOS DE DESARROLLO SOSTENIBLE 108

9. BIBLIOGRAFÍA 109

INDICE DE ILUSTRACIONES

Ilustración 1 Diagrama de funcionamiento de Predix GE 12

Ilustración 2 Diagrama funcionamiento de Mindsphere Siemens 14

Ilustración 3 Desarrollo tecnológico durante las revoluciones industriales 15

Ilustración 4 Tipos de tecnologías inalámbricas 17

Ilustración 5 Comparativa de las tecnologías inalámbricas según su rango y su ancho de banda 17

Ilustración 6 Relaciones del IIoT con la Industria 4.0, IoT, IT y OT 18

Ilustración 7 Comparativa de eficiencia de los protocolos IIoT 19

Ilustración 8 Arquitectura de publicación y suscripción del protocolo MQTT .. 20

Ilustración 9 Dataframe de un mensaje MQTT 20

Ilustración 10 OPCUA es la pirámide CIM 23

Ilustración 11 Diagrama de arquitectura OPCUA 24

Ilustración 12 Pirámide CIM: Niveles de Integración y Automatización 25

Ilustración 13 Comparación de los modelos de arquitectura IoT más extendidas 27

Ilustración 14 Clasificación según el modelo de servicio 29

Ilustración 15 Clasificación según el modelo de 30

Ilustración 16 Diagrama de funcionamiento web en Reflex.dev 35

Ilustración 17 Diagrama UML de implementación de la plataforma con Reflex y Python 46

Ilustración 18 Diagrama de alto nivel de la infraestructura del sistema IIoT diseñado 50

Ilustración 19 Diagrama de bloques de simulación mediante la Raspberry Pi. 50

Ilustración 20 Diagrama de bloques de bajo nivel del PC que actúa como servidor 51

Ilustración 21 Diagrama de bloques de la interacción de usuarios con la plataforma IIoT 52

Ilustración 22 Diagrama UML de casos de uso de la plataforma 53

Ilustración 23 Diagrama de bloques d navegación por las páginas 54

Ilustración 24 Diagrama UML de actividad para autentificación y autorización 55

Ilustración 25 Diagrama UML de actividad para gestión de fábricas y fuentes de datos 56

Ilustración 26 Diagrama UML de actividad para visualización de datos..... 57

Ilustración 27 Diagrama del diseño por capas del dato en la plataforma 58

Ilustración 28 Diagrama del módulo de comunicación MQTT 59

Ilustración 29 Diagrama del módulo de comunicación OPC UA..... 59

Ilustración 30 Diagrama de integración de módulos de comunicación 60

Ilustración 31 Modelo ER de gestión de datos 61

Ilustración 32 Modelo ER de gestión de usuarios 64

Ilustración 33 Modelo MR alto nivel de gestión de datos 65

Ilustración 34 Modelo MR bajo nivel fábrica-dispositivo 66

Ilustración 35 Modelo MR bajo nivel dispositivo-variable-parámetros..... 66

Ilustración 36 Modelo MR bajo nivel dispositivo-variable-datos 67

Ilustración 37 Modelo MR de gestión de usuarios 67

Ilustración 38 Módulo de predicción 68

Ilustración 39 Diagrama completo del backend de la plataforma IIoT 68

Ilustración 40 Diagrama UML de actividad de mqtt_client_publish 69

Ilustración 41 Información básica del Broker MQTT 71

Ilustración 42 Diagrama UML de actividad de OPCUA_server 72
Ilustración 43 Tablas de base de datos 74
Ilustración 44 Diagrama UML de clases del módulo MQTT 76
Ilustración 45 Diagrama UML de secuencia del módulo MQTT 77
Ilustración 46 Modulo OPCUA diagrama UML de clases 78
Ilustración 47 Modulo OPCUA diagrama UML de secuencia 79
Ilustración 48 Diagrama UML de clases del módulo de predicción 80
Ilustración 49 Diagrama UML de clases del módulo de predicción 80
Ilustración 50 Welcome page 81
Ilustración 51 Diagrama de componentes de Welcome page 83
Ilustración 52 Diagrama de secuencias de Welcome page 83
Ilustración 53 Auth page con login tab 84
Ilustración 54 Register page con regsiter tab 84
Ilustración 55 Diagrama de componentes de Auth page 85
Ilustración 56 Mapa de estados de Auth page 86
Ilustración 57 Diagrama de secuencia del proceso de login 87
Ilustración 58 Diagrama de secuencia de proceso de registro 88
Ilustración 59 Diagrama de componentes de Verify page 89
Ilustración 60 Diagrama de secuencia de verificación de nuevos usuarios 89
Ilustración 61 Home page 90
Ilustración 62 Diagrama de componentes de Home page 91
Ilustración 63 Diagrama de secuencia del on_load para el sidebar del template 91
Ilustración 64 Dashboard page 92
Ilustración 65 Diagrama de componentes de Dashboard page 93

Ilustración 66 Mapa de estados de carga de datos en Dashboard Page 93

Ilustración 67 Diagrama de secuencia de selección y carga de datos en Dashboard page 94

Ilustración 68 Diagrama de secuencia de actualización de datos en Dashboard page..... 94

Ilustración 69 New Factory page..... 95

Ilustración 70 Diagrama de componentes de New Factory page..... 95

Ilustración 71Diagrma de secuencia de New Factory page 96

Ilustración 72 New Source page - mqtt section 97

Ilustración 73 New Source page - opcua section 97

Ilustración 74 Diagrama de componentes de New Source page 98

Ilustración 75 Diagrama de estados de configuración de nuevos dispositivos 99

Ilustración 76 Diagrama de secuencia creación de dispositivos en New Source page.....100

INDICE DE TABLAS

Tabla 1 Tipos de mensajes empleados en el protocolo MQTT [16] 20

Tabla 2 Requisitos funcionales del sistema 40

Tabla 3 Requisitos no funcionales del sistema 41

Tabla 4 Comparativa entre los principales protocolos de comunicación IIoT 43

Tabla 5 Tabla comparativa de los principales broker MQTT gratuitos 44

Tabla 6 Comparativa de los modelos no supervisados de machine learning 47

Tabla 7 Especificaciones técnicas del proyecto 48

Tabla 8 Configuración de los usuarios y accesos al Broker MQTT 71

Tabla 9 Resultados obtenidos tras la implementación de la captura de datos 101

Tabla 10 Resultados de la implementación de la procesamiento de datos.....102

Tabla 11 Resultados de la implementación del almacenamiento de datos.....102

Tabla 12 Resultados obtenidos tras la implementación de visualización de datos 102

Tabla 13 Resultados obtenidos tras la implementación de predicción de datos 103

Tabla 14 Resultados de la implementación de autenticación de usuarios.....103

Tabla 15 Resultados de la implementación de rendimiento 104

Tabla 16 Resultados de la implementación de la procesamiento de datos.....104

Tabla 17 Resultados de la implementación del almacenamiento de datos.....104

Tabla 18 Resultados de la implementación de visualización de datos.....105

Tabla 19 Resultados de la implementación de predicción de datos 105

Tabla 20 Tabla de posibles mejoras de la plataforma IoT.....106

1. RESUMEN

En el contexto industrial actual, muchas empresas líderes han descuidado la implementación de estrategias de optimización y automatización de procesos, lo que ha resultado en una pérdida de competitividad. Este descuido ha generado una brecha en la eficiencia operativa que es crucial abordar para mantener o mejorar la posición en el mercado. A través de estas estrategias, se busca optimizar los procesos productivos para mejorar la calidad de los productos finales, reducir costes y tiempos de producción, y alargar la vida útil de la maquinaria. Los datos se han convertido en la herramienta principal para llevar a cabo estas tareas de optimización de parámetros configurables, consumos eléctricos y cualquier variable relacionada con la producción.

En este proyecto, se ha diseñado una plataforma integral de Industria 4.0 basada en tecnologías IoT, que se integra paralelamente con los procesos de producción para ofrecer un sistema de control y análisis de datos industriales. La plataforma aborda de manera eficiente la problemática de la dispersión de herramientas en el mercado y los altos costes de implementación de soluciones ad-hoc, ofreciendo una solución global y escalable.

La plataforma diseñada realiza un recorrido completo desde la extracción hasta la predicción y visualización de datos en entornos industriales. La arquitectura de la plataforma IIoT se conecta a dispositivos a través de protocolos de comunicación MQTT y OPCUA, procesando, almacenando y prediciendo el comportamiento de los datos. Un servicio de machine learning para predicción basado en el modelo random forest y los resultados se visualizan en la plataforma.

Uno de los aspectos más destacables de la plataforma es el uso de Python como único lenguaje de programación para desarrollar una aplicación completa y su enfoque modular, que permite una alta escalabilidad. Esto se traduce en la capacidad de incorporar nuevos tipos de comunicaciones con dispositivos y de implementar diferentes modelos de predicción según las necesidades específicas de cada empresa. Este diseño modular facilita el viaje del dato desde su extracción, procesamiento y almacenamiento, hasta su análisis y visualización, asegurando que las empresas puedan adaptar la plataforma a sus necesidades cambiantes y seguir siendo competitivas en el mercado.

1.1. PALABRAS CLAVE

Industria 4.0, IoT Industrial (IIoT), automatización industrial, análisis del dato, Machine Learning.

2. ABSTRACT

In the current industrial context, many leading companies have neglected the implementation of process optimization and automation strategies, resulting in a loss of competitiveness. This oversight has created a gap in operational efficiency that is crucial to address in order to maintain or improve market position. Through these strategies, the goal is to optimize production processes to improve the quality of final products, reduce costs and production times, and extend the life of machinery. Data has become the main tool to carry out tasks such as optimizing configurable parameters, reducing electrical consumption, and monitoring any variable related to production.

In this project, a comprehensive Industry 4.0 platform based on IoT technologies has been designed. It integrates seamlessly with production processes to offer a control and data analysis system for industrial operations. The platform efficiently addresses the issue of fragmented tools in the market and the high costs of implementing ad-hoc solutions, providing a global and scalable solution.

The designed platform covers the entire journey from data extraction to prediction and visualization in industrial environments. The IIoT platform architecture connects to devices via MQTT and OPCUA communication protocols, processing, storing, and predicting data behavior. A machine learning prediction service based on the random forest model is implemented, and the results are visualized on the platform.

One of the platform's most notable aspects is the use of Python as the sole programming language for developing a complete application, along with its modular approach, which allows for high scalability. This translates into the ability to incorporate new types of communication with devices and implement different prediction models according to the specific needs of each company. This modular design facilitates the data journey from extraction, processing, and storage to analysis and visualization, ensuring that companies can adapt the platform to their changing needs and remain competitive in the market.

2.1. KEY WORDS

Industry 4.0, Industrial IoT (IIoT), industrial automation, data analysis, machine learning.

3. INTRODUCCIÓN

3.1. JUSTIFICACIÓN DEL TRABAJO

En el contexto industrial, muchas empresas líderes en sus respectivos sectores han descuidado la implementación de estrategias de optimización y automatización de procesos, lo que las ha llevado a perder competitividad frente a sus competidores. Este descuido histórico, hasta hace poco o incluso en la actualidad, ha creado una brecha en la eficiencia operativa que es vital abordar para mantener o mejorar su posición en el mercado.

A través de estas estrategias se está poniendo foco en optimizar sus procesos productivos para mejorar la calidad de sus productos finales, reducir costes y tiempos de producción; y alargar la vida útil de la maquinaria.

Para ello, los datos son la principal herramienta de trabajo y es necesario disponer de ellos para llevar a cabo estas tareas de optimización de parámetros configurables, consumos eléctricos o cualquier variable directa o indirectamente relacionada con la producción.

Las herramientas que existen en el mercado para abordar esta problemática son muy difusas o requieren de despliegues ad-hoc debido a que no se ha desarrollado una solución global que cubra todas estas necesidades por lo que las empresas tienen que gastarse grandes inversiones en desarrollar sus propios sistemas o soluciones.

En este proyecto se propone contemplar el análisis y diseño de una plataforma de industria 4.0 basada en tecnologías IoT y paralela a producción que permita a estas empresas disponer de un sistema de control y análisis de datos industriales de sus procesos productivos para obtener estas mejoras y poder ser competitivas frente a su competencia.

3.2. OBJETIVOS

El objetivo fundamental del proyecto consiste en diseñar una plataforma de aplicación industrial que permita el control, monitorización y análisis de los datos manejados en cualquiera de los niveles de la pirámide CIM. Este sistema estará orientado a mejorar la eficiencia y efectividad de los procesos industriales mediante la integración de tecnologías avanzadas de Internet of Things (IoT) y técnicas de análisis de datos.

De este objetivo fundamental se derivan otros objetivos secundarios, pero igualmente necesarios para la consecución del objetivo principal.

- Desarrollar un proyecto completo mediante programación en Python
- Desarrollo scripts y aplicaciones en Python que faciliten la extracción, procesamiento y análisis de datos industriales.
-
- Diseñar plataformas web
- Creación de interfaces web que permitan la visualización y control de los datos en tiempo real, e interacción con el sistema por parte de los usuarios.
-
- Manejar protocolos de extracción de datos industriales
- Implementación y gestión protocolos estandarizados para la recolección de datos desde distintos dispositivos y sensores industriales, asegurando la interoperabilidad y precisión.
-
- Crear y manejar de bases de datos
- Diseño y gestión de bases de datos que almacenen los datos industriales de manera eficiente, asegurando la integridad y accesibilidad de la información.
-
- Diseñar de una arquitectura IoT para el proyecto
- Diseño de una arquitectura IoT que integre sensores, actuadores y nodos de comunicación para la recopilación y transmisión de datos en tiempo real.
-
- Predecir datos mediante machine learning
- Implementación de modelos de ml que permitan realizar predicciones sobre los datos industriales, mejorando la capacidad de anticipación y toma de decisiones.

3.3. ALCANCE

Con el objetivo de crear una plataforma funcional y escalable que permita su ampliación futura mediante la adición de nuevas funcionalidades o módulos, se han realizado las etapas de analizar, diseñar e implementar y documentar el proyecto.

A continuación, se exponen los alcances del proyecto. Como la plataforma incluye diferentes áreas de trabajo, se ha tenido que limitar la profundidad de cada uno de los apartados en cierta medida para poder abordar todas ellas.

Arquitectura y despliegue

- Desarrollo de una plataforma exclusivamente web.
- Implementación de una arquitectura basada en contenedores Docker para facilitar el despliegue y la escalabilidad.
- Diseño para su instalación en entornos locales o privados, con posibilidad de despliegue en máquinas virtuales para mayor flexibilidad.

Conectividad y protocolos

- Integración de los protocolos de comunicación industrial OPCUA y MQTT para la extracción de datos.
- Diseño de una arquitectura IoT que permita la conexión con PLC, dispositivos y máquinas industriales para la recolección de datos.

Funcionalidades principales

- Desarrollo de la lógica completa en Python para la extracción, procesamiento y análisis de datos industriales.
- Creación de una interfaz web para la visualización y control de datos en tiempo real.
- Diseño e implementación de un sistema de gestión de bases de datos para el almacenamiento eficiente de la información industrial.
- Desarrollo de modelos de predicción basados en regresión lineal para el análisis predictivo de datos industriales.

Seguridad y acceso

- Implementación de un sistema de autenticación con usuarios y login para garantizar el acceso seguro a la plataforma.

- Diseño de la plataforma para su uso en entornos seguros y aislados, minimizando los riesgos de seguridad.

Escalabilidad y rendimiento

- Consideración de requisitos funcionales y no funcionales específicos para garantizar el rendimiento en entornos industriales.

- Diseño de una arquitectura software que permita la escalabilidad futura, incluyendo la posibilidad de integrar más protocolos de comunicación en el futuro.

Limitaciones y consideraciones

- El proyecto se limita a la implementación de los protocolos OPCUA y MQTT, dejando abierta la posibilidad de expansión futura.

- No se incluye una fase de prueba o implementación piloto en un entorno industrial real, pero se realizarán pruebas simuladas utilizando una Raspberry Pi 5 para simular dispositivos conectados.

3.4. FASES DE TRABAJO

Análisis de requisitos

La web responsiva, diseñada en Python con Reflex y usando PyCharm, se enfoca en la extracción, procesamiento, análisis y predicción de datos industriales conectando sensores y PLC. Implementa autenticación de usuarios con bcrypt y passlib, y se instala como servicio en redes locales empresariales. Sus requisitos más importantes incluyen rendimiento y capacidad para manejar grandes volúmenes de datos en tiempo real, escalabilidad, seguridad y una interfaz de usuario intuitiva y responsiva.

Diseño del sistema y arquitecturas

El diseño del sistema se organiza en una arquitectura de múltiples niveles, cada uno con una función específica: adquisición, procesamiento, almacenamiento, visualización y predicción de datos.

- Para la adquisición de datos se utilizan los protocolos de comunicación MQTT y OPC UA.
- El procesamiento y análisis de datos se realizan mediante librerías de Python como Pandas, Numpy y Scikit-learn.
- En la programación del almacenamiento se emplea la librería sqlmodel para crear un modelo de datos sincronizado con el código. generales de conexión a bases de datos.
- La interfaz se utiliza componentes de visualización de React.js a través de librerías extendidas y punteras como Tailwind y Chakra.

Esta estructura modular permite una integración fluida entre los niveles, asegurando eficiencia, seguridad y escalabilidad. El enfoque modular facilita el desarrollo, la implementación y la futura expansión del sistema.

Desarrollo de la solución

El desarrollo se enfoca en la implementación práctica de la propuesta, abarcando la construcción de componentes y sistemas. Incluye actividades como programación de aplicaciones, configuración de infraestructuras, integración de sistemas y pruebas.

Documentación del proyecto

Esta última fase proporciona una visión detallada de todas las etapas y aspectos específicos del proyecto. Se incluyen descripciones detalladas de las metodologías empleadas y las técnicas utilizadas en cada fase de desarrollo particular. Además, ofrece una explicación completa del proceso de desarrollo que abarca desde la concepción inicial hasta la implementación final del sistema que será presentado ante el tribunal.

4. ESTADO DEL ARTE

En este apartado se pretende realizar una revisión y análisis de los desarrollos más recientes sobre plataformas industriales, así como marcar y explicar los fundamentos teóricos principales que ayuden a comprender los aspectos técnicos más relevantes del proyecto. Dado que el proyecto abarca diversas áreas temáticas, cada una de ellas será explorada y explicada de manera independiente para facilitar una comprensión integral y detallada.

4.1. ANTECEDENTES

Trabajando en el departamento de “IoT e Industria 4.0” en “Integra Tecnología y Estrategia”, se observa en repetidas ocasiones que las empresas enfrentan dificultades para integrar datos, desde aquellos extraídos a nivel de campo como máquinas de producción hasta los niveles de gestión de la empresa como sistemas ERP (Enterprise Resource Planning).

Actualmente, en el mercado existen diversas plataformas diseñadas para abordar necesidades similares, conectándose a diversas fuentes de datos empresariales. Sin embargo, estas herramientas suelen presentar limitaciones y a menudo se centran en niveles específicos de la pirámide CIM (Computer Integrated Manufacturing).

Aunque estas plataformas están destinadas a facilitar la aplicación de Big Data en la automatización de procesos, optimización de la producción y gestión eficiente, la mayoría de ellas son genéricas y poco adaptables. Esto dificulta que cada empresa pueda personalizar su plataforma para satisfacer sus necesidades específicas por lo que terminan generando desarrollos integrales para cubrir esta necesidad.

Las plataformas más empleadas en este ámbito son Predix de General Electrics y Mindsphere de Siemens.

4.1.1. Predix de General Electric

Plataforma de software de General Electric reconocida por Gartner como líder mundial en “2023 Magic Quadrant for Manufacturing Execution Systems”. Permite a las empresas industriales construir, desplegar y operar aplicaciones y soluciones de IoT industrial. La plataforma se enfoca en la gestión de grandes volúmenes de datos generados por sensores y dispositivos industriales, ofreciendo capacidades avanzadas de análisis y optimización.

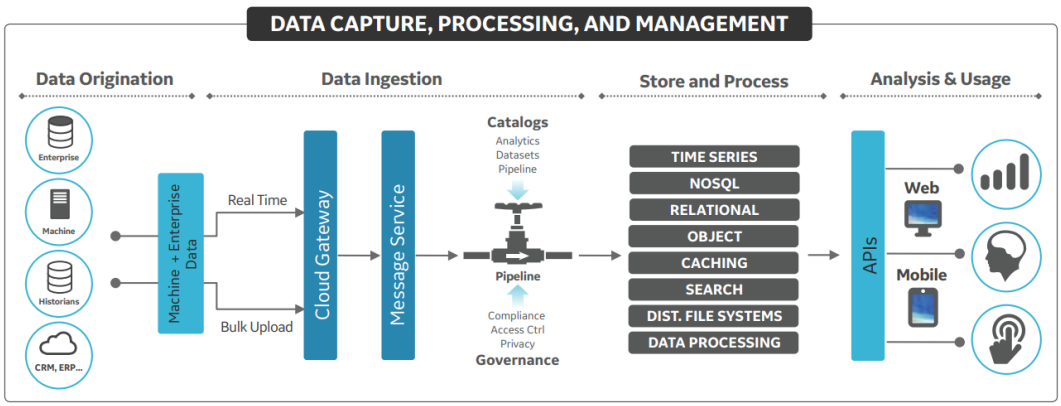

Ilustración 1 Diagrama de funcionamiento de Predix GE [1]

Características principales

Conectividad	Desde dispositivos de campo hasta sistemas MES. Soporta protocolos industriales estándar como OPC UA, Modbus y MQTT.
Ingestión de datos	Utiliza un Data Pipeline que permite la recolección y procesamiento en tiempo real. Además, cuenta con servicios de ingesta de datos que facilitan la captura desde múltiples fuentes.
Almacenamiento y gestión de datos	Utiliza una base de datos de series temporales, que maneja eficientemente datos históricos y en tiempo real. También incluye un Data Lake para almacenar grandes volúmenes de datos estructurados y no estructurados.
Análisis de datos	Capacidad para realizar análisis descriptivos, predictivos y prescriptivos. Incorpora tecnologías de Machine Learning para la detección de patrones y predicción de fallos. Además, utiliza Digital Twins para simular el comportamiento de activos en tiempo real.
Seguridad	Implementa mecanismos robustos de autenticación y autorización. Los datos se encriptan tanto en tránsito como en reposo. La plataforma cumple con estándares de seguridad como ISO 27001.
Ecosistema	Plataforma extensible a través de APIs y SDKs para integración y desarrollo. Incluye un Marketplace con aplicaciones y servicios que pueden integrarse fácilmente para ampliar su funcionalidad. [2], [3]

4.1.2. Mindsphere de Siemens

Plataforma industrial basada en la nube desarrollada por Siemens, diseñada para el Internet de las Cosas Industriales (IIoT). MindSphere es una plataforma integral IIoT, diseñada para conectar, gestionar y optimizar activos industriales a través de una infraestructura basada en la nube.

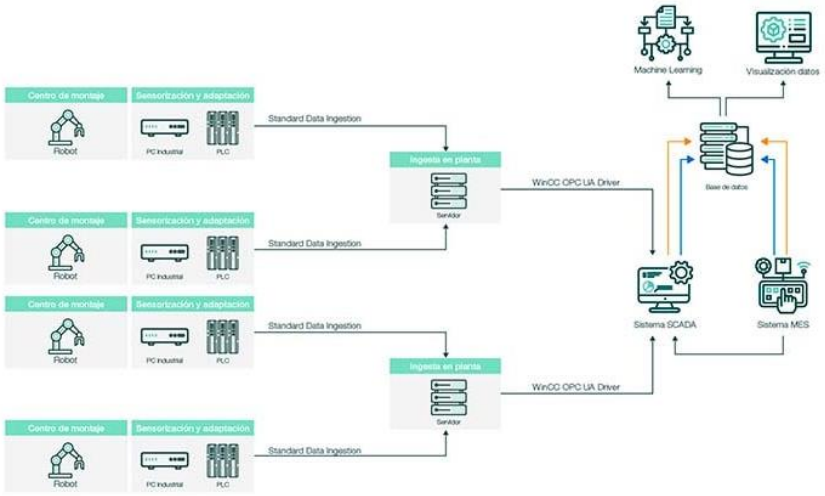

Ilustración 2 Diagrama funcionamiento de Mindsphere Siemens [4]

Características principales

Conectividad	Conecta dispositivos industriales a la nube. Usa MindConnect para protocolos OPC UA y MQTT.
Ingestión de datos	Ofrece ingesta de datos en tiempo real. Usa pipelines optimizados para seguridad e integridad.
Almacenamiento y gestión de datos	Almacena datos en bases de series temporales. Incluye Data Lake para datos estructurados y no estructurados.
Análisis de datos	Proporciona análisis avanzado con machine learning e IA. Permite crear Digital Twins para simular activos físicos.
Seguridad	Implementa seguridad robusta. Cumple con ISO 27001. Encripta datos en tránsito y reposo.
Ecosistema	Plataforma y adicionalmente soporta desarrollo de aplicaciones con arquitectura de microservicios. Ofrece IDE, APIs y SDKs. MindSphere Store provee aplicaciones y servicios preconstruidos. Facilita integración y extensión mediante APIs.

4.2. MARCO TEÓRICO

4.2.1. Industria 4.0

Este término trata de describir la digitalización y procesos industriales mediante la interconexión de IoT en el sector industrial como gran revolución del siglo XXI para modernizar y optimizar los procesos productivos.

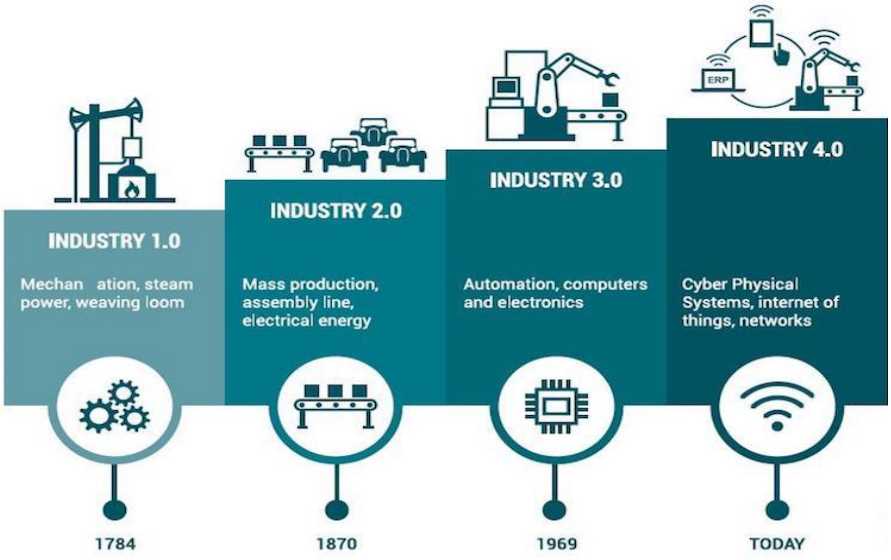

Ilustración 3 Desarrollo tecnológico durante las revoluciones industriales [5]

Es una visión de fábrica inteligente con la integración de nuevas tecnologías disruptivas como IoT, Big Data, Cloud, ciberseguridad o inteligencia artificial entre otras que han surgido durante las últimas décadas.

La conectividad en un mundo globalizado y el análisis de grandes volúmenes de datos en tiempo real permiten moldear nuevos modelos de producción y sistemas de fabricación. En el mundo existen actualmente más de 50.000 millones de dispositivos conectados que generan datos constantemente y son almacenados por los usuarios a través de máquinas, ordenadores, wearables, smartphones, dispositivos, etc.

4.2.2. Internet of Things (IoT)

Según la Unión Internacional de Telecomunicaciones (ITU), que es el organismo encargado de la estandarización y normativas de las telecomunicaciones móviles a nivel mundial, se puede definir IoT como:

“Infraestructura mundial para la sociedad de la información que propicia la prestación de servicios avanzados mediante la interconexión de objetos (físicos y virtuales) gracias a la interoperatividad de tecnologías de la información y la comunicación presentes y futuras.

Aprovechando las capacidades de identificación, adquisición de datos, procesamiento y comunicación, IoT utiliza plenamente las "cosas" para ofrecer servicios a todos los tipos de aplicaciones, garantizando a su vez el cumplimiento de los requisitos de seguridad y privacidad”. [6]

Como consecuencia de lo anterior, se puede decir que es el resultado de las diferentes tendencias tecnológicas que convergen y trabajan juntas para tender puentes entre el mundo físico y mundo virtual.

En el ámbito del IoT, los objetos pueden ser elementos del mundo físico o del mundo virtual que se identifican y se integran en redes de comunicación. Estos objetos poseen información relacionada, que puede ser tanto estática como dinámica.

Los objetos físicos son aquellos que existen en el mundo real y pueden ser detectados, manipulados y conectados a redes. Ejemplos de estos objetos incluyen el entorno que nos rodea, robots industriales, bienes materiales y equipos eléctricos.

Por otro lado, los objetos virtuales pertenecen al mundo de la información y pueden ser almacenados, procesados y accesibles digitalmente. Ejemplos de estos objetos son el contenido multimedia y el software de aplicaciones.

Las tecnologías más utilizadas son: sensores y actuadores, tecnologías de comunicación de corto alcance como LoRaWAN, Bluetooth, Wifi o de largo alcance como redes móviles como NB-IoT, 4G-LTE y EC-GSM; protocolos de mensajería ligeros como MQTT, CoAP o AMQP; y métodos de visualización y analítica de datos como gemelos digitales, inteligencia artificial, sistemas de almacenamiento en la nube y Big Data.

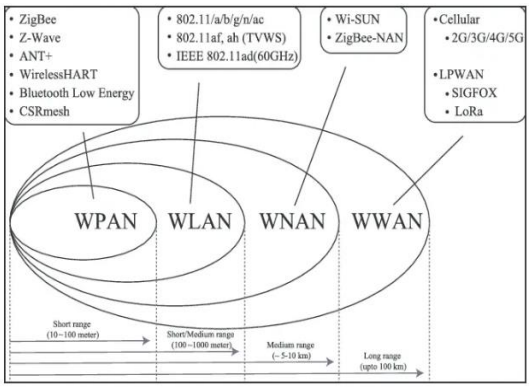

Ilustración 4 Tipos de tecnologías inalámbricas [7]

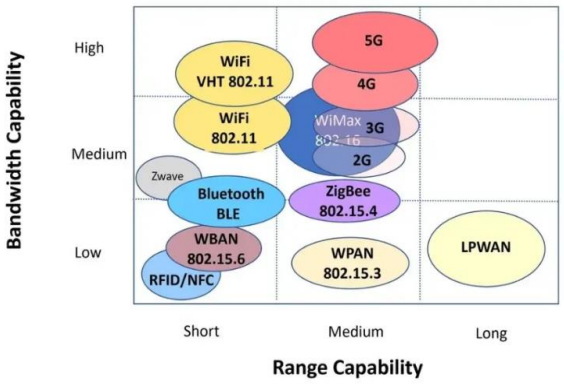

Ilustración 5 Comparativa de las tecnologías inalámbricas según su rango y su ancho de banda [7]

Actualmente, IoT está mejorando la eficiencia operativa, reducción de costes y creación de nuevas oportunidades de negocio. Algunas tendencias y avances recientes en el IoT incluyen:

- Interoperabilidad y Estándares: Adopción de estándares ha facilitado la integración de diferentes dispositivos y sistemas, permitiendo una comunicación más eficiente y segura.

- Seguridad y Privacidad: Nuevos enfoques y tecnologías para proteger los datos y asegurar la privacidad de los usuarios.

- Inteligencia Artificial y Machine Learning: Permite el análisis avanzado de datos y la toma de decisiones automatizada, mejorando la capacidad de respuesta y predicción en diversas aplicaciones.

- Conectividad Mejorada: Ampliación de las posibilidades de conectividad, permitiendo la implementación de IoT en áreas remotas y en dispositivos con baja demanda energética. [8]

El IoT tiene aplicaciones diversas y un impacto significativo en todos los sectores empresariales. En la manufactura, automatiza y monitoriza procesos en tiempo real, mejorando la eficiencia. En la logística, permite el seguimiento preciso de envíos y la optimización de rutas. En la atención médica, facilita el monitoreo remoto de pacientes y la gestión avanzada de dispositivos médicos, mejorando la calidad del cuidado. En la agricultura, los sensores IoT gestionan eficientemente recursos como el agua y los fertilizantes, aumentando la productividad y reduciendo el impacto ambiental. Este amplio alcance demuestra cómo el IoT adapta soluciones específicas para modernizar y aumentar la competitividad en diversas industrias.

4.2.3. Industrial Internet of Things (IIoT)

En 2012 la multinacional General Electric utilizó el término Internet Industrial para referirse a una red abierta y global que conecta máquinas, personas y datos, con el objetivo de afrontar la nueva revolución industrial que posteriormente se denominó Industria 4.0.

A finales del siglo XX, en la industria, la herramienta principal de procesos industriales fueron los SCADA, que se han visto obligado a evolucionar integrando y adaptando el IIoT gradualmente para potenciar el funcionamiento de la creación de fábricas inteligentes.

Hoy en día con el IIoT las empresas pueden recolectar los datos y transformarlos, a partir de la analítica de datos, en información y conocimiento para la toma de decisiones que les permita resolver desafíos como la reducción de costes de fabricación, selección de materias primas más sostenibles y una respuesta ágil a los requerimientos de los mercados, todo ello conectado a máquinas, personas y objetos.

Por tanto, IIoT es la convergencia de las Tecnologías Operativas (OT) y las Tecnologías de la Información (IT) en las organizaciones industriales. Se podría ver como una extensión del IoT en entornos industriales que incluye la conexión de sensores y actuadores de máquinas industriales a procesos locales y a Internet, y la futura conexión a otras redes industriales que puedan generar valores de manera independiente.

Al igual que el IoT, su ámbito de aplicación es muy amplio y transparente al sector o a la actividad como fabricación, logística, transporte, energía, etc. Se centra en la eficiencia de los procesos productivos, de gestión y de mantenimiento en todos los aspectos de procesos industriales. [9]

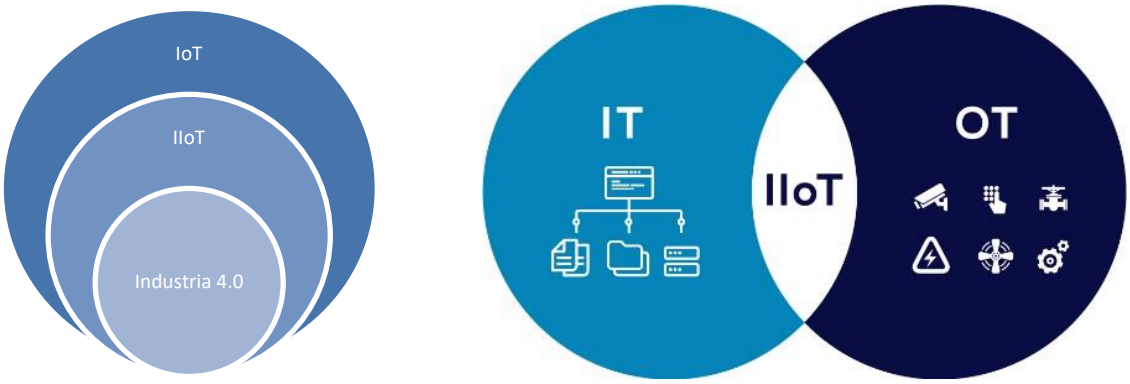

Ilustración 6 Relaciones del IIoT con la Industria 4.0, IoT, IT y OT [10]

4.2.3.1. Protocolos de comunicación IIoT

Los estándares más extendidos en la Industria 4.0 actualmente son MQTT y OPC UA. Ambos protocolos de comunicación se han consolidado para dar forma a sistemas robustos y fiables para la extracción de datos de PLC y dispositivos industriales. [11]

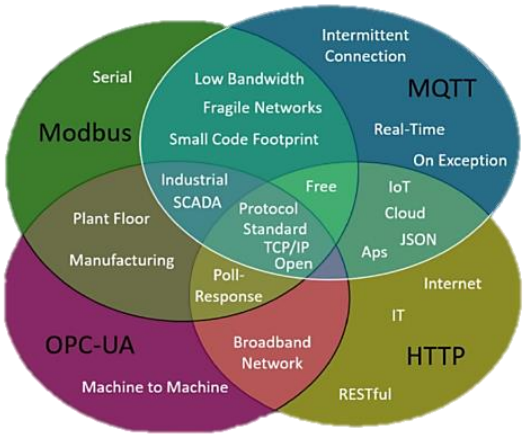

Ilustración 7 Comparativa de eficiencia de los protocolos IIoT [12]

4.2.3.1.1. MQTT (Message Queue Telemetry Transport)

Protocolo de mensajería M2M estándar para IoT. Está diseñado como un transporte de mensajería de publicación/suscripción extremadamente ligero ideal para conectar dispositivos remotos con una huella de código pequeña y un ancho de banda de red mínimo. [13]

Inicialmente diseñado por IBM en 1999, fue estandarizado por OASIS (Organization for the Advancement of Structured Information Standards) en 2013, que es una organización sin ánimo de lucro que se compone de un comité técnico con representantes de las principales empresas líderes en tecnología de la información, tales como Cisco, IBM y Microsoft. Este comité es responsable del desarrollo y la actualización continua del protocolo MQTT, asegurando que cumpla con los requisitos de eficiencia, seguridad y fiabilidad necesarios para aplicaciones en el ámbito del IoT y otros campos relacionados con la comunicación de datos en entornos con recursos limitados. [14]

Se basa en una arquitectura publicador-suscriptor sobre TCP/IP, donde el servidor se denomina “broker” y se emplea como servidor central encargado de recibir, filtrar y distribuir los mensajes. Utiliza “topics” como canales de comunicación organizados jerárquicamente y “payloads” como canal de datos con objetos en formato Json.

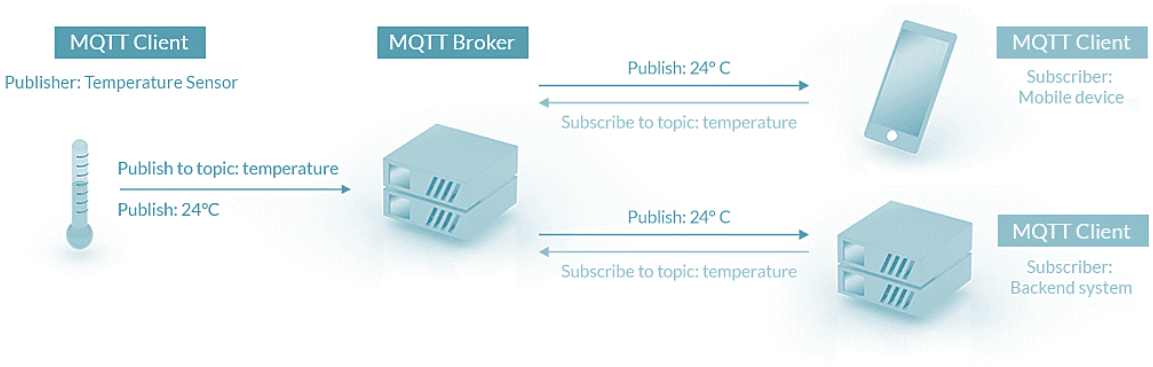

Ilustración 8 Arquitectura de publicación y suscripción del protocolo MQTT [15]

Formato de mensaje

Los mensajes en MQTT tienen un encabezado fijo, otro variable y un payload.

- Header fijo contiene el tipo de mensaje.
- Header variable contiene información específica del mensaje.
- Payload contiene datos del mensaje. [16]

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Meaning	Header	Remaining Length	Length of prtocol name		Protocol Name +Version					Connect Flags	Keep Alive		Length								
Hex	0x10	0x13	0x0	0x4	0x4d	0x51	0x54	0x54	0x4	0x2	0x0	0x3c	0x0	0x7	0x70	0x79	0x74	0x68	0x6F	6E	0x31
Ascii		19		4	M	Q	T	T	4			60		7	P	Y	T	H	O	N	1

Ilustración 9 Dataframe de un mensaje MQTT [17]

Tipos de mensajes

Tabla 1 Tipos de mensajes empleados en el protocolo MQTT [16]

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)

PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

La configuración de conexión para cualquier cliente MQTT debe tener en cuenta los parámetros. Se van a clasificar según:

Conexión

ClientID	Identificador único del cliente.
Keep Alive Interval	Tiempo máximo entre dos comunicaciones con el broker para mantener la conexión activa.
Clean Session	Flag que indica si se debe recuperar el estado anterior al reconectar la sesión.
Will message	Mensaje a enviar en caso de desconexión inesperada (opcional)

El bróker responde al cliente con un mensaje CONNACK para confirmar la conexión y la desconexión puede ser limpia (DISCONNECT) o inesperada. En el segundo caso, el bróker maneja el “Will Message” si está configurado.

Seguridad

Autenticación	Uso de usuarios y contraseñas en el mensaje CONNECT.
Autorización	Control de accesos que maneja el broker para permitir/denegar la publicación/suscripción basado en Topics.

TLS/SSL	Cifrado de la comunicación entre clientes y broker.

Retención de mensajes

RetainFlag	Almacenamiento del último mensaje por el broker.

Calidad de servicio

QoS 0	Entrega del mensaje no garantizada.
QoS 1	Entrega del mensaje al menos una vez. Requiere un mensaje de confirmación de recepción PUBACK. Puede causar duplicados.
QoS 2	Entrega del mensaje exactamente una vez utilizando un proceso de handshake de cuatro pasos (dos pares de mensajes PUBREC, PUBREL y PUBCOMP). Es el más seguro y complejo.

4.2.3.1.2. Open Protocol Communication Unified Architecture

Open Protocol Communication (OPC) es un estándar de comunicación M2M diseñado por Microsoft con el objetivo de obtener un protocolo de comunicación abierto e independiente para el ámbito industrial. La tecnología OPC se fundamenta en básicamente dos tecnologías. [18]

En primer lugar, una arquitectura cliente–servidor. Un servidor que se conecte a la fuente de datos y los exponga, y un cliente que haga las consultas pertinentes para hacer uso de los datos. Por ejemplo, servidores conectados a PLC, que comuniquen los datos a un cliente como un SCADA. En segundo lugar, una plataforma de comunicaciones basada en tecnología Windows mediante la tecnología COM/DCOM para llevar los paquetes de comunicación entre aplicaciones.

OPC tiene limitaciones que lo incapacitan para su uso en tecnologías IoT. Incompatibilidad con cualquier sistema operativo distinto de Windows y uso de servidores adicionales mediante drivers.

OPC UA es un protocolo de comunicación industrial desarrollado por la OPC Foundation, diseñado para garantizar la interoperabilidad entre dispositivos y sistemas en entornos industriales. Este protocolo es una evolución del clásico OPC, construido para satisfacer las demandas de la industria moderna, proporcionando una plataforma unificada para el intercambio de información.

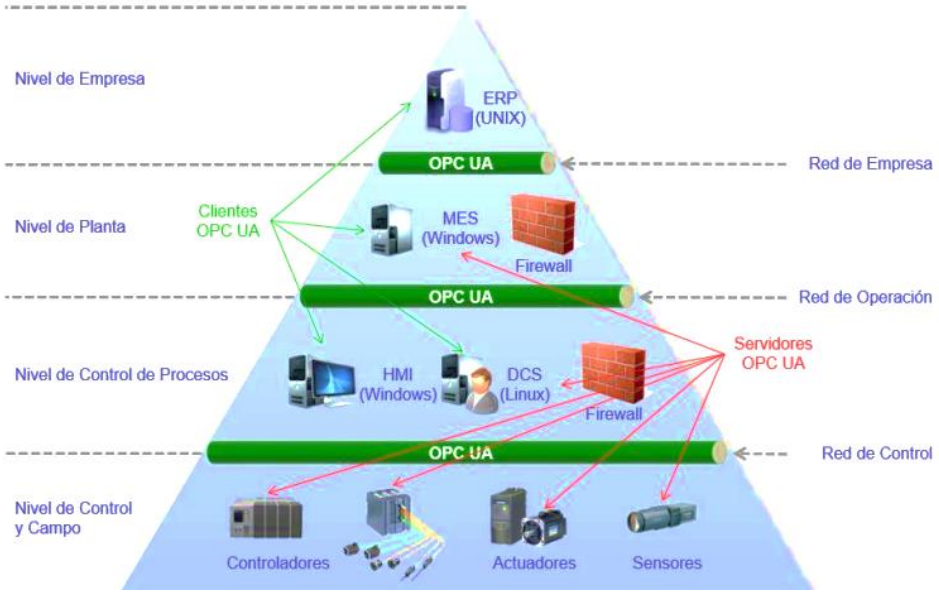

Ilustración 10 OPCUA es la pirámide CIM [19]

Las principales características de OPC UA incluyen:

- Interoperabilidad: Facilita la comunicación entre dispositivos y sistemas de diferentes fabricantes, eliminando las barreras de compatibilidad.
- Multiplataforma: Funciona en diversas plataformas, incluyendo Windows, Linux, y sistemas embebidos, lo que asegura su flexibilidad y adaptabilidad.
- Escalabilidad: Puede ser implementado en diferentes niveles de la pirámide CIM (Computer Integrated Manufacturing), desde dispositivos de campo hasta sistemas de gestión empresarial.
- Seguridad: Integra avanzadas características de seguridad como encriptación y autenticación, para proteger la integridad y confidencialidad de los datos.
- Modelado de Información: Soporta la modelización de datos compleja, permitiendo la representación estructurada y jerárquica de la información.

A bajo nivel, OPC UA opera mediante una arquitectura cliente-servidor y puede también emplear un modelo pub/sub (publicador/suscriptor) para ciertas aplicaciones. Aquí se desglosan los aspectos técnicos clave de su funcionamiento:

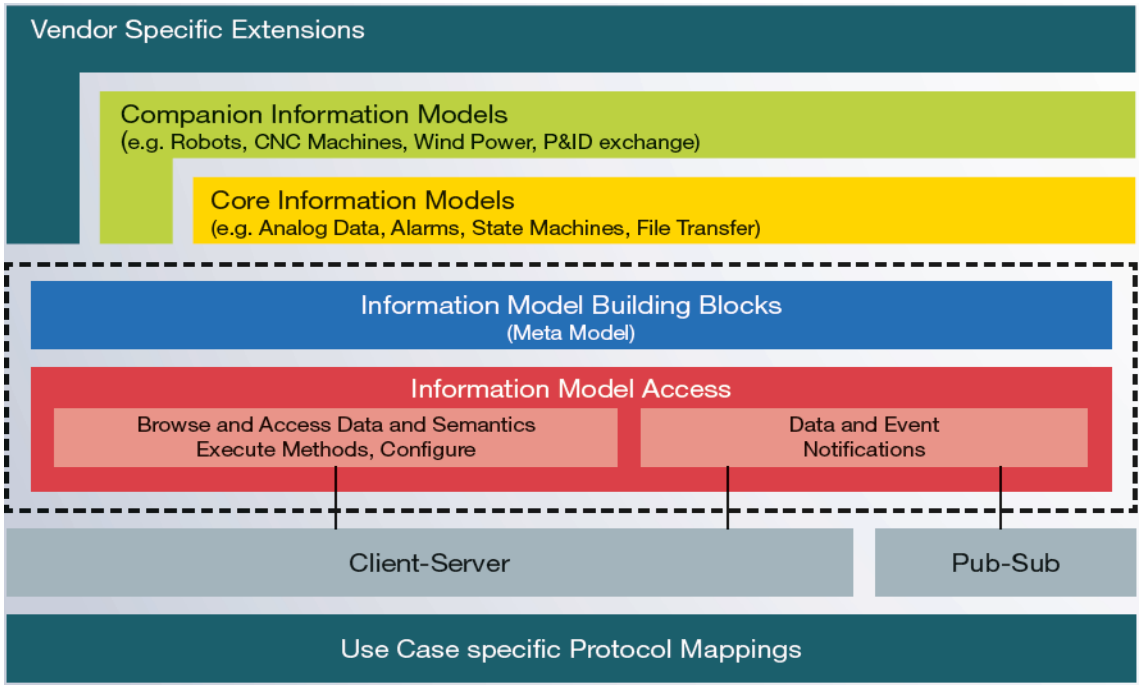

Ilustración 11 Diagrama de arquitectura OPCUA

Arquitectura Cliente-Servidor

- Cliente OPC UA: Solicita servicios y datos al servidor. Los clientes pueden ser aplicaciones SCADA, HMI, sistemas MES o cualquier otra aplicación que necesite interactuar con datos industriales.

- Servidor OPC UA: Proporciona servicios y datos a los clientes. Los servidores son típicamente dispositivos de campo como PLCs, sensores, actuadores o cualquier otro dispositivo industrial capaz de comunicarse mediante OPC UA.

Modelo de Información

- Nodos y Referencias: La información se organiza en una estructura jerárquica de nodos. Cada nodo puede representar un dispositivo, un dato o una variable, y las referencias definen la relación entre los nodos.

- Tipos de Nodos: Incluyen objetos, variables, métodos, datos y referencias, permitiendo una representación detallada y estructurada de cualquier sistema industrial.

Transporte y Codificación

- Protocolos de Transporte: Soporta múltiples protocolos de transporte, como TCP/IP para conexiones seguras y eficientes, y HTTP/HTTPS para aplicaciones web.

- Codificación de Datos: Utiliza binarios y XML para la codificación de datos, optimizando la transmisión de información según las necesidades de la aplicación.

Seguridad

- Autenticación y Autorización: Implementa mecanismos robustos de autenticación (certificados X.509, credenciales de usuario) y autorización para controlar el acceso a los datos.

- Encriptación: Los datos transmitidos son cifrados para asegurar la privacidad y la integridad durante la comunicación.

Modelo Pub/Sub

- Publicadores: Envían mensajes de datos a través de un medio de transporte común sin necesidad de que los suscriptores conozcan la identidad del publicador.

- Suscriptores: Reciben los mensajes de datos enviados por los publicadores. Este modelo es eficiente para aplicaciones que requieren distribución de datos a múltiples destinatarios.

4.2.4. Pirámide CIM

Sistema de organización y clasificación jerárquica que divide los componentes de una empresa industrial por niveles conectados para aumentar la eficiencia mejorando la producción a través de operaciones automatizadas y una correcta gestión de procesos.

El principal objetivo del CIM es englobar todos los procesos de una planta de fabricación bajo el control informático para integrarlos en la automatización completa de una empresa y poder segmentar las operaciones según la necesidad específica de cada departamento [20].

La pirámide se puede dividir en las siguientes categorías:

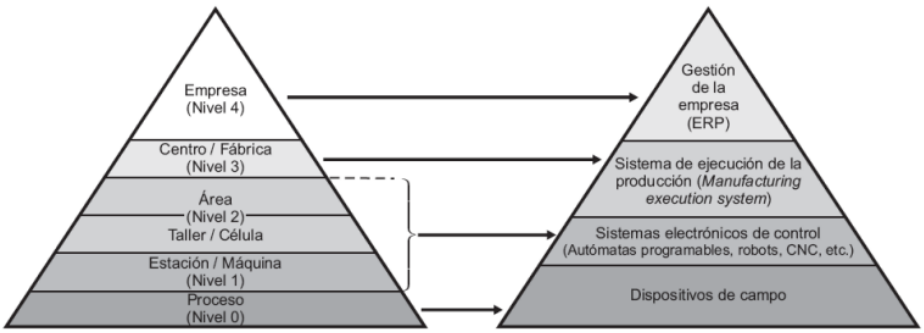

Ilustración 12 Pirámide CIM: Niveles de Integración y Automatización en la empresa [21]

0 - Nivel de proceso

Agrupa todos los dispositivos que interactúan directamente con el proceso, es decir, los dispositivos de campo. Aquí se incluyen principalmente los sensores y actuadores, que son elementos esenciales para la automatización del proceso [22].

1 - Nivel de control

Agrupa los diversos controladores que supervisan los componentes descritos en el nivel de proceso. Este nivel incluye sistemas de control distribuido, PLC (controladores lógicos programables) y ordenadores o máquinas virtuales, entre otros. Los controladores en este nivel son responsables de recopilar y analizar los datos de los dispositivos del nivel de proceso y tienen la capacidad de programar sus acciones.

2 - Nivel de supervisión

Agrupa los sistemas SCADA y los componentes HMI permitiendo el control y la supervisión remota de los procesos. Esta sección se encarga de recopilar diversos datos y supervisar los elementos de los dos niveles anteriores.

3 - Nivel de planificación o de detección

Agrupa los softwares MESH (Manufacturing Execution System) que supervisan y coordinan las actividades de producción, gestionan la ejecución de órdenes de trabajo, controlan el flujo de materiales y recopilan datos de producción para análisis y mejora continua.

4 - Nivel de gestión

Agrupa diferentes servidores y ordenadores integrados en sistemas como el ERP (Enterprise Resource Planning) un software que permite un control total sobre todas las acciones del proceso, desde el punto menos significativo al más significativo. Este nivel se encarga de supervisar y analizar cada proceso dentro del ámbito empresarial para garantizar su éxito. El software utilizado tiene diversas funcionalidades para monitorear el proceso como compras, ventas, producción, beneficios y pérdidas.

Cada nivel maneja un volumen de información mayor al anterior siendo este último el de mayor volumen de datos, por lo que el objetivo de las empresas es integrar todo dentro de su ERP.

4.2.5. Arquitectura de sistemas IoT

No existe un estándar o modelo definido para diseñar sistemas, aunque las principales organizaciones y empresas mundiales dedicadas a ello han propuesto sus respectivas arquitecturas. Los modelos más adoptados y extendidos son de tres y cincos capas. Ambos modelos están basados en los modelos de internet OSI y TPC/IP. [9]

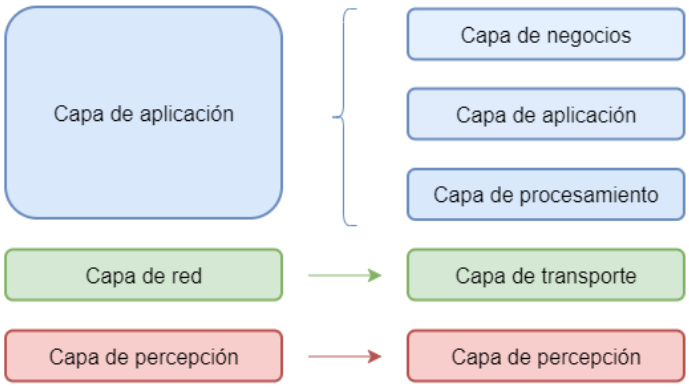

Ilustración 13 Comparación de los modelos de arquitectura IoT más extendidas

La arquitectura de 5 capas como se pude observar en su última capa está enfocada adicionalmente en los servicios y la rentabilidad del negocio por lo que se va a explicar detenidamente la arquitectura de 3 capas:

Capa de percepción

Capa física que integra los dispositivos para recopilar la información de diferentes fuentes de datos del entorno de forma eficiente y fiable. Por otro lado, incluye los dispositivos de actuación para aquellos sistemas que requieran comunicación bidireccional. Ambos tipos de dispositivos se pueden integrar en cualquier nivel dentro de la pirámide CIM. [9]

Capa de red

Capa de comunicaciones responsable de la extracción de datos de sistemas IoT que actúa como puente entre las capas de percepción y de aplicación.

Las comunicaciones IIoT han ganado relevancia en los últimos años permitiendo que las tecnologías inalámbricas faciliten la conectividad de dispositivos y sistemas a través de redes de baja latencia y alto rendimiento. Permiten sistemas de mayor escalabilidad y flexibilidad en la recopilación y transmisión de datos con los que se mejora la eficiencia operativa y se reducen costes. La capacidad de integrar múltiples dispositivos y sistemas de manera sencilla y eficiente convierte a las comunicaciones IIoT en una opción óptima para los sistemas que busquen la modernización y optimización de entornos industriales

Capa de aplicación

Capa de servicios enfocada en generar para el usuario un entorno virtual o aplicación web que permita la explotación y monitorización del sistema IoT. Incluye los procesos de transformación, almacenamiento, análisis y visualización del dato.

Suele integrar un middleware para realizar las transformaciones necesarias de los datos recibidos de los dispositivos físicos y una o varias aplicaciones de servicios interconectadas similares a las de internet.

4.2.6. Arquitectura de soluciones de software

Se van a estudiar los conceptos básicos de los principales patrones de diseño de arquitecturas de software empleadas en desarrollo de plataformas web para poder seleccionar durante el desarrollo del proyecto la opción ideal.

4.2.6.1. Descripción general

La arquitectura de aplicaciones web se basa en un modelo cliente-servidor, en el cual un servidor web proporciona páginas de información a los clientes (navegadores) conectados a través de una red. Esta arquitectura incluye tres componentes principales: el servidor web, la conexión de red y uno o más clientes.

4.2.6.2. Elementos de una web

4.2.6.3. Tipos de arquitecturas web

Existe una gran cantidad de patrones distintos que se adaptan a distintas necesidades según el tipo de web, su finalidad y las herramientas de desarrollo por lo que se van a clasificar según:

4.2.6.3.1. Clasificación por modelo de servicio

Arquitectura monolítica

Arquitectura tradicional donde todas las funcionalidades de una aplicación están integradas en una única base de código. Todas las funcionalidades y componentes de una aplicación se despliegan como una sola unidad. Esta estructura incluye típicamente tres capas principales:

- Capa de Presentación (Frontend): Maneja la interacción con el usuario, mostrando interfaces gráficas o páginas web.
- Capa de Lógica de Negocio (Backend): Implementa las reglas de negocio y la lógica de la aplicación.
- Capa de Acceso a Datos: Gestiona la interacción con la base de datos o sistemas de almacenamiento. [23]

Arquitectura de microservicios

La aplicación se divide en servicios pequeños e independientes, cada uno con su propia lógica de negocio y base de datos. Cada servicio se divide en módulos/bases de códigos y corresponde a una característica/tarea específica.

Esta arquitectura facilita el mantenimiento de una aplicación, el desarrollo de características, las pruebas y la implementación en comparación con una arquitectura monolítica. [24]

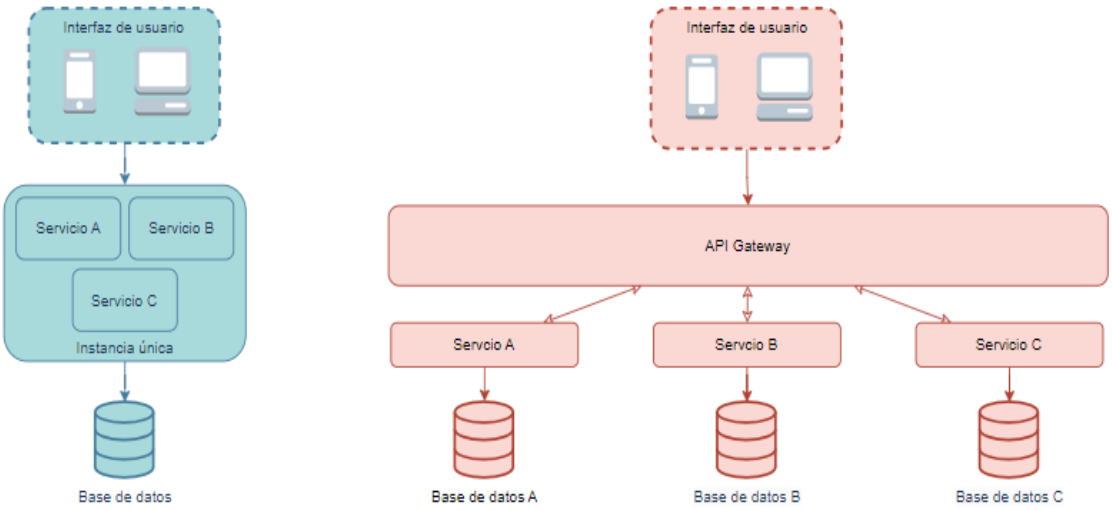

Ilustración 14 Clasificación según el modelo de servicio (monolítica vs microservicios)

4.2.6.3.2. Clasificación por modelo de comunicación

Comunicación síncrona

Modelo de interacción entre componentes en el que un componente emite una solicitud y espera la respuesta antes de continuar con cualquier otra operación. Durante este periodo de espera, el componente solicitante puede quedar bloqueado, impidiendo la ejecución de otras tareas hasta recibir la respuesta.

Los principales ejemplos de este modelo son las llamadas a procedimientos remotos (RPC) o servicios web REST, donde el cliente llama a un procedimiento o solicitud HTTP en el servidor y espera a que termine y le devuelva el resultado.

Comunicación asíncrona

Modelo de interacción entre componentes en el que un componente emite una solicitud y continúa con otras tareas sin esperar una respuesta inmediata. La respuesta, cuando esté disponible, será manejada de forma independiente, permitiendo al componente solicitante seguir operativo y evitando el bloqueo.

Los principales ejemplos de este modelo son la mensajería basada en colas, donde un componente envía un mensaje a una cola y otro componente receptor procesa el mensaje en su propio tiempo, y el modelo de publicación-suscripción, en el cual los clientes se suscriben a canales específicos para recibir mensajes que han sido publicados en cualquier momento anterior.

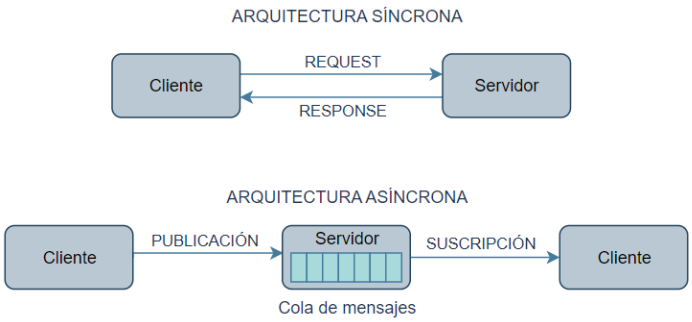

Ilustración 15 Clasificación según el modelo de comunicación (síncrono vs asíncrono)

4.2.7. Bases de datos

Una base de datos es una recopilación de información o datos estructurados organizada que se almacenan electrónicamente en un sistema informático. La complejidad de las bases de datos depende de la estructura y organización de los datos. Un sistema de base de datos se compone la mayoría de veces de la información en sí y un sistema de gestión de base de datos (SGBD), permitiendo al usuario acceder, actualizar, analizar y gestionar fácilmente la información. [25]

Existen varios tipos de bases de datos, los cuales se van a presentar a continuación, cada uno con características y usos específicos.

Las bases de datos SQL se utilizan para administrar datos de forma dinámica y las bases de datos NoSQL son idóneas para aplicaciones que requieran variabilidad entre la información almacenada y velocidad de recuperación.

Relacionales

El lenguaje de consulta estructurada, también conocida como SQL, es un lenguaje de programación que utilizan la mayoría de bases de datos relacionales para consultar, manipular y definir los datos, además de para proporcionar control de acceso.

Se caracterizan por tener unas estructuras de datos planas unidas con operaciones de tipo join y esquemas rígidos. Están enfocadas en las relaciones entre datos, los cuales son almacenados en registros que son organizados en tablas y evitan los datos duplicados. Permite crear todo tipo de datos, relacionándolos entre sí y garantizando la total consistencia de los datos, no dando posibilidad de error. Se puede acceder a los datos con gran rapidez y son de fácil gestión por lo que pueden ser utilizados por cualquier persona. [25], [26]

Las bases de datos relacionales más usadas son MySQL, SQL Server, Oracle, y PostgreSQL.

No relacionales

Recientemente, las bases de datos NoSQL han surgido como respuesta al crecimiento de Internet y la necesidad de acelerar la velocidad y el procesamiento de los datos no estructurados. [25]

Una base de datos NoSQL o base de datos no relacional, permite almacenar y manipular datos no estructurados y semiestructurados.

Se caracterizan por emplear colecciones de elementos en vez de tablas, por lo que éstos pueden ser heterogéneos, pudiendo almacenar registros con juegos de datos distintos entre sí en una colección. Se emplea programación funcional para hacer filtrados y otros tipos de operaciones. [27]

Las bases de datos no relacionales más usadas son MongoDB, CassandraDB, InfluxDB o Redis.

4.2.7.1. PostgreSQL

La base de datos PostgreSQL ofrece varias ventajas significativas frente a otras bases de datos: [28], [29]

1. Es una base de datos relacional de código abierto y gratuito, por lo que no hay pagos asociados ni con su uso ni necesidad de una licencia. Tiene una larga historia de desarrollo, lo que garantiza que sea muy confiable.

Es compatible con una amplia variedad de sistemas operativos y servidores web, lo que lo hace muy versátil y fácil de implementar en diferentes entornos.

2. La configuración es sencilla y puede realizarse de forma rápida y eficiente gracias a herramientas de gestión de bases de datos como DBeaver.

3. Ofrece una gran cantidad de opciones avanzadas y permite la creación de funciones personalizadas utilizando varios lenguajes de programación.

Es muy escalable y puede manejar grandes cantidades de datos y usuarios. Tiene un sistema de gestión de concurrencia muy robusto que garantiza la integridad de los datos y evita conflictos entre transacciones.

Puede ser utilizado como almacén de datos relacionales y NoSQL al mismo tiempo, lo que lo hace muy útil para proyectos que requieren manejar diferentes tipos de datos y estructuras.

4.2.8. Inteligencia artificial

La inteligencia artificial (IA) es la combinación de algoritmos diseñada para inventar máquinas que presentan las mismas capacidades que el ser humano, tanto a nivel intelectual como físico. Esta tecnología ha evolucionado desde su inicio en 1950 y ha sido ampliamente utilizada en diversas industrias, tales como la medicina, la financiación, el transporte, la educación, etc. Se pueden distinguir dos enfoques diferentes para entrenar modelos de IA: [30]

4.2.8.1. Aprendizaje supervisado

Utiliza conjuntos de datos etiquetados para entrenar algoritmos que clasifican o predicen resultados con precisión. Un conjunto de entrenamiento enseña a los modelos a generar resultados correctos, ajustando el algoritmo para minimizar el error mediante la función de pérdida. Este enfoque puede lograr alta precisión si hay suficientes datos etiquetados y es aplicable a tareas como clasificación, regresión o predicción.

Existen varios tipos de aprendizaje como la regresión logística, random forest, naive bayes o la regresión lineal supervisada. Esta última predice un valor continuo (variable objetivo)

basado en características o variables de predicción, asumiendo una relación lineal entre ellas. Cada variable predictora altera la variable objetivo en una cantidad constante. [32]

Sus ventajas incluyen ser fácil de entender e implementar, alta precisión en predicciones continuas y coeficientes de regresión fáciles de interpretar, lo que facilita comprender cómo las variables predictoras afectan el resultado. [33]

4.2.8.2. Aprendizaje no supervisado

Este enfoque utiliza datos no etiquetados, y los algoritmos descubren patrones o agrupaciones sin intervención humana, siendo útil cuando hay grandes volúmenes de datos sin etiquetas.

Entre sus ventajas destacan el descubrimiento de patrones sin necesidad de etiquetas, y su aplicabilidad en tareas como agrupación en clústeres, asociación o reducción de dimensionalidad. Sin embargo, suelen ser menos precisos y más difíciles de interpretar.

Algunas aplicaciones incluyen secciones de noticias como Google News, detección de anomalías en equipos informáticos y creación de imágenes médicas radiológicas. [31]

4.2.8.1. Modelo Random Forest

Random Forest es un modelo supervisado de machine learning que combina múltiples árboles de decisión para mejorar la precisión y evitar el sobreajuste. Cada árbol se entrena con diferentes datos, y el modelo final toma la decisión basada en la combinación de sus predicciones.

Entre sus ventajas están la alta precisión, robustez frente al sobreajuste y capacidad para manejar datos faltantes. Sin embargo, es más complejo de interpretar y requiere más recursos computacionales.

Se usa en clasificación, como detección de fraudes y reconocimiento de imágenes, en regresión, como predicción de valores, y para identificar las variables más importantes en un conjunto de datos.

4.2.9. Herramientas

4.2.9.1. Docker y contenedores

Docker es una plataforma de código abierto que permite crear, desplegar y gestionar aplicaciones dentro de contenedores. Los contenedores son entornos ligeros y portátiles que contienen todo lo necesario para ejecutar una aplicación, como el código, bibliotecas, dependencias y configuraciones, asegurando que funcione de manera uniforme en cualquier entorno.

Entre sus ventajas destacan su eficiencia, ya que los contenedores son más ligeros que las máquinas virtuales, y su portabilidad, permitiendo trasladar aplicaciones entre diferentes sistemas sin problemas de compatibilidad. Además, Docker facilita la gestión de entornos de desarrollo y producción, acelerando los despliegues y mejorando la escalabilidad.

Se usa ampliamente en desarrollo y despliegue de aplicaciones, pruebas automatizadas, creación de entornos de desarrollo reproducibles, y en la infraestructura de microservicios, permitiendo que los equipos trabajen de manera más ágil y eficiente.

4.2.9.2. Python

Python es un lenguaje de programación basado en C de alto nivel, interpretado y de propósito general, reconocido por su sintaxis clara y legibilidad. Su facilidad de uso y extensa biblioteca lo hacen ideal para el desarrollo de aplicaciones web, ciencia de datos, inteligencia artificial y más.

4.2.9.3. Reflex.dev

Framework de Python creado a finales de 2024 evolucionando a partir del framework Pynecone, que se encuentra en fase beta y utiliza tecnologías avanzadas como FastAPI en el backend y Tailwind de React.js en el frontend. Este framework ha recibido financiación de empresas debido a su potencial para competir con Django y Flask, los frameworks más utilizados en el desarrollo web con Python.

Reflex se destaca por unificar el desarrollo frontend y backend en una sola base de código en Python, eliminando la necesidad de mantener aplicaciones separadas y reduciendo la repetición de código. Combina la accesibilidad de los frameworks de bajo código con la flexibilidad y personalización del desarrollo web tradicional.

El objetivo de Reflex es simplificar el proceso de desarrollo, permitiendo a los desarrolladores enfocarse en la lógica de la aplicación en lugar de los detalles técnicos. Además, facilita el despliegue de nuevas aplicaciones web en minutos mediante comandos simples, con opciones de auto-hospedaje o alojamiento en servidores propios. [34]

4.2.9.3.1. Arquitectura y diseño

Las aplicaciones desarrolladas con Reflex se dividen en una parte frontend basada en React.js y una parte backend utilizando FastAPI. Aunque la interfaz de usuario se traduce a Javascript, toda la lógica de la aplicación y la gestión del estado permanecen en Python y se ejecutan en el servidor. Emplea WebSockets para la comunicación bidireccional, permitiendo que los eventos se envíen desde el frontend al backend y que las actualizaciones de estado se envíen de vuelta desde el backend al frontend.

Cuando se despliega un proyecto, Reflex compila la interfaz en una aplicación Next.js de una sola página y la entrega en un puerto (de forma predeterminada 3000) al que se puede acceder a través del navegador. El frontend refleja el estado de la aplicación y envía eventos al backend cuando el usuario interactúa con la interfaz de usuario, pero no se ejecuta ninguna lógica real en la interfaz. [34]

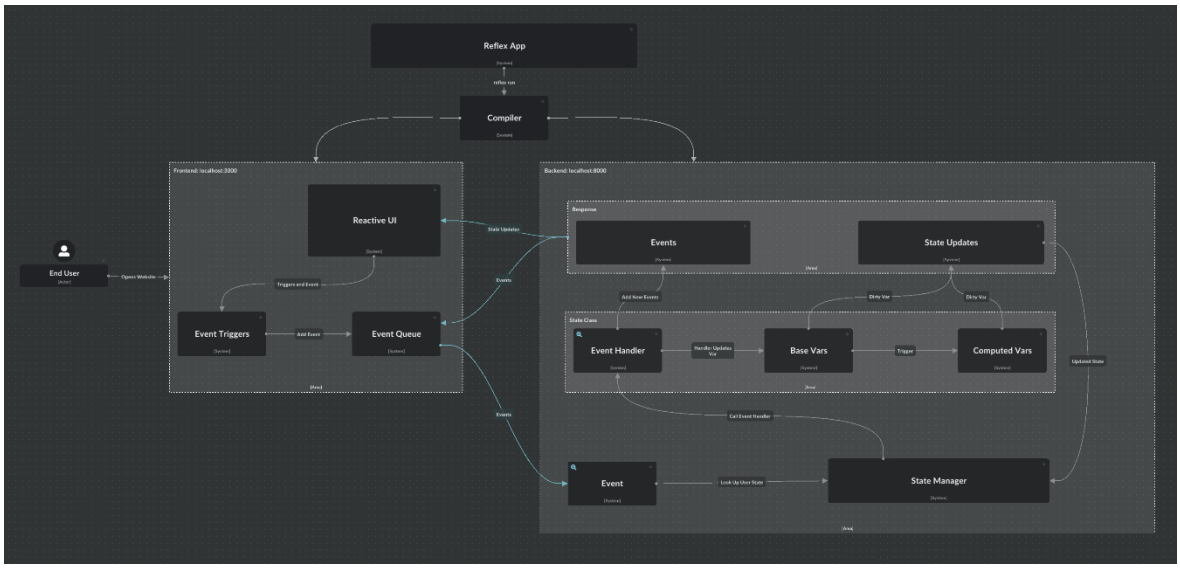

Ilustración 16 Diagrama de funcionamiento web en Reflex.dev

4.2.9.3.2. Frontend

Las interfaces de Reflex se crean mediante componentes de Python, en lugar de usar un lenguaje de plantillas que combine HTML y Python. Estos componentes se compilan en componentes de React basados en Radix, una popular biblioteca que ofrece muchos otros componentes para gráficos, tablas de datos, y más. Adicionalmente los usuarios pueden integrar sus propios componentes si Reflex no proporciona lo que necesitan.

En cuanto al estilo, Reflex se ha diseñado para garantizar aplicaciones atractivas desde el primer momento, ofreciendo a los desarrolladores control total sobre su apariencia. Los componentes de Reflex se pueden diseñar utilizando CSS, gracias a la biblioteca Emotion, que permite el estilo "CSS-in-Python". Esto permite pasar cualquier propiedad de CSS como argumento de palabra clave a un componente, incluyendo propiedades receptivas

mediante una lista de valores, facilitando la creación de diseños responsivos y personalizados. [34]

4.2.9.3.3. Backend

Los estados y la lógica se definen dentro de una clase State que permanece en Python y se ejecuta en el servidor. Al desplegar la web, se inicia un servidor FastAPI (por defecto en el puerto 8000) al que se conecta la interfaz a través de un WebSocket.

El estado se compone de variables y controladores de eventos:

- Las variables son valores en su aplicación que pueden cambiar con el tiempo. Se definen como atributos de clase y pueden ser cualquier tipo de Python que pueda serializarse a JSON.

- Los controladores de eventos son métodos de clase que se llaman cuando el usuario interactúa con la interfaz de usuario. Son la única forma de modificar las variables y se pueden llamar en respuesta a las acciones del usuario, como hacer clic en un botón o escribir en un cuadro de texto.

Dado que los controladores de eventos se ejecutan en el backend, es posible usar cualquier biblioteca de Python dentro de ellos. [34]

4.2.9.3.4. Procesamiento de eventos

El procesamiento de eventos y las actualizaciones de estado eliminan la necesidad de escribir una gran cantidad de código repetitivo para conectar el frontend y el backend. Reflex maneja la comunicación entre el frontend y el backend haciendo que cuando las variables se actualicen, la interfaz de usuario se actualice automáticamente. [34]

Activadores de Eventos

Se denominan desencadenadores de eventos a las diversas maneras en que el usuario puede interactuar con la interfaz de usuario: haciendo clic en un botón, escribiendo en un cuadro de texto o pasando el cursor sobre un elemento. [34]

Cola de Eventos

En la interfaz, se mantiene una cola de todos los eventos pendientes. Un evento consta de tres datos principales:

- Token de identificación del cliente que permite al backend saber qué estado actualizar.

- Controlador de eventos que se ejecutará en el estado.

- Argumentos que se pasarán al controlador de eventos.

Cuando se activa un evento, se agrega a la cola. A través de un flag de procesamiento se asegura que solo se procese un evento a la vez. Esto garantiza que el estado sea siempre coherente y que no haya condiciones de carrera con dos controladores de eventos modificando el estado al mismo tiempo. Una vez que el evento está listo para ser procesado, se envía al backend a través de una conexión WebSocket. [34]

Controlador de eventos

Una vez recibido el evento, se procesa en el backend. Reflex utiliza un administrador de estado que mantiene un mapeo entre los tokens del cliente y su estado. Por defecto, el administrador de estado es un diccionario en memoria, pero se puede ampliar para usar una base de datos o caché. [34]

Administrador del estado

Una vez se ha obtenido el estado del usuario, el siguiente paso es ejecutar el controlador de eventos con los argumentos. Cada vez que un controlador de eventos se ejecuta, se guarda el estado en el administrador de estado y ese envía las actualizaciones de estado a la interfaz para actualizar la interfaz de usuario.

Para mantener el rendimiento a medida que crece el estado, internamente Reflex realiza un seguimiento de las variables que se actualizaron durante el controlador de eventos (variaciones sucias). Cuando el controlador de eventos termina de procesar, se encuentran todas las variables sucias.

Por último, se almacena el nuevo estado en el administrador de estado y se envía la actualización del estado a la interfaz. Luego, la interfaz actualiza la interfaz de usuario para reflejar el nuevo estado. [34]

4.2.9.4. Librería paho-mqtt

Librería de Python diseñada para implementar el protocolo MQTT creada y mantenida por Eclipse. Permite manejar la conexión y reconexión al broker, así como la publicación y suscripción a tópicos. Los mensajes publicados pueden tener diferentes niveles de calidad de servicio (QoS), y la suscripción a tópicos permite recibir mensajes en tiempo real. Además, Paho-MQTT soporta la personalización de callbacks para manejar eventos como la conexión, la recepción de mensajes y la entrega de publicaciones.

4.2.9.5. Librería opcua

Librería de Python de código abierto para la comunicación y control en entornos industriales, basada en el protocolo OPC Unified Architecture. Facilita la interoperabilidad

entre diversos dispositivos y sistemas de control, independientemente del fabricante, permitiendo una integración eficiente en entornos industriales.

Soporta características avanzadas de seguridad como autenticación y encriptación, y proporciona un modelo de datos rico y extensible. Esto permite representar la información de manera estructurada y jerárquica, mejorando la gestión y análisis de datos. La librería es escalable y compatible con múltiples plataformas, siendo ideal para aplicaciones industriales de cualquier tamaño.

4.2.9.1. Librerías Pandas y Numpy

Librerías de Python utilizada para el análisis y la manipulación de datos. Son ampliamente utilizadas en ciencia de datos, finanzas, estadística y otras áreas que requieren el manejo eficiente de datos tabulares y de series temporales.

4.2.9.2. Librerías SQLAlchemy y SQLModel

Librerías de Python de código abierto que proporcionan un conjunto completo de herramientas para trabajar con bases de datos relacionales.

Ofrecen una interfaz de alto nivel para el mapeo objeto-relacional (ORM), permitiendo a los desarrolladores interactuar con bases de datos utilizando objetos de Python en lugar de escribir SQL directamente.

Soporta tanto el uso del ORM (Object-Relational Mapping) como la escritura directa de SQL, ofreciendo flexibilidad para distintos casos de uso. Además, incluye características avanzadas para la gestión de transacciones, conexiones y consultas, facilitando el desarrollo y mantenimiento de aplicaciones con bases de datos complejas.

4.2.9.3. Librerías Passlib y Bcrypt

Passlib es una librería de Python de código abierto para la gestión de contraseñas, diseñada para facilitar la implementación de técnicas seguras de hashing y almacenamiento de contraseñas. Ofrece una interfaz unificada para diversos algoritmos de hashing, permitiendo a los desarrolladores manejar contraseñas de manera segura y eficiente.

Bcrypt es un algoritmo de hashing robusto y adaptativo, también de código abierto, diseñado para proteger contraseñas mediante la incorporación de un factor de costo que incrementa la complejidad del hash. Esto lo hace resistente a ataques de fuerza bruta. Bcrypt se integra fácilmente con Passlib, proporcionando una solución completa para la seguridad de contraseñas en aplicaciones Python.

4.2.9.1. Librería Scikit-learn

Scikit-learn es una librería de código abierto en Python, construida sobre NumPy, SciPy, Matplotlib y Cython, diseñada para tareas de aprendizaje automático como clasificación, regresión, agrupamiento, y preprocesamiento de datos.

Es ampliamente utilizada en IA por su facilidad de uso y eficiencia. En este proyecto se ha elegido por su sencilla implementación de la regresión lineal, una técnica que modela la relación entre variables dependientes e independientes ajustando una línea o plano que minimiza la distancia entre los puntos de datos, facilitando predicciones básicas en campos como ingeniería, economía y biología. [35]

En este proyecto se ha decidido utilizar esta biblioteca debido a las facilidades que ofrece para la implementación de la regresión lineal. La regresión lineal es un método estadístico empleado para modelar la relación entre una variable dependiente y una o varias variables independientes. Esto implica ajustar una línea recta (en el caso de una variable) o plano (en el caso de múltiples variables) a los datos, de forma que se minimice la distancia entre los puntos

de datos y la mencionada línea o plano. Esta técnica es fundamental para la predicción básica de datos y es ampliamente utilizada en diversos campos dentro de la ingeniería, así como en economía o incluso biología.

Scikit-learn facilita en gran medida el uso de la regresión lineal mediante una serie de herramientas que simplifican el proceso, permitiendo a los usuarios centrarse en el análisis y la interpretación de los resultados. [36]

5. DESARROLLO

5.1. ESPECIFICACIONES DEL SISTEMA IOT

En este apartado se detallan las especificaciones del sistema IoT diseñado para entornos industriales. El sistema abarca la extracción, procesamiento, visualización y predicción de datos.

La sección se estructura en cuatro subapartados:

- Requisitos Funcionales
- Requisitos No Funcionales
- Especificaciones Técnicas
- Limitaciones y Suposiciones

Estos subapartados cubren desde las funcionalidades esenciales hasta los parámetros de rendimiento y seguridad del sistema. Se presentan los aspectos críticos que garantizarán su funcionamiento eficiente y fiable en el entorno industrial.

5.1.1. Requisitos funcionales

Los requisitos funcionales describen las funcionalidades específicas que el sistema debe cumplir para lograr sus objetivos en un entorno industrial. A continuación, se detallan los principales requisitos funcionales que se han identificado para el sistema IoT de extracción, procesamiento, visualización y predicción de datos:

Tabla 2 Requisitos funcionales del sistema

Categoría	Requisito	Descripción
Captura y Transmisión de Datos	**Captura de Datos**	El sistema debe capturar datos de variables industriales en PLCs y dispositivos conectados.
	Frecuencia de Captura	La frecuencia de captura de datos puede variar dependiendo del tipo de dispositivo y la necesidad, con una frecuencia mínima establecida cada segundo.
	Protocolos de Comunicación	Utilización de protocolos industriales para la transmisión en tiempo real de los datos capturados por los sensores.
	Frecuencia de Transmisión	Los datos deben ser transmitidos cada segundo al servidor central.
Procesamiento de Datos	**Algoritmos de Procesamiento**	Implementación de un sistema de transformación o unificación de datos de las distintas fuentes y tipos.

Categoría	Requisito	Descripción
	Tiempo de Procesamiento	Procesamiento en tiempo real con una latencia máxima de 2 segundos desde la recepción de los datos.
Almacenamiento de Datos	**Base de Datos**	Implementación de una base de datos SQL para el almacenamiento estructurado de datos. La base de datos debe ser escalable y capaz de manejar grandes volúmenes de datos.
	Accesibilidad y Consulta	Los datos deben ser fácilmente accesibles para consultas y análisis por parte de los usuarios autorizados. Implementación de índices y optimización de consultas para asegurar un acceso rápido a la información almacenada.
Visualización de Datos	**Dashboard Interactivo**	El sistema debe proporcionar una interfaz de usuario que permita la visualización en tiempo real de los datos a través de gráficos y widgets interactivos.
Predicción de Datos	**Modelos Predictivos**	Implementación de modelos de regresión y clasificación para predecir fallos en los equipos basándose en datos históricos.
Gestión de Usuarios y Autenticación	**Autenticación de Usuarios**	El sistema debe implementar un mecanismo de autenticación que requiera que los usuarios se identifiquen antes de acceder. Utilización de tokens de seguridad (JWT) para la gestión de sesiones de usuario.
	Gestión de Usuarios	El sistema debe permitir la creación, edición y eliminación de cuentas de usuario.

5.1.2. Requisitos no funcionales

Los requisitos no funcionales describen los criterios que determinan cómo debe operar el sistema para cumplir con los estándares de rendimiento, seguridad, fiabilidad, usabilidad, mantenibilidad y compatibilidad. Estos requisitos son esenciales para asegurar que el sistema IoT no solo funcione correctamente, sino que también sea robusto, seguro y fácil de usar y mantener. A continuación, se detallan los principales requisitos no funcionales:

Tabla 3 Requisitos no funcionales del sistema

Categoría	Requisito	Descripción
Seguridad	**Autenticación**	Implementación de autenticación basada en tokens (JWT) para asegurar que solo los usuarios autorizados puedan acceder al sistema.
Usabilidad	**Interfaz Intuitiva**	La interfaz de usuario debe ser intuitiva y fácil de usar, permitiendo una navegación sencilla y eficiente por las diferentes funcionalidades del sistema.
Mantenibilidad	**Documentación Completa**	Provisión de documentación completa y detallada para facilitar el mantenimiento y la actualización del sistema.
	Modularidad del Código	El sistema debe estar desarrollado de manera modular para facilitar el mantenimiento y la actualización de sus componentes individuales.
Compatibilidad	**Integración con Herramientas**	Capacidad para ampliar sus funcionalidades de extracción con nuevos tipos de fuentes de datos, almacenamiento con otros

Categoría	Requisito	Descripción
		tipos de bases de datos, predicción mediante otros modelos, etc.

5.2. ANÁLISIS Y TECNOLOGÍAS

En esta sección se lleva a cabo un estudio detallado del entorno en el que se implementará el sistema IoT, así como una evaluación de las tecnologías y requisitos necesarios para su desarrollo.

5.2.1. Análisis del problema

Como se ha podido observar durante los apartados introductorios del proyecto, el objetivo es desarrollar una plataforma industrial que simplifique los grandes obstáculos que tienen las empresas para unificar un entorno de recolección de todos los datos que recogen o generan.

Más en detalle, las empresas industriales se encuentran en una etapa de digitalización y en muchos casos su maquinaria o cadenas de producción no tienen un reporte de los datos, o es estricta e independiente para cada una de ellas. Por ejemplo, en un proyecto en el que he participado como técnico del área de IoT e Industria 4.0 de Integra Tecnología y Estrategia, el cliente solicitaba un método de extracción de datos para tres máquinas con más de 20 años de antigüedad que debían continuar funcionando los próximos 30 años debido al gran coste, tanto económico como logístico, que supondría cambiarlas.

El enfoque se ha llevado a cabo desde la perspectiva de acceso a datos de las capas base de la pirámide CIM, ya que en la mayoría de casos son los datos que las empresas no tienen tan controlados. Para ello, simular un PLC es la opción ideal, ya que en ellos se recoge gran parte de la información de estos procesos.

Para que sea una plataforma de utilidad para las empresas, teniendo en cuenta otros antecedentes y las facilidades actuales en el campo de la inteligencia artificial, además de la extracción y visualización de datos, se ha añadido un bloque de predicciones que pretende analizar cada variable recibida de forma individual.

Esta plataforma está ideada para un proyecto de fin de grado por lo que el alcance es reducido, aunque pretende ser escalable para dar la posibilidad de adición de nuevos módulos de comunicación, ampliaciones en modelos predictivos, etc.

5.2.2. Análisis del entorno

5.2.2.1. Protocolos de comunicación

Existen diversos protocolos que se utilizan para la transmisión de datos en entornos industriales, cada uno con sus propias ventajas y desventajas. Se presenta una tabla comparativa de los principales protocolos industriales para identificar las características clave de cada protocolo.

Tabla 4 Comparativa entre los principales protocolos de comunicación IIoT

Característica	OPC UA	Modbus TCP/IP	PROFINET
Interoperabilidad	Alta, estándar abierto	Moderada, estándar abierto	Alta, específico de la industria
Seguridad	Alta (cifrado, autenticación)	Baja (básico, requiere capas adicionales)	Moderada (seguridad a nivel de red)
Topología	Jerárquica y flexible	Maestro-esclavo	Jerárquica
Protocolo de Transporte	TCP/IP	TCP/IP	TCP/IP, UDP
Compatibilidad IIoT	Alta, diseñado para IIoT	Baja, no diseñado para IIoT	Moderada, puede integrarse con IIoT
Manejo de Datos Complejos	Soporta datos complejos y modelos de información	Limitado a datos simples	Soporta datos complejos
Estandarización	IEC 62541	IEC 61158, IEC 61784	IEC 61158, IEC 61784
Aplicación Típica	Integración de sistemas, automatización industrial	Automatización simple, dispositivos antiguos	Automatización industrial de alta precisión
Latencia	Baja a moderada	Baja	Muy baja
Capacidad de Tiempo Real	Moderada	Baja	Alta
Escalabilidad	Alta, adaptable a sistemas crecientes	Baja, limitado a dispositivos y redes	Alta, pero requiere configuración específica
Soporte para Redundancia	Sí	Limitado	Sí
Fácil de Implementar	Moderada	Alta	Moderada

Los modelos más actuales de PLC de marcas líderes como Siemens (S7-1200, S7-1500), Rockwell (ControlLogix, CompactLogix) y Schneider (Modicon M241, M251, M262) ya integran tecnologías de extracción de datos inalámbricas y utilizan OPC UA como el principal protocolo de comunicación externa a través de TCP/IP. Esto facilita la

incorporación del IIoT en entornos industriales, permitiendo a las empresas mantener y optimizar sus operaciones con tecnologías avanzadas y conectividad mejorada.

Por otro lado, MQTT se ha convertido en el protocolo por excelencia para IoT debido a su diseño ligero y eficiente. La mayoría de los dispositivos IoT modernos, desde sensores simples hasta complejos sistemas de automatización, ya integran soporte para MQTT, lo que lo convierte en un estándar en el ecosistema IoT. Para gestionar la comunicación MQTT, se va a comparar los principales brokers MQTT disponibles de forma gratuita:

Tabla 5 Tabla comparativa de los principales broker MQTT gratuitos

Característica	HiveMQ	Eclipse Mosquitto	AWS IoT Core	Google Cloud IoT Core	Adafruit IO
Modelo de Implementación	Serverless	Servidor local	Serverless	Serverless	Serverless
Facilidad de Uso	Muy alta	Alta	Media	Media	Alta
Escalabilidad	Alta, automático	Media, manual	Alta, automático	Alta, automático	Media, limitado
Gestión de Conexiones	Automática	Manual	Automática	Automática	Limitado
Coste	Gratuito (plan básico)	Gratuito	Gratuito (con limitaciones)	Gratuito (con limitaciones)	Gratuito (con limitaciones)
Límite de Conexiones	Alta, soporta múltiples conexiones	Media, depende de la configuración	Alta, depende del plan	Alta, depende del plan	Limitado a un número pequeño
Facilidad de Configuración	Muy alta	Alta	Media	Media	Alta
Seguridad	Alta, integrada	Alta, requiere configuración	Alta, integrada	Alta, integrada	Media, básica
Documentación y Soporte	Extensa y detallada	Buena, comunidad activa	Extensa, soporte oficial de AWS	Extensa, soporte oficial de Google	Buena, comunidad activa

Entre las opciones, se va a utilizar HiveMQ en formato serverless a través del plan básico. Este broker MQTT es ideal para proyectos académicos y de prueba debido a su facilidad de uso, su capacidad de manejar múltiples conexiones simultáneas y no incurrir en costes adicionales. Además, al no contar con infraestructura para alojar el servidor, se elimina la necesidad del despliegue, mantenimiento y administración del mismo.

Ambos protocolos, OPC UA y MQTT, son los que se han escogido para utilizar como herramienta de comunicación entre cualquier dispositivo del entorno industrial y la plataforma.

5.2.2.2. Hardware

5.2.2.2.1. Simulación de dispositivos

Para simular un PLC o un dispositivo se ha planteado utilizar una Raspberry Pi que simule datos de producción y actúe como cliente para la plataforma. Su bajo coste y alta flexibilidad facilitan disponer de un simulador de datos de producción similares a los de un PLC real. Además, gracias a su capacidad para ejecutar múltiples scripts mediante contendores Docker que aíslan el funcionamiento, una Raspberry Pi puede simultáneamente actuar como cliente MQTT y servidor OPC UA, enviando datos a la plataforma IIoT para su procesamiento y análisis. Esto permite crear un entorno de pruebas realista y funcional sin la necesidad de invertir en hardware industrial costoso.

5.2.2.2.2. Alojamiento de la plataforma

Como se trata de entornos industriales donde la seguridad es crítica se ha decidido que el alojamiento de la plataforma debe realizarse en un entorno On-premise dentro de la red local de la empresa, a través de una VM con IP interna para que se le asocie un dominio y únicamente se pueda acceder estando conectado a la red interna de la empresa.

Por tanto, la implementación del proyecto se va a realizar en servidor local como un ordenador que se sustituiría por la VM sin necesidad de acceso a internet y únicamente con necesidad de virtualización para ejecutar los contendores.

5.2.2.3. Software

La plataforma pretende ser robusta y escalable por lo que se han investigado distintas herramientas que permitan desarrollarla adecuadamente. La principal problemática en esto es que las herramientas de LowCode se quedaban cortas para obtener una plataforma en condiciones y las plataformas de desarrollo completo requerían una programación full stack que integrara la programación del frontend y del backend.

Durante el grado se ha programado en Python y microcontroladores en C++ como lenguajes de backend, pero no se ha aprendido ningún lenguaje de programación para el frontend.

Desde un principio se ha visto una opción ideal la programación del backend en Python por su alto nivel y el amplio entorno de librerías con las que cuenta para comunicación, análisis y sobre todo predicciones. Por ello, la solución es el desarrollo con un framework reciente llamado Reflex.dev que permite la programación completa de la web en Python, e internamente convierte el frontend a través de React.js a código HTML, CSS y JS.

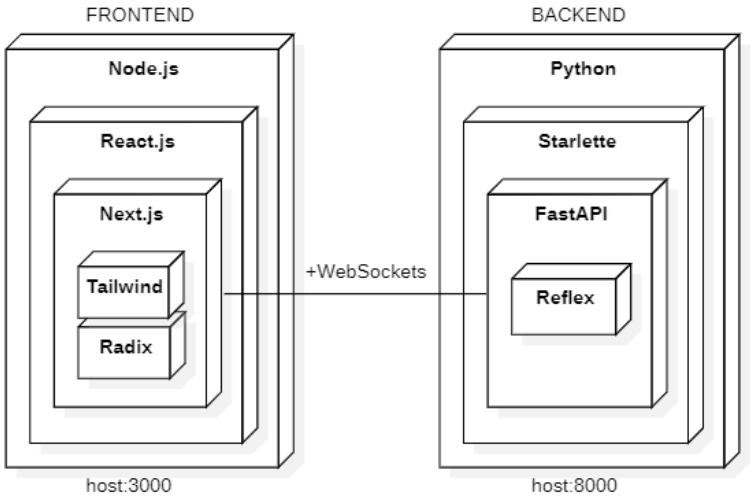

Ilustración 17 Diagrama UML de implementación de la plataforma con Reflex y Python

5.2.2.4. Almacenamiento de datos

Las bases de datos SQL son preferidas por su capacidad para manejar relaciones complejas y garantizar la integridad referencial, lo que es crucial para integrar datos de múltiples fuentes industriales. Aunque las bases de datos NoSQL son más flexibles y escalables, su estructura no relacional es menos adecuada para gestionar relaciones directas entre datos. PostgreSQL se ha escogido debido a la experiencia previa.

5.2.2.5. Predicción de datos

Para el desarrollo del módulo de predicciones, se realizó una investigación exhaustiva de las principales librerías de Python que facilitan la implementación de modelos de machine learning. Las opciones consideradas fueron TensorFlow, Keras, PyTorch y Scikit-learn.

Se ha seleccionado Scikit-learn debido a su facilidad de uso, amplia documentación y robustez para manejar tareas comunes de aprendizaje automático. Scikit-learn es ideal para el desarrollo rápido y eficiente de modelos predictivos que simplifican el flujo de trabajo de machine learning.

En cuanto al modelo, se ha optado por un enfoque no supervisado, dado que la plataforma debe autogestionar las predicciones sin intervención humana en la clasificación de datos. A continuación, se realiza el análisis comparativo de los principales modelos de machine learning no supervisados de regresión y cálculo de predicciones:

Tabla 6 Comparativa de los modelos no supervisados de machine learning

Característica	Regresión lineal	ARIMA	Prophet	Random Forest	Gradient Boosting	LSTM
Complejidad	Muy baja	Baja	Moderada	Moderada - Alta	Alta	Muy alta
Precisión en predicciones a corto plazo	Moderada	Alta	Alta	Alta	Muy alta	Muy alta
Precisión en predicciones a largo plazo	Baja	Moderada	Alta	Moderada	Moderada	Alta
Manejo de datos	Limitado	Limitado	Bueno	Excelente	Excelente	Excelente
Capacidad de capturar estacionalidad	Limitado	Excelente	Excelente	Buena	Buena	Excelente
Robustez frente a outliers	Baja	Bajo	Moderada	Alta	Moderada	Moderada
Manejo de múltiples variables	Bueno	Limitado	Moderada	Excelente	Excelente	Excelente
Interoperabilidad	Muy alta	Alta	Alta	Moderada	Baja	Baja
Velocidad de entrenamiento	Muy rápida	Rápida	Rápida	Moderada	Lenta	Muy lenta
Requisitos de datos	Bajos	Bajos	Bajos	Moderados	Altos	Muy altos
Capacidad de automatización	Alta	Baja	Alta	Alta	Moderada	Moderada
Facilidad de implementación y mantenimiento	Muy alta	Moderada	Alta	Alta	Moderada	Baja

Entre los modelos disponibles, se seleccionó el Random Forest por su equilibrio entre rendimiento y complejidad. Ofrece alta precisión a corto plazo, maneja datos no lineales, es robusto ante outliers y moderadamente interpretable. Sus requisitos de datos manejables, alta automatización y fácil implementación lo aventajan sobre modelos más simples o complejos. Proporciona rendimiento avanzado manteniendo la practicidad necesaria en entornos industriales, donde eficiencia, confiabilidad y facilidad de mantenimiento son cruciales.

La combinación de Scikit-learn y Random Forest proporciona una solución robusta y eficiente para el módulo de predicciones de la plataforma, permitiendo un análisis predictivo automatizado y adaptable a diversos escenarios industriales.

5.2.1. Especificaciones técnicas

En esta sección se presentan las especificaciones técnicas detalladas del sistema IoT para entornos industriales, abarcando hardware, software, protocolos de comunicación, seguridad, y configuración de red. Además, se detallan las estrategias de despliegue y escalabilidad, asegurando que el sistema pueda manejar cargas crecientes de datos y usuarios sin comprometer el rendimiento.

Tabla 7 Especificaciones técnicas del proyecto

Componente	Especificación
Hardware	Simulación PLC y dispositivos: Se despliegan dos clientes distintos que envían de forma paralela al cliente MQTT y al cliente OPC UA por conexión inalámbrica desde una Raspberry Pi. Plataforma IoT: Se despliega en un servidor Windows 10 y se aloja el host de Reflex para control de versiones adecuado.
Software	Python, Reflex.dev, Pandas, Scikit-learn, SQLAlchemy, Mqtt-client, Opcua, Passlib y Bcrypt, broker MQTT EMQX.
Protocolos de Comunicación	OPC UA para comunicación entre PLCs y el servidor central, MQTT para transmisión de datos en tiempo real entre sensores y el servidor central, configuración de red segura.
Base de Datos	PostgreSQL con tablas para datos recogidos y configuraciones.
Seguridad	Autenticación basada en tokens JWT, cifrado AES-256 para datos en reposo.
Infraestructura de Despliegue	Despliegue en host local, uso de Gunicorn para servidor WSGI. Alojamiento del código en GitHub y actualizaciones con Git.
Documentación y Mantenimiento	Documentación detallada del sistema, diagramas de estructura y comportamiento, así como de los modelos de base de datos.

5.3. DISEÑO DEL SISTEMA IIOT

Este apartado proporciona una guía técnica detallada para la estructura y desarrollo del sistema IIoT. Se abordan los componentes principales, incluyendo backend, frontend, base de datos y estrategias de integración y comunicación mediante protocolos MQTT y OPC UA.

El diseño de la plataforma del sistema IIoT implica una representación abstracta o con bocetos de la solución propuesta. La estructura del proceso de diseño de cualquier producto generalmente sigue las etapas de:

DISEÑO > PROTOTIPO > INDUSTRIALIZACIÓN

En el diseño inicial de proyectos de ingeniería destacan los modelos iterativos, donde se realizan modificaciones continuas sobre el producto a medida que se avanza. Un modelo adecuado para este tipo de proyectos es el modelo de prototipado evolutivo, en el cual se desarrollan prototipos incrementales que se refinan continuamente con base en el feedback recibido.

El prototipo se realizará inicialmente en forma de simulación. El último apartado de industrialización, que incluye diseño de negocio y otros aspectos adicionales, no se abordará en este proyecto.

5.3.1. Arquitectura de infraestructura

Para comenzar el diseño, es fundamental tener clara la infraestructura necesaria que compondrá el sistema IIoT. El diseño va a ser completo y escalonado, comenzando por diagramas de bloques de alto nivel que se van a profundizar hasta obtener diagramas complejos de bajo nivel.

El sistema se va a componer principalmente de dos componentes clave de hardware: una Raspberry Pi y un servidor local.

- **Raspberry Pi:** Va a actuar como servidor de simulación para dos dispositivos, un cliente MQTT y un servidor OPC UA.

- **Servidor local:** Se va a utilizar un PC personal con Windows 10 como sistema operativo. Este servidor local va a alojar y ejecutar los servicios de la plataforma, incluyendo el servidor backend y la base de datos como contendores Docker. Además, va a permitir la interacción y visualización de la plataforma a través de un navegador web accediendo a la interfaz de usuario.

A continuación, se muestra un diagrama a alto nivel de la solución propuesta:

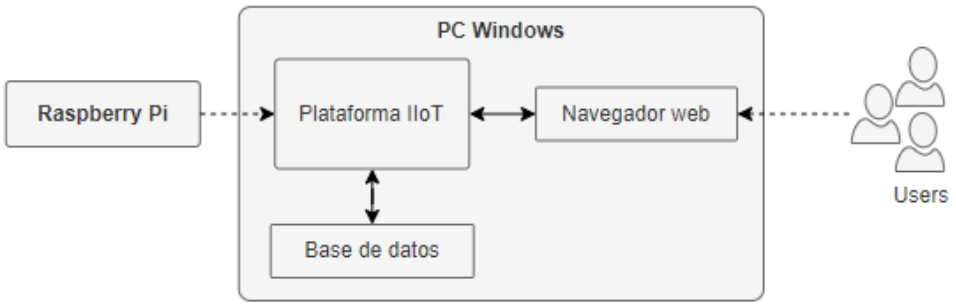

Ilustración 18 Diagrama de alto nivel de la infraestructura del sistema IIoT diseñado

5.3.1.1. Raspberry PI

La simulación de dispositivos se va a realizar mediante un servidor que contenga los dos clientes y simule datos sobre la plataforma. En un entorno industrial serían generados durante los procesos de producción y recogidas por los dispositivos y PLC.

Ambas simulaciones se van a realizar mediante scripts de Python que se ejecutarán de forma simultanea e independiente empleando contendores Docker:

- **Dispositivo MQTT**: Cliente Python usando la librería mqtt-client para conectar con un bróker en alojado en la nube de HiveMQ.
- **Servidor OPC UA en PLC**: Cliente Python usando la librería opcua para enviar al cliente alojado en el PC.

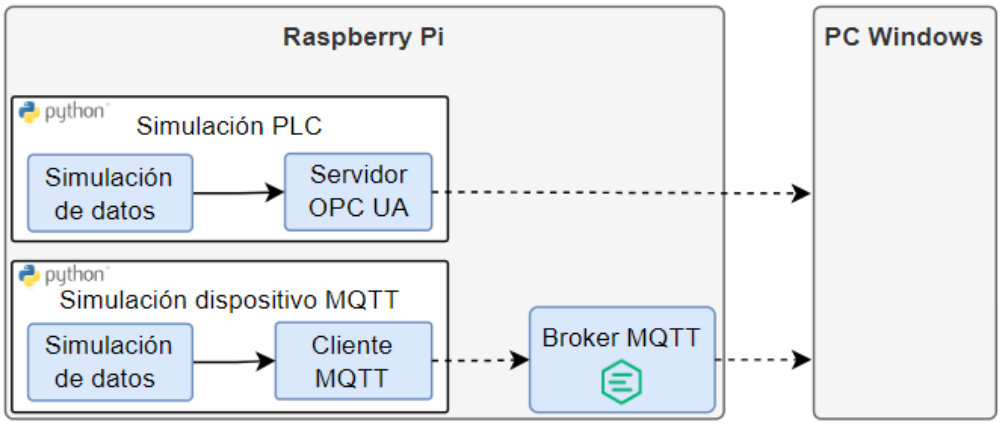

Ilustración 19 Diagrama de bloques de simulación mediante la Raspberry Pi

5.3.1.2. PC Windows

El PC Windows actúa como el servidor principal de la plataforma IIoT integrando varios servicios y módulos que se describen a continuación:

- **Cliente OPC UA:** Este módulo se conecta al servidor OPC UA externo y recoge datos de producción en tiempo real, enviándolos al servidor Backend para su procesamiento y almacenamiento.

- **Cliente MQTT:** Este módulo se conecta al Broker MQTT y suscribe a los mensajes de datos enviados por los dispositivos IoT. Los datos recibidos son enviados al servidor Backend para su procesamiento y almacenamiento.

- **Servidor Backend:** Es el núcleo de la plataforma IIoT. Procesa los datos recibidos de los clientes OPC UA y MQTT, realiza operaciones de almacenamiento en la base de datos y coordina con otros módulos de la plataforma.

  - **Módulo de predicciones:** Este módulo se integra con el servidor Backend para realizar análisis predictivos sobre los datos de producción. Utiliza algoritmos de machine learning para predecir comportamientos futuros basados en datos históricos y actuales.

  - **Almacenamiento de datos:** Una base de datos relacional que guarda todos los datos de producción, históricos y predicciones. Este almacenamiento es crucial para realizar análisis y generar informes.

- **Servidor Frontend**: Este módulo maneja la interfaz de usuario de la plataforma IIoT. Genera las páginas web que los usuarios acceden a través de un navegador web para interactuar con la plataforma. Proporciona funcionalidades como el inicio de sesión, la visualización de datos de producción, y el acceso a las predicciones y reportes generados por el sistema.

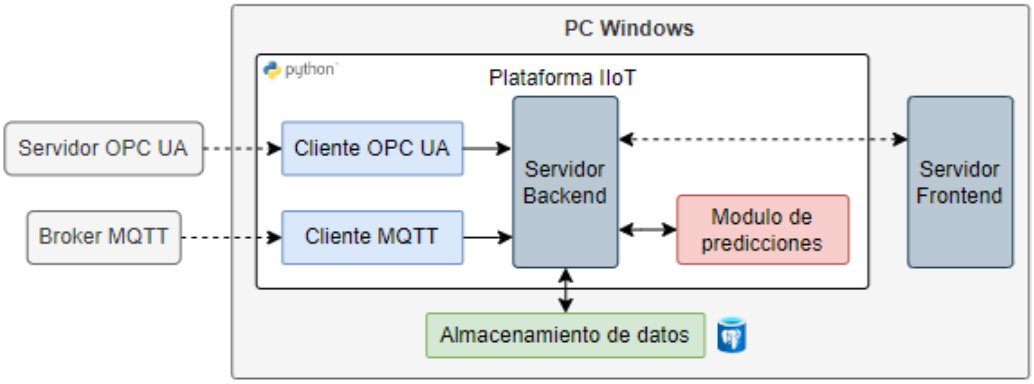

Ilustración 20 Diagrama de bloques de bajo nivel del PC que actúa como servidor

5.3.1.3. Navegador web

El navegador web es el punto de interacción principal para los usuarios. Aloja la interfaz de usuario de la plataforma IIoT, permitiendo el acceso y la gestión de los datos de producción y predicciones.

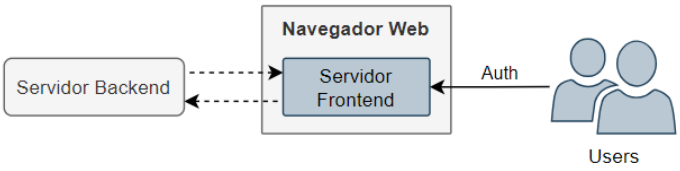

Ilustración 21 Diagrama de bloques de la interacción de usuarios con la plataforma IIoT

5.3.2. Arquitectura de plataforma

Una vez diseñada la infraestructura del proyecto se va a estudiar y diseñar la plataforma IIoT. Además, se ha decidido segmentar el diseño para explicar detalladamente cada elemento que la constituye y para facilitar la comprensión del lector.

El diseño se basa enteramente en Python, aprovechando su versatilidad y robusto ecosistema de bibliotecas. Se ha querido utilizar el nuevo framework Reflex.dev, una elección innovadora que unifica el desarrollo de backend y frontend, simplificando significativamente el proceso de creación de aplicaciones web interactivas. La plataforma se ha planteado con una arquitectura monolítica, ya que principalmente el código se va a desarrollar en un único servicio.

Para el almacenamiento de datos, se ha optado por PostgreSQL, una base de datos relacional potente y confiable, ideal para entornos industriales que requieren alta seguridad y rendimiento. La implementación se ha realizado en un entorno local utilizando contenedores Docker, lo que facilita la portabilidad y el despliegue en infraestructuras locales como máquinas virtuales, cumpliendo con los requisitos de seguridad de entornos industriales.

5.3.2.1. Diseño funcional

En este apartado se describen las funcionalidades principales de la plataforma IIoT y se detallan los casos de uso que representan las interacciones entre los usuarios, dispositivos y el sistema.

5.3.2.1.1. Casos de uso

El siguiente diagrama UML proporciona una representación visual de los casos de uso basados en las funcionalidades descritas, destacando las interacciones entre los usuarios, dispositivos y el servidor. Cada caso de uso está etiquetado con las acciones principales y las relaciones de inclusión y extensión entre ellos.

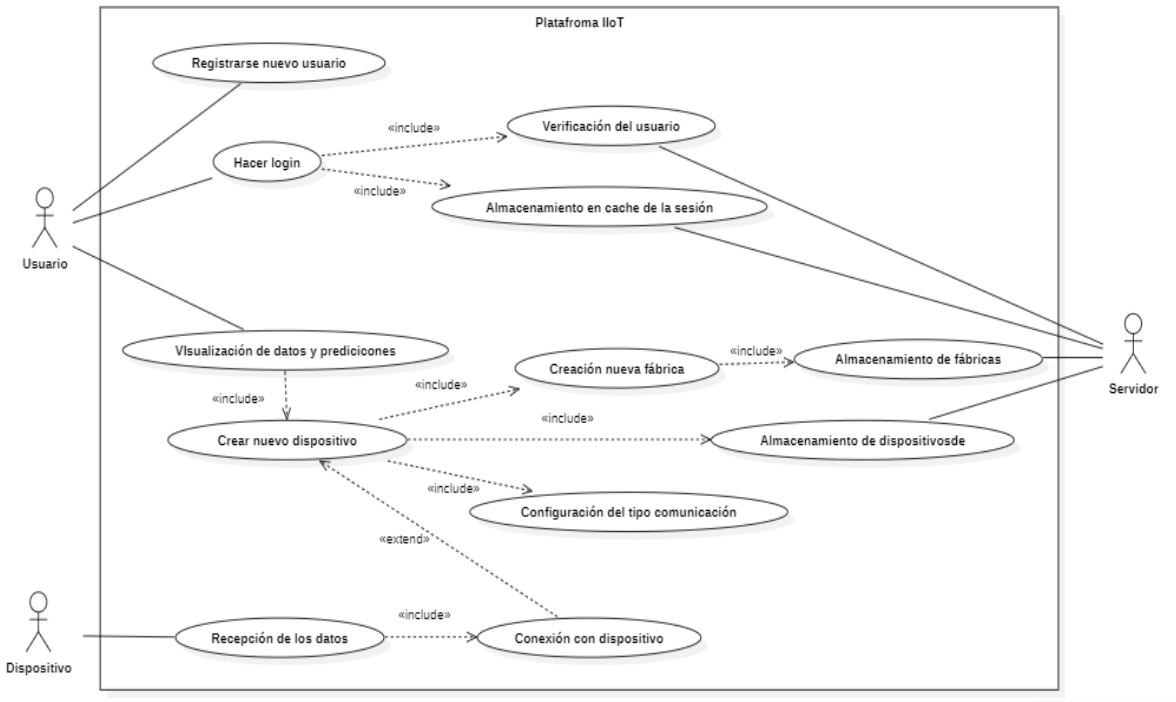

Ilustración 22 Diagrama UML de casos de uso de la plataforma

Registrar nuevo usuario

El usuario es el actor que completa un formulario de registro para crear una nueva cuenta en la plataforma. Este caso de uso incluye la verificación del usuario y el almacenamiento de la información de registro en la base de datos.

Hacer login

El usuario es el actor que ingresa sus credenciales (nombre de usuario y contraseña) para iniciar sesión en la plataforma, y ésta verifica las credenciales y almacena la sesión en caché.

Visualización de datos y predicciones

El usuario es el actor que accede al dashboard para visualizar los datos históricos y en tiempo real de los dispositivos conectados. También puede ver gráficos de predicciones generados a partir de los datos históricos.

Crear nueva fábrica

El usuario es el actor que completa un formulario para agregar una nueva fábrica a la plataforma incluyendo información como el nombre, ubicación y detalles del gerente. Este caso de uso incluye la creación y almacenamiento de la nueva fábrica en la base de datos.

Crear nuevo dispositivo

El usuario agrega un nuevo dispositivo a la plataforma seleccionando el tipo de dispositivo (MQTT u OPC UA) y configurando los parámetros necesarios para la conexión. Este caso de uso incluye la configuración del tipo de comunicación y la conexión con el dispositivo.

Recepción de los Datos

El dispositivo es el actor que envía datos a la plataforma utilizando el protocolo configurado (MQTT u OPC UA). La plataforma recibe y almacena estos datos en la base de datos.

5.3.2.2. Diseño de la interfaz

La interfaz de usuario de la plataforma IIoT pretender ser intuitiva, accesible y eficiente. El diagrama de navegación de pantallas refleja un flujo de usuario lógico, comenzando desde la bienvenida y registro, pasando por la autenticación, y luego permitiendo el acceso a la funcionalidad principal de la plataforma a través de la página principal. Las opciones para visualizar datos históricos y predicciones, así como la capacidad de agregar nuevas fuentes de datos y fábricas, están claramente accesibles desde la página principal, facilitando la navegación y el uso de la plataforma.

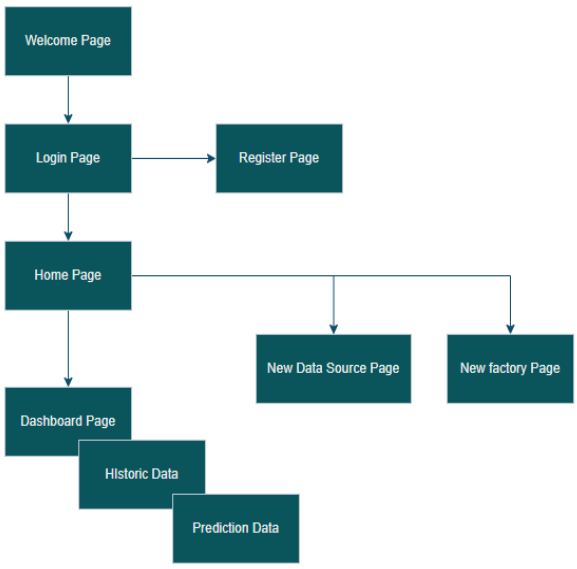

Ilustración 23 Diagrama de bloques d navegación por las páginas

5.3.2.2.1. Funcionalidades principales

La plataforma IIoT proporciona una serie de funcionalidades esenciales para la gestión y visualización de datos industriales. Las principales rutinas de actividad en la plataforma son la autentificación de usuarios, la gestión de fábricas y dispositivos; y la visualización de datos:

5.3.2.2.1.1. *Autenticación de usuarios*

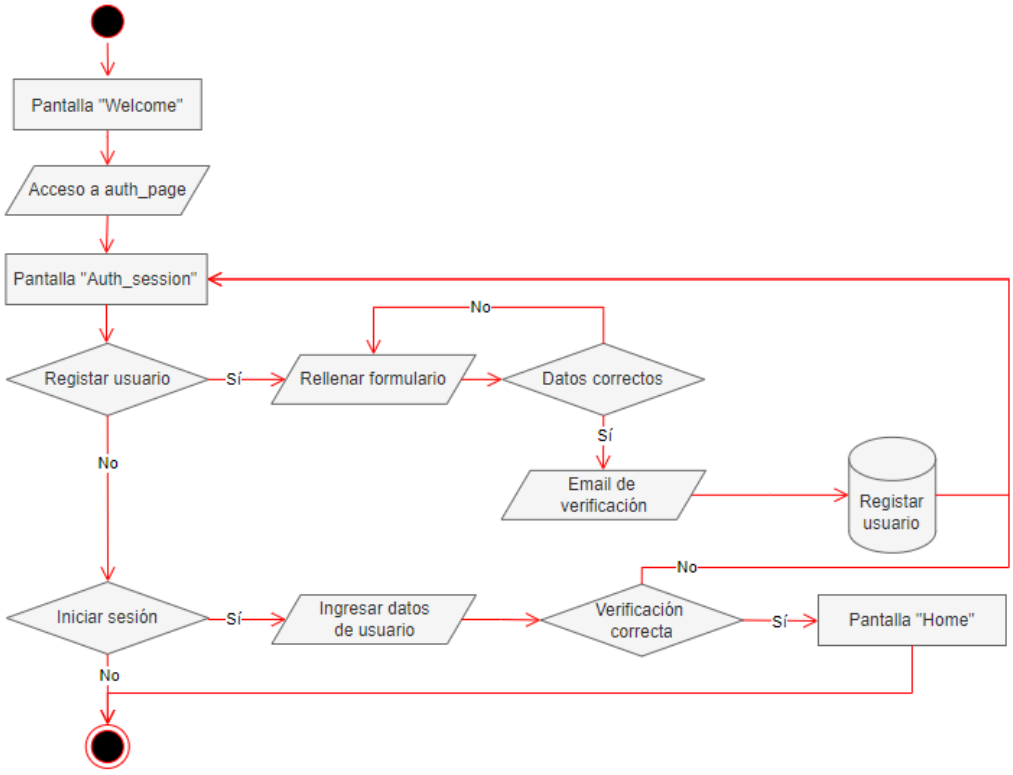

Ilustración 24 Diagrama UML de actividad para autentificación y autorización

Registro de Usuarios: Permite a los nuevos usuarios crear una cuenta en la plataforma.

Inicio de Sesión (Login): Facilita el acceso de los usuarios registrados mediante la verificación de credenciales.

Gestión de Sesiones: Almacenamiento en caché de la sesión del usuario para mantener su estado durante la interacción con la plataforma.

5.3.2.2.1.2. *Gestión de fábricas y fuentes de datos*

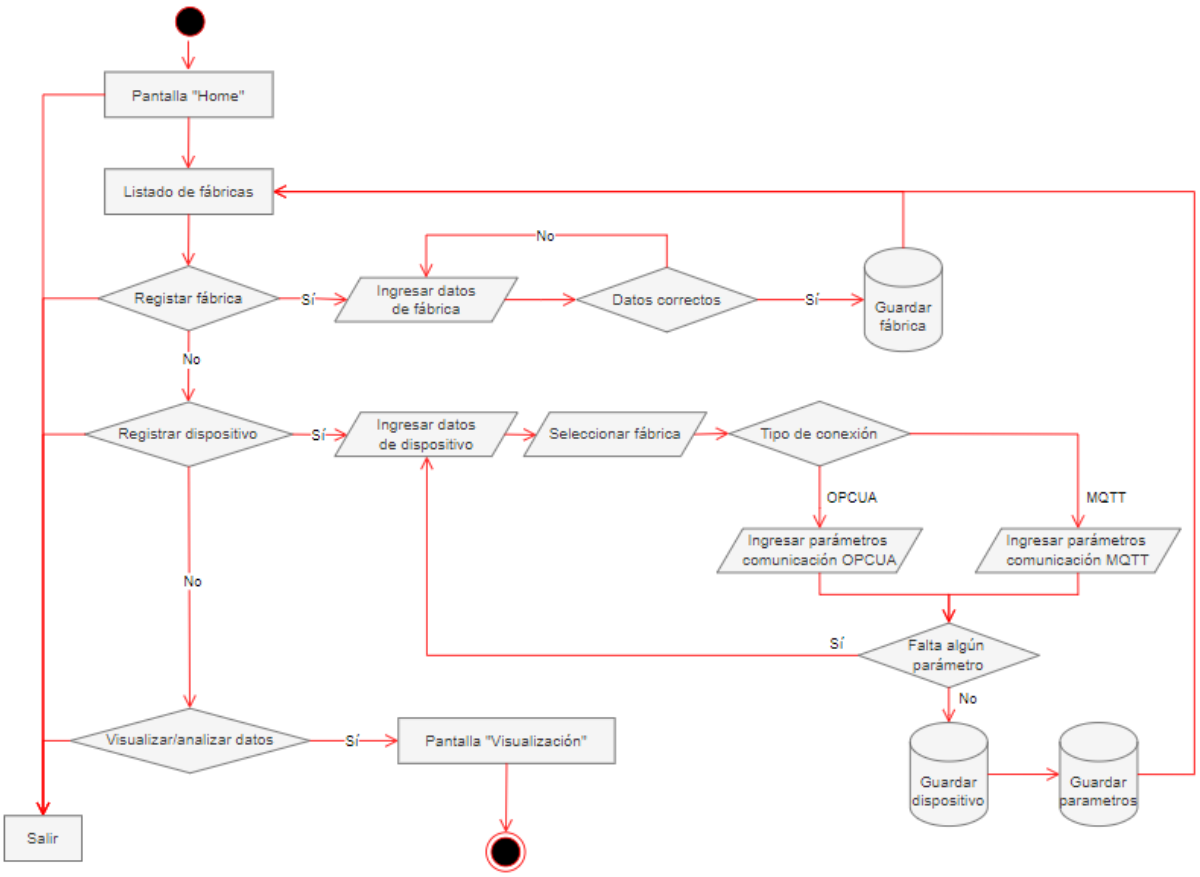

Ilustración 25 Diagrama UML de actividad para gestión de fábricas y fuentes de datos

Creación y administración de fábricas: Permite a los usuarios agregar y configurar nuevas fábricas en la plataforma.

Creación y administración de dispositivos: Facilita la adición y configuración de nuevos dispositivos IoT, especificando los parámetros de conexión (MQTT u OPC UA).

Conexión de dispositivos: Configuración del tipo de comunicación (MQTT u OPC UA) y establecimiento de conexiones con los dispositivos.

Recepción y almacenamiento de datos: Permite la recepción de datos de los dispositivos y su almacenamiento en la base de datos.

5.3.2.2.1.3. *Visualización de datos*

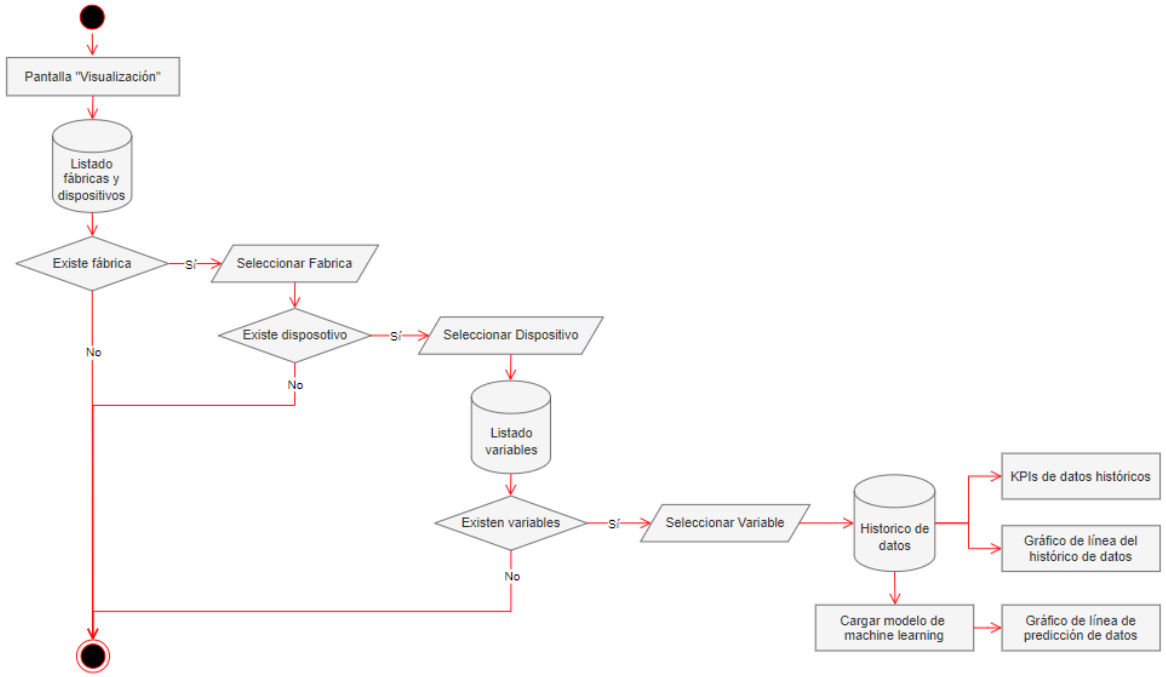

Ilustración 26 Diagrama UML de actividad para visualización de datos

Dashboard de datos: Proporciona una gráfica de históricos y KPIs para visualizar datos históricos y en tiempo real de los dispositivos conectados.

Predicciones: Genera un gráfico de predicciones basadas en datos históricos para ayudar en la toma de decisiones mediante el modelo de machine learning de random forest.

5.3.2.3. Diseño de seguridad

La seguridad es una prioridad en cualquier plataforma IIoT. Se ha centrado en la autenticación, gestionando las sesiones de usuario y los permisos de acceso de manera segura. Las contraseñas de los usuarios van a ser hasheadas utilizando AES-256 mediante las librerías bcrypt y passlib para asegurar su almacenamiento seguro.

5.3.3. Arquitectura del dato

Como se mencionaba al comienzo del diseño, las comunicaciones se van a añadir mediante módulos de programación independientes para obtener una plataforma escalable que permita añadir en el futuro nuevos tipos de conexiones.

El modelo de capas de un proyecto de IoT implica cinco capas desde que se recibe el dato hasta que se visualiza o analiza:

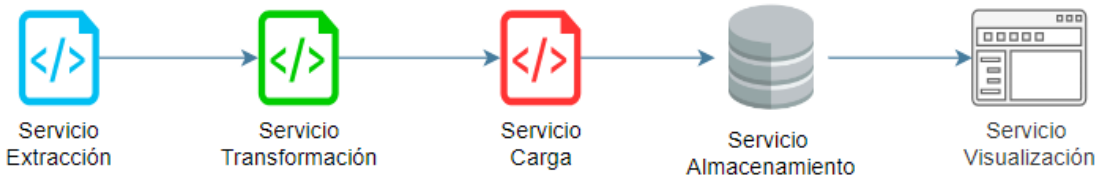

Ilustración 27 Diagrama del diseño por capas del dato en la plataforma

Inicialmente se va a desarrollar un sistema de comunicación con dos protocolos: MQTT y OPC UA. Como se pretende que la plataforma sea escalable y se puedan añadir nuevos protocolos sin necesidad de reprogramar cada uno de los servicios se va a modularizar las capas de extracción y transformación.

Por tanto, el servicio de carga será agnóstico al tipo de conexión y protocolo empleados para recibir los datos del dispositivo. A esta capa va a llegar siempre el dato en formato Json para que sea procesado y almacenado en la base de datos.

5.3.3.1. Módulo MQTT

Para conectar la plataforma a los dispositivos por protocolo de mensajería MQTT se necesita un bróker intermedio que haga de puente entre los clientes.

Para establecer los mensajes es necesario un sistema de Topics ordenado y jerárquico que englobe todas las variables que se quieren monitorizar del dispositivo y que serán recibidas y procesadas por el módulo MQTT para su posterior almacenamiento.

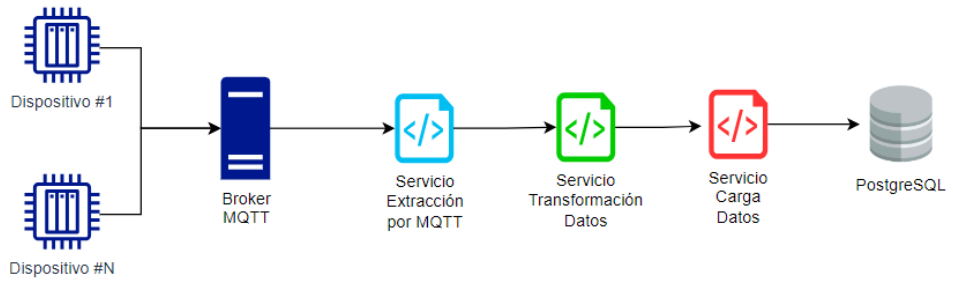

Ilustración 28 Diagrama del módulo de comunicación MQTT

5.3.3.2. Módulo OPC UA

Para conectar la plataforma a los PLC instalados en una red local de empresa será necesario configurar el servidor OPC UA y añadir las variables que requieran ser obtenidas por la plataforma.

El formato de acceso a las variables en un servidor OPC UA es a través de nodos, cada uno identificado por un nodo único que sigue una estructura jerárquica. Cada variable tiene un Identificador de Nodo (NodeId) que permite a la plataforma realizar solicitudes de lectura.

La plataforma empleando el módulo OPC UA realizará una escucha continua de las modificaciones de los valores asociados a cada nodo generando eventos. Cada evento en la plataforma recibirá el dato y será procesado para su posterior almacenamiento.

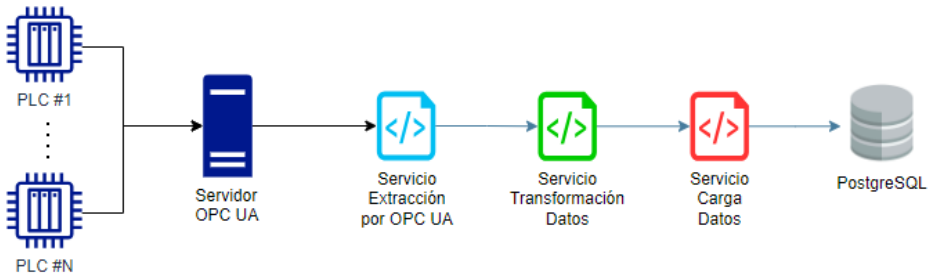

Ilustración 29 Diagrama del módulo de comunicación OPC UA

5.3.3.3. Viaje del dato

Para integrar ambos módulos en la plataforma se unifican los diagramas anteriores y se puede observar como la extracción del dato se realiza de forma paralela convergiendo en la etapa de transformación A partir de ella, se unifican los formatos y se realiza el almacenamiento en base de datos para su posterior lectura por el usuario mediante una interfaz gráfica amigable.

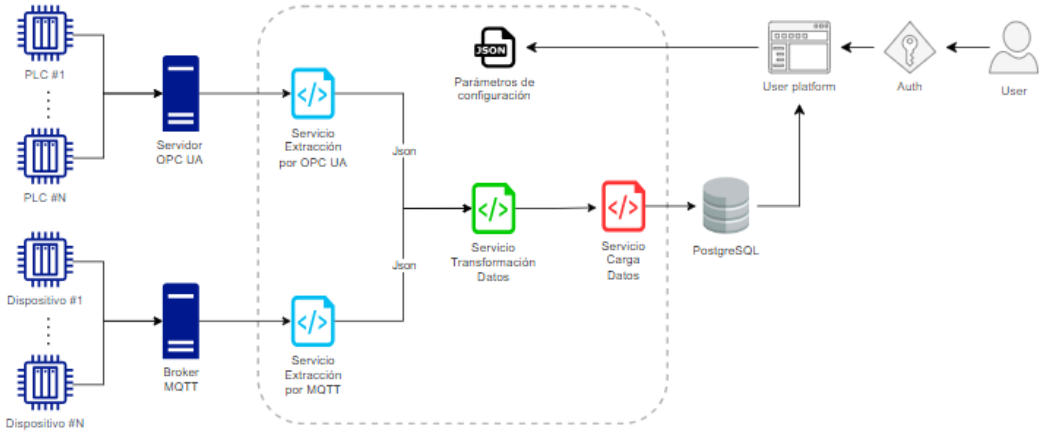

Ilustración 30 Diagrama de integración de módulos de comunicación

5.3.4. Arquitectura de almacenamiento

El diseño de la base de datos de la plataforma se ha estructurado a partir de un modelo entidad-relación (ER), el cual ha sido transformado en un modelo relacional. Este enfoque abarca la identificación detallada de las entidades y sus atributos, la definición precisa de las relaciones entre ellas, y la creación de tablas con sus respectivas claves primarias y tipos de datos.

5.3.4.1. Modelo de entidad-relación

El modelo ER para la base de datos de la plataforma se detalla a continuación, reflejando las entidades principales, sus atributos y las relaciones entre ellas.

La base de datos está diseñada para soportar un sistema que gestiona usuarios, así como la recopilación y almacenamiento de datos de dispositivos en fábricas. Esto permite tanto la administración de acceso como el monitoreo de dispositivos en tiempo real.

Gestión de datos

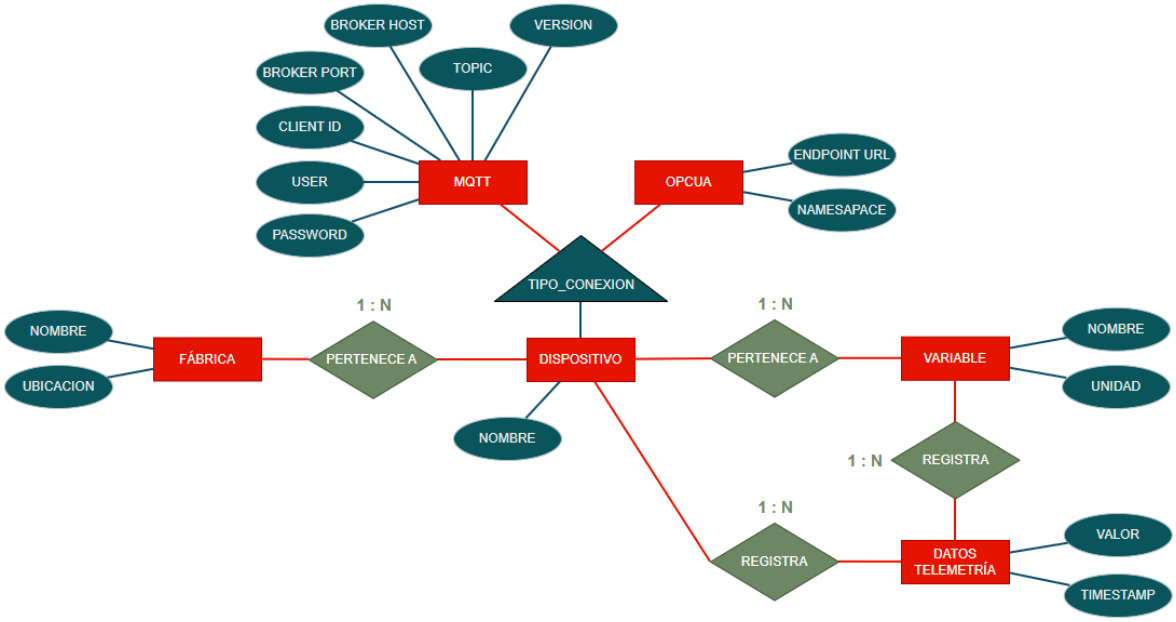

Ilustración 31 Modelo ER de gestión de datos

La entidad **FÁBRICA** es el mayor componente que representa una fábrica que contiene dispositivos e incluye los siguientes atributos:

- id: Clave primaria e identificador único de la fábrica.
- nombre: Nombre de la fábrica.
- ubicacion: Ubicación geográfica de la fábrica.

La entidad **DISPOSITIVO** representa cada dispositivo o fuente de datos dentro de una fábrica y está compuesta por los atributos:

- id: Clave primaria y se utiliza como identificador único del dispositivo.
- nombre: Nombre descriptivo del dispositivo.
- fabrica_id: Referencia a la fábrica a la que pertenece el dispositivo.
- tipo_conexion: Tipo de comunicación utilizado por el dispositivo y puede ser de dos tipos MQTT y OPCUA.

La entidad **VARIABLE** representa una variable medida por un dispositivo y consta de los siguientes atributos:

- id: Clave primaria y actúa como identificador único de la variable.
- nombre: Nombre de la variable.

- unidad: Unidad de medida de la variable.
- dispositivo_id: Referencia al dispositivo que mida la variable.

La entidad **DATOS TELEMETRÍA** almacena datos de telemetría capturados por dispositivos y consta de los siguientes atributos:

- id: Clave primaria y actúa como identificador único del dato.
- dispositivo_id: Referencia al dispositivo que mida la variable.
- variable_id: Referencia a la variable medida.
- timestamp: Marca de tiempo del dato de telemetría.
- valor: Valor medido.

La entidad **PARÁMETROS OPCUA** almacena parámetros de conexión de los dispositivos con tipo_conexion OPCUA y está compuesta por los atributos:

- id: Clave primaria y actúa como identificador único de los parámetros.
- dispositivo_id: Referencia al dispositivo que mida la variable.
- endpoint_url: URL de endpoint OPCUA.
- node_ids: Identificadores de nodos OPCUA.

La entidad **PARÁMETROS MQTT** almacena parámetros de conexión de los dispositivos con tipo_conexion MQTT y está compuesta por los atributos:

- id: Clave primaria y actúa como identificador único de los parámetros.
- dispositivo_id: Referencia al dispositivo que mida la variable.
- bróker_host: Host de conexión al broker MQTT.
- broker_port: Host de conexión al broker MQTT.
- client_id: ID del cliente que se va a utilizar para recibir datos del broker MQTT.
- versión: Versión del protocolo MQTT empleado.
- mqtt_user: Credenciales de usuario para acceso a mensajes.
- mqtt_password: Credenciales de contraseña para acceso a mensajes.

- topic: Canal de comunicación del broker MQTT donde se publican los mensajes.

Las relaciones entre las entidades se estructuran de la siguiente manera: una FÁBRICA puede tener múltiples DISPOSITIVOS representando una relación de uno a muchos. Asimismo, un DISPOSITIVO puede tener múltiples DATOS asociados, lo que también representa una relación de uno a muchos. (1: N)

La relación entre DISPOSITIVO y DATOSTELEMETRIA es de uno a muchos (1: N), lo que significa que cada dispositivo puede generar múltiples datos de telemetría, pero cada dato de telemetría es generado por un único dispositivo. Esta relación se podría etiquetar como "GENERA".

La relación entre VARIABLE y DATOSTELEMETRIA es de uno a muchos (1: N), lo que significa que cada variable puede tener múltiples datos de telemetría asociados, pero cada dato de telemetría está asociado con una única variable. Esta relación se podría etiquetar como "REGISTRA".

La relación entre DISPOSITIVO y PARAMETROSOPCUA es de uno a uno (1:1), lo que significa que cada dispositivo puede tener un único conjunto de parámetros OPCUA y cada conjunto de parámetros OPCUA está asociado con un único dispositivo. Esta relación se podría etiquetar como "CONFIGURA". La relación entre DISPOSITIVO Y PARAMETROSMQTT tiene la misma lógica.

Gestión de usuarios

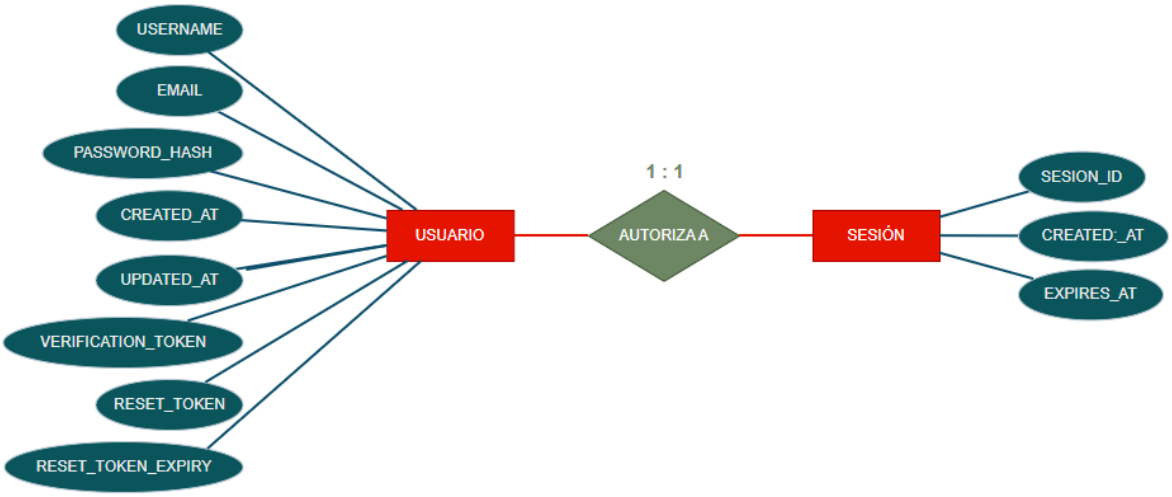

Ilustración 32 Modelo ER de gestión de usuarios

La entidad **USUARIO** representa a un usuario del sistema e incluye los siguientes atributos:

- id: Clave primaria e identificador único del usuario.
- nombre: Nombre de usuario único.
- email: Correo electrónico único del usuario.
- password_hash: Hash de la contraseña establecida por el usuario.
- created_at: Fecha y hora de creación del usuario.
- updated_at: Fecha y hora de la última actualización del usuario.
- email_verified: Booleano que indica el estado de verificación del usuario.
- verifiication_token: Token autogenerado para verificar nuevos usuarios.
- reset_token: Token de expiración para reestablecer la contraseña.
- reset_token_expiry: Fecha de expiración del token de restablecimiento.

La entidad **SESIÓN** representa una sesión activa de un usuario y está compuesta por los atributos:

- id: Clave primaria y se utiliza como identificador único de la sesión.
- user_id: Referencia al usuario asociado a la sesión.

- session_id: Identificador único de sesión utilizado para validar sesiones activas.
- created_at: Fecha y hora de creación de la sesión.
- expires_at: Fecha y hora en que la sesión expirará asegurando el control de tiempo de las sesiones activas.

La relación entre USUARIO y SESIÓN es de uno a uno (1:1), lo que significa que cada usuario autoriza una sola sesión a la vez y cada sesión está asociada con un único usuario. Esta relación se etiqueta como "AUTORIZA".

5.3.4.2. Modelo relacional

El modelo relacional (MR) se detalla a continuación mostrando la transformación del modelo ER en un esquema de tablas relacionales. Los diagramas muestran las tablas principales, cada una con sus atributos identificados previamente, junto con sus respectivos tipos de datos y claves primarias y foráneas.

Gestión de datos

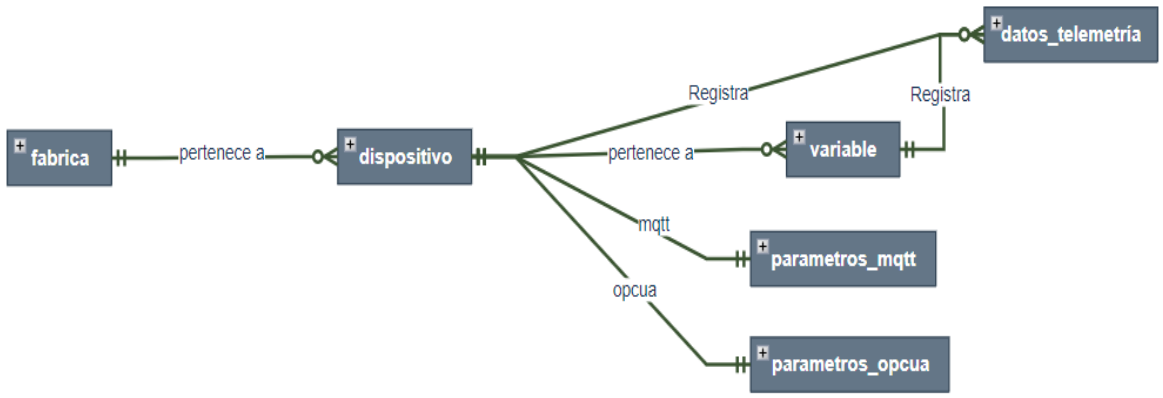

Ilustración 33 Modelo MR alto nivel de gestión de datos

Como se trata de un diagrama complejo que no se puede ver bien en una página se ha segmentado según las relaciones explicadas en el modelo ER. Se puede ver el diagrama completo de bajo nivel en el anexo 1.

Fábrica – Dispositivo

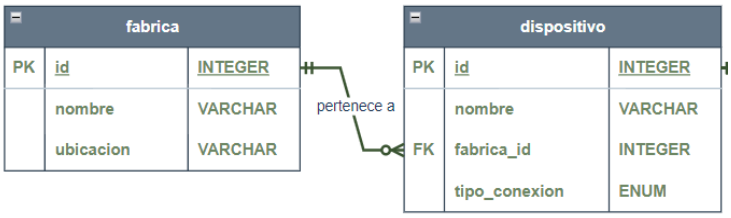

Ilustración 34 Modelo MR bajo nivel fábrica-dispositivo

Dispositivo – Variable

Dispositivo – Parámetros

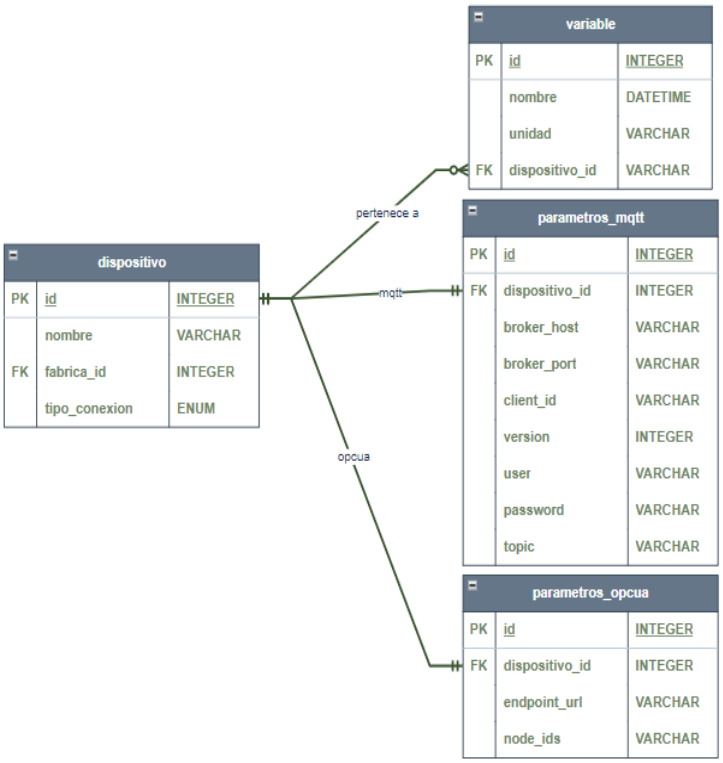

Ilustración 35 Modelo MR bajo nivel dispositivo-variable-parámetros

DatosTelemetría

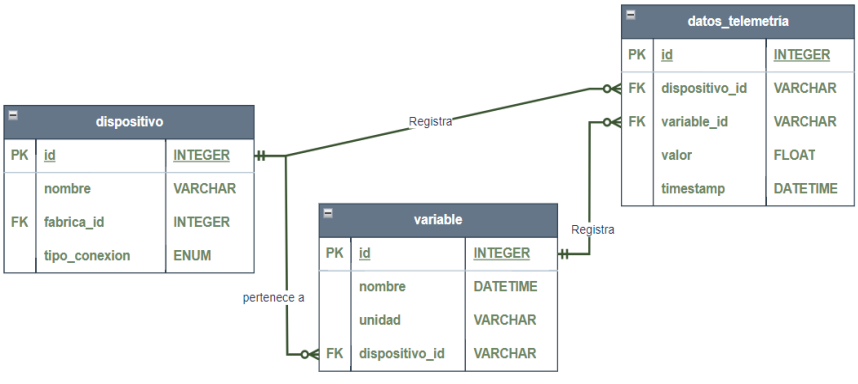

Ilustración 36 Modelo MR bajo nivel dispositivo-variable-datos

Gestión de usuarios

El modelo relacional de la gestión de usuarios incluye dos tablas principales: USER y AUTH_SESSION.

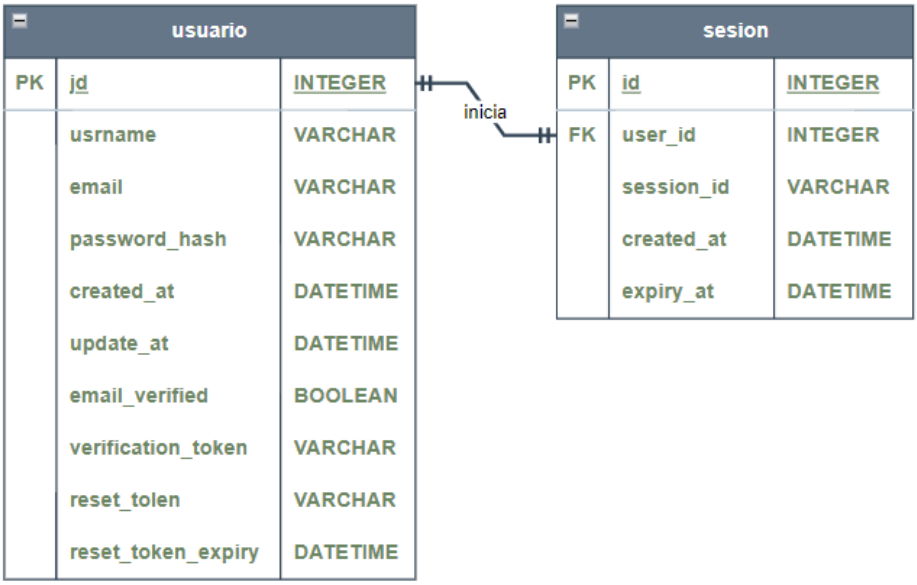

Ilustración 37 Modelo MR de gestión de usuarios

5.3.4.3. Normalización

La base de datos está normalizada hasta la Tercera Forma Normal (3NF). Esto significa que:

1- Se ha minimizado la redundancia de datos.
2- Se ha asegurado la integridad de los datos.
3- Se han eliminado las dependencias parciales y transitivas.

Este alto grado de normalización es adecuado para la mayoría de las aplicaciones, ya que equilibra la eficiencia del almacenamiento con la integridad de los datos.

5.3.1. Arquitectura de predicciones

Al diseño del viaje del dato anterior se le debe añadir el módulo de predicciones que va a realizar modelos de datos que serán entrenados mediante el pool de datos almacenado en la base de datos. El proceso va a consistir en conectarse a los históricos de la base de datos y ejecutar un servicio que entrenará un modelo de regresión lineal para finalmente obtener una gráfica en la plataforma web.

Ilustración 38 Módulo de predicción

Esto debe implementarse junto al resto del proyecto y al integrar en el diseño de comunicación que el esquema final del backend de la plataforma:

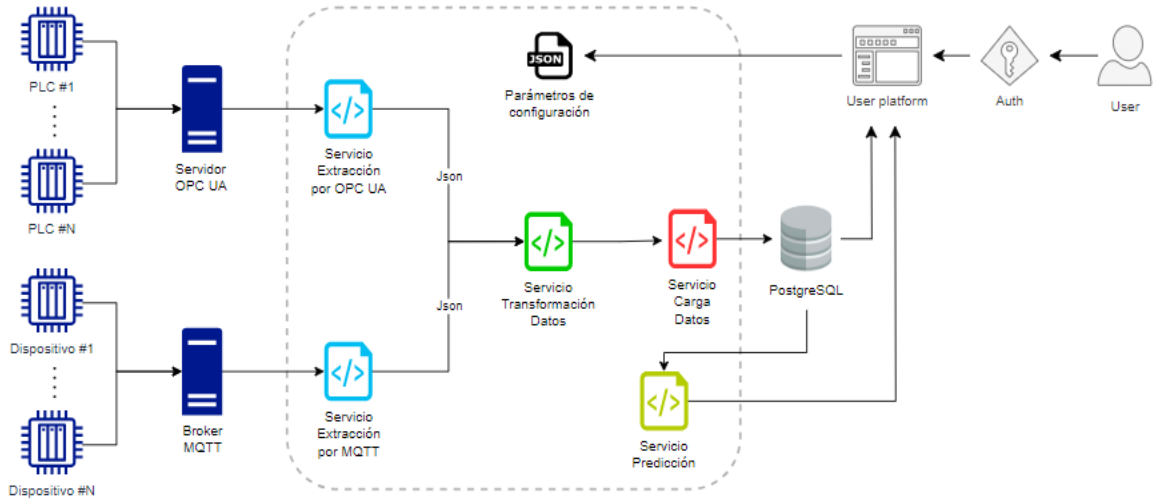

Ilustración 39 Diagrama completo del backend de la plataforma IIoT

Como se puede observar se integra de forma que funciona de forma independiente al proceso de extracción, procesado y carga de datos bebiendo directamente de la base de datos. Esto permite modularizar la plataforma y poder añadir nuevos módulos en paralelo más sofisticados de predicción.

5.4. Implementación del sistema IIoT

5.4.1. Simulación de entornos industriales

5.4.1.1. Dispositivo MQTT

```
MQTT_device/
│
├── config/
│   ├── load_config.json
│   ├── log_system.py
│   ├── mqtt_config.json
│   └── telemetry_config.json
│
├── Logs/
│   └── aaaa/
│       └── mm/
│           └── dd.log
│
├── mqtt_client_publish.py
├── requirements.txt
└── README.md
```

Se ha ejecutado en una Raspberry Pi un proyecto en Python llamado “MQTT_device” para simular la telemetría de un dispositivo real, generando valores aleatorios con formatos y rangos preestablecidos. Su estructura de archivos es la siguiente:

El esquema del proyecto se basa en un archivo principal “mqtt_client_publish.py” que se explica a continuación mediante un diagrama UML de actividad y se nutre de los archivos contenidos en el subdirectorio “config” para su configuración.

A continuación, se detalla el funcionamiento del dispositivo mediante un diagrama UML de actividad:

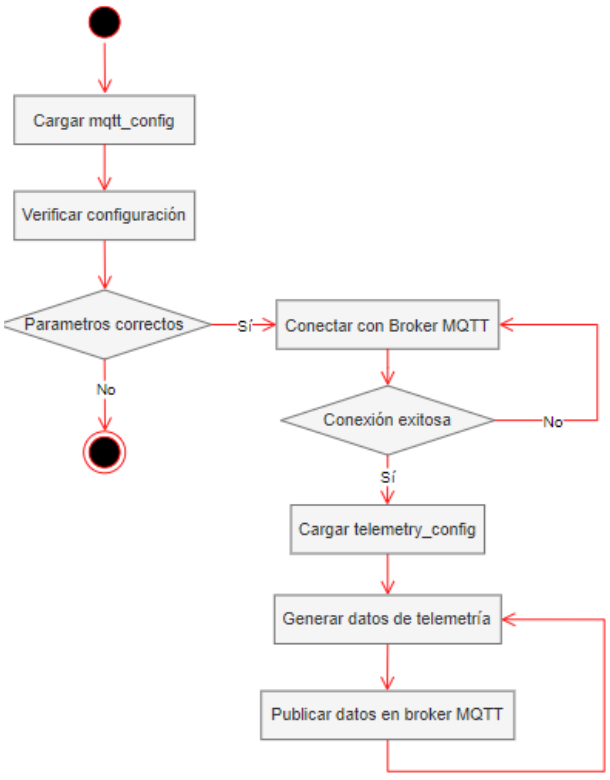

Ilustración 40 Diagrama UML de actividad de mqtt_client_publish

Se ha optimizado un sistema de logs para reportar información completa del dispositivo y se verifica automáticamente la presencia de todos los parámetros necesarios, notificando cualquier ausencia en el subdirectorio “Logs” ordenado por subpaths en jerarquía temporal aaaa/mm/dd.log.

Se ha implementado un sistema try/except de reintento en caso de fallo de conexión para asegurar la fiabilidad del envío de datos.

Además, se han añadido un archivo “requirements.txt” con las librerías necesarias para su correcto funcionamiento y un “README.md” con la información básica del proyecto.

Está diseñado para ser configurable tanto en el formato y variables del payload (permitiendo múltiples niveles en el JSON) como en los parámetros de conexión al broker MQTT.

MQTT_config.json

```json
{
  "hostname": "0a4d3c9443f740958ffe23276066bb09.s1.eu.hivemq.cloud",
  "port": 8883,
  "topic": "industria/telemetria",
  "username": "erevilla",
  "password": "tfg",
  "interval": 5
}
```

Telemetry_config.json

```json
{
    "estado_dispositivo": {
        "type": "choice",
        "values": ["operativo", "apagado", "error"]
    },
    "temperatura": {
        "type": "range",
        "min": 20.0,
        "max": 100.0
    },
    "humedad": {
        "type": "range",
        "min": 30.0,
        "max": 90.0
    },
    "consumo_energia": {
        "type": "object",
        "fields": {
            "voltaje": {
                "type": "range",
                "min": 220.0,
                "max": 240.0
            },
            "corriente": {
                "type": "range",
                "min": 0.5,
                "max": 10.0
            },
            "potencia": {
                "type": "range",
                "min": 100.0,
                "max": 2000.0
            }
        }
    }
}
```

5.4.1.2. Broker MQTT

El bróker MQTT empleado es HiveMQ en su formato serverless que emplea AWS, por el cual a través de una suscripción gratuita se ha generado un bróker al que se puede acceder mediante conexión a Internet. Esto no es lo ideal en entornos industriales ya que debería alojarse todos los recursos y herramientas en el entorno local, pero servía como demostración para la conexión sencilla con la plataforma.

La configuración de dominio, puertos, etc; viene predefinida y se ha configurado con TLS para securizar la transmisión de los datos con un certificado de CA que ofrece el propio servidor a través del puerto 8883.

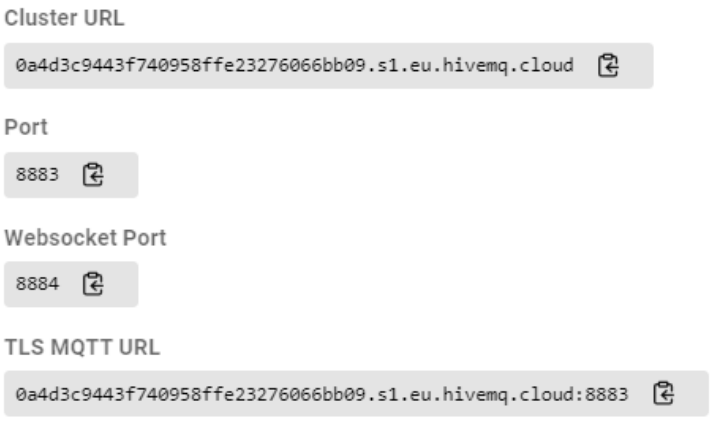

Ilustración 41 Información básica del Broker MQTT

Adicionalmente, se han configurado credenciales de usuario y contraseña obligatorios para conectarse al bróker con dos roles distintos, el primero de ellos “erevilla” para publicar por el dispositivo, y un segundo “factoryhub” para suscribirse desde la plataforma.

Tabla 8 Configuración de los usuarios y accesos al Broker MQTT

Username	Permission type	Actions
erevilla	Publish Only	DELETE
factoryhub	Subscribe Only	DELETE

5.4.1.3. Servidor OPC UA

```
OCPUA_server/
├── config/
│   ├── load_config.json
│   ├── log_system.py
│   ├── opcua_config.json
│   └── telemetry_config.json
├── Logs/
│   └── aaaa/
│       └── mm/
│           └── dd.log
├── opcua_server.py
├── requirements.txt
└── README.md
```

Se ha ejecutado en una Raspberry Pi un proyecto en Python llamado “OPCUA_server” para simular el servidor OPC UA instalado para uno o varios PLCs que estén conectado a una red profinet/profibus y de los cuales se necesite extraer datos de la producción, consumos o consignas. Para ello, se ha generado un pool de datos variado que ejemplifique la situación.

El esquema del proyecto se basa en un archivo principal “opcua_server.py” que se explica a continuación mediante un diagrama UML de actividad y se nutre de los archivos contenidos en el subdirectorio “config” para su configuración.

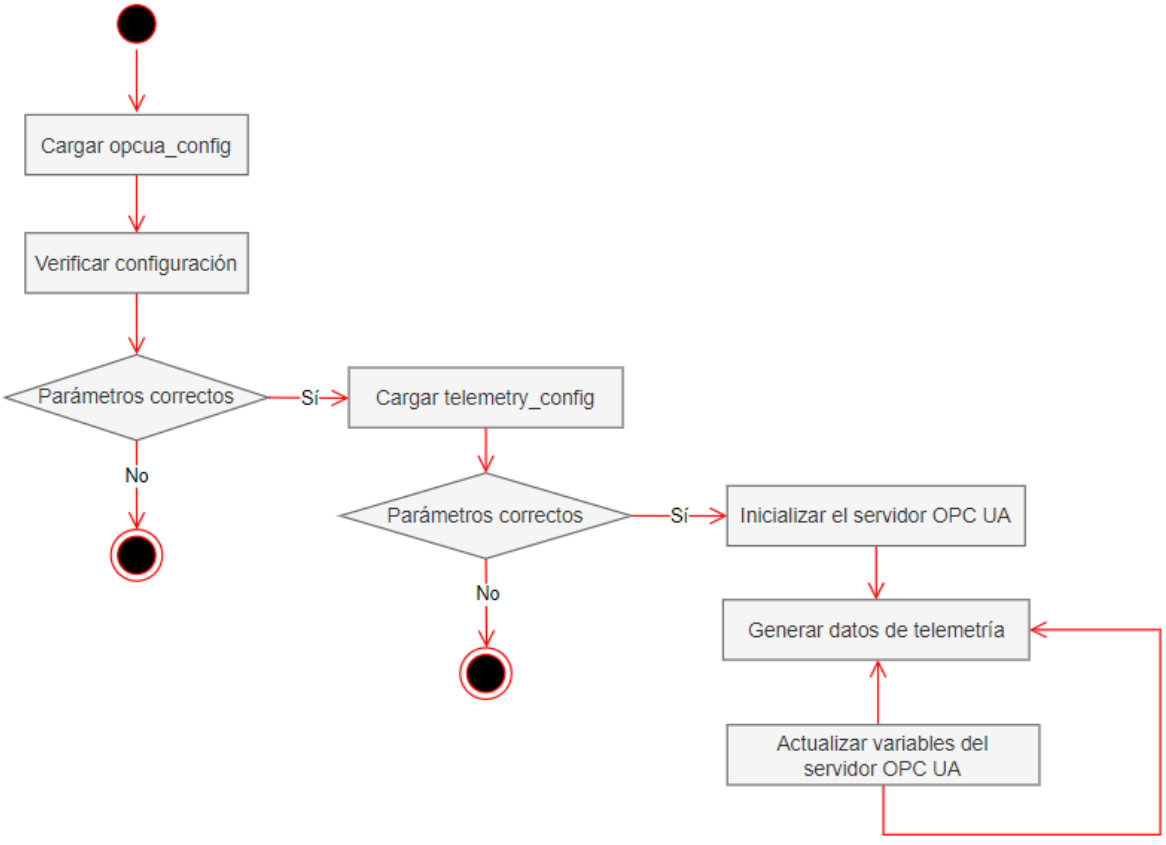

Ilustración 42 Diagrama UML de actividad de OPCUA_server

Se ha optimizado un sistema de logs para reportar información completa del dispositivo y se verifica automáticamente la presencia de todos los parámetros necesarios, notificando cualquier ausencia en el subdirectorio “Logs” ordenado por subpaths en jerarquía temporal aaaa/mm/dd.log.

Se ha implementado un sistema try/except de reintento en caso de fallo de conexión para asegurar la fiabilidad del envío de datos.

Además, se han añadido un archivo “requirements.txt” con las librerías necesarias para su correcto funcionamiento y un “README.md” con la información básica del proyecto.

Está diseñado para ser configurable tanto en el formato y tipos de variables, accediendo a ellas mediante namespace y nodos, como en los parámetros de conexión al broker MQTT.

OPCUA_config.json

```
{
    "endpoint": "opc.tcp://localhost:4848",
    "server_name": "OPC UA Simulated Server",
    "read_frequency": 15,
    "telemetry_config": "telemetry_config.json"
}
```

Telemetry_config.json

```
{
    "ArcTemperature": {
        "type": "range",
        "min": 1500.0,
        "max": 3000.0
    },
    "WeldingCurrent": {
        "type": "range",
        "min": 50.0,
        "max": 500.0
    },
    "WireSpeed": {
        "type": "range",
        "min": 0.0,
        "max": 20.0
    },
    "ShieldingGasFlowRate": {
        "type": "range",
        "min": 10.0,
        "max": 50.0
    },
    "Voltage": {
        "type": "range",
        "min": 10.0,
        "max": 30.0
    },
    "OperationalStatus": {
        "type": "choice",
        "values": ["idle", "welding", "error"]
    },
    "IsOperational": {
        "type": "bool"
    },
    "LastMaintenanceDate": {
        "type": "datetime"
    },
    "RobotID": {
        "type": "string"
    }
}
```

5.4.2. Plataforma IIoT

La implementación de la plataforma se ha realizado de forma escalona, comenzando por la base de datos y terminando por las visualizaciones del frontend. A continuación, se detallan los módulos por orden de desarrollo.

5.4.2.1. Base de datos

La base de datos actúa como pieza central donde la web y el servicio de predicciones interactúan con los datos.

Una vez diseñados los modelos Entidad-Relación y Relacional, se ha realizado el despliegue de la base de datos. Reflex trabaja conjuntamente con las librerías SQLModel y SQLAlchemy de Python para permitir el uso de modelos con migración directa a la base de datos para que pueda ser programada desde el código. Para aprovechar esta funcionalidad se ha añadido un archivo “models.py” que se muestra en el código del anexo.

Para desplegar la base de datos PostgreSQL en un contendor Docker se ha utilizado la imagen oficial con la versión 12.2, que es la última versión estable (pt3.3 del anexo).

Tras la migración del modelo a la base de datos se genera automáticamente la tabla “alembic_version” para manejar las migraciones del proyecto.

Para acceder, hacer la gestión y monitorizar la base de datos durante el desarrollo se ha empleado el software DBeaver.

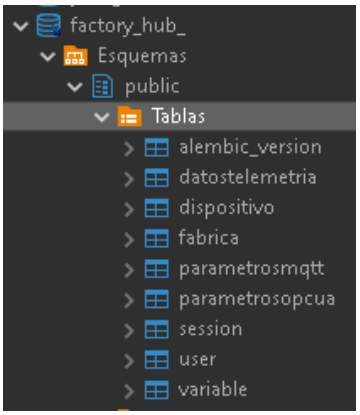

Ilustración 43 Tablas de base de datos

5.4.2.2. Estructura de archivos

Reflex proporciona un inicializador para crear proyectos que establece una estructura básica de carpetas y archivos, a partir de la cual se puede comenzar a desarrollar. La estructura de archivos de este proyecto sigue la metodología de Reflex, incluyendo una carpeta "assets" para recursos estáticos como imágenes, una carpeta principal del proyecto llamada "factory_hub" que contiene toda la lógica de programación, y un archivo "rxconfig.py" que configura las conexiones externas, como la base de datos. A partir de esta base, se ha desarrollado una estructura optimizada para la organización y jerarquía de cada componente de la plataforma.

La carpeta "factory_hub" está organizada en las siguientes subcarpetas principales:

1. Backend: Contiene la lógica central de la plataforma, dividida en cuatro subcarpetas:

2. Models: Alberga el archivo models.py, que define los modelos de la base de datos.
3. States: Incluye las subcarpetas auth_states y protected_states, que manejan los estados de las distintas páginas.
4. Protocols: Gestiona la extracción de datos para cada tipo de comunicación, con carpetas separadas para mqtt_module y opcua_module.
5. Predictions: Contiene los archivos para cada modelo de predicción. Se han desarrollado dos modelos: regresión lineal y random forest, aunque finalmente se utiliza solo el segundo por su mayor eficiencia.
6. Components: Almacena los componentes reutilizables de la interfaz de usuario.
7. Views: Contiene las implementaciones de las vistas y la lógica de presentación.
8. Pages: Alberga las definiciones de cada página de la plataforma.

Además, la carpeta "factory_hub" incluye los archivos init.py, app_routes.py y factory_hub.py, que son esenciales para la configuración y el enrutamiento de la aplicación.

5.4.2.3. Backend

Los servicios de extracción de datos de MQTT y OPC UA realizan las conexiones a los distintos dispositivos de forma asíncrona y recogen los valores actuales para que después sean procesado y almacenados. Estos se comunican con el backend de la plataforma que recibe los datos almacenados y con ellos crea las visualizaciones. Además, integra la lógica de cada página mediante estados.

5.4.2.3.1. Módulo de extracción MQTT

De forma muy similar al cliente del dispositivo de publicación de telemetría, se ha desarrollado un módulo llamado “MQTT_module” que se suscribe al bróker para escuchar todos los topics configurados para cada dispositivo.

El cliente es una clase “MQTTConnectionState” que se utiliza en paralelo asíncronamente para cada dispositivo configurado en la plataforma. Esto ofrece la posibilidad de múltiples dispositivos conectados simultáneamente de forma eficiente.

A continuación, se detalla la estructura de la clase mediante un diagrama UML de clases y su funcionamiento mediante un diagrama de secuencia.

Diagrama de clases (Estático)

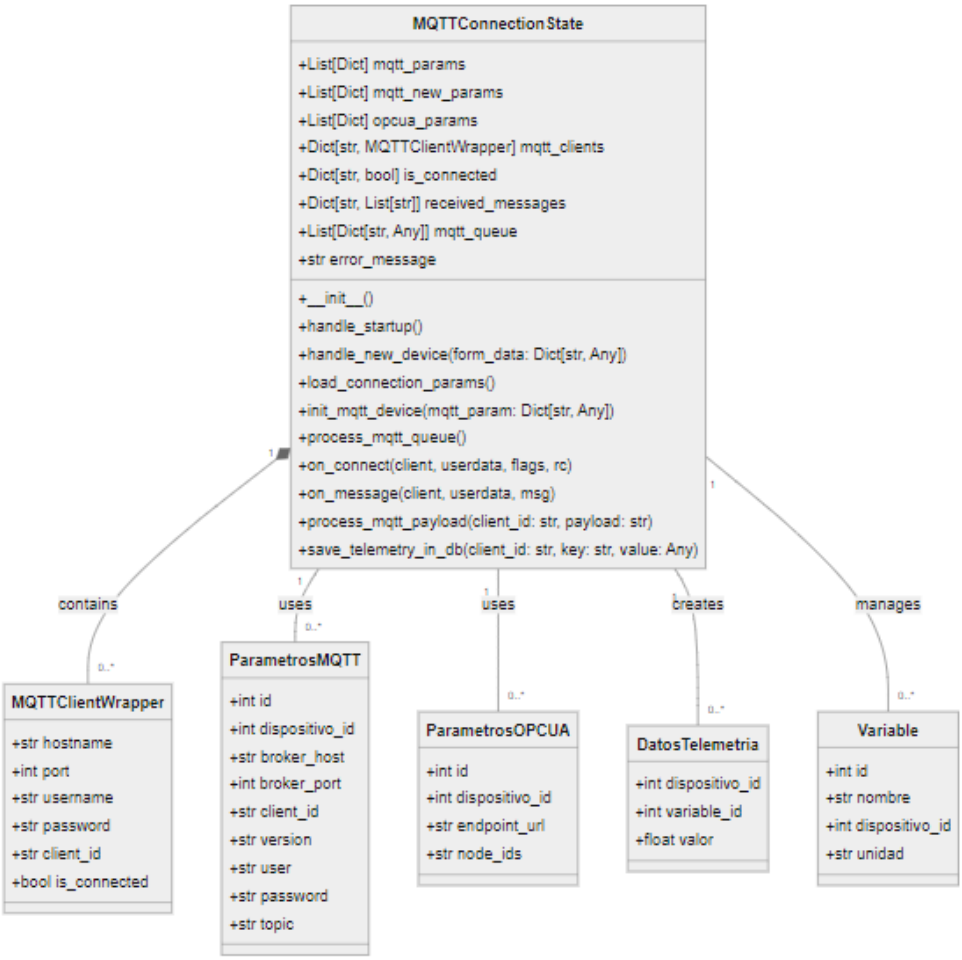

Ilustración 44 Diagrama UML de clases del módulo MQTT

Diagrama de secuencia (Dinámico)

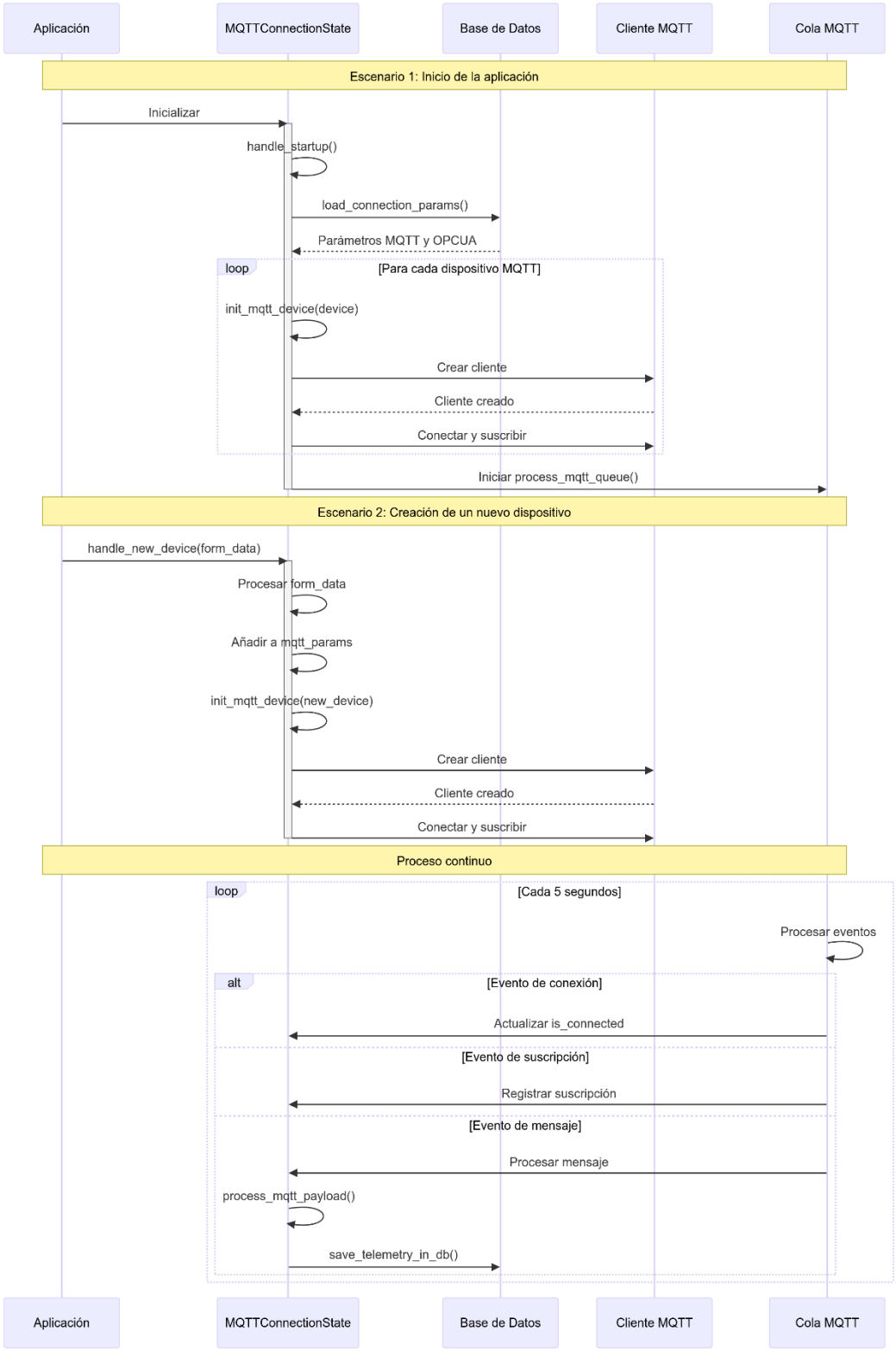

Ilustración 45 Diagrama UML de secuencia del módulo MQTT

5.4.2.3.1. Módulo de extracción OPC UA

El módulo OPCUA consta de dos clases OPCUAConnectionState, que maneja la conexión individual de cada dispositivo, y OPCUAConnectionManager, que actúa como gestor central para todas las conexiones. A través de un diccionario se registran las conexiones activas y se pueden añadir los nuevos dispositivos mediante los métodos implementados dentro del manejador de conexiones.

A continuación, se detalla la estructura de la clase mediante un diagrama UML de clases y su funcionamiento mediante un diagrama de secuencia:

A continuación, se detalla la estructura de la clase mediante un diagrama UML de clases y su funcionamiento mediante un diagrama de secuencia:

Diagrama de clases

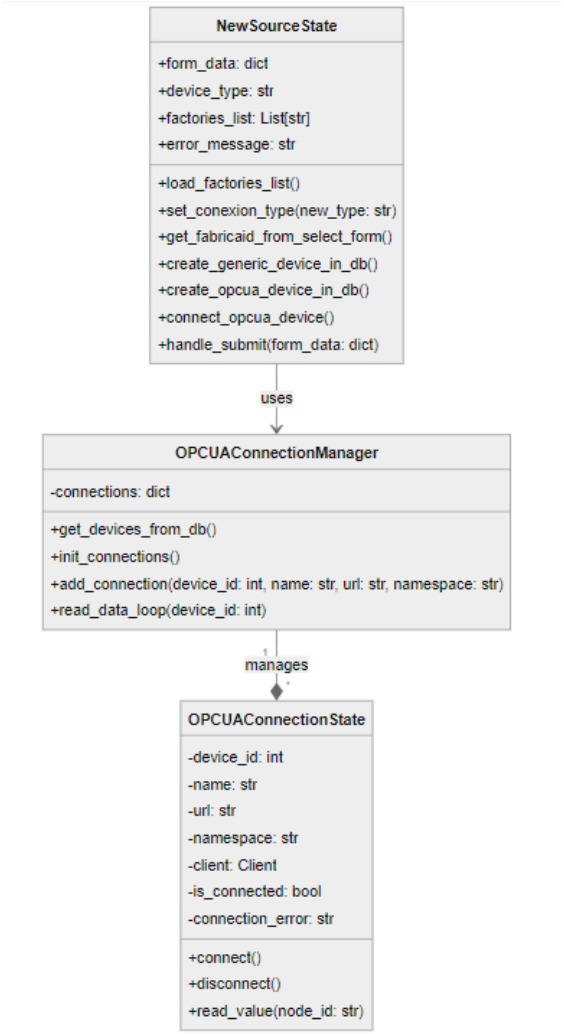

Ilustración 46 Modulo OPCUA diagrama UML de clases

Diagrama de secuencia

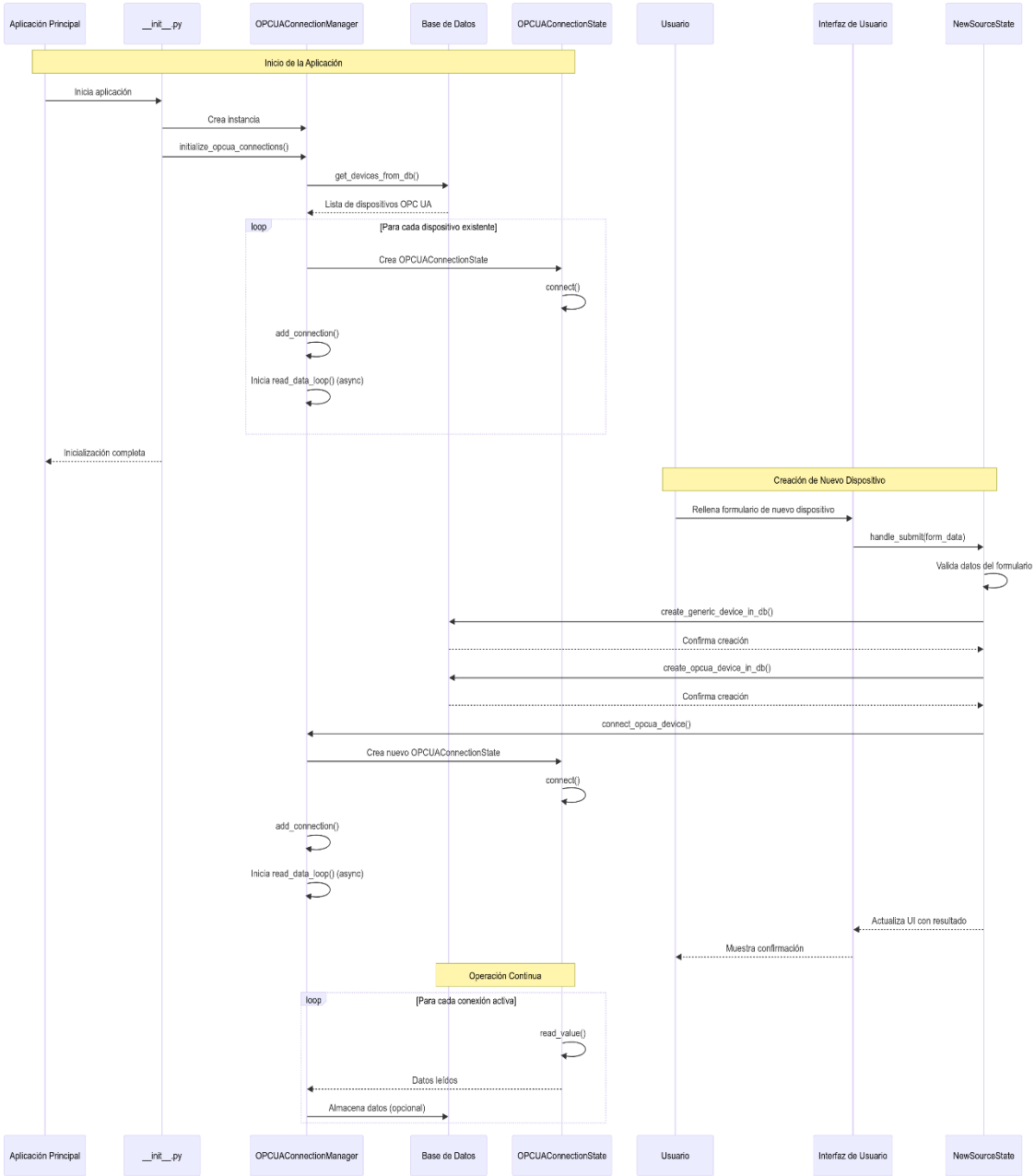

Ilustración 47 Modulo OPCUA diagrama UML de secuencia

5.4.2.3.2. Módulo de predicciones

Se ha desarrollado un módulo de predicción de datos que realiza análisis mediante Random Forest Regression utilizando la librería Scikit-learn de Python. Este módulo se conecta a la base de datos para recoger los valores almacenados en la tabla de históricos, filtrando por dispositivo y variable, lo que permite entrenar el modelo y optimizar las respuestas de forma constante.

Al igual que los módulos anteriores se ha estructurado como un módulo independiente que pueda ser mejorando en el futuro o sustituido por otros modelos más completos. El script de ejecución está segmentado en pequeñas tareas a través de funciones, cuya secuencia se puede analizar mediante el siguiente diagrama UML de actividad.

Diagrama de clase

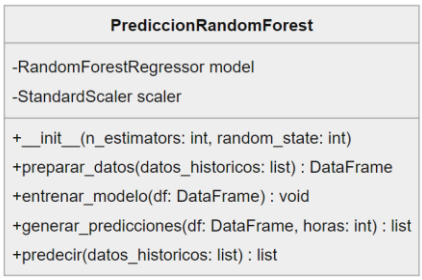

Ilustración 48 Diagrama UML de clases del módulo de predicción

Diagrama de secuencia

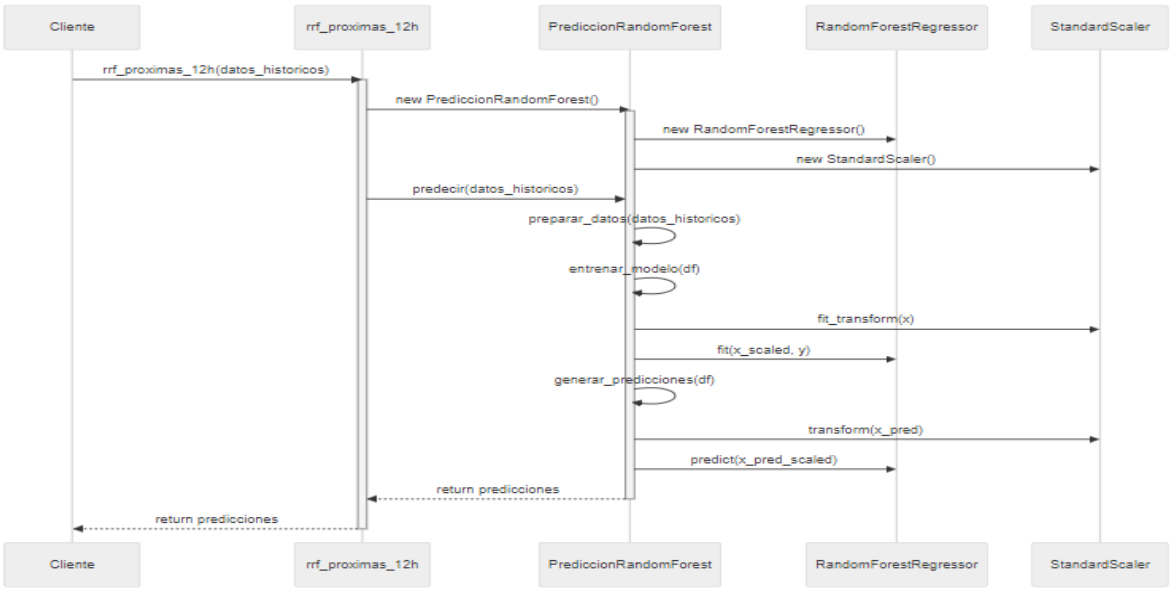

Ilustración 49 Diagrama UML de clases del módulo de predicción

5.4.2.4. Frontend y estados

Al tratarse un Singleton, se puede acceder a la plataforma mediante un único enlace que combina distintos paths para llamar a las rutas específicas para cada página. Las páginas en Reflex funcionan por componentes que se combinan y a las que se les asocian métodos de estado. Por ello, cada página generalmente lleva asociado un estado que aplique la lógica.

Como Reflex trabaja con componentes, se ha decidido utilizar diagramas de componentes para explicar la visualización y de clases, secuencias y mapas de estados dependiendo del objetivo y la complejidad, para los estados.

5.4.2.4.1. Welcome

Ilustración 50 Welcome page

Es la primera pantalla que ve el cliente al acceder a la plataforma. Desde ella, se accede a la página de autentificación de usuarios. Esta página se encuentra en la ruta "/" y se puede acceder a ella desde cualquier navegador dentro de la red local. Está compuesta por una imagen de fondo y un botón central que al clicar redirige a la página Auth_Session.

Esta página no tiene un estado propio, ya que su lógica es mínima y solo requiere la presentación visual y la redirección mediante un botón.

La página se organiza en un contenedor principal que incluye dos elementos clave:

- welcome_logo: utiliza una imagen del logo que se almacena en assets.

- welcome_button, que redirige a la ruta de autenticación definida en AUTH_ROUTE al ser clica que redirige al usuario a la página de autentificación de usuarios.

Diagrama de componentes

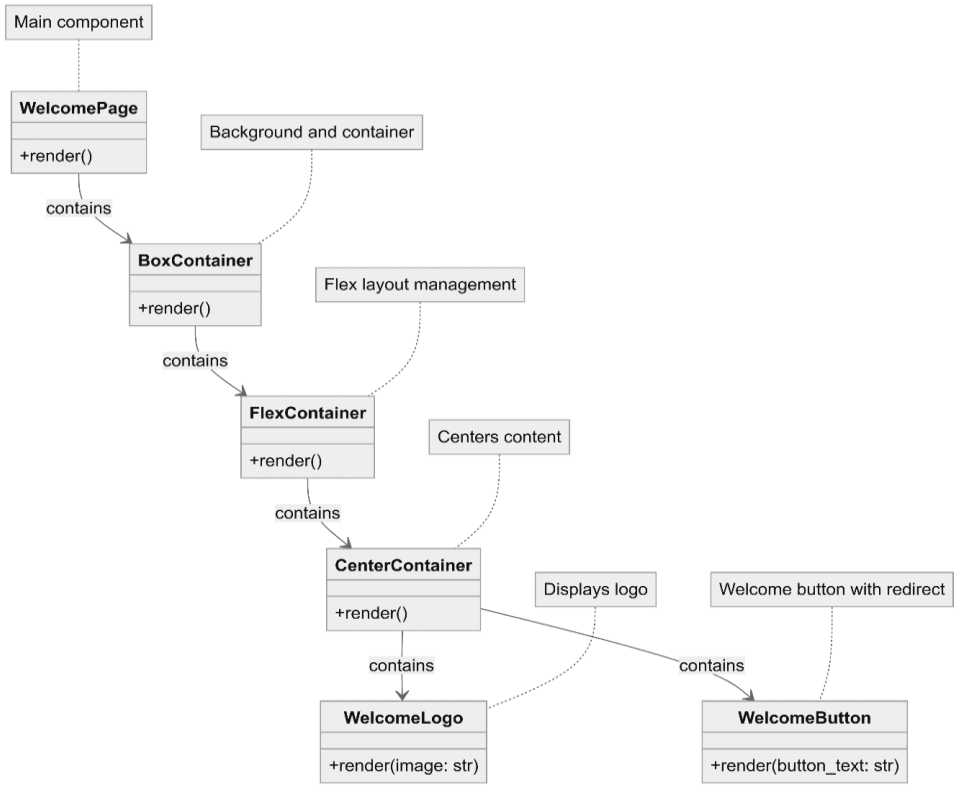

Ilustración 51 Diagrama de componentes de Welcome page

Diagrama de secuencia de acceso a auth page

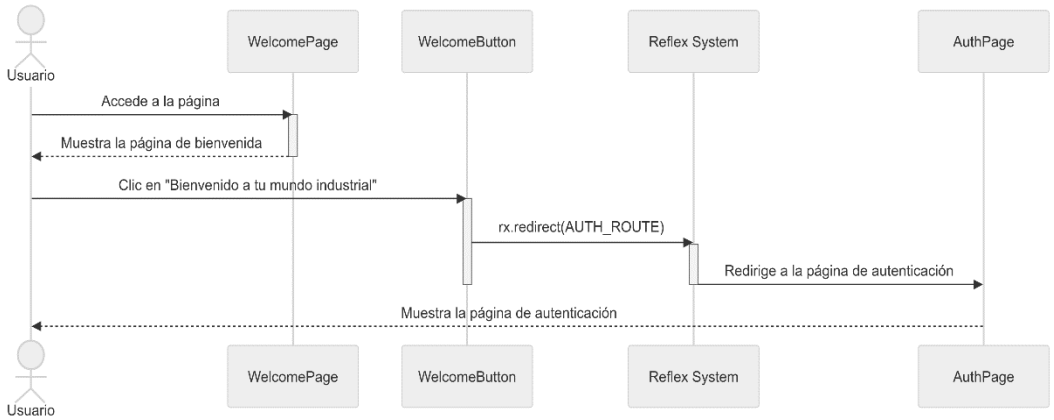

Ilustración 52 Diagrama de secuencias de Welcome page

5.4.2.4.1. Authentification

La página de autenticación permite a los usuarios acceder a la plataforma mediante el inicio de sesión o el registro de nuevos usuarios. Esta página se encuentra en la ruta /auth_session y es accesible desde la página anterior. Sigue su mismo formato y está compuesta por un panel central con dos secciones principales: una para el inicio de sesión (login) y otra para el registro de nuevos usuarios. Los usuarios pueden alternar fácilmente entre estas dos secciones utilizando los tableros disponibles.

Ilustración 53 Auth page con login tab

Ilustración 54 Register page con regsiter tab

En esta página, los clientes pueden completar formularios para registrar nuevos usuarios o autenticarse si ya están registrados. Tras completar el formulario de registro, la información del nuevo usuario se almacena en la base de datos. Por otro lado, al iniciar sesión, las credenciales proporcionadas se verifican contra la base de datos para autenticar al usuario.

Una vez que la verificación es exitosa, se crea una sesión para el usuario y se le redirige al Home de la plataforma, donde tiene acceso completo a todas las páginas y funcionalidades. Tanto el proceso de registro como el de inicio de sesión trabajan con la base de datos para registrar y verificar usuarios y sesiones activas.

Adicionalmente, para la verificación de nuevos usuarios, se ha implementado un servidor SMTP con la herramienta Brevo, que permite el envío correos electrónicos de confirmación con un enlace para activar cada cuenta del usuario. Este proceso garantiza que solo los usuarios verificados puedan acceder a la plataforma y asegura una gestión adecuada de las sesiones activas.

Diagrama de componentes

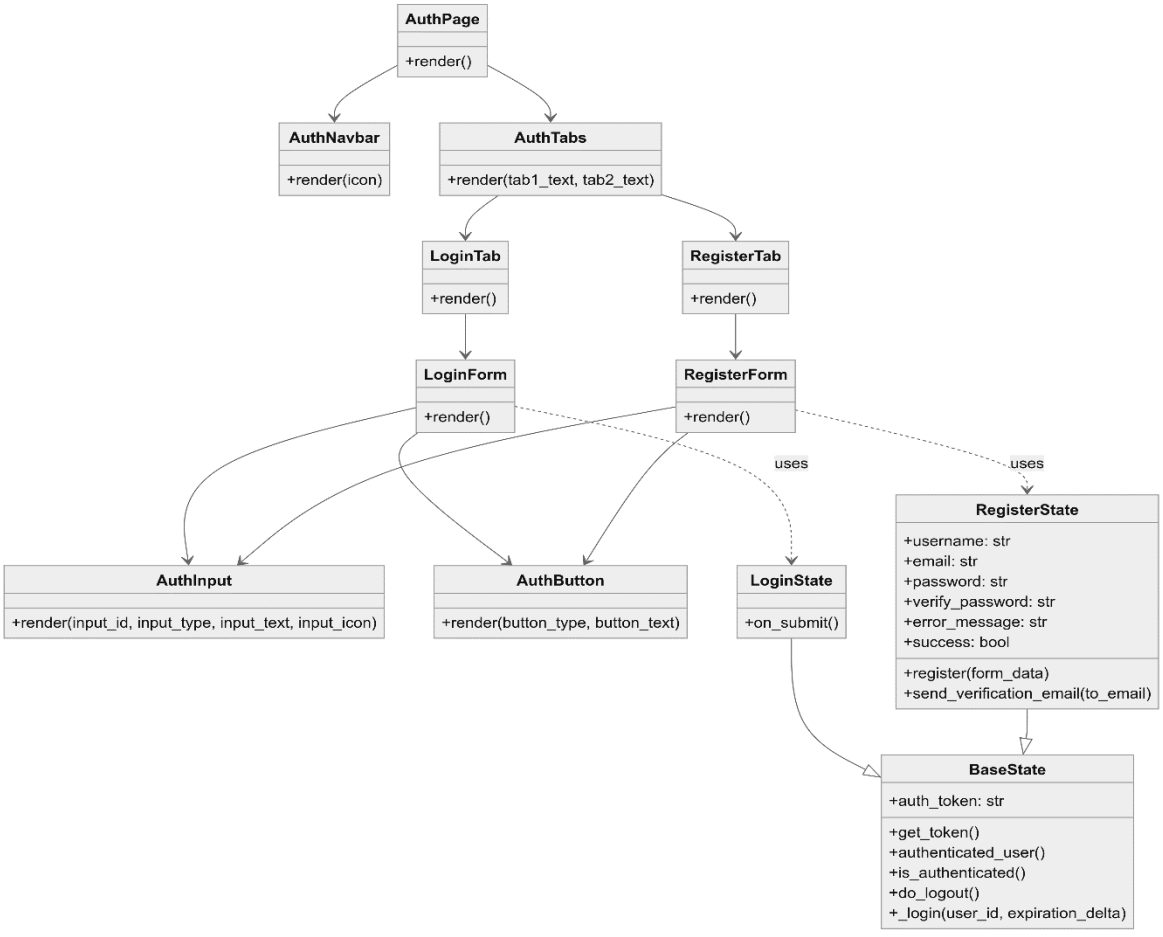

Ilustración 55 Diagrama de componentes de Auth page

Mapa de estados completo de autenticación

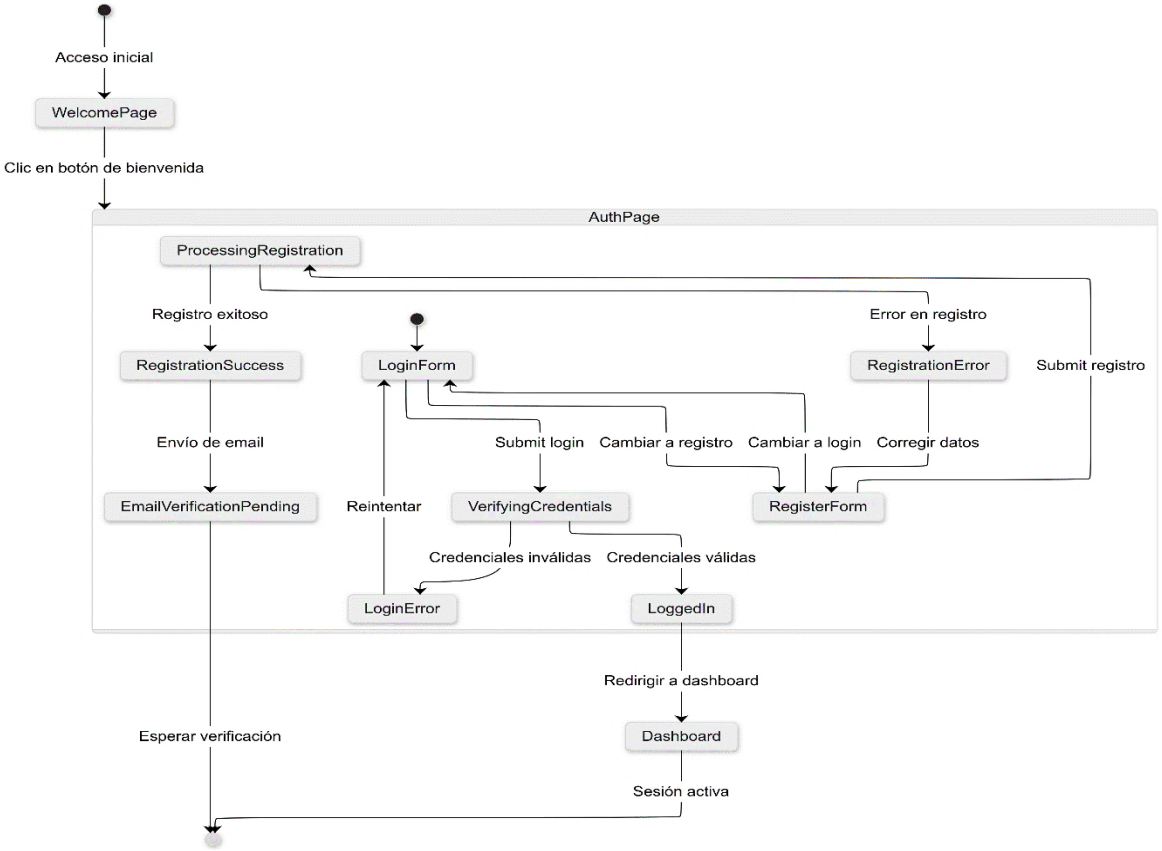

Ilustración 56 Mapa de estados de Auth page

Diagrama de secuencia del proceso de login

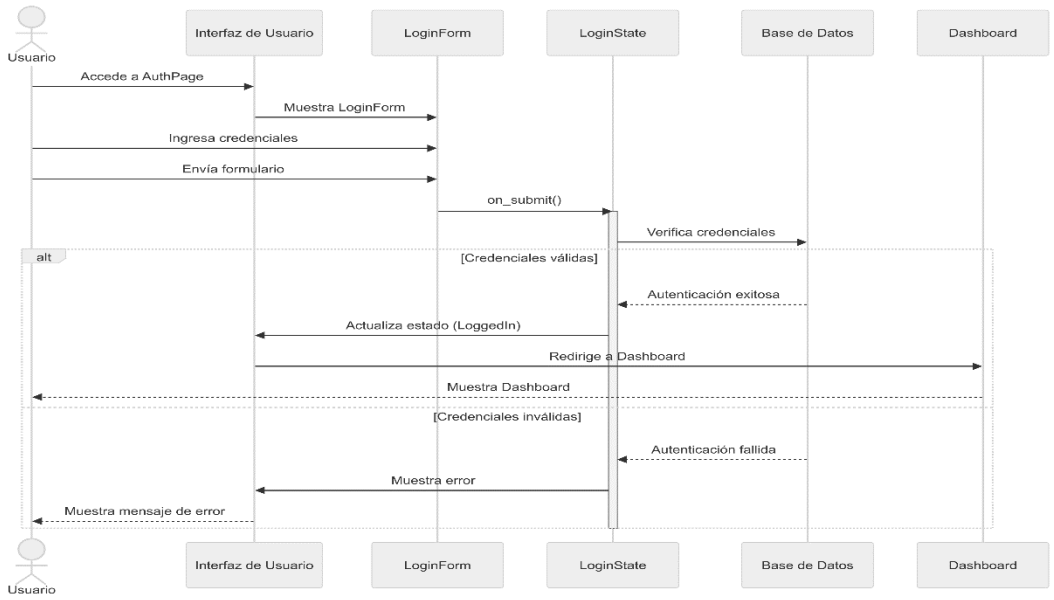

Ilustración 57 Diagrama de secuencia del proceso de login

Diagrama de secuencia del proceso de registro

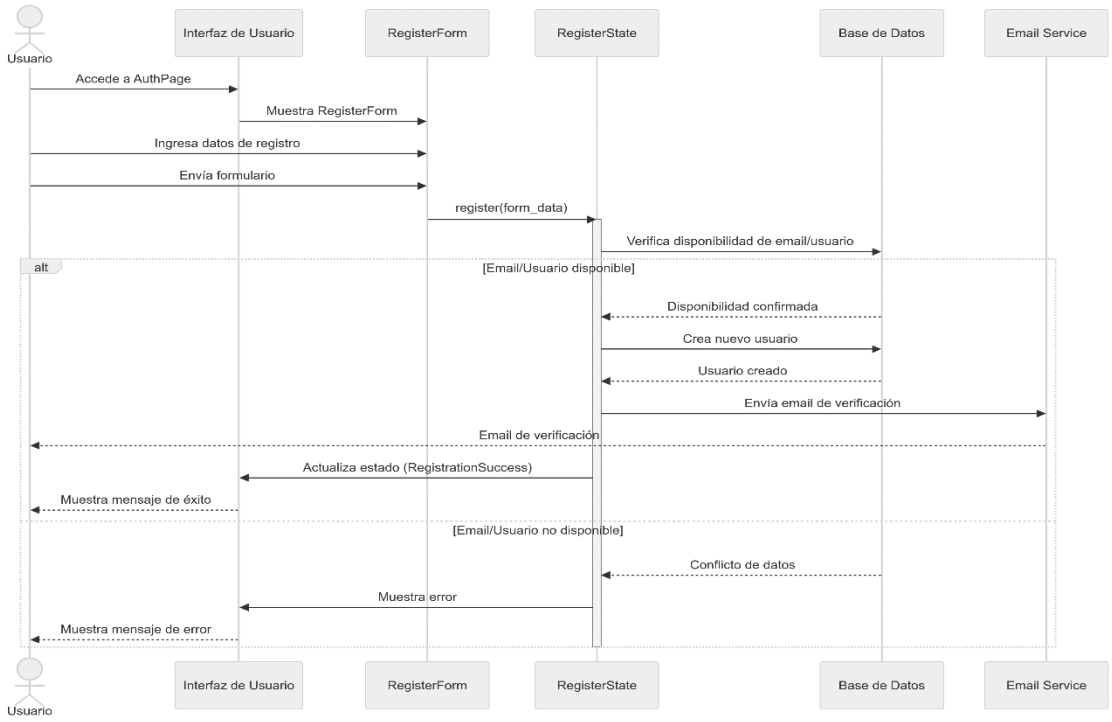

Ilustración 58 Diagrama de secuencia de proceso de registro

5.4.2.4.1. Verify

La Página de Verificación es una página simple sin interfaz visual, diseñada para confirmar la verificación de usuarios tras el envío de un correo electrónico. Este correo contiene un enlace con un token único, cuya estructura es “/verify/[token]”.

Al acceder a esta URL, la aplicación busca en la base de datos el registro asociado a dicho token y, si es válido, actualiza el campo email_verified, confirmando que el correo del usuario ha sido verificado. Una vez completado este proceso, el usuario ya puede iniciar sesión en la plataforma de manera normal.

La verificación se realiza automáticamente al acceder al enlace, sin necesidad de interacción adicional por parte del usuario.

Diagrama de componentes

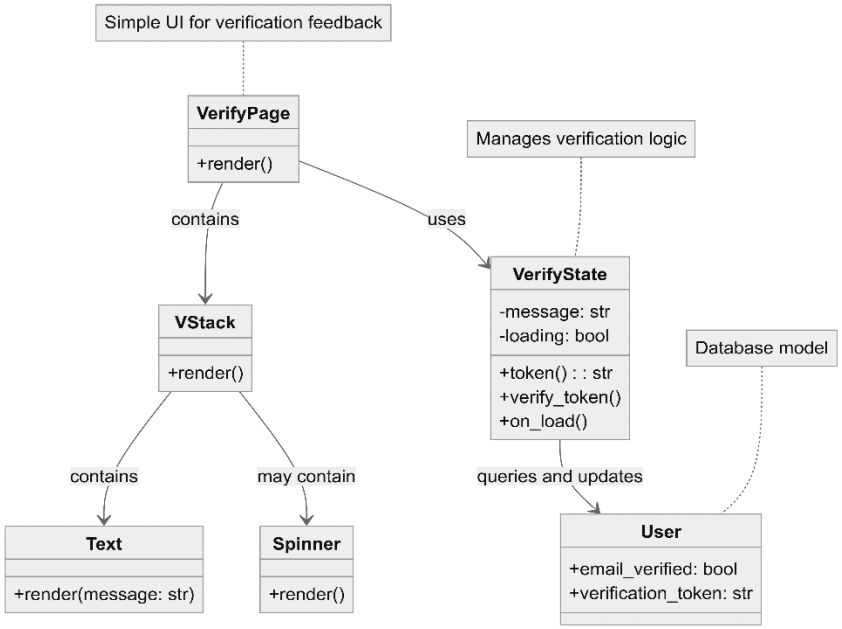

Ilustración 59 Diagrama de componentes de Verify page

Diagrama de secuencia del proceso de verificación

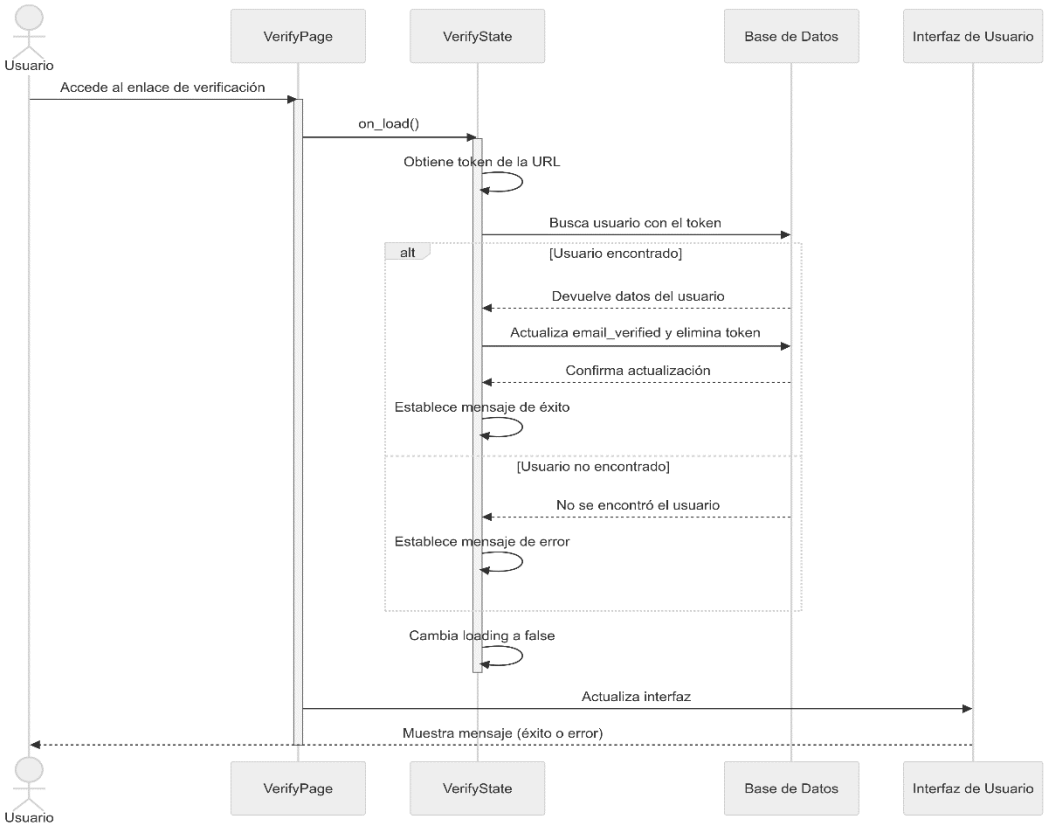

Ilustración 60 Diagrama de secuencia de verificación de nuevos usuarios

5.4.2.4.2. Home

La Home Page es la página de bienvenida a la plataforma, a la que se accede tras iniciar sesión correctamente. Dado que es una página protegida, requiere que el usuario esté autenticado; de lo contrario, el sistema redirige automáticamente a la página de autenticación auth_sesion. Como todas las páginas protegidas, la Home Page utiliza un template estándar que incluye una navbar, un sidebar y un footer.

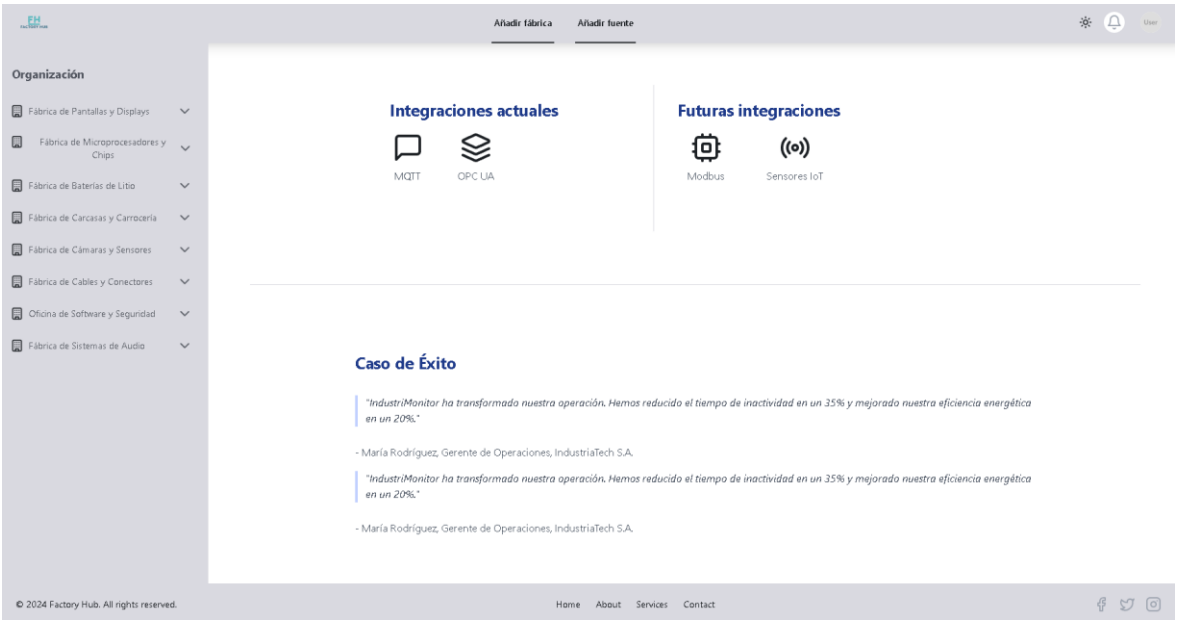

Ilustración 61 Home page

El sidebar muestra un listado de las fábricas y dispositivos configurados por el usuario, facilitando el acceso rápido a estos recursos. Además, la navbar permite navegar hacia páginas de configuración, donde el usuario puede añadir nuevos dispositivos o fábricas, extendiendo la funcionalidad de la plataforma. Esta estructura asegura una experiencia de navegación coherente y accesible dentro de la plataforma.

Diagrama de componentes

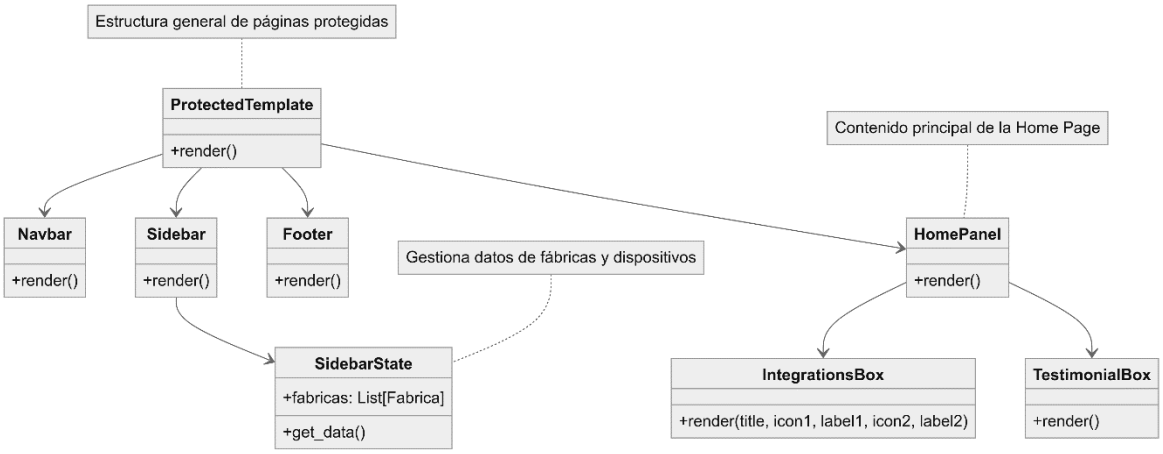

Ilustración 62 Diagrama de componentes de Home page

Diagrama de secuencia on_load del sidebar

*Todas las páginas autenticadas que utilizan el template tienen asociado este estado inicial.

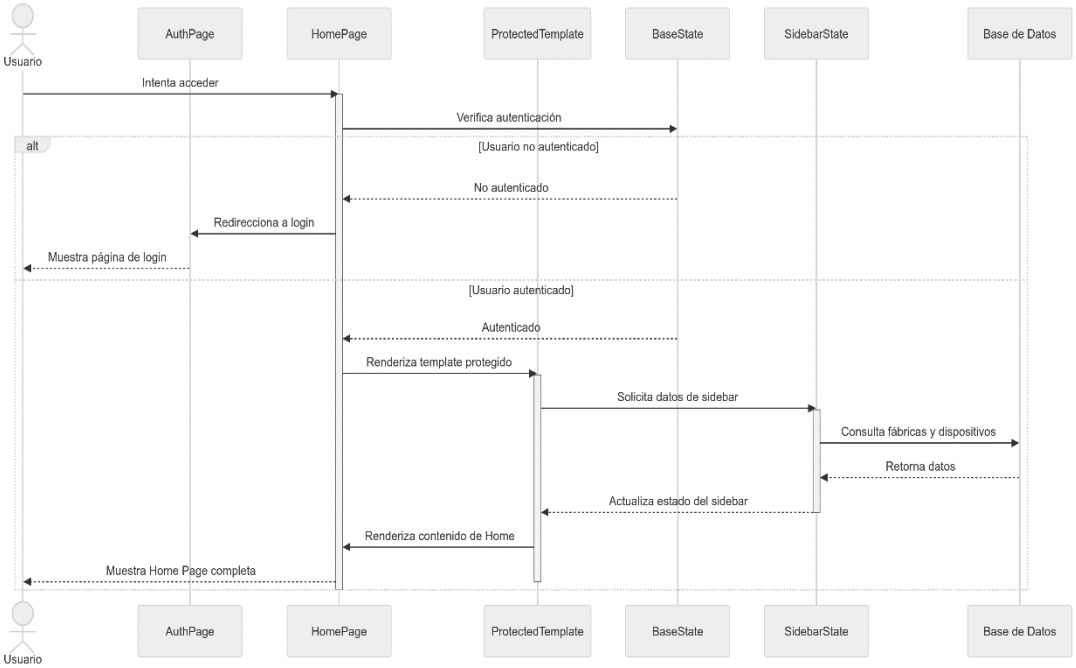

Ilustración 63 Diagrama de secuencia del on_load para el sidebar del template

5.4.2.4.3. Dashboard

Página principal de la plataforma, accesible tras iniciar sesión a través de la ruta "/dashboard", utiliza el template estándar que incluye la navegación entre los dispositivos mediante el sidebar. Al hacer clic en cualquier dispositivo del sidebar, se actualiza la página y se carga un desplegable con las variables asociadas a ese dispositivo. Al seleccionar una variable, se actualiza dinámicamente el dashboard.

El dashboard está compuesto por tres secciones clave:

- Cards que muestran los KPIs, como el último valor registrado, la hora de ese último valor, y el valor promedio de los últimos 30 días.
- Un gráfico de líneas que presenta el histórico de datos del dispositivo.
- Otro gráfico de líneas que muestra la predicción de los datos a 122 horas, generada mediante el módulo de predicciones.

Esta página permite al usuario monitorear y analizar los datos de sus dispositivos de manera detallada, proporcionando tanto información histórica como proyecciones futuras.

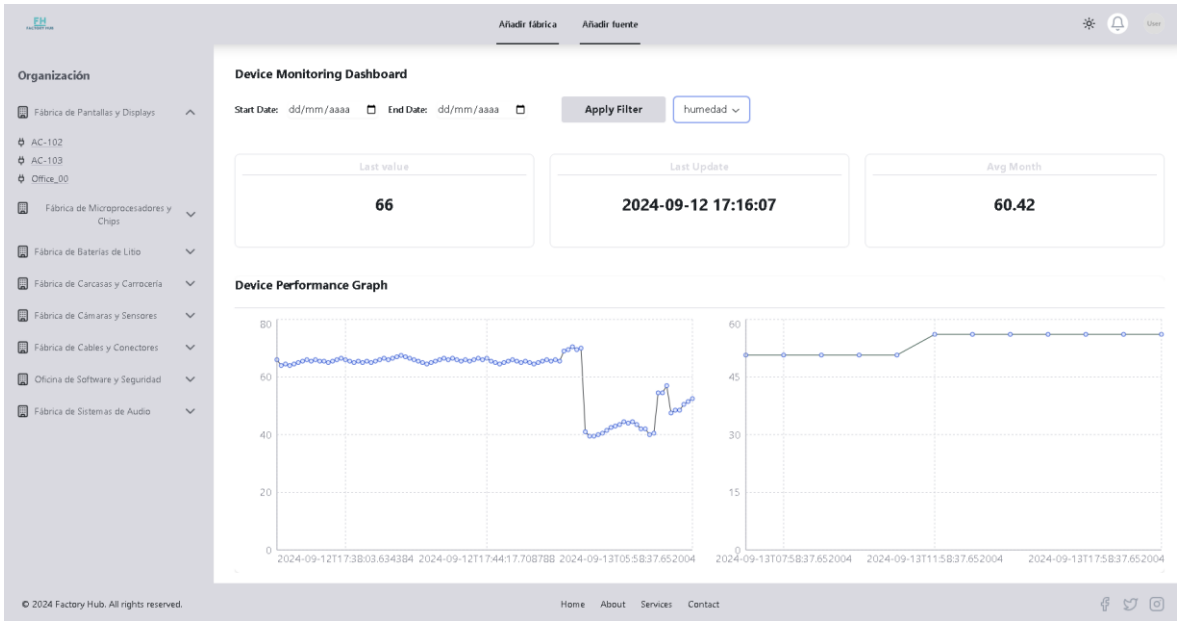

Ilustración 64 Dashboard page

Diagrama de componentes

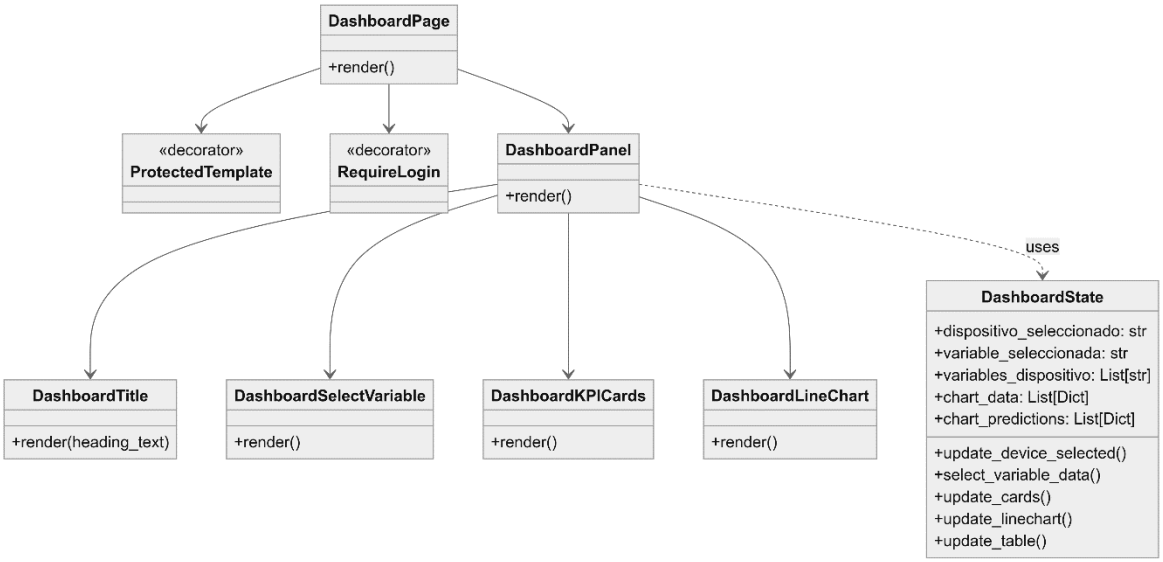

Ilustración 65 Diagrama de componentes de Dashboard page

Mapa de estados de carga de datos

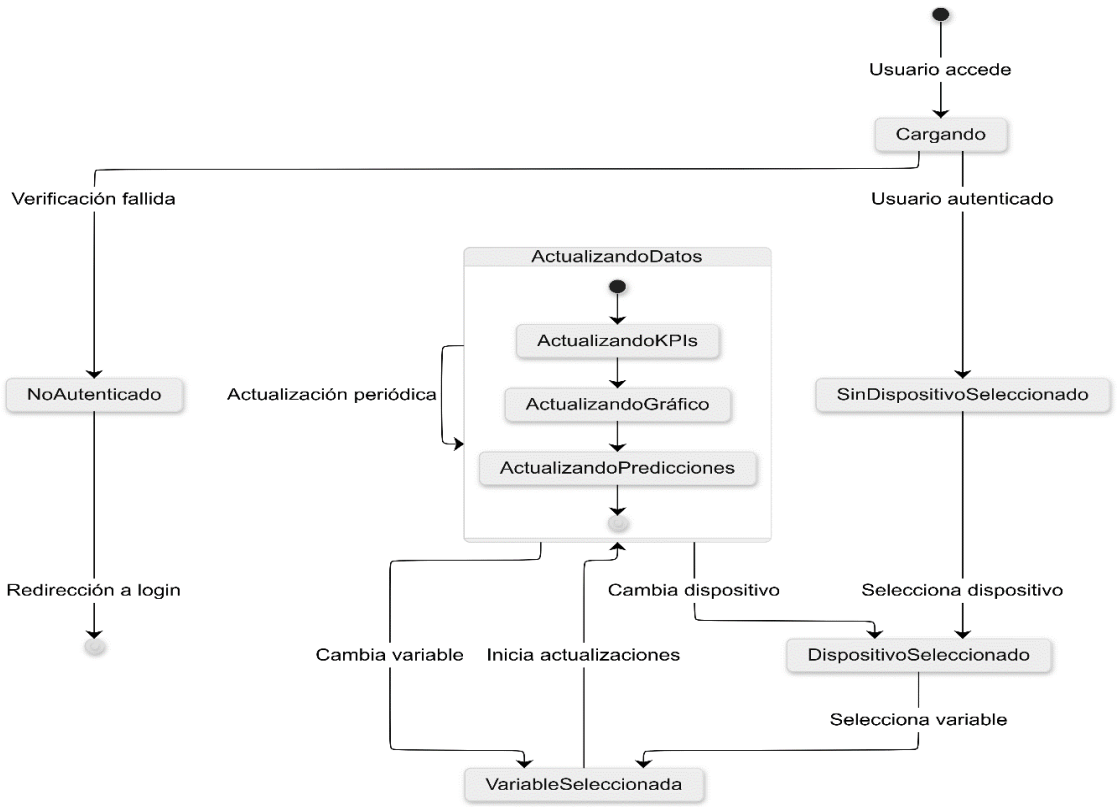

Ilustración 66 Mapa de estados de carga de datos en Dashboard Page

Diagrama de secuencia de carga de datos

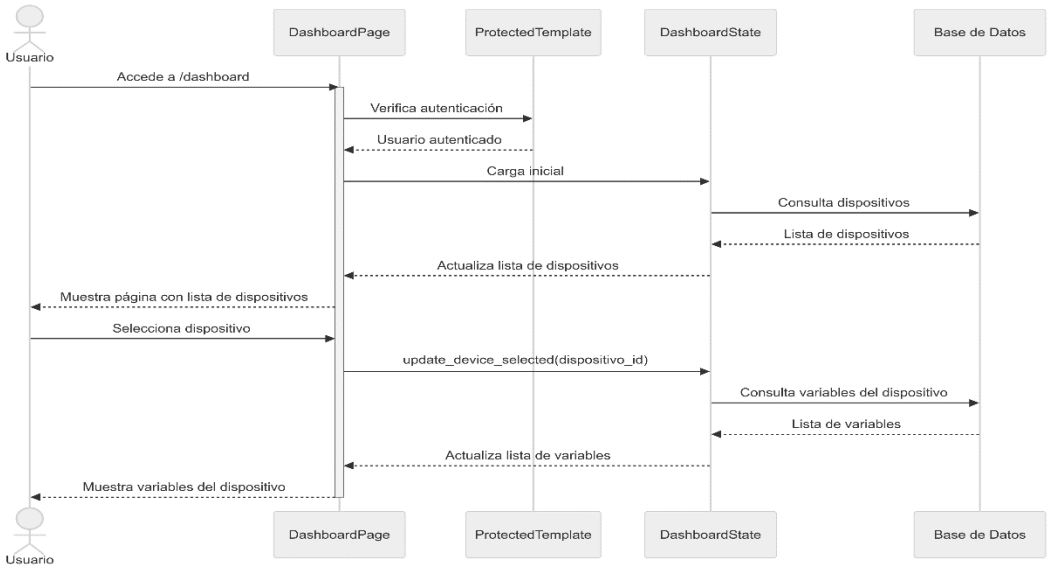

Ilustración 67 Diagrama de secuencia de selección y carga de datos en Dashboard page

Diagrama de secuencia de actualización de datos

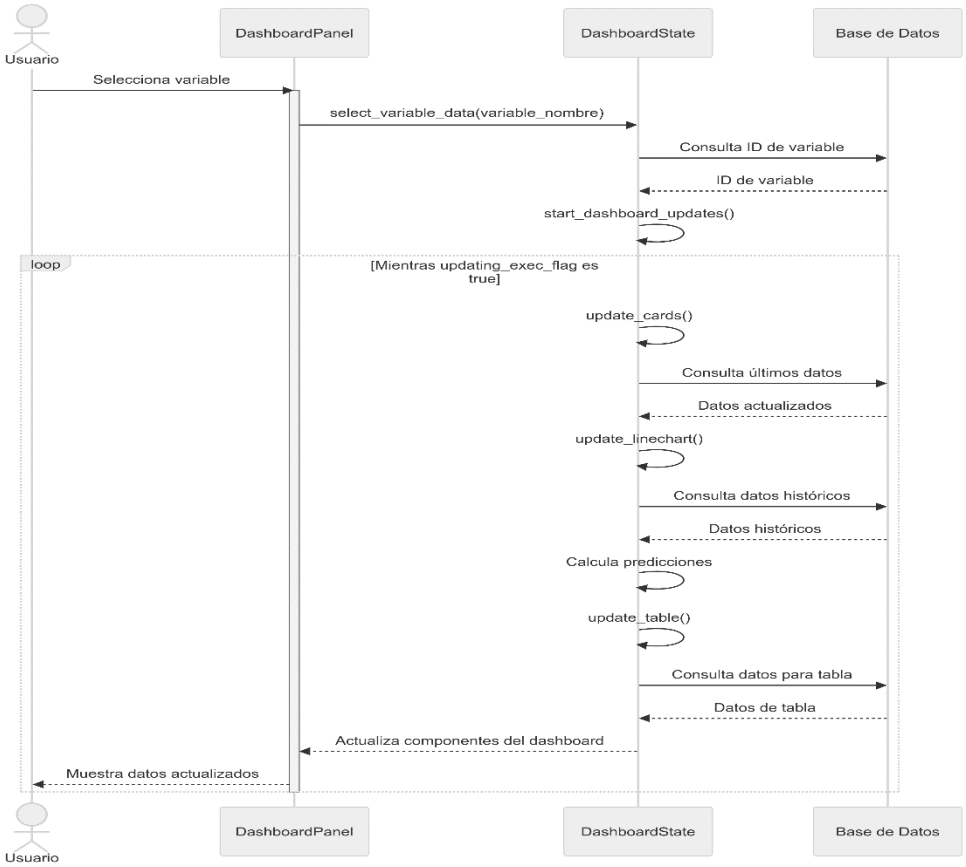

Ilustración 68 Diagrama de secuencia de actualización de datos en Dashboard page

5.4.2.4.4. New Factory

Accesible a través de la ruta protegida “/new_factory”, por lo que se requiere autenticación para ingresar. Esta página permite al usuario registrar nuevas fábricas, que son el nivel más alto dentro de la jerarquía de elementos en la plataforma. El formulario de registro incluye los campos necesarios para crear una fábrica, proporcionando una interfaz sencilla para la adición de este tipo de elementos.

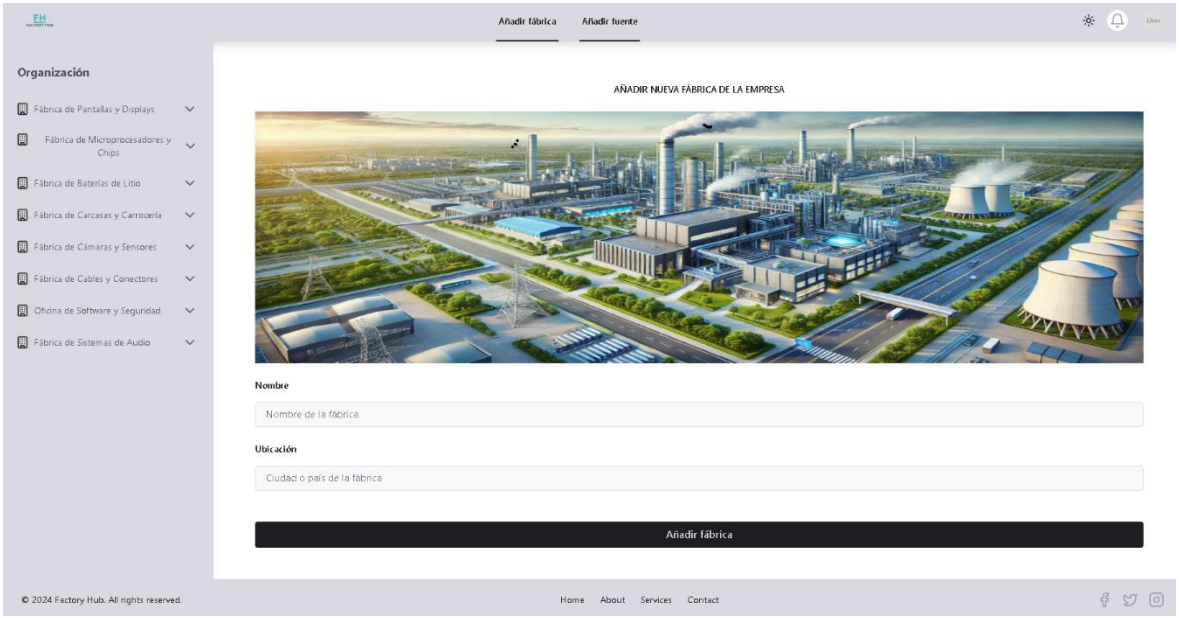

Ilustración 69 New Factory page

Diagrama de componentes

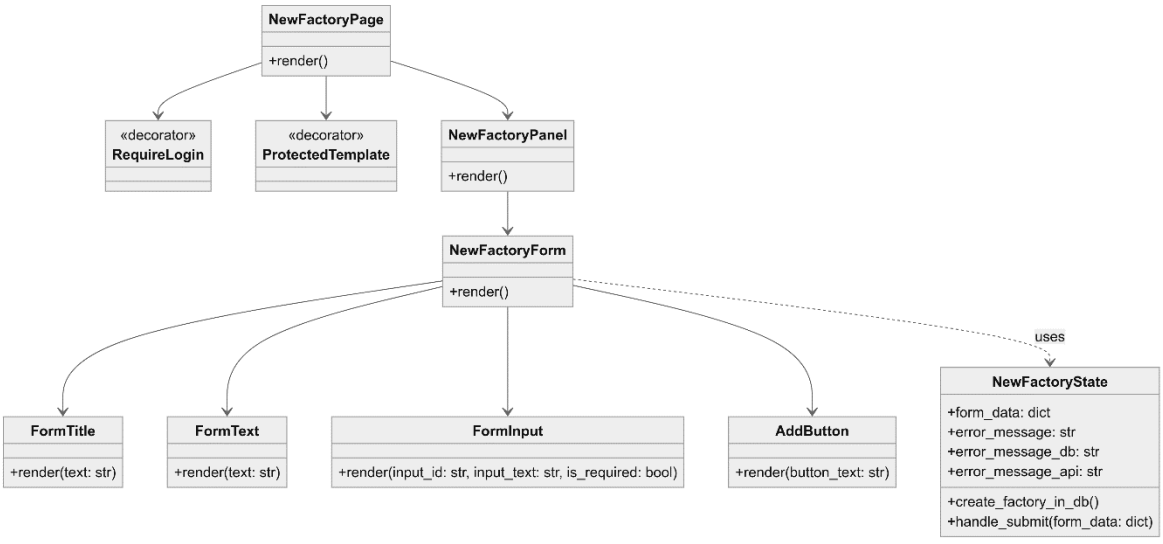

Ilustración 70 Diagrama de componentes de New Factory page

Diagrama de secuencia de adición de fábricas

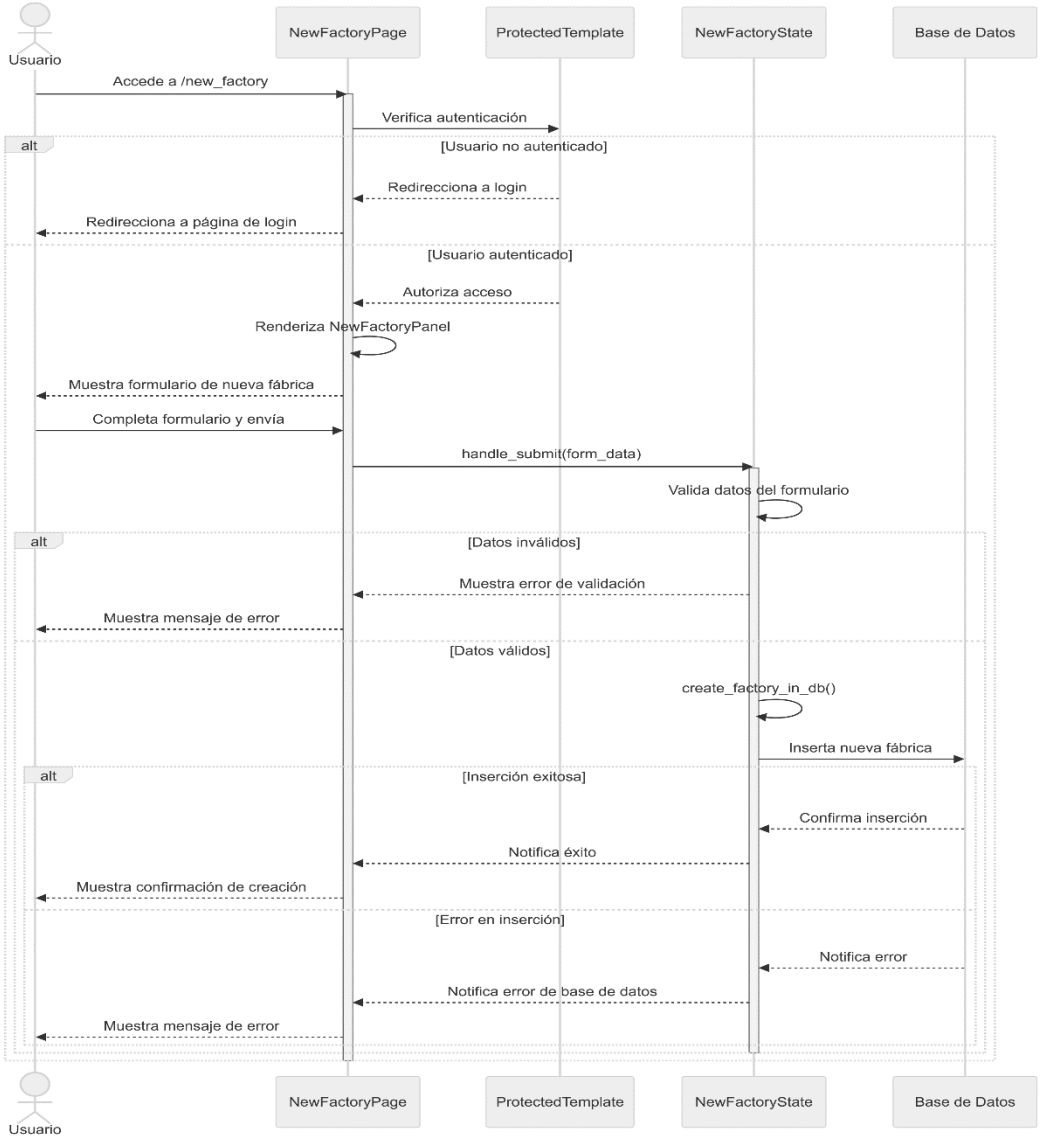

Ilustración 71Diagrma de secuencia de New Factory page

5.4.2.4.5. New Source

Accesible mediante la ruta protegida “/new_source”, por lo que también requiere que el usuario esté autenticado. Esta página contiene un formulario más complejo que permite registrar nuevos dispositivos, los cuales son elementos hijos de las fábricas previamente creadas.

Para facilitar la configuración de los dispositivos, se ha incorporado un selector que permite elegir el tipo de conexión (MQTT u OPC-UA). Según la opción seleccionada, el formulario se actualiza dinámicamente para mostrar los campos específicos requeridos para la configuración de la conexión elegida.

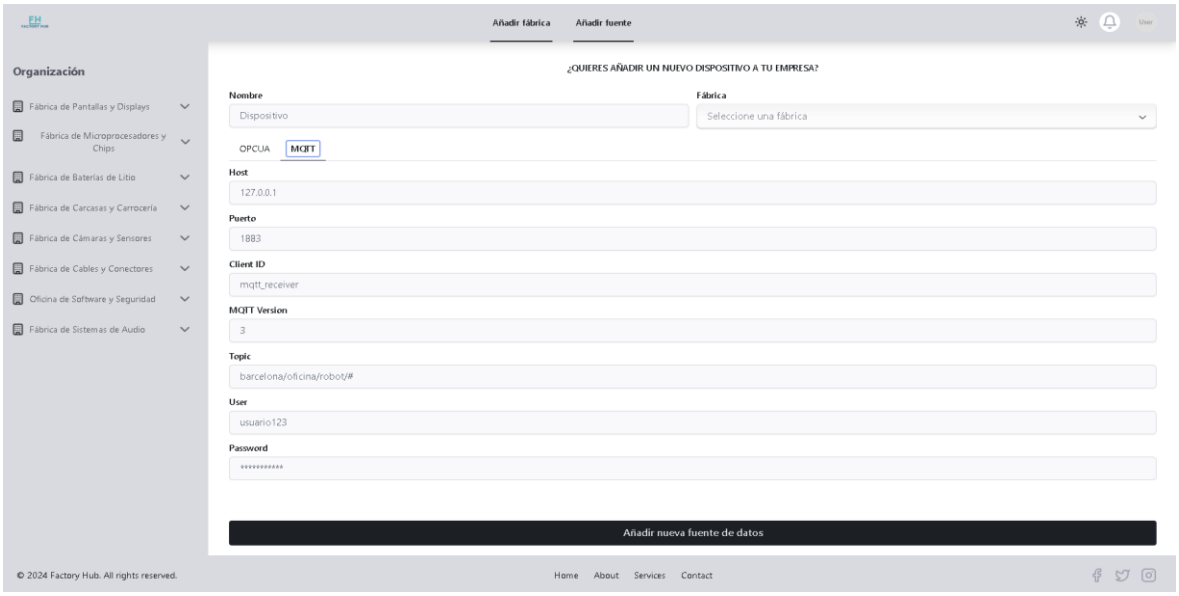

Ilustración 72 New Source page - mqtt section

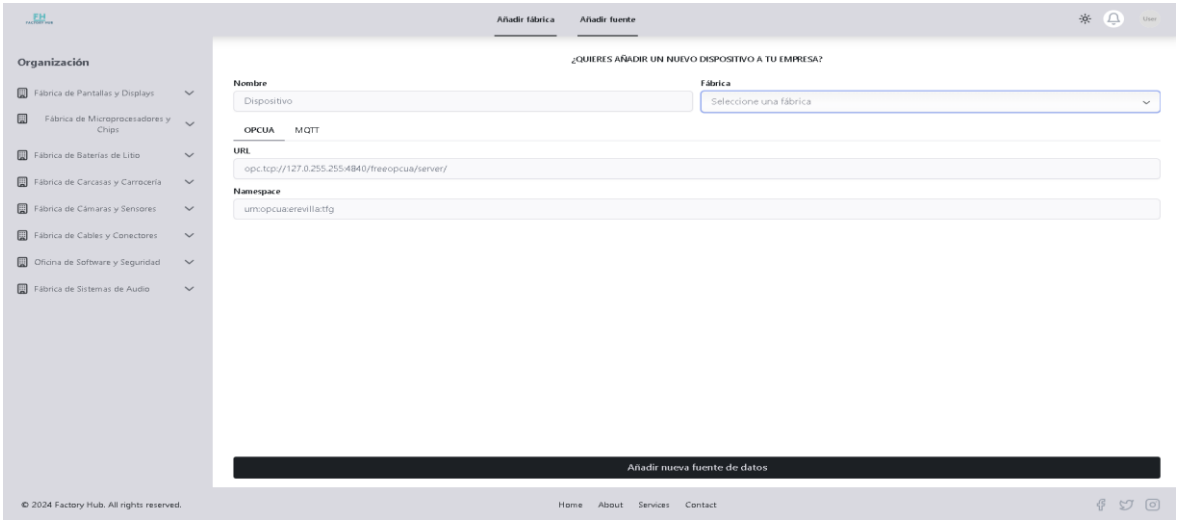

Ilustración 73 New Source page - opcua section

Diagrama de componentes

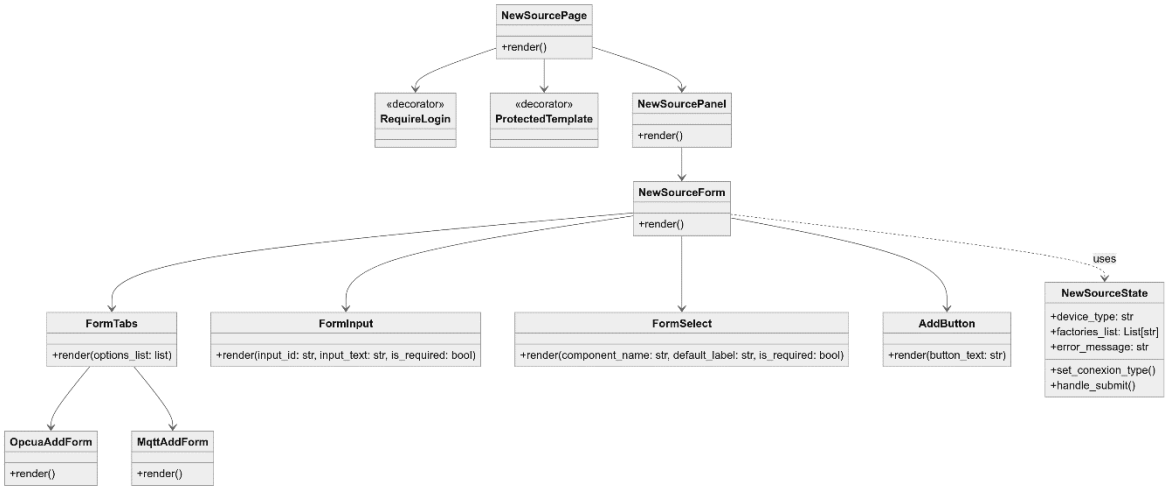

Ilustración 74 Diagrama de componentes de New Source page

Mapa de estados de creación de dispositivos

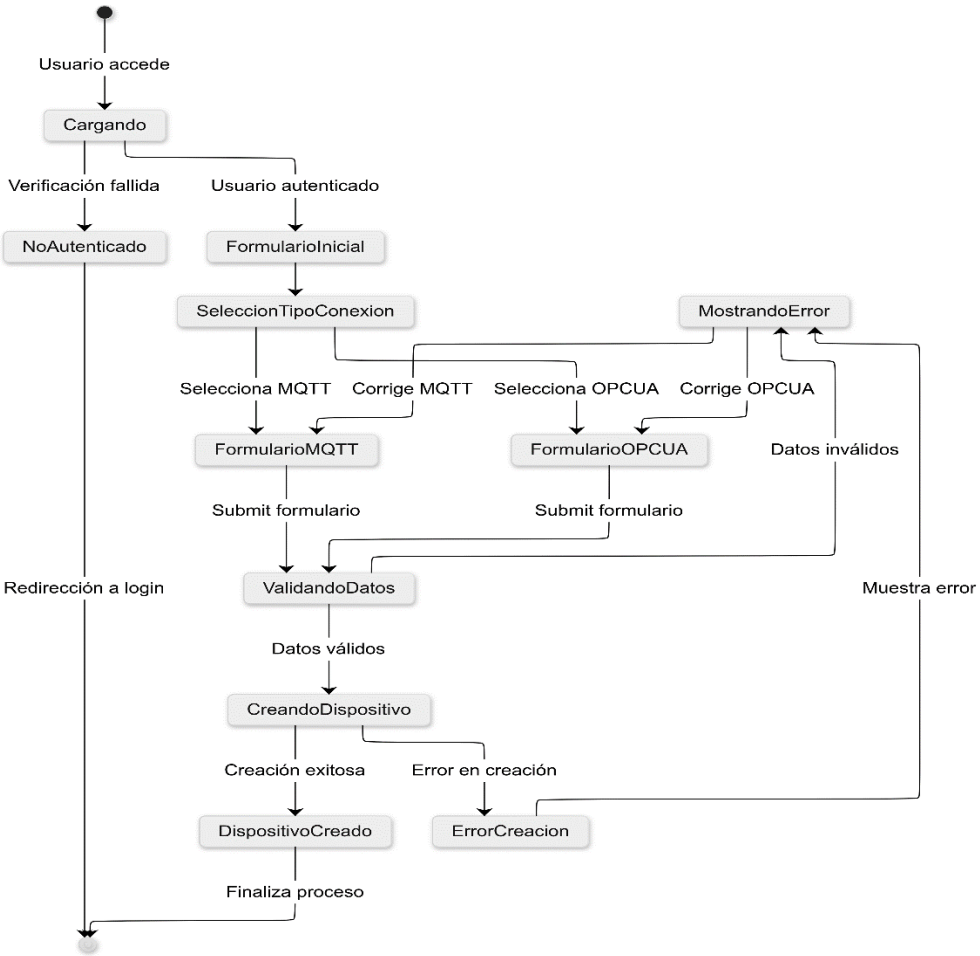

Ilustración 75 Diagrama de estados de configuración de nuevos dispositivos

Diagrama de secuencia de dispositivos

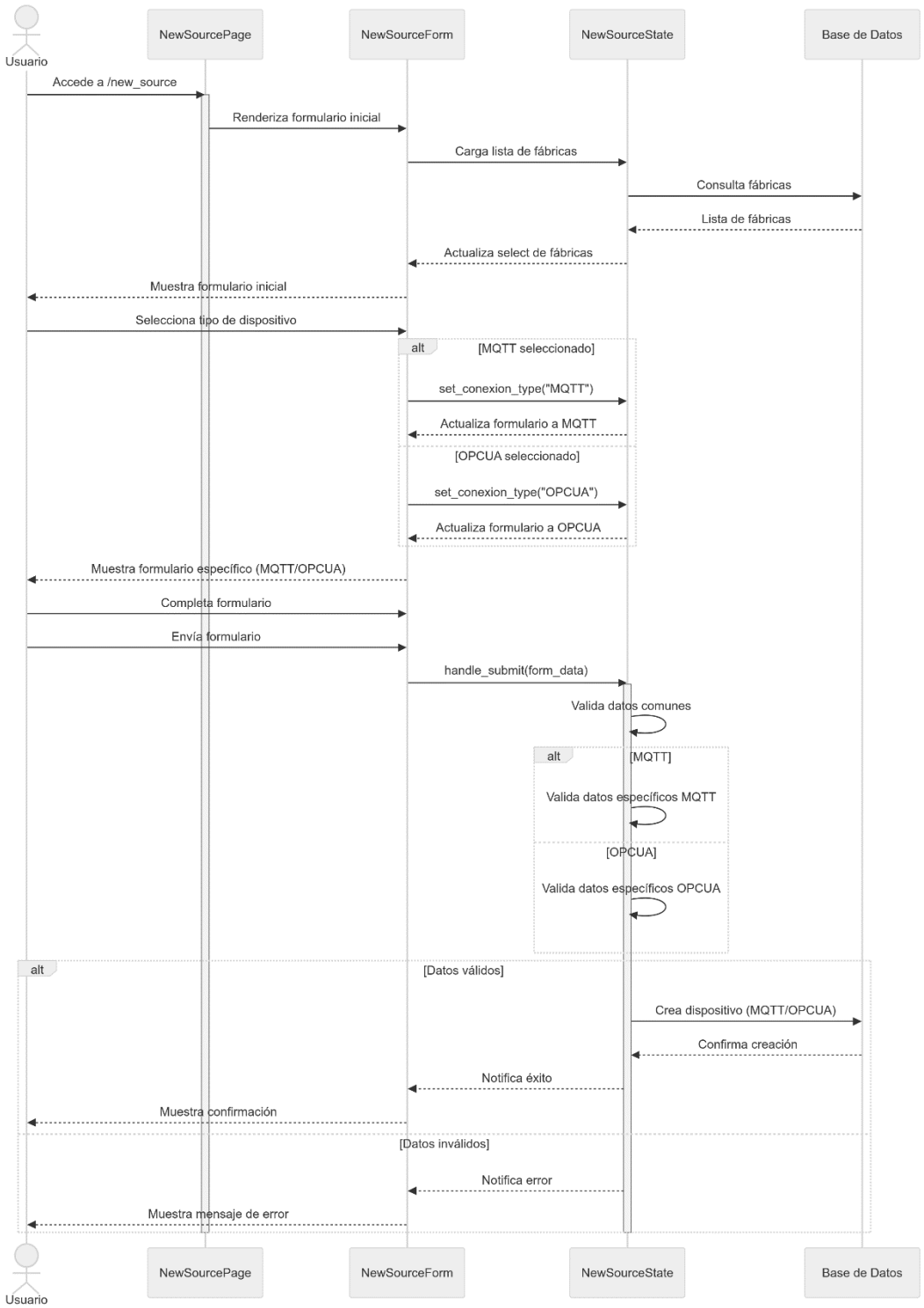

Ilustración 76 Diagrama de secuencia creación de dispositivos en New Source page

6. RESULTADOS

En este apartado se va a comprobar que tras la implementación de la plataforma IIoT se hayan cumplido los requisitos y especificaciones iniciales. Se va a abordar la tabla de requisitos funcionales para revisar las funcionalidades obtenidas y los requisitos no funcionales para revisar el entorno de ejecución y desarrollo empleado.

Entre los requisitos funcionales se habían especificado distintas categorías: captura y transmisión de datos, procesamiento de datos, almacenamiento de datos, visualización de datos, predicción de datos y gestión de usuarios y contraseñas.

6.1. CAPTURA DE DATOS

Tabla 9 Resultados obtenidos tras la implementación de la captura de datos

Requisito	Descripción	Resultado
Captura de Datos	El sistema debe capturar datos de variables industriales desde PLCs y dispositivos conectados.	El sistema permite lectura y captura de cualquier tipo de dato en formato Json con infinitos niveles recogiendo como variable la clave del último nivel.
Frecuencia de Captura	La frecuencia de captura de datos puede variar dependiendo del tipo de dispositivo y la necesidad, con una frecuencia mínima establecida cada segundo.	Se realiza una escucha en tiempo real y se procesan los mensajes cada segundo, por aumentar la robustez de los procesos, aunque es posible reducir este intervalo de tiempo.
Protocolos de Comunicación	Utilización de los protocolos industriales para la transmisión en tiempo real de los datos capturados por los sensores.	Se ha implementado un sistema de recolección de datos modular que actualmente cuenta con comunicación por protocolos MQTT y OPC UA, pero permite añadir nuevos protocolos de forma sencilla implementando nuevos servicios de extracción a la solución.
Frecuencia de Transmisión	Los datos deben ser transmitidos cada segundo al servidor central.	Los datos son recibidos de los dispositivos son procesados sin ningún tiempo o intervalo de retraso. Sus modificaciones generan eventos inmediatos de inserción en la base de datos.

6.2. PROCESAMIENTO DE DATOS

Tabla 10 Resultados obtenidos tras la implementación de la procesamiento de datos

Requisito	Descripción	Resultado
Algoritmos de Procesamiento	Implementación de un sistema de transformación o unificación de datos de las distintas fuentes y tipos.	Se ha establecido como formato json la entrada de datos y son almacenados en los estados, donde se procesan y obtienen todos datos contenidos en ellos para su correcta inserción en DB.
Tiempo de Procesamiento	Procesamiento en tiempo real con una latencia máxima de 2 segundos desde la recepción de los datos.	Al estar realizando escuchas con un watchdog de las modificaciones de los archivos json, la latencia es mínima e inferior a los dos segundos preestablecidos.

6.3. ALMACENAMIENTO DE DATOS

Tabla 11 Resultados obtenidos tras la implementación del almacenamiento de datos

Requisito	Descripción	Resultado
Base de datos	Implementación de una base de datos SQL para el almacenamiento estructurado de datos. La base de datos debe ser escalable y capaz de manejar grandes volúmenes de datos.	Se ha diseñado e implementado una base de datos con un modelo de ER y otro relacional que permiten que la plataforma se adapte a las distintas situaciones previstas para la plataforma.
Accesibilidad y consulta	Los datos deben ser fácilmente accesibles para consultas y análisis por parte de los usuarios autorizados. Implementación de índices y optimización de consultas para asegurar un acceso rápido a la información almacenada.	Se están realizando consultas desde el módulo de procesamiento y carga para insertar datos, desde el módulo de predicciones para obtener el histórico de datos y entrenar los modelos, y desde la propia plataforma para gestionar la interacción.

6.4. VISUALIZACIÓN DE DATOS

Tabla 12 Resultados obtenidos tras la implementación de visualización de datos

Requisito	Descripción	Resultado
Dashboard Interactivo	El sistema debe proporcionar una interfaz de usuario que permita la visualización en tiempo real de los datos a través de gráficos y widgets interactivos.	Se han diseñado un dashboard sencillo y práctico de visualización de datos que permitan la visualización de datos en tiempo real, históricos y predicciones.

6.5. PREDICCIÓN DE DATOS

Tabla 13 Resultados obtenidos tras la implementación de predicción de datos

Requisito	Descripción	Resultado
Modelos Predictivos	Implementación de modelos de regresión y clasificación para predecir fallos en los equipos basándose en datos históricos.	Se ha configurado el módulo de predicción de datos para que se entrene automáticamente con los históricos cada vez que realice una predicción. Esto no supone inconveniente en la latencia debido a que es un modelo muy sencillo de predicción que no requiere gran cantidad de procesamiento. No se ha abordado el tema de predicción de fallos.

6.6. GESTIÓN DE USUARIOS Y AUTENTICACIÓN

Tabla 14 Resultados obtenidos tras la implementación de autenticación de usuarios

Requisito	Descripción	Resultado
Autenticación de Usuarios	El sistema debe implementar un mecanismo de autenticación que requiera que los usuarios se identifiquen antes de acceder. Utilización de tokens de seguridad (JWT) para la gestión de sesiones de usuario.	Cada usuario tiene un ID único y el almacenamiento de las contraseñas se realiza mediante hash para securizar los datos.
Gestión de Usuarios	El sistema debe permitir la creación, edición y eliminación de cuentas de usuario.	Se ha implementado un sistema de login en la plataforma que permite registrar usuarios y acceder mediante user y password de cada usuario. No se ha tenido en cuenta durante la ejecución la posibilidad directa de borrar usuarios, pero se puede realizar mediante queries a la base de datos

Entre los requisitos no funcionales se habían especificado distintas categorías: captura y transmisión de datos, procesamiento de datos, almacenamiento de datos, visualización de datos, predicción de datos y gestión de usuarios y contraseñas.

6.7. RENDIMIENTO

Tabla 15 Resultados obtenidos tras la implementación de rendimiento

Requisito	Descripción	Resultado
Throughput	El sistema debe ser capaz de procesar un volumen elevado de datos por segundo para asegurar un rendimiento adecuado.	No se ha llegado a testear un volumen de datos muy grande por segundo ya que únicamente se han simulado dos dispositivos simultáneos enviando periódicamente con intervalos varios segundos, pero no se ha producido ningún fallo de rendimiento.
Latencia	La latencia máxima para el procesamiento de datos debe ser de 1 segundo para garantizar respuestas rápidas y eficientes.	Como se ha mencionado anteriormente, no se produce latencia aparente de procesado de datos, ni de respuesta de la web tras interacciones con el backend.

6.8. SEGURIDAD

Tabla 16 Resultados obtenidos tras la implementación de la procesamiento de datos

Requisito	Descripción	Resultado
Autenticación	Implementación de autenticación basada en tokens (JWT) para asegurar que solo los usuarios autorizados puedan acceder al sistema.	Se ha diseñad o un sistema de accesos por usuarios que se almacena de forma segura empleando hash de securización de contraseñas.

6.9. USABILIDAD

Tabla 17 Resultados obtenidos tras la implementación del almacenamiento de datos

Requisito	Descripción	Resultado
Interfaz Intuitiva	La interfaz de usuario debe ser intuitiva y fácil de usar, permitiendo una navegación sencilla y eficiente por las diferentes funcionalidades del sistema.	Se ha planteado un diseño UI / UX sencillo para la gestión y movimiento rápido entre pantallas que permita moverse de forma ágil por la plataforma.
Accesibilidad	Acceso a la plataforma desde entornos de la empresas de forma adecuada	Se ha diseñado para ser implantado en entornos locales en VM dentro de la red local de la empresa que permite acceso por dominio local a cualquier dispositivo conectado en la misma red.

6.10. MANTENIBILIDAD

Tabla 18 Resultados obtenidos tras la implementación de visualización de datos

Requisito	Descripción	Resultado
Dashboard Interactivo	El sistema debe proporcionar una interfaz de usuario que permita la visualización en tiempo real de los datos a través de gráficos y widgets interactivos.	Se han diseñado una pantalla de visualización de datos que permita la visualización de datos en tiempo real, históricos y predicciones. Es interactivo al poder seleccionar dispositivo y variables a representar.

6.11. COMPATIBILIDAD

Tabla 19 Resultados obtenidos tras la implementación de predicción de datos

Requisito	Descripción	Resultado
Integración con Herramientas	Capacidad para ampliar sus funcionalidades de extracción con nuevos tipos de fuentes de datos, almacenamiento con otros tipos de bases de datos, predicción mediante otros modelos, etc.	Al proponer el diseño modular y por capas permite flexibilizar prácticamente cada parte de la plataforma introduciendo nuevas funcionalidades en cualquiera de ellas y no únicamente en la extracción y predicción.

7. CONCLUSIONES

Tras la finalización del proyecto se va a hacer una revisión de limitaciones actuales de la plataforma y posibles mejoras.

Al ser un proyecto individual que no se ha aplicado en un caso de uso real, ni ha sido asistido por ninguna empresa, se han aplicado las limitaciones que se han creído convenientes por parte del alumno, supervisadas por el tutor.

El proyecto tal y como se planteó inicialmente pretendía ser una respuesta sencilla al problema propuesto de obtención de un hub de datos industriales accesible por cualquier empleado de la empresa (o a decisión de la propia empresa). Durante el desarrollo del mismo, se ha comprobado que las funcionalidades de la plataforma y sistema IIoT completo se podía ampliar indefinidamente.

Por tanto, los focos principales del proyecto han sido la extracción de los datos de fuentes simuladas de entornos industriales y la plataforma a nivel funcional.

Para que la plataforma realmente usable por las empresas necesitaría una segunda fase de rediseño para mejorar los posibles defectos y añadir nuevas funcionalidades, y otra fase de industrialización que implique testeos de rendimiento, seguridad, etc; más detallados.

Entre las posibles mejorar de la plataforma se van a nombrar algunas que se han visto durante el desarrollo:

Tabla 20 Tabla de posibles mejoras de la plataforma IoT

Categoría	Descripción
Captura y Transmisión de Datos	Adición de módulos de otras tecnologías de comunicación IIoT como Modbus TPC/IP, tecnologías de comunicación IoT como LoRaWAN, CoAP o comunicación con APIs de terceros por HTTP/HTTPS
	Testeo del rendimiento de recepción de grandes cantidades de dispositivos o de volúmenes de datos por segundo.
	Añadir la funcionalidad de comunicación bidireccional entre dispositivos, APIs, etc.
Procesamiento de Datos	Adición de un procesador de eventos con Kafka para mantener un buffer de mensajes y que no se pierdan, además de particiones para gestionar el manejo de mensajes para cada usuario.
	Testeo del rendimiento del procesamiento de grandes cantidades de dispositivos o de volúmenes de datos por segundo.
Almacenamiento de Datos	Mejora de los modelos ER y relacional de la base de datos y rediseño del formato de acceso a datos.

Categoría	Descripción
Visualización de Datos	Mejora del diseño UI / UX para que las pantallas sean más completas y se puedan gestionar y navegar óptimamente.
	Añadir funcionalidades de visualización: - Pantalla de logs de la plataforma - Panel de notificaciones o alertas. - Creación de informes de reporte de datos
Predicción de Datos	Implementación de modelos de regresión y clasificación para predecir fallos en los equipos basándose en datos históricos.
Gestión de Usuarios y Autenticación	Realizar un sistema de autenticación mejorado con mayor seguridad como el 2FA.

8. OBJETIVOS DE DESARROLLO SOSTENIBLE

Los objetivos de este Trabajo Fin de Grado están alineados con los siguientes Objetivos de Desarrollo Sostenible (ODS) y metas, de la Agenda 2030:

Objetivo 9 - Industria, Innovación e Infraestructura

Meta 9.4: Para 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficiencia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales, con todos los países actuando en función de sus respectivas capacidades.

Meta 9.5: Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales en todos los países, en particular los países en desarrollo, entre otros, fomentando la innovación y aumentando sustancialmente el número de personas que trabajan en investigación y desarrollo por millón de habitantes, así como los gastos en investigación y desarrollo del sector público y privado.

Objetivo 12 - Producción y Consumo Responsables

Meta 12.2: Para 2030, lograr la gestión sostenible y el uso eficiente de los recursos naturales.

Meta 12.5: Para 2030, reducir sustancialmente la generación de desechos mediante actividades de prevención, reducción, reciclado y reutilización.

Meta 12.6: Alentar a las empresas, especialmente las grandes y transnacionales, a que adopten prácticas sostenibles e incorporen información sobre sostenibilidad en su ciclo de presentación de informes.

9. BIBLIOGRAFÍA

[1] «Snapshot». Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://www.opertek.com/blog/plataforma-predix/

[2] «Get Started for Predix Developers».

[3] «GE_Predix_Architecture_and_Services-20161128.pdf». Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://d154rjc49kgakj.cloudfront.net/GE_Predix_Architecture_and_Services-20161128.pdf

[4] «¿Cuáles son los puntos clave en un proceso de Ingesta de Dat». Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://www.mytra.es/blog-post/proceso-ingesta-datos

[5] «Snapshot». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.velatia.com/es/blog/cuales-son-las-claves-de-la-industria-4-0/

[6] «Y.2060 : Visión general de la Internet de las cosas». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.itu.int/rec/T-REC-Y.2060-201206-I/es

[7] «Snapshot». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://www.mokolora.com/es/lora-and-wireless-technologies/

[8] «Cobos - Diseño e implementación de una arquitectura IoT ba.pdf». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://biblus.us.es/bibing/proyectos/abreproy/70884/fichero/TFM-Antonio+Cobos+Dominguez.pdf

[9] L. Joyanes Aguilar, *Internet de las cosas Un futuro hiperconectado: 5G, inteligencia artificial, big data, cloud, blockchain y ciberseguridad*, Primera edición: MARCOMBO S.L. 2021. MARCOMBO S.L., 2021.

[10] «Imagen relación IT, OT e IIoT». [En línea]. Disponible en: https://www.secior.com/wp-content/uploads/2021/05/Secior_IT-OT_5.jpg

[11] Chris, «Efficient IIoT Communications: A Comparison of MQTT, OPC-UA, HTTP, and Modbus», Cirrus Link. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://cirrus-link.com/efficient-iiot-communications-a-comparison-of-mqtt-opc-ua-http-and-modbus/

[12] «Snapshot». Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://cirrus-link.com/efficient-iiot-communications-a-comparison-of-mqtt-opc-ua-http-and-modbus/

[13] «MQTT - The Standard for IoT Messaging». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://mqtt.org/

[14] «MQTT - The Standard for IoT Messaging». Accedido: 21 de mayo de 2024. [En línea]. Disponible en: https://mqtt.org/

[15] «MQTT - The Standard for IoT Messaging». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://mqtt.org/

[16] «mqtt-v5.0.pdf». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

[17] steve, «Understanding the MQTT Protocol Packet Structure». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

[18] Satoshi, «¿No sabes qué es OPC? Aprende ya sobre esta tecnología», Opiron. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://www.opiron.com/que-es-opc/

[19] «Estévez - Análisis de OPC-UA y MQTT como protocolos de comun.pdf». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://crea.ujaen.es/bitstream/10953.1/18901/1/TFGJavifinal%20-%20Javier%20Del%20Moral%20Conde.pdf

[20] «Computer Integrated Manufacturing (CIM)». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://publichealth.buffalo.edu/cat/kt4tt/best-practices/need-to-knowledge-ntk-model/ntk-commercial-devices/master-list-of-tools/mechanical-engineering-/computer-integrated-manufacturing--cim--.html

[21] «cim0910.pdf». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: http://www.etitudela.com/fpm/comind/downloads/cim0910.pdf

[22] «Memoria TFG-Joel Fernandez Torrado.pdf». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://upcommons.upc.edu/bitstream/handle/2117/394561/Memoria%20TFG-Joel%20Fernandez%20Torrado.pdf?sequence=6

[23] «Monolítico frente a microservicios: diferencia entre arquitecturas de desarrollo de software - AWS», Amazon Web Services, Inc. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/

[24] F. Valdes, «La guía definitiva de la arquitectura del software», Medium. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://medium.com/@ktufernando/la-gu%C3%ADa-definitiva-de-la-arquitectura-del-software-f419db9c6bf7

[25] «Qué es una base de datos | Oracle España». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.oracle.com/es/database/what-is-database/

[26] «¿Qué es una base de datos relacionales? - IONOS España». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos-relacionales/

[27] «¿Qué es una base de datos NoSQL? | IBM». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/nosql-databases

[28] «Qué es PostgreSQL y sus principales ventajas | Ayuda Ley Protección Datos». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://ayudaleyprotecciondatos.es/bases-de-datos/que-es-postgresql-ventajas/

[29] «PostgreSQL vs. MySQL: sistemas de gestión de bases de datos - IONOS España». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.ionos.es/digitalguide/servidores/know-how/postgresql-vs-mysql/

[30] «¿Qué es la Inteligencia Artificial? - Iberdrola». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.iberdrola.com/innovacion/que-es-inteligencia-artificial

[31] «¿Qué es el aprendizaje no supervisado? | IBM». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/unsupervised-learning

[32] «Aprendizaje Supervisado y No Supervisado». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.cs.us.es/~fsancho/Blog/posts/Aprendizaje_Supervisado_No_Supervisado.md.html

[33] «¿Qué es el aprendizaje supervisado? | IBM». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/supervised-learning

[34] «Diseño de un marco web Pure Python · Blog Reflex». Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://reflex.dev/blog/2024-03-21-reflex-architecture/

[35] F. Pedregosa *et al.*, «Scikit-learn: Machine Learning in Python», *Mach. Learn. PYTHON*.

[36] Raschka, S., & Mirjalili, V., *Python Machine Learning: Machine Learning and Deep learning with Python, scikit-learn, and TensorFlow 2.*, 3rd Edition. Packt Publishing, 2019.

[37] «pgAdmin 4 — pgAdmin 4 8.7 documentation». Accedido: 5 de junio de 2024. [En línea]. Disponible en: https://www.pgadmin.org/docs/pgadmin4/latest/

[38] E. E. Estévez, «Análisis de OPC-UA y MQTT como protocolos de comunicación en la Industria conectada».

[39] I. J. González-Hernández, B. Armas-Alvarez, M. Coronel-Lazcano, N. Maldonado-López, O. Vergara-Martínez, y R. Granillo-Macías, «El desarrollo tecnológico en las revoluciones industriales», *Ingenio Concienc. Bol. Científico Esc. Super. Ciudad Sahagún*, vol. 8, n.º 16, Art. n.º 16, jul. 2021, doi: 10.29057/escs.v8i16.7118.

[40] A. Cobos, «Diseño e implementación de una arquitectura IoT basada en tecnologías Open Source.».

[41] «Claves de la industria 4.0: las cinco piezas que la definen», Velatia. Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.velatia.com/es/blog/cuales-son-las-claves-de-la-industria-4-0/

[42] Satoshi, «¿Que es OPC UA? Aprende en menos de 3 minutos», Opiron. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://www.opiron.com/que-es-opc-ua/

[43] «Documento sin título», Google Docs. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://docs.google.com/document/d/1adpaDpkeW6OQ6Qtdc8UfaUpR1hZvsNJprDd

RkS4wHxw/edit?usp=drive_web&ouid=104915408341617048345&usp=embed_facebook

[44] «OPC-UA_Overview_ES.pdf». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://opcfoundation.org/wp-content/uploads/2014/05/OPC-UA_Overview_ES.pdf

[45] «Comparación entre LoRa y otras tecnologías inalámbricas». Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://www.mokolora.com/es/lora-and-wireless-technologies/

[46] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, y M. Ayyash, «Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications», *IEEE Commun. Surv. Tutor.*, vol. 17, n.º 4, pp. 2347-2376, 2015, doi: 10.1109/COMST.2015.2444095.

[47] C. Team, «What is the Difference Between IT and OT? | Coolfire Blog», Coolfire. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://coolfiresolutions.com/blog/difference-between-it-ot/

[48] E. Team, «MQTT Control Packets: A Beginner's Guide», www.emqx.com. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://www.emqx.com/en/blog/introduction-to-mqtt-control-packets

[49] «Fig. 2. Structure of an MQTT message.», ResearchGate. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/Structure-of-an-MQTT-message_fig2_328606808

[50] E. Isa, «OPC UA – interoperabilidad en el nivel semántico».

[51] «Unified Architecture», OPC Foundation. Accedido: 25 de mayo de 2024. [En línea]. Disponible en: https://opcfoundation.org/about/opc-technologies/opc-ua/

[52] E. E. Estévez, «Análisis de OPC-UA y MQTT como protocolos de comunicación en la Industria conectada».

[53] K. Gos y W. Zabierowski, «The Comparison of Microservice and Monolithic Architecture», abr. 2020, pp. 150-153. doi: 10.1109/MEMSTECH49584.2020.9109514.

[54] A. Schiaffarino, «Modelo cliente servidor: ¿Qué es? Características, Ventajas y Desventajas», Infranetworking. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://blog.infranetworking.com/modelo-cliente-servidor/

[55] www.dreams.es, «Patrones de arquitectura de software Desarrolladores Páginas Web», dreams. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/patrones-de-arquitectura-de-software

[56] Slack, «mensajería instantánea: diferencias entre la comunicación sincrónica y asincrónica.», Slack. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://slack.com/intl/es-es/blog/collaboration/mensajeria-instantanea-comunicacion-sincronica-y-asincronica-diferencias

[57] G. Corzo, «Programación Asíncrona», Laboratoria Developers. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://medium.com/laboratoria-how-to/programacion-asincrona-cea3bad7c3c6

[58] «Predix, cloud para Internet Industrial de General Electric», Opertek. Accedido: 28 de mayo de 2024. [En línea]. Disponible en: https://www.opertek.com/blog/plataforma-predix/

[59] M. A. O. Pérez, «Material docente en abierto para las asignaturas "Aplicaciones Telemáticas" y "Desarrollo de Aplicaciones Telemáticas"».

[60] «módulo cliente — documentación de Eclipse paho-mqtt». Accedido: 29 de mayo de 2024. [En línea]. Disponible en: https://eclipse.dev/paho/files/paho.mqtt.python/html/client.html

[61] A. H.-M. Zufía, «Desarrollo de un Sistema para la Extracción, Análisis y Visualización de Estadísticas de la Liga de Fútbol Profesional».

[62] A. Kocamuftuoglu, O. Akbay, y S. Kaba, «A Comparative Study on Industrial Communication Protocols Using IoT Platforms», *Eurasia Proc. Sci. Technol. Eng. Math.*, vol. 14, pp. 57-65, dic. 2021, doi: 10.55549/epstem.1050178.

[63] domingo, «IoT, según Gartner», PrensarioHub. Accedido: 30 de mayo de 2024. [En línea]. Disponible en: https://www.prensariohub.com/iot-segun-gartner/

Relación de documentos

(X) Memoria 129 páginas

La Almunia, a 05 de 06 de 202x

Firmado: Revilla Mata, Eduardo