{ 2606: datos-noja_SW/HW_SDID }

SDITD

Revisión: { 00 }
Fecha: { 09/06/2025 }

DOCUMENTO DE DISEÑO, IMPLEMENTACIÓN Y PRUEBAS

para

VISUALIZACIÓN DE DATOS EQUIPO NOJA

Fecha de entrega:	*23/06/2025*
Fecha de inicio del Proyecto:	*19/05/2025*
Duración:	*1*
Asunto:	*Obtención y visualización de datos del equipo reconectador NOJA Power.*
Socio/s para este entregable:	*Juan Carlos Amati* *Pablo Solivellas* *Daniel Anunziata*

Nivel de difusión		
PU	Público.	
UI	Uso interno, para uso interno de *IPSEP*. No está permitida la distribución libre de la información fuera de la organización.	
RS	Restringido, restringido bajo las condiciones establecidas en el Modelo de Acuerdo de Trabajo. La versión que proporcionará *F3-Telecomunicaciones* tendrá toda la información necesaria para realizar su evaluación.	**X**
CO	Confidencial, *IPSEP* solo proveerá una versión del documento mediante un acuerdo de confidencialidad previo por escrito a tal fin.	

SOBRE ESTE DOCUMENTO

Preparado por:

Equipo *F3-Telecomunicaciones*

Responsable:

Milanesio, Valentin *09/06/2025*

Revisado por:

Solivellas, Pablo *09/06/2025*

Anunziata, Daniel *09/06/2025*

Aprobado por:

Solivellas, Pablo *09/06/2025*

Anunziata, Daniel *09/06/2025*

NOTA IMPORTANTE

LISTA DE MODIFICACIONES

VERSIÓN	FECHA	PÁGINAS	CAMBIOS	OBSERVACIONES
00	*09/06/2025*	*TODAS*	*REVISIÓN INICIAL*	

LISTA DE DISTRIBUCIÓN

FECHA	NOMBRE	FUNCIÓN	ORGANIZACIÓN	Nº COPIAS
09/06/2025	*Juan Carlos Amati*	*BENEFICIARIO*	*IPSEP*	*1*
09/06/2025	*Pablo Solivellas*	*REVISIÓN*	*LABREDES*	*1*
09/06/2025	*Daniel Anunziata*	*REVISIÓN*	*LABREDES*	*1*

ÍNDICE

1. INTRODUCCIÓN

Este documento describe el diseño y la implementación del software para el proyecto *Visualización de datos equipo NOJA*, desarrollado en el marco de *contrato* y destinado a *IPSEP*. Su objetivo es presentar:

- La arquitectura global del sistema, detallando los distintos módulos y su interacción.
- Los patrones de diseño y las decisiones tecnológicas empleadas.
- El diseño de datos: modelos, esquemas y flujos de información.
- El plan de implementación: estructura del código, estándares de calidad y herramientas de compilación e integración continua.
- La estrategia de pruebas y el despliegue en los entornos previstos.

El documento ha sido producido siguiendo las pautas de los estándares de documentación previstos en las normas vigentes y en la documentación de soporte del proyecto. El mismo está organizado en las siguientes secciones:

Sección 1	**Introducción**. Propósito, alcance y referencias bibliográficas.
Sección 2	**Arquitectura del Sistema**. Diagrama de componentes, descripción de subsistemas y comunicaciones.
Sección 3	**Diseño de Componentes**. Detalle de cada módulo, sus interfaces y responsabilidades.
Sección 4	**Diseño de Datos**. Detallado de las variables y de su base de datos correspondiente.
Sección 5	**Implementación**. Estructura del repositorio, convenciones de codificación, entorno de desarrollo y automatización de compilación.
Sección 6	**Pruebas y Calidad**. Estrategia de pruebas unitarias, de integración y de aceptación, así como métricas de cobertura.
Sección 7	**Despliegue y Operación**. Procedimiento de instalación, configuración de entornos y monitoreo en producción.
Bibliografía	Bibliografía asociada al documento.
Anexos	Documentación anexa.

1.1. Alcance

Este documento abarca las fases de diseño conceptual, diseño detallado e implementación del software, incluyendo la definición de la arquitectura, la construcción de los componentes, sus pruebas y su puesta en marcha. Aplica a todo el equipo de desarrollo y al personal de operaciones, de modo que cualquier cambio en la arquitectura o el código debe reflejarse aquí para garantizar:

- La **trazabilidad** de las decisiones de diseño y de los requerimientos hasta el código fuente.
- La **coherencia** entre la documentación, los artefactos de diseño y la implantación.
- La **interfaz** clara entre los stakeholders (usuarios finales y gestores de proyecto) y el equipo técnico.

1.2 . Términos, definiciones y términos abreviados

En esta sección se presenta de manera ordenada el glosario de conceptos, definiciones y abreviaturas utilizados a lo largo del documento. Su finalidad es aclarar el significado de los términos clave y asegurar que todos los lectores compartan un mismo entendimiento de la nomenclatura empleada.

1.2.1. Términos

Docker	Plataforma de virtualización ligera que permite empaquetar una aplicación y todas sus dependencias en contenedores.
Modbus TCP	Versión del protocolo Modbus que utiliza TCP/IP para la comunicación sobre redes Ethernet.
Streamlit	Framework de Python de código abierto que permite construir aplicaciones web interactivas orientadas a la visualización de datos.
CSV	Formato de archivo de texto plano que almacena datos en forma tabular, donde cada línea representa una fila y los valores están separados por comas (u otros delimitadores).
InfluxDB	Base de datos orientada a series temporales, optimizada para almacenar, consultar y analizar datos.

1.2.2. Términos abreviados

SDITD	Documento de Diseño, Implementación y Pruebas de Software. (Software Design, Implementation and Testing Document)
HW	Hardware.
SW	Software.
IPSEP	Instituto de Protecciones de Sistemas Eléctricos de Potencia

2. ARQUITECTURA DEL SISTEMA

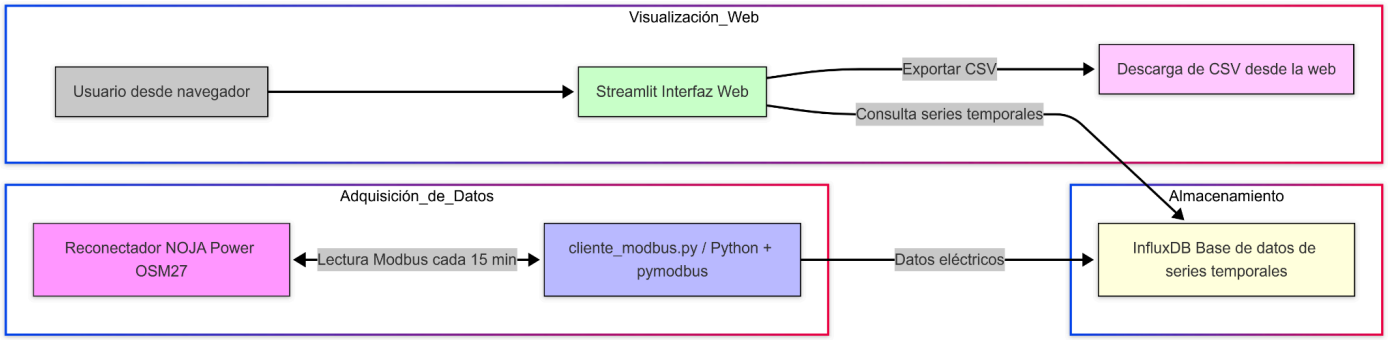
Este apartado describe la visión global de la solución, mostrando cómo se organiza el sistema en componentes y subsistemas. Incluye diagramas de alto nivel, las tecnologías y protocolos empleados, y explica las comunicaciones e integraciones entre los distintos módulos y servicios.

2.1. Visión general

El sistema está diseñado para realizar el monitoreo automatizado del reconectador NOJA Power OSM27 a través de una arquitectura modular dockerizada. Utiliza el protocolo **ModbusTCP** para comunicarse con el equipo, adquiere métricas eléctricas de interés y las almacena en una base de datos **InfluxDB**.

El sistema está compuesto por distintos servicios que se orquestan mediante **Docker Compose**, permitiendo portabilidad, despliegue sencillo y escalabilidad. La visualización de datos se realiza mediante una página web mediante **Streamlit**, y también se ofrece la posibilidad de exportar los datos en formato **CSV** para análisis externo. Toda la información es accesible desde la red del IPSEP y el sistema opera en ciclos de adquisición cada 15 minutos.

2.2. Diagrama de componentes



2.3. Subsistemas y módulos

El sistema está compuesto por los siguientes módulos principales:

- **Cliente Modbus**: servicio en Python que actúa como cliente ModbusTCP. Se conecta periódicamente al OSM27 para adquirir variables eléctricas como tensiones, corrientes, potencias y eventos. Emplea la librería *pymodbus*.

- **InfluxDB**: base de datos de series temporales. Recibe e indexa los datos eléctricos adquiridos, facilitando consultas eficientes y almacenamiento optimizado por tiempo. Se utiliza la estructura de mediciones con etiquetas y campos personalizables.

- **Exportador CSV**: módulo que permite la exportación de datos históricos desde InfluxDB a archivos *.csv*, bajo demanda o como tarea periódica. Es útil para respaldos o análisis externos.

- **Streamlit Interfaz Web**: aplicación web creada en **Streamlit**, encargada de presentar las métricas de forma gráfica e intuitiva. Permite acceder a los datos en tiempo real desde cualquier navegador conectado a la red, y soporta visualización por fecha, variable y estado.

Todos los servicios -salvo el Cliente Modbus- están containerizados y definidos en *docker-compose.yml*, comunicándose entre sí mediante red Docker interna.

2.4. Comunicaciones e integraciones

El sistema se basa en una arquitectura distribuida con los siguientes protocolos y tecnologías de integración:

- **ModbusTCP**: utilizado por el módulo *cliente_modbus* para comunicarse con el reconectador NOJA OSM27. El equipo actúa como servidor Modbus, mientras que el módulo Python realiza lecturas sobre registros configurados.

- **InfluxDB**: almacena los datos adquiridos en forma de series temporales. El cliente Modbus formatea los datos y los inserta utilizando la API de escritura de InfluxDB (vía HTTP o cliente Python).

- **Streamlit**: se encarga de la interfaz gráfica. Se conecta a InfluxDB usando la API de consulta o librerías cliente para recuperar y visualizar los datos dinámicamente en tiempo real.

- **Exportación CSV**: los datos pueden ser exportados en archivos CSV para procesamiento adicional o respaldo externo.

- **Docker Compose**: todos los servicios están containerizados para facilitar su integración y despliegue. La red Docker interna garantiza la comunicación entre contenedores sin necesidad de configuración adicional.

3. DISEÑO DE COMPONENTES

Esta sección profundiza en el diseño interno de cada módulo, describiendo su lógica de funcionamiento, las interfaces que expone y los patrones de diseño aplicados.

3.1. Módulo 1: *Cliente Modbus*

Función:

Este módulo es el encargado de adquirir las métricas eléctricas del reconectador NOJA Power OSM27 utilizando el protocolo **ModbusTCP**. Realiza la conexión como cliente, consulta registros específicos y procesa los datos para su posterior almacenamiento.

Contexto de uso:

Cada 15 minutos, ejecuta un ciclo de lectura, adquiere los datos y los publica en la base de datos InfluxDB.

Tecnologías y librerías utilizadas:

- Python 3.12
- *pymodbus* para comunicación Modbus.
- *influxdb-client* para insertar datos.

Flujo interno:

1. Cargar configuración (IP, puerto, registros, etc.)
2. Conectar al servidor Modbus (OSM27)
3. Leer los registros configurados (tensiones, corrientes, potencias, etc.)
4. Formatear los datos como puntos de serie temporal
5. Escribir en InfluxDB

Interfaces expuestas:

Este módulo no expone una API pública, ya que actúa como proceso autónomo. Su salida son los puntos insertados en la base de datos.

3.2. Módulo 2: InfluxDB (motor de almacenamiento)

Función:

Almacenar series temporales de datos eléctricos provenientes del módulo de adquisición.

Contexto de uso:

Ejecutado como contenedor independiente gestionado por Docker Compose. Permite almacenamiento, indexación y consultas eficientes sobre datos con marcas de tiempo.

Características clave:

- Organización por *measurement*, *fields* y *tags*
- Compresión de datos y manejo eficiente de grandes volúmenes
- Consultas por rango de tiempo.

Interfaces expuestas:

- API HTTP REST para escritura y lectura (8086)
- Cliente Python (*influxdb-client*) utilizado por los otros módulos

3.3. Módulo 3: Streamlit Interfaz Web

Función:

Interfaz gráfica para visualización de métricas históricas y en tiempo real. Permite al usuario explorar los datos de forma intuitiva mediante gráficos, filtros y selección de variables.

Contexto de uso:

Módulo implementado como contenedor Docker independiente, y ejecutado mediante Docker Compose como aplicación web accesible por navegador desde la red del IPSEP.

Tecnologías:

- Python 3.12-slim
- Streamlit
- influxdb-client (para consultas)

Flujo de funcionamiento:

1. El usuario accede a la interfaz en el navegador.
2. La aplicación solicita datos a InfluxDB según los filtros seleccionados.
3. Los datos se muestran en gráficos interactivos (líneas, áreas, etc.)
4. Permite exportar datos visualizados a CSV desde la interfaz.

Interfaces expuestas:

- Página web accesible vía *http://localhost:8501* (o IP del servidor)

4. DISEÑO DE DATOS

En este apartado se define el modelo de datos que sustenta la aplicación, detallando entidades, relaciones y estructura de almacenamiento.

4.1. Esquemas de Base de Datos

El sistema utiliza **InfluxDB** como base de datos especializada para el almacenamiento de series temporales. A diferencia de las bases de datos relacionales, InfluxDB no emplea tablas ni claves primarias/foráneas tradicionales. En su lugar, los datos se estructuran en **measurements**, que contienen puntos con:

- **timestamp** (marca temporal obligatoria),
- **fields** (valores numéricos medidos), y
- **tags** (atributos categóricos que permiten segmentación y filtrado).

4.2. Diccionario de Datos

Se detalla a continuación la estructura lógica de la entidad principal del sistema, mediciones, incluyendo campos y etiquetas relevantes, sus tipos de datos, restricciones y descripciones:

Tabla	Columna	Tipo de dato	Descripciones
mediciones	Ia	integer	Corriente en fase A (Amperes)
mediciones	Ib	integer	Corriente en fase B (Amperes)
mediciones	Ic	integer	Corriente en fase C (Amperes)
mediciones	Ua	integer	Tensión en fase A (Voltios)
mediciones	Ub	integer	Tensión en fase B (Voltios)
mediciones	Uc	integer	Tensión en fase C (Voltios)

mediciones	Ur	integer	Tensión en secuencia R (Voltios)
mediciones	Us	integer	Tensión en secuencia S (Voltios)
mediciones	Ut	integer	Tensión en secuencia T (Voltios)
mediciones	Uab	integer	Tensión entre fases A-B (Voltios)
mediciones	Ubc	integer	Tensión entre fases B-C (Voltios)
mediciones	Uca	integer	Tensión entre fases C-A (Voltios)
mediciones	Urs	integer	Tensión entre secuencias R-S (Voltios)
mediciones	Ust	integer	Tensión entre secuencias S-T (Voltios)
mediciones	Utr	integer	Tensión entre secuencias T-R (Voltios)
mediciones	kVA A	integer	Potencia aparente fase A (VA)
mediciones	kVA B	integer	Potencia aparente fase B (VA)
mediciones	kVA C	integer	Potencia aparente fase C (VA)
mediciones	kW A	integer	Potencia activa fase A (Watt)
mediciones	kW B	integer	Potencia activa fase B (Watt)
mediciones	kW C	integer	Potencia activa fase C (Watt)
mediciones	kVAr A	integer	Potencia reactiva fase A (VAR)

mediciones	kVAr B	integer	Potencia reactiva fase B (VAR)
mediciones	kVAr C	integer	Potencia reactiva fase C (VAR)
mediciones	kVA total	integer	Potencia aparente total (VA)
mediciones	kW total	integer	Potencia activa total (Watt)
mediciones	kVAr total	integer	Potencia reactiva total (VAR)
mediciones	Freq abc	float	Frecuencia (Hz)
mediciones	equipo	string (TAG)	Identificador del equipo (ej. “OSM27”)
mediciones	ubicación	string (TAG)	Ubicación geográfica o lógica del dispositivo

5. IMPLEMENTACIÓN

La totalidad de los archivos nombrados se encuentran en un repositorio de la plataforma *Github* bajo el nombre *danunziata/TP_FINAL-TCP_IP_2025-Grupo_2*.

5.1. Estructura del Repositorio

```
TP_FINAL-TCP_IP_2025-Grupo_2/
├── Documentacion/          # Sitio MKDocs con manuales, diagramas e imágenes
├── Streamlit/              # Aplicación web para visualizar datos
├── client_modbusTCP/       # Cliente Modbus que guarda mediciones en InfluxDB
├── database_InfluxDB/      # Configuración de InfluxDB (Docker Compose, datos)
├── server_modbusTCP/       # Servidor Modbus TCP simulado (Dockerfile y scripts)
├── iniciar_entorno.sh      # Script para levantar cliente, servidor y base de datos
├── detener_entorno.sh      # Script para detener servicios en ejecución
└── .github/workflows/      # Workflow de GitHub Actions para notificaciones
```

Carpetas principales

client_modbusTCP

Cliente Modbus escrito en Python que se conecta al servidor y guarda registros en *InfluxDB*. Incluye los scripts *client.py*, *client_once.py* y su *requirements.txt*.

database_InfluxDB

Configuración de la base de datos *InfluxDB*. Contiene el *docker-compose.yaml* para levantar la base y una carpeta *data/* con archivos persistentes. Su *README* explica cómo instalar y operar InfluxDB.

Streamlit

Aplicación de visualización en Streamlit. Incluye *streamlit_app.py*, el archivo *config.yaml* con usuarios y un *requirements.txt* con dependencias. También alberga imágenes y recursos para la interfaz.

Documentación

Documentos del proyecto en formato *Markdown* y recursos para *MkDocs*. El archivo *mkdocs.yml* define la navegación del sitio con secciones para diagramas, requerimientos e implementación del sistema.

La carpeta *server_modbusTCP* no es analizada ni tenida en cuenta ya que el servidor será el propio equipo para el cual fue desarrollada la aplicación. Esta carpeta presenta un contenedor docker que simula la generación de datos del equipo para que el cliente pueda consultar y enviarlos hacia la base de datos, siguiendo con el flujo presentado.

Scripts y automatización

iniciar_entorno.sh

Automatiza la puesta en marcha de todo el entorno: verifica e instala dependencias, inicia el cliente a partir de su script de python, levanta InfluxDB y lanza la app de Streamlit.

detener_entorno.sh

Detiene y elimina los contenedores de Docker y finaliza el cliente Modbus en ejecución.

Integración continua

En *.github/workflows/deploy.yml* se define un workflow de GitHub Actions que envía notificaciones a un webhook de Discord cada vez que se realiza un push o una pull request. El flujo no compila ni ejecuta pruebas; únicamente envía mensajes.

5.2. Entorno de Desarrollo

El entorno de desarrollo del proyecto se configura automáticamente mediante el script iniciar_entorno.sh*, incluido en el repositorio. Este script instala todas las dependencias necesarias y levanta los servicios esenciales para la operación del sistema, permitiendo una inicialización reproducible del entorno completo.*

Lenguaje base y contenedor principal

- ***Python 3.11 (imagen base: python:3.11-slim)***
  Utilizado como entorno de ejecución principal para el servidor Modbus TCP simulado. Esta imagen ligera de Python garantiza compatibilidad con librerías de networking industrial y bajo consumo de recursos.

Paquetes Python requeridos

Para el funcionamiento completo del sistema, tanto en su backend como en la interfaz web, se utilizan las siguientes librerías:

- ***influxdb-client==1.38.0**: Cliente oficial para InfluxDB v2, utilizado para la escritura y consulta de series temporales desde el servidor Modbus simulado.*

- ***streamlit** (última versión estable: **1.35.0**)*
  Framework para construir la interfaz web interactiva de visualización.

- ***pandas** (**v2.2.2**)*
  Para manipulación y procesamiento de datos tabulares.

- ***plotly** (**v5.21.0**)*
  Generación de gráficos dinámicos y visualizaciones interactivas.

- ***openpyxl** (**v3.1.2**)*
  Soporte para exportación e importación de archivos Excel desde la interfaz.

- ***mitosheet** (**v0.1.502**)*
  Visualización tipo hoja de cálculo para edición rápida de datasets en Streamlit.

- ***streamlit-autorefresh** (**v0.0.4**)*
  Permite actualizar automáticamente la interfaz en intervalos definidos.

- ***streamlit-authenticator** (**v0.2.2**)*
  Módulo de autenticación para proteger el acceso a la interfaz Streamlit.

- ***pymodbus** (**v3.6.6**)*
  Librería utilizada para simular el servidor Modbus TCP y manejar los registros expuestos por el mismo.

Contenedores y orquestación

- ***Docker** (**v25.0.3**)*
  Plataforma para contenerización de la aplicación Python, la base de datos InfluxDB y servicios auxiliares.

- ***Docker Compose** (**v2.27.0**)*
  Herramienta de orquestación para levantar y gestionar los servicios de forma declarativa (archivo

docker-compose.yml). Se encarga de iniciar tanto el servidor Modbus como la base de datos InfluxDB.

6. PRUEBAS Y CALIDAD

En esta sección se define la estrategia completa de verificación y validación del software.

6.1. Checklist para verificación de obtención de datos

- Correcta conexión con el equipo NOJA Power
- Obtención de datos a partir del protocolo Modbus TCP.
- Guardado de datos en base de datos InfluxDB.
- Visualización de datos en interfaz web a través de Streamlit.
- Posibilidad de exportar datos a partir del formato ".csv".
- Acceso desde la red interna de la facultad.

7. DESPLIEGUE Y OPERACIÓN

Esta sección establece los procedimientos para la instalación, configuración y mantenimiento del sistema en los entornos de pruebas y producción.

7.1. Pre-requisitos

Para instalar y operar la solución de monitoreo es necesario contar con un entorno que satisfaga ciertos requisitos de hardware, software, red y permisos. A continuación se describen los elementos esenciales basados en la documentación y scripts del repositorio.

Software básico

El software básico necesario para la implementación del sistema es el detallado en la sección anterior. El mismo se instala automáticamente al ejecutar el script *iniciar_entorno.sh.*

Configuración de red y puertos

Los distintos componentes utilizan los siguientes puertos locales, por lo que deben estar disponibles y habilitados en la máquina de despliegue:

***5020** – Puerto del servidor Modbus TCP según docker-compose.yaml del servidor:*

8086 *– Puerto para la interfaz web y API de InfluxDB*

8501 *– Puerto por defecto de la aplicación Streamlit, indicado en la guía de uso.*

Requisitos de hardware y permisos

Se recomienda contar con un equipo capaz de ejecutar Docker y Docker Compose con al menos algunos gigabytes de memoria disponible para los contenedores de InfluxDB y del simulador Modbus. Además, será necesario:

- ***Permisos de administrador o pertenencia al grupo docker*** *para ejecutar los contenedores.*
- ***Conexión de red local*** *que permita acceder a los puertos indicados (5020, 8086 y 8501).*
- ***Python 3.x*** *(se usa la imagen python:3.11-slim en el Dockerfile del servidor) para ejecutar los scripts fuera de contenedores si se desea.*

Con estos pre-requisitos cubiertos, es posible iniciar el entorno mediante el script iniciar_entorno.sh, que automatiza la instalación de dependencias y el arranque de los componentes del sistema

7.2. Procedimiento de Despliegue

Se describen de forma detallada los pasos, scripts y comandos necesarios para instalar y actualizar el sistema (realizado para un sistema operativo *Linux Xubuntu 22.04* y similares):

- Poseer el conjunto completo de scripts detallados en la sección 5, los cuales se encuentran en el repositorio de *GitHub* nombrado.
- Instalar los requerimientos tanto del cliente modbus como de la interfaz web Streamlit, a partir de los comandos *pip install -r client_modbusTCP/requirements.txt* para el cliente, y *pip install -r Streamlit/requirements.txt*
- Levantar la base de datos mediante docker compose, a través del comando *docker compose up* dentro de la carpeta "*database_InfluxDB*".
- Iniciar el cliente modbus, el cual debe apuntar a la dirección IP y puerto correspondientes al equipo, mediante el comando *python3 client.py* dentro de la carpeta *"client_modbusTCP"*.
- Iniciar finalmente la aplicación web para poder visualizar los datos correspondientes, mediante el comando *streamlit run streamlit_app.py* dentro de la carpeta *"Streamlit"*.

Para detener el entorno, se deben ejecutar los comandos *docker compose down* (para dar de baja un contenedor docker) y *pkill -f* (para matar un proceso python) según corresponda.

Todo esto ha sido automatizado a partir de dos scripts de ejecución y detención: *iniciar_entorno.sh* y *detener_entorno.sh* respectivamente. Mediante la ejecución de estos dos códigos *bash* se puede realizar la inicialización y detención automática de todo el entorno, ejecutandolos desde la carpeta raíz del repositorio *GitHub*.

8. BIBLIOGRAFÍA

IEEE Standard 1016-2009: “Software Design Descriptions (SDD)”

Pressman, Roger S. *Ingeniería del Software - Un Enfoque Práctico*. Cuarta Edición ed., McGraw-Hill Companies, 1998.

Sommerville, Ian. *Ingeniería de software*. Novena Edición ed., Pearson Education Inc., 2012.

Thyler, Richard H., et al. *Software Requirements Engineering*. Second Edition ed., Wiley-IEEE Computer Society Pr., 1997.

Roos, Patrick (2023). “The Ultimate Guide To Software Architecture Documentation.”

Mezzalira, Luca (2024). “How to Document Software Architecture: Techniques and Best Practices.”

NOJA Power (2024). *Guía de comunicación Ethernet con el reconectador NOJA OSM15-16-800*.

NOJA Power (2021). *Modbus in Reclosers*. https://www.nojapower.com/expertise/2021/modbus-in-reclosers

InfluxData (2025). *InfluxDB Client Library for Python – Documentation*. https://github.com/influxdata/influxdb-client-python

Streamlit Inc. (2025). *Streamlit Documentation*. https://docs.streamlit.io

Modbus.org. (2024). *Modbus Application Protocol Specification V1.1b3*. https://modbus.org