UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

# INTEGRACIÓN DE UN MODELO DE INTELIGENCIA ARTIFICIAL LLM CON ENFOQUE RAG PARA DAR SOPORTE AL MANTENIMIENTO, OPERACIÓN Y DIAGNÓSTICO DE LA PLANTA DE PROCESOS ANÁLOGOS (PPA)


**Autores**

Christian David Puerto Ramírez — 20242583011

Daniel Felipe Cárdenas Sánchez — 20242573014

**Director del proyecto**

Jorge Eduardo Porras Bohada

**Universidad Distrital Francisco José de Caldas**

Facultad Tecnológica

Ingeniería en Control y Automatización



**Bogotá D.C., Colombia**

**Septiembre de 2026**

# DEDICATORIA

Dedicamos este trabajo, en primer lugar, a nuestras familias, por su apoyo incondicional, comprensión y confianza durante todo nuestro proceso de formación profesional. Su acompañamiento y motivación fueron fundamentales para superar cada uno de los retos que se presentaron a lo largo de este camino.

A nuestros padres, por sus enseñanzas, sacrificios y por inculcarnos los valores de responsabilidad, perseverancia y compromiso que nos permitieron alcanzar esta meta.

A todas las personas que estuvieron presentes durante nuestra formación académica y que, de una u otra manera, contribuyeron a que este proyecto se hiciera realidad.

Finalmente, dedicamos este logro a nuestro propio esfuerzo, como reconocimiento a la dedicación, la paciencia y la constancia que cada uno aportó para llegar hasta este momento.

# AGRADECIMIENTOS

Agradecemos principalmente a nuestras familias por el apoyo, la comprensión y la motivación que nos brindaron a lo largo de nuestra formación profesional. Su acompañamiento fue fundamental para superar las dificultades y continuar avanzando hasta alcanzar esta meta.

A la Universidad Distrital Francisco José de Caldas, por brindarnos la formación académica, los conocimientos y los espacios necesarios para nuestro desarrollo profesional y para la realización de este proyecto de grado.

A nuestros docentes, por compartir sus conocimientos y experiencias, y por contribuir significativamente a nuestra formación académica y profesional.

De manera especial, agradecemos a nuestro director de proyecto, el profesor Jorge Eduardo Porras Bohada, por su orientación, acompañamiento, disposición y aportes durante el desarrollo de este trabajo. Sus recomendaciones fueron fundamentales para avanzar en las diferentes etapas del proyecto. Agradecemos también al evaluador del proyecto, el profesor Henry Alberto Hernández Martínez, cuyas observaciones permitieron mejorar la estructura, el rigor y la claridad de este documento.

También agradecemos a nuestros compañeros y a todas las personas que, directa o indirectamente, contribuyeron con sus conocimientos, colaboración y palabras de motivación durante el desarrollo de este proyecto.

# CONTENIDO

**DEDICATORIA** ..... **2**
**AGRADECIMIENTOS** ..... **3**
**LISTA DE SIGLAS Y ABREVIATURAS** ..... **8**
**RESUMEN** ..... **9**
**ABSTRACT** ..... **10**
**INTRODUCCIÓN** ..... **11**
**1. PLANTEAMIENTO DEL PROBLEMA** ..... **12**
1.1 DESCRIPCIÓN DEL CASO DE ESTUDIO ..... 12
1.2 PREGUNTA DE INVESTIGACIÓN ..... 15
1.3 JUSTIFICACIÓN ..... 16
**2. OBJETIVOS** ..... **17**
2.1 OBJETIVO GENERAL ..... 17
2.2 OBJETIVOS ESPECÍFICOS ..... 17
**3. ALCANCE Y LIMITACIONES** ..... **18**
3.1 ALCANCE ..... 18
3.2 LIMITACIONES ASUMIDAS ..... 18
**4. ANTECEDENTES Y ESTADO DEL ARTE** ..... **19**
4.1 TRABAJOS PREVIOS SOBRE LA PPA ..... 19
4.2 DIAGNÓSTICO BASADO EN SEÑALES Y MONITOREO CONTINUO ..... 19
4.3 LLM Y RAG APLICADOS AL MANTENIMIENTO ..... 20
4.4 INFRAESTRUCTURA DE ADQUISICIÓN Y ALMACENAMIENTO VECTORIAL ..... 20
4.5 SÍNTESIS Y BRECHA IDENTIFICADA ..... 20
**5. MARCO TEÓRICO** ..... **22**
5.1 MANTENIMIENTO BASADO EN CONDICIÓN ..... 22
5.2 MODELOS DE LENGUAJE DE GRAN TAMAÑO ..... 22
5.3 EMBEDDINGS Y SIMILITUD SEMÁNTICA ..... 23
5.4 RECUPERACIÓN VECTORIAL ..... 24
5.5 GENERACIÓN AUMENTADA POR RECUPERACIÓN ..... 24
5.6 TOOL CALLING Y AGENTES ..... 25
5.7 ALUCINACIÓN EN MODELOS DE LENGUAJE ..... 25
5.8 EVALUACIÓN DE SISTEMAS RAG ..... 26
5.9 INTEGRACIÓN INDUSTRIAL Y ADQUISICIÓN DE DATOS ..... 27
5.10 PROGRAMACIÓN ASISTIDA POR IA ..... 27
**6. MARCO LEGAL Y NORMATIVO** ..... **29**
6.1 SEGURIDAD Y SALUD EN EL TRABAJO ..... 29
6.2 PROTECCIÓN DE DATOS PERSONALES ..... 30
6.3 PROPIEDAD INTELECTUAL ..... 30
6.4 SEGURIDAD INFORMÁTICA Y CIBERSEGURIDAD INDUSTRIAL ..... 30
6.5 MONITOREO DE CONDICIÓN ..... 31

**7. METODOLOGÍA** ..... **32**
7.1 ENFOQUE METODOLÓGICO ..... 32
7.2 FASE 1: ANÁLISIS DE REQUISITOS Y RESTRICCIONES ..... 33
7.2.1 Restricción de partida y caracterización del hardware ..... 33
7.2.2 Requisitos del sistema ..... 34
7.2.3 Inventario y diagnóstico de la documentación ..... 35
7.3 FASE 2: DISEÑO DE LA ARQUITECTURA ..... 35
7.3.1 Bloques funcionales ..... 35
7.3.2 Decisiones técnicas ..... 37
7.3.3 Principio de red ..... 38
7.4 FASE 3: IMPLEMENTACIÓN E INTEGRACIÓN ..... 38
7.4.1 Experimentos descartados ..... 39
7.5 FASE 4: PROTOCOLO EXPERIMENTAL Y VALIDACIÓN ..... 40
7.5.1 Banco de preguntas y unidad de análisis ..... 41
7.5.2 Condiciones de ejecución ..... 41
7.5.3 Rúbricas y criterios de calificación ..... 41
7.5.4 Métricas ..... 42
7.5.5 Tratamiento estadístico ..... 43
7.5.6 Consistencia del evaluador ..... 43
7.5.7 Criterios de validación ..... 44
7.5.8 Parámetros de reproducibilidad ..... 44
**8. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA** ..... **46**
8.1 MODELOS DE LENGUAJE Y CONFIGURACIÓN DE LA INFERENCIA ..... 46
8.2 MOTOR DE RAZONAMIENTO: TOOL CALLING Y CAPA DETERMINISTA ..... 47
8.2.1 Herramientas disponibles ..... 49
8.2.2 Capa de respuestas deterministas ..... 49
8.2.3 Conocimiento curado de la planta ..... 50
8.3 PIPELINE RAG: DE LOS DOCUMENTOS A LA RESPUESTA ..... 51
8.3.1 Ingesta de documentos ..... 51
8.3.2 Fragmentación del texto ..... 52
8.3.3 Embeddings e indexación idempotente ..... 53
8.3.4 Recuperación híbrida ..... 54
8.4 BASE DE DATOS DEL HISTÓRICO DE SENSORES ..... 56
8.5 BACKEND DE SERVICIOS ..... 58
8.6 INTERFAZ DE USUARIO ..... 60
8.6.1 Pestaña Documentos: trazabilidad hasta la línea de origen ..... 61
8.6.2 Filtro por sensor en Datos reales del PLC ..... 64
8.7 TRAZABILIDAD DE UNA CONSULTA ..... 65
8.8 ADQUISICIÓN DE DATOS DEL PLC Y SIMULADOR ..... 67
8.8.1 Servicio de adquisición EtherNet/IP ..... 67
8.8.2 Simulador en Node-RED ..... 69
8.9 MOTOR DE ALERTAS DE MANTENIMIENTO ..... 70
8.9.1 Reglas implementadas ..... 71
8.9.2 Correlación temporal entre tags ..... 72

8.9.3 Validación con sesiones reales ... 73
8.9.4 Ubicación física y fotografía del elemento en alerta ... 77
8.10 VERIFICACIÓN DE LA INTEGRACIÓN ... 78
8.11 APOYO DE HERRAMIENTAS DE IA EN LA CONSTRUCCIÓN DEL SISTEMA ... 79
**9. ANÁLISIS DE RESULTADOS ... 81**
9.1 PROTOCOLO APLICADO ... 81
9.2 EVOLUCIÓN DE LA EXACTITUD GLOBAL ... 81
9.3 RESULTADO DETALLADO DE LA PRUEBA 6 ... 82
9.4 HALLAZGO METODOLÓGICO: EL RUIDO DEL CRITERIO DE REFERENCIA ... 83
9.5 SELECCIÓN DE HERRAMIENTA FRENTE A CALIDAD DE LA SÍNTESIS ... 84
9.6 PRUEBA DE CONSISTENCIA ... 85
9.6.1 Ejemplo de respuesta determinista ... 86
9.6.2 Ejemplo de pregunta fuera de dominio ... 86
9.6.3 Ejemplo de solicitud de contenido creativo ... 86
9.7 CASOS DE ESTUDIO: FALLAS ENCONTRADAS Y CORREGIDAS ... 87
9.8 SEGUNDO CICLO DE EVALUACIÓN: RÚBRICA FLEXIBLE Y MEJORA DIRIGIDA POR PATRONES ... 87
9.8.1 Resultado de la Prueba 7 y patrones sistémicos ... 88
9.8.2 Correcciones aplicadas ... 88
9.8.3 Resultado de la Prueba 8 ... 89
9.8.4 Estado de los patrones tras las correcciones ... 90
9.9 CUMPLIMIENTO DE LOS OBJETIVOS ... 91
9.10 DISCUSIÓN ... 92
9.11 LIMITACIONES ... 92
**10. APORTES Y RECOMENDACIONES ... 94**
10.1 APORTES ... 94
10.2 RECOMENDACIONES Y TRABAJO FUTURO ... 94
**11. CONCLUSIONES ... 96**
**BIBLIOGRAFÍA ... 98**
**ANEXO A. BITÁCORA DE DESARROLLO ... 106**
**ANEXO B. ÍNDICE DE IMÁGENES DEL SISTEMA ... 109**
**ANEXO C. CÓDIGO FUENTE: api/config.py ... 110**
**ANEXO D. CÓDIGO FUENTE: api/conocimiento_plc.py ... 111**
**ANEXO E. CÓDIGO FUENTE: api/db.py ... 113**
**ANEXO F. CÓDIGO FUENTE: api/main.py ... 115**
**ANEXO G. CÓDIGO FUENTE: api/ollama_client.py ... 122**
**ANEXO H. CÓDIGO FUENTE: api/plc_store.py ... 125**
**ANEXO I. CÓDIGO FUENTE: api/rag.py ... 128**
**ANEXO J. CÓDIGO FUENTE: ingest/chunker.py ... 148**
**ANEXO K. CÓDIGO FUENTE: ingest/config.py ... 150**
**ANEXO L. CÓDIGO FUENTE: ingest/embeddings.py ... 151**
**ANEXO M. CÓDIGO FUENTE: ingest/ingest.py ... 152**

**ANEXO N. CÓDIGO FUENTE: ingest/loaders.py** ..... **154**
**ANEXO O. CÓDIGO FUENTE: ingest/make_dummy_pdf.py** ..... **157**
**ANEXO P. CÓDIGO FUENTE: ingest/qdrant_store.py** ..... **158**
**ANEXO Q. CÓDIGO FUENTE: ui/app.py** ..... **160**
**ANEXO R. FUENTES PRIMARIAS DEL PROYECTO** ..... **165**
**ANEXO S. BANCO DE 145 PREGUNTAS DE EVALUACIÓN** ..... **166**
**ANEXO T. PRUEBA DE CONSISTENCIA: LAS CINCO RESPUESTAS POR PREGUNTA** ..... **172**

# LISTA DE SIGLAS Y ABREVIATURAS

**Tabla 1.** Siglas y abreviaturas utilizadas en el documento.

| Sigla | Significado |
|---|---|
| API | Interfaz de programación de aplicaciones (*Application Programming Interface*) |
| CBM | Mantenimiento basado en condición (*Condition-Based Maintenance*) |
| CIP | Protocolo Industrial Común (*Common Industrial Protocol*) |
| CPU | Unidad central de procesamiento |
| GPU | Unidad de procesamiento gráfico |
| HMI | Interfaz humano-máquina |
| HNSW | Grafos jerárquicos navegables de mundo pequeño (*Hierarchical Navigable Small World*) |
| IC | Intervalo de confianza |
| IIoT | Internet industrial de las cosas |
| LLM | Modelo de lenguaje de gran tamaño (*Large Language Model*) |
| MCSA | Análisis de firma de corriente del motor (*Motor Current Signature Analysis*) |
| MTTD | Tiempo medio de detección (*Mean Time To Detect*) |
| PLC | Controlador lógico programable |
| PPA | Planta de Procesos Análogos |
| RAG | Generación aumentada por recuperación (*Retrieval-Augmented Generation*) |
| REST | Transferencia de estado representacional |
| RF / RNF | Requisito funcional / Requisito no funcional |
| SG-SST | Sistema de Gestión de Seguridad y Salud en el Trabajo |
| SQL | Lenguaje de consulta estructurado |

*Fuente: elaboración propia.*

# RESUMEN

El mantenimiento industrial ha evolucionado desde esquemas reactivos o basados en calendarios fijos hacia estrategias que utilizan información real de los equipos para decidir cuándo intervenir. La Planta de Procesos Análogos (PPA) de la Universidad Distrital Francisco José de Caldas, un banco didáctico con tres tanques interconectados, sensores de nivel, temperatura y presión diferencial y un PLC industrial Allen-Bradley, presenta una dificultad común a las plantas con años de uso: su documentación técnica se acumuló en distintas épocas (manuales, bitácoras de práctica y diccionarios de tags) y su consulta manual resulta lenta, en especial para quien se enfrenta al sistema por primera vez.

En este trabajo se diseñó, implementó y validó un asistente conversacional que combina un modelo de lenguaje de gran tamaño (LLM) con un enfoque de generación aumentada por recuperación (RAG). El asistente responde preguntas técnicas sobre la PPA citando el documento de origen de cada afirmación y consulta, mediante *tool calling*, el histórico real de sensores capturado desde el PLC por EtherNet/IP. La totalidad del sistema se ejecuta en local, en un equipo sin GPU; esta restricción de diseño condicionó la mayoría de las decisiones de arquitectura.

La validación se realizó con un banco propio de 145 preguntas distribuidas en tres grupos (documentación, base de datos y preguntas combinadas), calificadas con una rúbrica de tres niveles y, posteriormente, con una rúbrica de seis niveles. La versión base alcanzó una exactitud global de 0.29. La incorporación de una capa de respuestas deterministas, que impide que el modelo redacte libremente los datos numéricos ya recuperados, elevó la exactitud a 0.624 (IC 95 %: 0.552–0.696) con la rúbrica de tres niveles. Un segundo ciclo de mejora, dirigido por nueve patrones sistémicos de error, llevó la exactitud de 0.618 a 0.723 con la rúbrica de seis niveles. La selección de herramienta superó el 84 % de acierto desde la primera versión evaluada, lo que indica que la principal fuente de error no fue la decisión sobre qué consultar, sino la redacción final de la respuesta. Una prueba exploratoria de consistencia (15 preguntas, 5 repeticiones) no registró alucinaciones en las respuestas

deterministas, frente a un comportamiento intermitente en un tercio de las preguntas redactadas por el modelo.

**Palabras clave:** modelos de lenguaje de gran tamaño; generación aumentada por recuperación (RAG); mantenimiento predictivo; controlador lógico programable (PLC); alucinación; *tool calling*.

# ABSTRACT

Industrial maintenance has evolved from reactive or fixed-schedule schemes toward strategies that use real equipment data to decide when to intervene. The Analog Process Plant (PPA) at Universidad Distrital Francisco José de Caldas, a teaching test bench with three interconnected tanks, level, temperature and differential pressure sensors, and an Allen-Bradley industrial PLC, faces a problem common to plants with years of operation: its technical documentation accumulated over different periods (manuals, lab logs and tag dictionaries), and consulting it manually is slow, particularly for first-time users.

This work designed, implemented and validated a conversational assistant that combines a large language model (LLM) with a retrieval-augmented generation (RAG) approach. The assistant answers technical questions about the PPA, citing the source document of each statement, and uses tool calling to query the real sensor history captured from the PLC over EtherNet/IP. The entire system runs locally on a computer without a GPU, a design constraint that shaped most architectural decisions.

Validation used a custom bank of 145 questions divided into three groups (documentation, database and combined questions), graded first with a three-level rubric and later with a six-level rubric. The baseline version reached an overall accuracy of 0.29. Adding a deterministic answer layer, which prevents the model from freely rewording numeric data that has already been retrieved, raised accuracy to 0.624 (95 % CI: 0.552–0.696) under the three-level rubric. A second improvement cycle, driven by nine systemic error patterns, increased accuracy from 0.618 to 0.723 under the six-level rubric. Tool selection exceeded 84 % accuracy from the first evaluated version, indicating that the main source of error was not deciding what to query but writing the final answer. An exploratory consistency test (15 questions, 5 repetitions) recorded no hallucinations in deterministic answers, compared with intermittent behavior in one third of the model-written answers.


**Keywords:** large language models; retrieval-augmented generation (RAG); predictive maintenance; programmable logic controller (PLC); hallucination; tool calling.

# INTRODUCCIÓN

Este documento presenta el desarrollo de un sistema de inteligencia artificial basado en un modelo de lenguaje de gran tamaño (LLM) con enfoque de generación aumentada por recuperación (RAG), construido para apoyar la comprensión, el diagnóstico y el mantenimiento de la Planta de Procesos Análogos (PPA) de la Universidad Distrital Francisco José de Caldas. El punto de partida es un problema concreto: la documentación técnica de la PPA se acumuló durante más de una década de prácticas y proyectos de grado sucesivos [1], [2], y hoy se encuentra distribuida en manuales, bitácoras y diccionarios de tags que no tienen la misma calidad ni el mismo grado de actualización. Para un estudiante u operador que llega por primera vez a la planta, comprender con rapidez y precisión su funcionamiento es una tarea más compleja de lo deseable.

Con el fin de atender esta problemática se diseñó un sistema que integra, en una única interfaz conversacional, la consulta semántica de la documentación técnica y la lectura de los datos operativos que la planta registra a través de su PLC. El sistema determina automáticamente cuál de las dos fuentes utilizar, o si requiere ambas, según la naturaleza de la consulta. La ejecución completamente local, sobre el hardware disponible en el laboratorio y sin servicios de inteligencia artificial en la nube, se estableció como restricción de diseño desde el inicio del proyecto y condicionó la mayoría de las decisiones de arquitectura.

El funcionamiento del sistema se verificó de manera cuantitativa: se construyó un banco de 145 preguntas técnicas sobre la planta, que se ejecutó en varias ocasiones contra el sistema en producción. Este procedimiento permitió medir la evolución de la exactitud a lo largo de las mejoras e identificar en qué tipo de pregunta fallaba el sistema y por qué.

El documento se organiza en once capítulos. El capítulo 1 plantea el problema y su justificación; el capítulo 2 presenta los objetivos; el capítulo 3 delimita el alcance y las limitaciones asumidas. El capítulo 4 revisa los antecedentes y el estado del arte, y el capítulo 5 desarrolla el marco teórico. El capítulo 6 expone el marco legal y normativo aplicable. El capítulo 7 describe la metodología en cuatro fases reproducibles y el

capítulo 8 detalla la implementación de cada componente. El capítulo 9 analiza los resultados de las rondas de evaluación; el capítulo 10 recoge los aportes y las recomendaciones, y el capítulo 11 presenta las conclusiones.

# 1. PLANTEAMIENTO DEL PROBLEMA

## 1.1 DESCRIPCIÓN DEL CASO DE ESTUDIO

La comprensión del funcionamiento de un sistema automatizado es un reto vigente tanto en la industria como en la academia. La complejidad creciente de los procesos y la dispersión de la información técnica en fuentes y formatos heterogéneos dificultan el diagnóstico y prolongan las labores de mantenimiento [3], [4]. La PPA ilustra esta situación: es el mismo banco de pruebas descrito en los trabajos de grado de 2016 y 2018 [1], [2], reconfigurado en 2024, de modo que su documentación reúne manuales de práctica, bitácoras de laboratorio, diccionarios de tags y planos eléctricos de distinta vigencia. Preguntas que deberían resolverse de inmediato, como la función del sensor LSH-100 o el rango de la variable PE-102, obligan a revisar varios documentos a la vez y conllevan el riesgo de encontrar información que ya no corresponde a la configuración actual de la planta.

Las figuras 1 a 3 muestran el estado físico de la planta durante el desarrollo de este proyecto: el conjunto de los tres tanques interconectados con su estructura de soporte, el gabinete de control con el PLC Allen-Bradley y la pantalla HMI, y el detalle del cableado y los relés donde se alojan los sensores de temperatura y flujo referenciados a lo largo del documento.

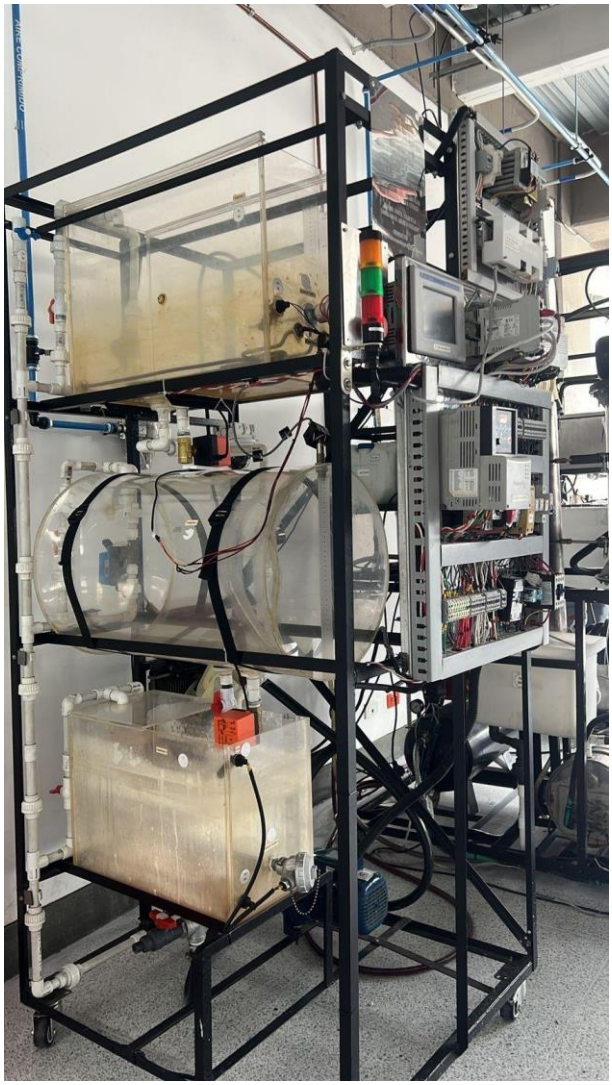


**Figura 1.** Vista general de la Planta de Procesos Análogos (PPA): los tres tanques interconectados sobre su estructura de soporte, con el gabinete de control a la derecha.

*Fuente: Elaboración propia (fotografía de la planta).*

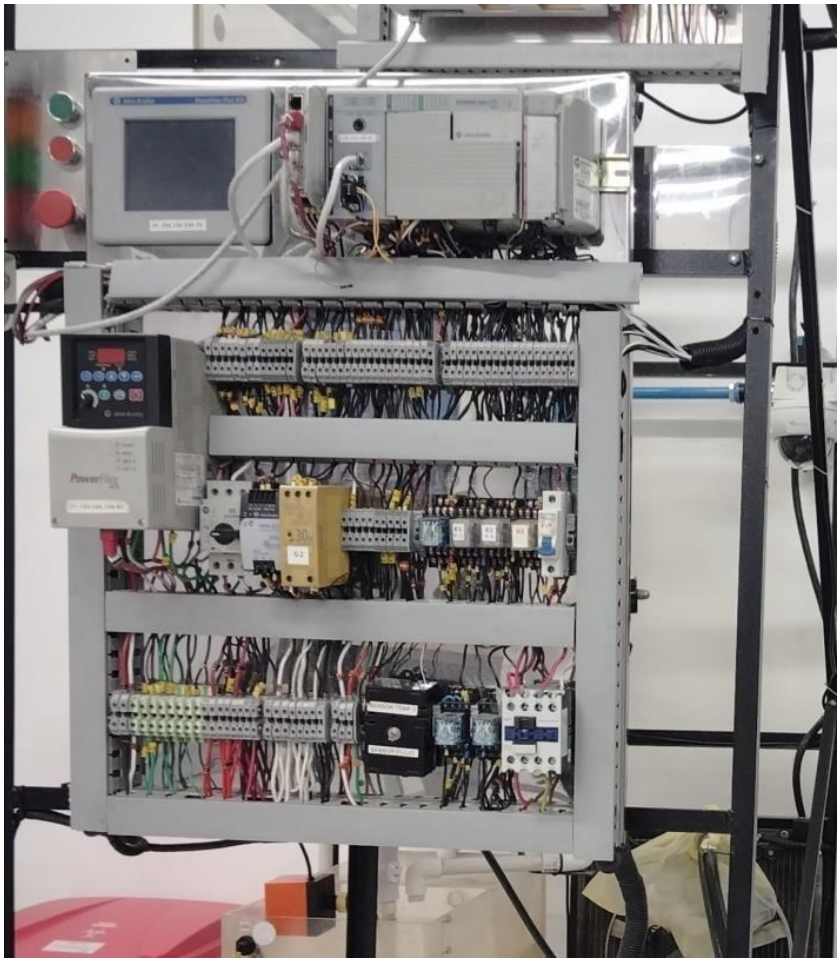


**Figura 2.** Gabinete de control: PLC Allen-Bradley CompactLogix, pantalla HMI PowerView Plus, variador PowerFlex y luces piloto de estado.

*Fuente: Elaboración propia (fotografía de la planta).*

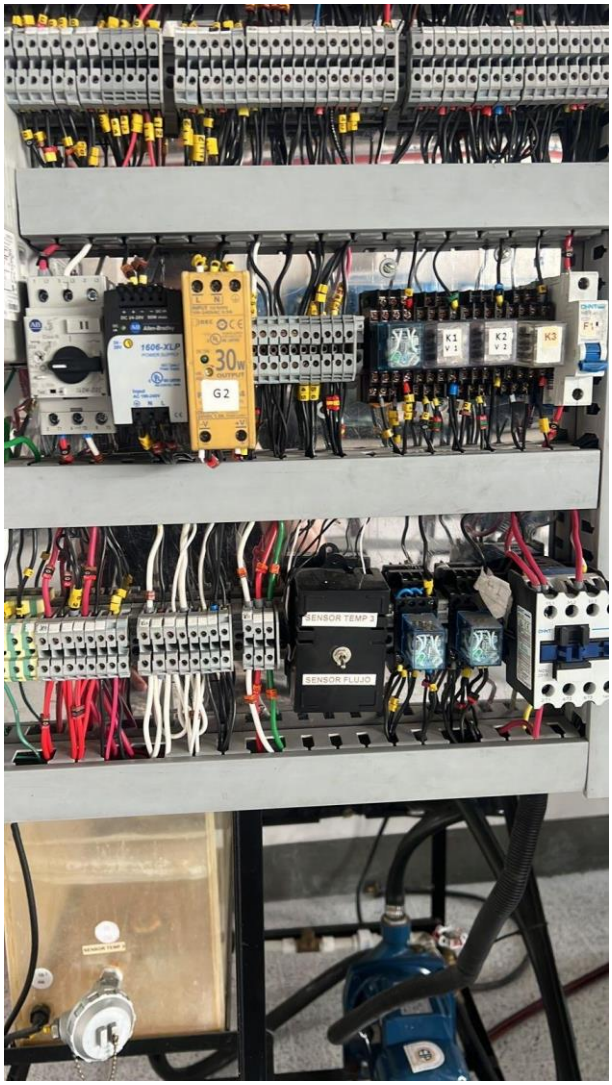



**Figura 3.** Detalle interno del gabinete: bornes, relés (K1, K2, K3), fuente de 24 V y sensores de temperatura y flujo.

*Fuente: Elaboración propia (fotografía de la planta).*

## 1.2 PREGUNTA DE INVESTIGACIÓN

Frente a esta situación, los modelos de lenguaje de gran tamaño con enfoque RAG constituyen una alternativa pertinente, pues permiten procesar y relacionar grandes volúmenes de texto, tanto para interpretar documentación técnica como para apoyar el análisis predictivo [5], [6]. Al incorporar la lectura de variables operativas reales y una interfaz de interacción, es posible concebir una herramienta que apoye la comprensión de la máquina y el razonamiento sobre su comportamiento. De ahí surge la pregunta que orienta este proyecto:

*¿Cómo puede un sistema basado en un modelo LLM con enfoque RAG integrar información documental y datos operativos en tiempo real, de forma que facilite la comprensión del funcionamiento de la máquina y genere alertas predictivas de mantenimiento en la planta PPA?*

## 1.3 JUSTIFICACIÓN

La mejora de la eficiencia y la confiabilidad de un proceso industrial requiere, en primer lugar, comprender el funcionamiento de la máquina. La literatura sobre mantenimiento basado en condición señala que el valor de los datos de monitoreo depende de que puedan interpretarse en el contexto técnico del equipo, y que la integración entre la información documental y las señales operativas es uno de los factores que más influye en la calidad del diagnóstico [4]. De forma complementaria, la conceptualización de *smart maintenance* propuesta por Bokrantz et al. identifica la toma de decisiones basada en datos y la integración del conocimiento técnico disperso como dimensiones centrales del mantenimiento en la industria actual [3].

En la PPA esta integración no existía. Los trabajos previos sobre la planta se concentraron en el control y la supervisión de variables mediante sistemas SCADA [1], [2], y cada uno aportó documentación propia sin una estructura común, de modo que la información técnica quedó dispersa entre documentos de distintas épocas. Además, no había ningún mecanismo que relacionara esa documentación con los datos operativos que la planta genera, lo que retrasa tanto la detección temprana de fallas como el mantenimiento. Las herramientas de monitoreo convencionales permiten visualizar variables, pero rara vez interpretan la información según su contexto ni ofrecen un soporte real para la toma de decisiones técnicas [7].

Por estas razones, el proyecto propone un sistema basado en un modelo LLM con enfoque RAG que integre la información documental y los datos en tiempo real en una única interfaz interactiva. El sistema busca facilitar la comprensión del comportamiento de la máquina, generar alertas de mantenimiento fundamentadas en la documentación y dejar un antecedente concreto de una solución inteligente aplicada al diagnóstico industrial dentro de la misma PPA. La elección de RAG, frente al reentrenamiento del modelo, responde además a una necesidad práctica: la base de conocimiento puede

actualizarse sin reentrenar el modelo cuando la documentación de la planta cambie [8].

# 2. OBJETIVOS

## 2.1 OBJETIVO GENERAL

Integrar un sistema de inteligencia artificial LLM con enfoque RAG que integre información documental y datos de la planta mediante una interfaz de usuario, para mejorar el diagnóstico, el mantenimiento predictivo y la comprensión del funcionamiento de la planta PPA.

## 2.2 OBJETIVOS ESPECÍFICOS

1. Integrar el modelo LLM con enfoque RAG, capaz de consultar, analizar y generar información contextualizada a partir de la documentación técnica y los registros operativos asociados a la planta PPA.
2. Desarrollar una interfaz de usuario interactiva que facilite la comunicación con el sistema, permitiendo la consulta inteligente de información, la visualización de datos y la comprensión del comportamiento de la máquina.
3. Implementar la lectura de variables operativas de la planta PPA para generar alertas automáticas de mantenimiento basadas en condiciones anómalas o desviaciones de los parámetros establecidos en la documentación técnica.

# 3. ALCANCE Y LIMITACIONES

## 3.1 ALCANCE

El proyecto comprende la implementación completa de los cinco bloques funcionales definidos en la propuesta de grado: lectura de sensores, análisis documental, modelo de lenguaje, módulo RAG e interfaz de usuario. Incluye además la instrumentación necesaria para medir con datos, y no por impresión, si el asistente cumple su propósito. Esto implicó registrar cada consulta desde el inicio del desarrollo (pregunta, fragmentos recuperados, modelo que respondió, tiempos y tokens generados) y construir un banco propio de 145 preguntas para evaluar la exactitud sobre el sistema en producción, no sobre una versión de prueba independiente.

Quedan fuera del alcance el reentrenamiento o ajuste fino (*fine-tuning*) de los modelos de lenguaje, la operación con servicios de inteligencia artificial en la nube, el control o la escritura de variables sobre el PLC (el sistema solo lee) y la modificación de los documentos fuente de la planta.

## 3.2 LIMITACIONES ASUMIDAS

El desarrollo se realizó bajo una restricción que se mantuvo sin excepción: el sistema debía operar completamente en local, sin ninguna API en la nube y sin exponer servicios fuera de una red privada, sobre un equipo Dell OptiPlex con procesador Intel i5-7500T, 12 GB de RAM y sin GPU dedicada. Esta restricción fijó el techo de exactitud alcanzable con modelos de 3 000 a 7 000 millones de parámetros ejecutados en CPU.

Se asumieron, además, dos limitaciones de contexto. La primera es que el servidor de desarrollo no dispuso de conectividad permanente hacia el PLC físico, por lo que parte de la lógica de adquisición se validó contra la base de datos y no contra el hardware de forma continua. La segunda es que el volumen de datos reales capturados es reducido (ocho sesiones al momento de la redacción), lo que limita la representatividad de los agregados estadísticos. Ambas limitaciones se retoman en los capítulos de resultados y conclusiones.

# 4. ANTECEDENTES Y ESTADO DEL ARTE

Este capítulo reúne los trabajos previos que sirven de referencia al proyecto, separados de las definiciones teóricas que se presentan en el capítulo 5. Los antecedentes se agrupan en cuatro líneas: los trabajos realizados sobre la propia PPA, el diagnóstico basado en señales y monitoreo continuo, la aplicación de LLM y RAG al mantenimiento, y la infraestructura de adquisición y almacenamiento que hace viable este tipo de sistemas.

## 4.1 TRABAJOS PREVIOS SOBRE LA PPA

La PPA de la Universidad Distrital cuenta con un historial de trabajos enfocados en su control, supervisión y adquisición de variables. Vega Correa y Meza Aguilar diseñaron e implementaron uno de los primeros sistemas SCADA de la planta, orientado al control del enfriamiento del tanque de agua fría, con sensores industriales, un PLC y una interfaz HMI de supervisión en tiempo real [1]. Este trabajo es relevante porque documenta la instrumentación de la planta y su comportamiento bajo condiciones controladas. Posteriormente, Torres Castillo y Suárez Chaparro ampliaron esa arquitectura hacia un control PID de nivel y temperatura en dos tanques, e incorporaron una Raspberry Pi y un microcontrolador PSoC 5 para la adquisición de señales analógicas y digitales [2]. Estos antecedentes confirman que la PPA admite capas adicionales de análisis como la que se propone en este trabajo, y son a la vez parte del origen de la dispersión documental descrita en el capítulo 1.

## 4.2 DIAGNÓSTICO BASADO EN SEÑALES Y MONITOREO CONTINUO

La literatura sobre diagnóstico basado en el análisis de corriente aporta antecedentes para el monitoreo no invasivo de motores y actuadores. El análisis de firma de corriente del motor (MCSA) se ha consolidado como técnica para detectar fallas eléctricas a partir de variaciones medibles en la señal, ya sea mediante patrones vectoriales de corriente en coordenadas estacionarias [9] o mediante modelos autorregresivos que superan las limitaciones de la transformada rápida de Fourier tradicional [10]. En la misma línea, sistemas de bajo costo basados en IoT han mostrado que es posible capturar información operativa distribuida en tiempo real para el mantenimiento

predictivo de sistemas de bombeo [11]. La literatura sobre acondicionamiento y procesamiento digital de señales insiste, además, en que la calidad del diagnóstico depende de la calidad de la señal capturada en origen [12], y los mecanismos de verificación temporal propuestos por Tian et al. muestran cómo reducir falsos positivos en entornos con ruido y variaciones transitorias [13]. Este último principio se refleja en la regla de sensor trabado del presente proyecto, que exige un número mínimo de lecturas antes de emitir una alerta.

### 4.3 LLM Y RAG APLICADOS AL MANTENIMIENTO

El antecedente más cercano al enfoque de este proyecto es PARAM, propuesto por Harbola y Purwar: un sistema de agentes con RAG que detecta fallas y genera planes de mantenimiento contextualizados a partir de una arquitectura de detección, conocimiento y acción [14]. Este planteamiento respalda la idea central que se persigue aquí: no basta con detectar una anomalía; es necesario cerrar el ciclo hasta una recomendación verificable. Jin et al. muestran, con TIME-LLM, que es posible adaptar modelos de lenguaje al análisis de series temporales sin modificar su arquitectura base [15], lo que abre la posibilidad de tratar las señales de proceso como secuencias interpretables.

En cuanto a la construcción de sistemas RAG, Collado Alonso describe en detalle la indexación, la generación de embeddings y la recuperación semántica en un contexto institucional [16], y García Herrera presenta el flujo de ingestión, fragmentación, embeddings y recuperación contextual sobre documentos PDF [17], que corresponde en lo esencial a la arquitectura base adoptada en este proyecto. Dettmers et al., con QLoRA, demuestran que es posible ajustar modelos grandes en hardware limitado mediante cuantización y adaptadores de bajo rango [18], lo cual respalda la viabilidad de ejecutar modelos cuantizados en equipos modestos.

### 4.4 INFRAESTRUCTURA DE ADQUISICIÓN Y ALMACENAMIENTO VECTORIAL

Nugraha et al. muestran cómo integrar PLC con plataformas IIoT mediante Node-RED y protocolos como Modbus TCP/IP [19], lo que confirma que la captura de datos rara vez es el componente más difícil de estos sistemas; el reto suele estar en la capa de

análisis. Pan et al., por su parte, analizan los sistemas de gestión de bases de datos vectoriales que permiten a un sistema RAG recuperar información relevante de forma eficiente [20].

## 4.5 SÍNTESIS Y BRECHA IDENTIFICADA

La tabla 2 resume los aportes de los antecedentes más cercanos y la relación de cada uno con este proyecto.

**Tabla 2.** Síntesis de antecedentes y relación con el proyecto.

| Trabajo | Aporte principal | Relación con este proyecto |
|---|---|---|
| Vega Correa y Meza Aguilar (2016) [1] | SCADA para enfriamiento del tanque de agua fría | Describe la instrumentación base de la PPA |
| Torres Castillo y Suárez Chaparro (2018) [2] | Control PID de nivel y temperatura; adquisición con Raspberry Pi y PSoC 5 | Confirma que la PPA admite capas adicionales; origen de parte de la documentación |
| Harbola y Purwar (2025) [14] | Agentes con RAG para mantenimiento prescriptivo | Referente directo de la integración entre detección, conocimiento y acción |
| García Herrera (2025) [17]; Collado Alonso (2024) [16] | Flujos RAG sobre documentos | Arquitectura base del pipeline documental |
| Nugraha et al. (2025) [19] | Integración PLC–IIoT con Node-RED y Modbus | Referente de la capa de adquisición; se adaptó a EtherNet/IP |
| Pan et al. (2023) [20] | Revisión de bases de datos vectoriales | Criterio para seleccionar la base vectorial |

*Fuente: elaboración propia.*

En conjunto, los antecedentes muestran que los LLM se emplean ya para integrar información heterogénea y apoyar decisiones en sistemas complejos. Sin embargo, no se identificó un trabajo que combine, sobre la PPA o un banco didáctico equivalente, la consulta documental con citación de fuentes, el acceso a datos reales de un PLC mediante *tool calling* y la operación completamente local en hardware sin GPU, con una evaluación cuantitativa de exactitud y alucinación. Esa combinación constituye la brecha que aborda el presente trabajo.

# 5. MARCO TEÓRICO

Este capítulo define los conceptos sobre los que se construye el sistema. Cada sección presenta la definición, el funcionamiento y la relevancia del concepto para el proyecto.

## 5.1 MANTENIMIENTO BASADO EN CONDICIÓN

El mantenimiento industrial se clasifica tradicionalmente en tres estrategias: correctivo, que interviene después de la falla; preventivo, que interviene según un calendario fijo; y predictivo, que interviene según el estado real del equipo [21]. El mantenimiento basado en condición (CBM) generaliza esta última estrategia: la decisión de mantenimiento se fundamenta en la información recogida mediante monitoreo de condición [4]. Jardine, Lin y Banjevic describen un programa CBM en tres etapas: adquisición de datos, procesamiento de datos y toma de decisiones de mantenimiento, esta última dividida en diagnóstico (detección, aislamiento e identificación de la falla) y pronóstico (estimación de su evolución) [4]. La norma ISO 17359 establece las directrices generales de un programa de monitoreo de condición, incluida la selección de variables, la definición de umbrales de alerta y alarma y la relación entre ambos [22].

En este marco, variables como la corriente, la temperatura o el nivel dejan de ser señales aisladas y pasan a ser indicadores del estado interno del equipo. En el contexto de la Industria 4.0, este enfoque se refuerza con IoT, analítica de datos e inteligencia artificial, con el propósito de anticipar la falla y sugerir la acción correctiva antes de que el problema escale [7], [23]. En el presente proyecto, las seis reglas de alerta (sección 8.9) implementan la etapa de diagnóstico: comparan las lecturas reales contra umbrales y enclavamientos documentados y contra el rango físico de cada instrumento.

## 5.2 MODELOS DE LENGUAJE DE GRAN TAMAÑO

Un modelo de lenguaje asigna una probabilidad a secuencias de *tokens* (fragmentos de palabras) y genera texto prediciendo, de forma iterativa, el siguiente *token* a partir de los anteriores. Los LLM actuales se basan en la arquitectura *transformer*, cuyo componente central es el mecanismo de atención [24], definido en la ecuación 1.

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

donde Q, K y V son las matrices de consultas, claves y valores obtenidas por proyección lineal de las representaciones de entrada, y $d_k$ es la dimensión de las claves. La atención permite que cada *token* pondere la relevancia de todos los demás dentro de la ventana de contexto, que es el número máximo de *tokens* que el modelo puede considerar a la vez; en este proyecto se fijó en 4 096 *tokens*.

Estos modelos se entrenan de forma autosupervisada sobre grandes volúmenes de texto, se ajustan después con supervisión y, en muchos casos, se refinan con aprendizaje por refuerzo a partir de retroalimentación humana [25]. Su tamaño se mide en número de parámetros: los modelos empleados aquí tienen 3 000 millones (llama3.2:3b [26]) y 7 000 millones (qwen2.5:7b [27]). Para ejecutarlos en CPU se utilizan versiones cuantizadas, que representan los pesos con menos bits (por ejemplo, 4 bits en lugar de 16) a cambio de una pérdida acotada de precisión [18].

En la generación, el modelo produce un vector de puntajes z sobre el vocabulario V y lo convierte en probabilidades mediante la función *softmax* con temperatura T (ecuación 2).

$$p_i = \frac{\exp(z_i/T)}{\sum_{j=1}^{|V|}\exp(z_j/T)} \tag{2}$$

Una temperatura baja concentra la probabilidad en los *tokens* más probables y reduce la variabilidad entre ejecuciones; una temperatura alta favorece respuestas más diversas, pero también más inconsistentes [28]. Este es el fundamento de la decisión de fijar T = 0.2 en el sistema (sección 8.1).

### 5.3 EMBEDDINGS Y SIMILITUD SEMÁNTICA

Un *embedding* es una representación vectorial densa de un texto en un espacio de n dimensiones, construida de manera que textos con significado similar queden próximos entre sí [29]. En este proyecto se emplea el modelo nomic-embed-text, que

produce vectores de 768 dimensiones y admite entradas de hasta 2 048 *tokens* [30]. La proximidad entre la consulta q y un fragmento d se mide con la similitud coseno (ecuación 3).

$$\mathrm{sim}(\mathbf{q},\mathbf{d}) = \cos\theta = \frac{\mathbf{q}\cdot\mathbf{d}}{\|\mathbf{q}\|\,\|\mathbf{d}\|} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\,\sqrt{\sum_{i=1}^{n} d_i^2}} \tag{3}$$

La similitud coseno toma valores entre −1 y 1; un valor cercano a 1 indica que ambos textos tratan un contenido semánticamente equivalente, aunque no compartan las mismas palabras. Esta propiedad permite que una pregunta como «¿qué protege a la resistencia de calentar en seco?» recupere el fragmento que describe el enclavamiento IL-01, aun cuando la pregunta no mencione el identificador del enclavamiento.

### 5.4 RECUPERACIÓN VECTORIAL

La recuperación vectorial consiste en encontrar, dentro de una colección de N vectores, los k más similares a la consulta (búsqueda de los k vecinos más cercanos). Una búsqueda exhaustiva compara la consulta con todos los vectores, con un costo proporcional a N; las bases de datos vectoriales emplean índices aproximados para reducir ese costo [20]. Qdrant, la base utilizada en este proyecto, emplea el índice HNSW, un grafo jerárquico de proximidad que permite búsquedas aproximadas con complejidad aproximadamente logarítmica respecto al tamaño de la colección [31].

Además del índice, la calidad de la recuperación depende de la fragmentación (*chunking*) de los documentos: fragmentos demasiado largos diluyen el significado del vector, y fragmentos que separan un dato de su contexto (por ejemplo, una fila de una tabla de su encabezado) producen vectores que no representan correctamente la información [6]. Este fenómeno se observó en el proyecto y motivó el fragmentador especializado en tablas descrito en la sección 8.3. La recuperación semántica puede complementarse con recuperación léxica, como BM25 [32], o con búsqueda exacta de identificadores; esta última se adoptó en el sistema para los nombres de tags.

## 5.5 GENERACIÓN AUMENTADA POR RECUPERACIÓN

La generación aumentada por recuperación (RAG) combina un modelo generativo con una base de conocimiento externa [5]. En lugar de responder únicamente con lo que el modelo memorizó durante su entrenamiento, el sistema recupera primero los fragmentos relevantes y luego redacta la respuesta condicionada a ellos. En la formulación original de Lewis et al., el recuperador y el generador se entrenan de forma conjunta, y el documento recuperado se trata como una variable latente sobre la que se marginaliza la generación [5]. La variante que se adoptó en este proyecto, y la más extendida en la práctica actual, simplifica ese esquema: el recuperador no se entrena, sino que se apoya en un modelo de embeddings ya entrenado (nomic-embed-text) y en una búsqueda por similitud; y en lugar de marginalizar sobre los documentos, los k fragmentos más relevantes se concatenan directamente en el *prompt* del modelo generador [6], [33].

RAG ofrece tres ventajas relevantes para la PPA. Primero, reduce la alucinación al anclar la respuesta en evidencia controlada. Segundo, permite citar la fuente exacta de cada afirmación, lo que hace verificable la respuesta. Tercero, permite actualizar el conocimiento del sistema sin reentrenar el modelo, simplemente reindexando la documentación [8]. No obstante, RAG no elimina la alucinación: si la recuperación trae un fragmento incorrecto, o si el modelo interpreta mal un fragmento correcto, la respuesta puede ser errónea aun con la fuente citada.

## 5.6 TOOL CALLING Y AGENTES

Un LLM por sí solo solo puede razonar sobre el texto de su ventana de contexto; no puede ejecutar una consulta SQL ni leer el valor actual de un sensor. El mecanismo de *tool calling* (también denominado *function calling*) resuelve esta limitación: junto con la pregunta, el modelo recibe la descripción estructurada de un conjunto de funciones disponibles (nombre, propósito y parámetros), y decide si la pregunta requiere invocar una, varias o ninguna [34]. Cuando el modelo solicita una herramienta, el sistema la ejecuta y devuelve el resultado al modelo, que lo incorpora a su respuesta. El patrón ReAct formaliza esta alternancia entre razonamiento y acción [35].

En el proyecto se definieron dos herramientas: buscar_documentacion, que ejecuta la recuperación sobre Qdrant, y consultar_sensores_plc, que consulta lecturas, resúmenes estadísticos o sesiones de captura en PostgreSQL. Este mecanismo permite que el asistente distinga, sin que el usuario lo indique, entre una pregunta documental y una que exige datos reales, y es el eje de la arquitectura descrita en el capítulo 8.

## 5.7 ALUCINACIÓN EN MODELOS DE LENGUAJE

Se denomina alucinación al contenido generado que no es fiel a la fuente proporcionada o que no es verificable [36]. Ji et al. distinguen entre alucinación intrínseca, cuando la salida contradice la fuente, y extrínseca, cuando la salida agrega información que la fuente no respalda [36]. Huang et al. proponen una taxonomía complementaria que separa la alucinación factual (afirmaciones falsas sobre el mundo) de la alucinación de fidelidad (desviaciones respecto a las instrucciones o al contexto dado) [37].

En este proyecto se adoptó la siguiente definición operativa: una respuesta contiene alucinación cuando incluye un dato, un umbral, un valor o una relación que no aparece en la documentación ni en la base de datos consultada, o cuando incumple una instrucción explícita del sistema (por ejemplo, responder una pregunta fuera del dominio de la planta). Con base en ella se clasificaron los errores observados en tres tipos: factual (valores o umbrales inventados), de interpretación (estados o comparaciones invertidas a partir de un dato correcto) y de comportamiento (incumplimiento de restricciones, como generar contenido creativo). La capa de respuestas deterministas descrita en la sección 8.2 es la principal medida de mitigación: elimina la redacción libre del modelo cuando el dato y la regla de decisión se conocen con certeza.

## 5.8 EVALUACIÓN DE SISTEMAS RAG

La evaluación de un sistema RAG debe considerar tanto la recuperación como la generación [38]. En la recuperación interesa si se obtuvo el fragmento pertinente; en la generación, si la respuesta es correcta, completa y fiel al contexto. En ausencia de

un conjunto de datos público para el dominio, una práctica habitual es construir un banco de preguntas con respuestas de referencia y calificarlas con una rúbrica ordinal. La calificación puede ser humana o automática mediante un LLM evaluador [39]; en este proyecto se optó por la calificación humana, por la especificidad del dominio y la limitada capacidad de los modelos locales.

Una rúbrica ordinal introduce el problema de la consistencia del evaluador. La medida de referencia para cuantificar el acuerdo entre dos evaluadores es el coeficiente kappa de Cohen [40], que en su versión ponderada es adecuado para escalas ordinales, y cuyos valores suelen interpretarse según la escala de Landis y Koch [41]. Por otra parte, cuando se estima una proporción a partir de una muestra pequeña, como la tasa de alucinación en pocas repeticiones, el intervalo de Wilson ofrece una estimación más adecuada que la aproximación normal [42]. Las ecuaciones concretas empleadas se presentan en la sección 7.5.

## 5.9 INTEGRACIÓN INDUSTRIAL Y ADQUISICIÓN DE DATOS

La integración industrial es el puente entre el mundo físico (sensores, actuadores y PLC) y el entorno digital donde se realiza el análisis. Los PLC se comunican mediante protocolos industriales; entre los más difundidos están Modbus TCP/IP y EtherNet/IP. EtherNet/IP es la adaptación del Protocolo Industrial Común (CIP) sobre Ethernet estándar, mantenida por ODVA, y es el protocolo nativo de los controladores Allen-Bradley de la familia Logix [43]. Nugraha et al. muestran la integración de PLC con plataformas IIoT mediante Node-RED y Modbus [19]; en este proyecto se empleó EtherNet/IP por ser el protocolo nativo del PLC disponible (sección 7.3).

Las señales analógicas de los transmisores industriales, típicamente en lazo de 4–20 mA, llegan al PLC como cuentas digitales (valor crudo) que deben convertirse a unidades de ingeniería mediante una linealización (ecuación 4).

$$y = m\,x + b \tag{4}$$

donde x es la cuenta cruda, y el valor en unidades de ingeniería, m la pendiente y b el término independiente. Una consecuencia práctica de esta relación, que el sistema aprovecha para diagnosticar fallas, es que una cuenta cruda igual a cero produce

exactamente y = b; por tanto, una lectura constante igual al término independiente sugiere un lazo abierto o un canal sin alimentación (sección 8.9).

## 5.10 PROGRAMACIÓN ASISTIDA POR IA

El término *vibe coding* fue propuesto en febrero de 2025 por el investigador Andrej Karpathy para describir una forma de programar en la que el desarrollador describe en lenguaje natural lo que necesita y un LLM genera, revisa y corrige el código correspondiente. A diferencia del autocompletado tradicional, esta práctica involucra herramientas de tipo agente, capaces de crear archivos, ejecutar comandos y correr pruebas, con el programador en un rol de supervisión.

Sarkar y Drosos presentaron uno de los primeros estudios empíricos de esta práctica y concluyen que no elimina la necesidad de saber programar, sino que redistribuye esa habilidad hacia la evaluación rápida del código generado, la gestión del contexto entregado a la herramienta y la decisión de cuándo intervenir manualmente [44]. Fawzy, Tahir y Blincoe, en una revisión de 101 fuentes de practicantes, identifican una paradoja entre velocidad y calidad: el código obtenido se percibe como rápido de producir y a la vez frágil, sobre todo cuando se omiten las pruebas [45]. Estos hallazgos fundamentan el protocolo de revisión adoptado en el proyecto (sección 8.11), en el que todo código generado con apoyo de IA se revisó y se probó contra el sistema real antes de aceptarse.

# 6. MARCO LEGAL Y NORMATIVO

Este capítulo no se limita a enumerar las normas aplicables: para cada disposición se explica cómo se relaciona con un componente concreto del sistema, sea la adquisición de datos, la seguridad industrial, el tratamiento de la información o la operación del sistema. La tabla 3 resume esa correspondencia y las secciones siguientes la desarrollan.

**Tabla 3.** Relación entre el marco normativo y los componentes del sistema.

| Norma | Disposición relevante | Relación con el sistema |
|---|---|---|
| Decreto 1072 de 2015 [46] | Identificación de peligros, evaluación y control de riesgos en el SG-SST | Las alertas de sobretemperatura, nivel y enclavamientos apoyan la identificación temprana de condiciones peligrosas; el sistema es de solo lectura sobre el PLC |
| Ley 1581 de 2012 y Decreto 1377 de 2013 [47], [48] | Principios de finalidad, circulación restringida y seguridad del dato personal | El registro de consultas no almacena identidad del usuario; los servicios solo son accesibles en red privada |
| Ley 23 de 1982 y Decisión Andina 351 de 1993 [49], [50] | Protección del software como obra y derechos patrimoniales | Titularidad del código y de la base documental derivada; uso sin alterar los documentos fuente |
| Ley 1273 de 2009 [51] | Delitos de acceso abusivo a sistemas informáticos e interceptación de datos | Justifica el control de acceso: ningún servicio escucha en 0.0.0.0 y el acceso al PLC se restringe a un usuario de sistema dedicado |
| ISA/IEC 62443 [52] | Segmentación en zonas y conductos, mínimo privilegio en sistemas de control | Separación entre la red de la planta y los servicios de análisis; el servicio de adquisición es el único que alcanza el PLC |
| ISO 17359:2018 [22] | Directrices de un programa de monitoreo de condición y definición de umbrales | Las reglas de alerta usan umbrales documentados y marcan como aproximadas las que no cuentan con la señal exacta |
| CONPES 3975 de 2019 [53] | Política nacional para la transformación digital e inteligencia artificial | Contexto de política pública que respalda la adopción de IA en entornos académicos y productivos |

*Fuente: elaboración propia.*

## 6.1 SEGURIDAD Y SALUD EN EL TRABAJO

El Decreto 1072 de 2015, Decreto Único Reglamentario del Sector Trabajo, obliga a implementar el Sistema de Gestión de Seguridad y Salud en el Trabajo (SG-SST) y exige identificar los peligros, evaluar los riesgos y establecer controles [46]. En la PPA existen peligros concretos asociados al proceso: el calentamiento de agua mediante una resistencia eléctrica, el riesgo de operar la resistencia sin agua y el riesgo de rebose o de marcha en seco de la bomba. Las reglas de alerta del sistema se diseñaron a partir de esos mismos escenarios documentados (alarma AL-01 y

enclavamientos IL-01, IL-03 e IL-04), de modo que el sistema contribuye a la identificación temprana de condiciones de riesgo. Como medida de control adicional, el sistema opera en modo de solo lectura: ninguna función puede escribir variables ni modificar la lógica del PLC, lo que evita introducir un nuevo peligro en el proceso.

### 6.2 PROTECCIÓN DE DATOS PERSONALES

La Ley 1581 de 2012 y su decreto reglamentario 1377 de 2013 regulan el tratamiento de datos personales bajo los principios de finalidad, libertad, circulación restringida y seguridad [47], [48]. El sistema trabaja con datos operativos de un banco de pruebas y no con información personal: el registro de consultas almacena la pregunta, las fuentes, el modelo, los tiempos y los tokens, pero no la identidad de quien pregunta. Si en el futuro el sistema cruzara el desempeño del equipo con el historial de un operario específico, ese tratamiento debería contar con autorización previa e informada. En aplicación del principio de seguridad, todos los servicios se publican únicamente en la interfaz local o en redes privadas (sección 7.3).

### 6.3 PROPIEDAD INTELECTUAL

En Colombia el software se protege como obra literaria mediante el derecho de autor, según la Ley 23 de 1982 y la Decisión Andina 351 de 1993 [49], [50]. El código desarrollado constituye una obra cuyos derechos patrimoniales deben definirse entre los proponentes, la Universidad Distrital y la entidad en la que eventualmente se implemente la solución. La base de conocimiento se construyó a partir de documentos de la planta que tienen sus propios autores; por esa razón, el sistema no modifica ni reescribe los documentos originales, y las inconsistencias detectadas entre documentos se mitigaron mediante reglas de priorización y no mediante la edición de las fuentes (sección 8.3).

### 6.4 SEGURIDAD INFORMÁTICA Y CIBERSEGURIDAD INDUSTRIAL

La Ley 1273 de 2009 tipifica como delito el acceso abusivo a un sistema informático y la interceptación de datos [51]. Aunque el sistema no maneja información sensible, conectar un servidor a la red de un PLC industrial exige controles de acceso. La serie ISA/IEC 62443 recomienda segmentar los sistemas de control en zonas y conductos

y aplicar el principio de mínimo privilegio [52]. El proyecto aplica ambos principios: ningún servicio escucha en todas las interfaces de red (0.0.0.0), el acceso remoto se realiza por una red privada cifrada (Tailscale) y el único componente que se comunica con el PLC es el servicio de adquisición, ejecutado bajo un usuario de sistema dedicado y separado del que ejecuta la API y la interfaz.

## 6.5 MONITOREO DE CONDICIÓN

La norma ISO 17359 establece que un programa de monitoreo de condición debe definir las variables que se miden, los umbrales de alerta y alarma y la acción asociada a cada uno [22]. Las seis reglas del motor de alertas siguen esta estructura: cada regla indica el tag, el umbral o condición documentada, la fuente de la que proviene y la acción recomendada. Cuando la documentación define una condición que depende de una señal que no se captura, la regla se implementa con una señal sustituta y se marca explícitamente como aproximada, en lugar de inventar un umbral.

# 7. METODOLOGÍA

## 7.1 ENFOQUE METODOLÓGICO

El proyecto se desarrolló con un enfoque iterativo e incremental organizado en cuatro fases: análisis de requisitos y restricciones, diseño de la arquitectura, implementación e integración, y protocolo experimental y validación (figura 4). Las tres primeras fases se ejecutaron de forma secuencial; la cuarta se conectó con la tercera mediante un ciclo de retroalimentación: los resultados de cada prueba se analizaron, los errores se agruparon en patrones y las correcciones derivadas se incorporaron al sistema antes de repetir la evaluación completa.

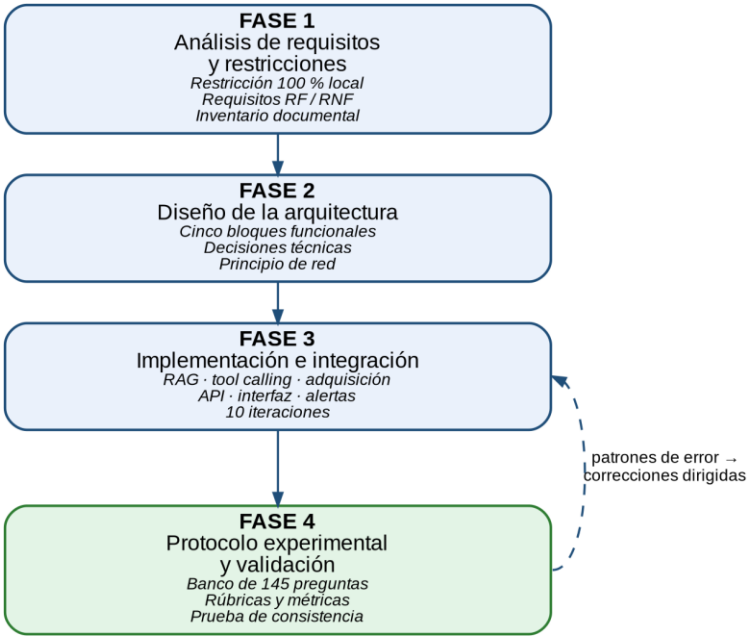



**Figura 4.** Fases de la metodología y ciclo de retroalimentación entre la validación y la implementación.

*Fuente: Elaboración propia.*

La tabla 4 resume, para cada fase, sus actividades, sus entregables y la sección del documento donde se desarrolla. El registro cronológico detallado del desarrollo, que en una versión anterior de este documento ocupaba el cuerpo de la metodología, se trasladó al Anexo A como bitácora de apoyo.

**Tabla 4.** Actividades, entregables y ubicación de cada fase metodológica.

| Fase | Actividades principales | Entregables | Sección |
|---|---|---|---|
| 1. Análisis de requisitos y restricciones | Caracterización del hardware; definición de requisitos funcionales y no funcionales; inventario y diagnóstico de la documentación | Tablas de requisitos; corpus documental en Markdown | 7.2 |
| 2. Diseño de la arquitectura | Definición de los cinco bloques funcionales; evaluación de alternativas tecnológicas; principio de red | Diagrama de arquitectura y de despliegue; tabla de decisiones técnicas | 7.3 |
| 3. Implementación e integración | Construcción de cada componente en diez iteraciones; pruebas de integración; experimentos descartados | Sistema funcional en producción; tabla de iteraciones | 7.4 y cap. 8 |
| 4. Protocolo experimental y validación | Construcción del banco de preguntas; definición de rúbricas y métricas; ejecución de ocho pruebas y una prueba de consistencia | Resultados cuantitativos y patrones de error | 7.5 y cap. 9 |

*Fuente: elaboración propia.*

## 7.2 FASE 1: ANÁLISIS DE REQUISITOS Y RESTRICCIONES

### *7.2.1 Restricción de partida y caracterización del hardware*

La restricción que gobernó todas las decisiones posteriores fue la operación completamente local: sin dependencia de API en la nube y sin exponer servicios fuera de una red privada. La tabla 5 caracteriza el equipo asignado al proyecto.

**Tabla 5.** Características del servidor asignado al proyecto.

| Componente | Especificación |
|---|---|
| Equipo | Dell OptiPlex |
| Procesador | Intel Core i5-7500T, 4 núcleos, hasta 3.3 GHz |
| Memoria RAM | 12 GB |
| Gráficos | Intel HD 630 integrada (no utilizable para inferencia) |
| Almacenamiento | 207 GB |
| Sistema operativo | Debian 13 (trixie) |
| Memoria de intercambio | Partición de 11.5 GB, vm.swappiness = 10 |
| Acceso remoto | Tailscale (red privada) y Ethernet del laboratorio con IP estática |

*Fuente: elaboración propia.*

Bajo esta restricción, lo que en un entorno empresarial se resolvería con un modelo de gran capacidad en la nube debía resolverse con modelos de 3 000 a 7 000 millones de parámetros ejecutados íntegramente en CPU. La tensión entre un modelo de capacidad limitada y la exigencia de respuestas correctas y sin alucinaciones es el hilo conductor del desarrollo y de la discusión de resultados.

### *7.2.2 Requisitos del sistema*

A partir de los objetivos específicos y de la restricción de partida se definieron los requisitos funcionales (tabla 6) y no funcionales (tabla 7). Cada requisito se asocia al objetivo que atiende y a la evidencia con la que se verificó.

**Tabla 6.** Requisitos funcionales.

| Código | Requisito | Objetivo | Verificación |
|---|---|---|---|
| RF-01 | Responder preguntas sobre la documentación técnica citando documento y sección de origen | OE1 | Grupo G1 del banco de preguntas |
| RF-02 | Consultar lecturas, estadísticos y sesiones del histórico real del PLC | OE1, OE3 | Grupo G2 del banco de preguntas |
| RF-03 | Decidir automáticamente qué fuente consultar (documentación, datos, ambas o ninguna) | OE1 | Tasa de selección correcta de herramienta |
| RF-04 | Visualizar datos reales por sesión y por variable | OE2 | Pestaña de datos del PLC |
| RF-05 | Iniciar y detener sesiones de captura desde la interfaz | OE2, OE3 | Endpoints /plc/sesiones |
| RF-06 | Incorporar nuevos documentos a la base de conocimiento desde la interfaz | OE2 | Endpoint /documentos/subir |
| RF-07 | Generar alertas a partir de umbrales y enclavamientos documentados | OE3 | Sesiones reales 16, 26, 27, 28 y 61 |
| RF-08 | Registrar cada consulta con sus fuentes, modelo, tiempos y tokens | Validación | Base consultas_log |

*Fuente: elaboración propia.*

**Tabla 7.** Requisitos no funcionales.

| Código | Requisito | Criterio de aceptación |
|---|---|---|
| RNF-01 | Operación 100 % local | Ninguna llamada a servicios externos de IA |
| RNF-02 | Ejecución en el hardware disponible, sin GPU | Memoria en reposo por debajo de la RAM física, sin intercambio sostenido |
| RNF-03 | Exposición exclusiva en redes privadas | Ningún servicio escucha en 0.0.0.0 |
| RNF-04 | Recuperación automática ante reinicios | Todos los servicios arrancan solos tras reiniciar el servidor |
| RNF-05 | Trazabilidad de cada respuesta | Toda respuesta documental muestra sus fuentes; toda consulta queda registrada |
| RNF-06 | No inventar umbrales ni valores | Las alertas y respuestas deterministas solo usan datos documentados o capturados |
| RNF-07 | Latencia aceptable en CPU | Respuesta en tiempo acotado y aviso visible al usuario durante la espera |

*Fuente: elaboración propia.*

### *7.2.3 Inventario y diagnóstico de la documentación*

El diagnóstico de la documentación de la planta arrojó un hallazgo que modificó la estrategia de ingesta: buena parte de los manuales originales en PDF eran imágenes escaneadas sin capa de texto, de modo que un extractor convencional los procesaba como páginas vacías. En consecuencia, la documentación clave se transcribió a Markdown estructurado, con tablas, conservando la numeración de los documentos fuente (por ejemplo, PPA_02_diccionario_tags.md, PPA_03_mapa_direcciones_io.md, PPA_05_interlocks_y_permissives.md y PPA_06_alarmas.md). Esta decisión resultó ventajosa, pues permitió construir un fragmentador que interpreta tablas, algo difícil de lograr sobre documentos escaneados. Se identificó también que la bitácora histórica contiene información que contradice el diccionario de tags vigente; este conflicto se trató en la fase de implementación sin alterar las fuentes (sección 8.3).

## 7.3 FASE 2: DISEÑO DE LA ARQUITECTURA

### *7.3.1 Bloques funcionales*

La arquitectura conserva los cinco bloques funcionales definidos en la propuesta de grado (figura 5): lectura de sensores, análisis documental, modelo de lenguaje, módulo RAG e interfaz de usuario. El primer bloque captura las lecturas de los sensores de la PPA; el segundo procesa manuales, planos y bitácoras como base documental; el tercero interpreta la información y redacta la respuesta; el cuarto conecta a los anteriores, recuperando el fragmento de documentación o el dato de base de datos antes de que el modelo redacte; y el quinto es el punto de interacción donde el usuario formula preguntas y recibe respuestas y alertas.

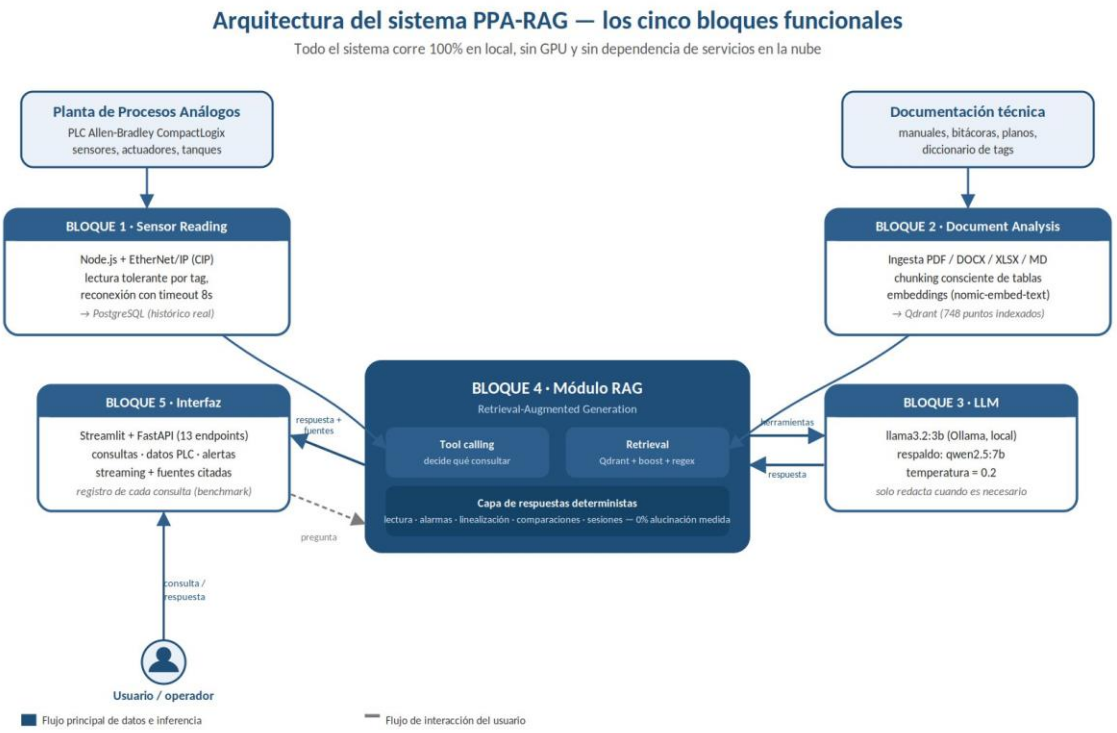



**Figura 5.** Arquitectura general del sistema PPA-RAG: cinco bloques funcionales, fuentes de información y flujo de una consulta hasta la respuesta con fuentes citadas.

*Fuente: Elaboración propia.*

Cada bloque se implementó de forma desacoplada, sin depender del código interno de los demás. Esta decisión permitió sustituir piezas puntuales durante el desarrollo, como el protocolo de comunicación con el PLC, el modelo principal o la estrategia de fragmentación, sin rediseñar el sistema completo. La figura 6 muestra el despliegue físico de los componentes, sus puertos y las redes por las que son accesibles.

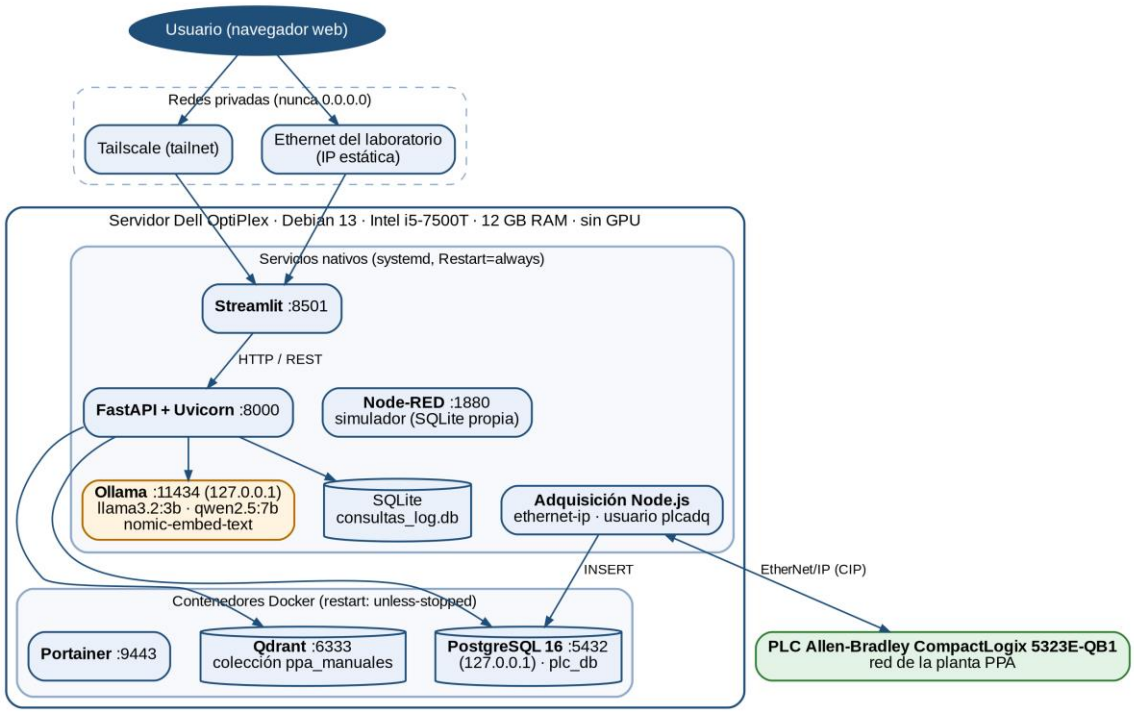



**Figura 6.** Diagrama de despliegue: servicios nativos, contenedores, puertos y redes privadas del sistema.

*Fuente: Elaboración propia.*

### *7.3.2 Decisiones técnicas*

Las decisiones técnicas se tomaron comparando alternativas frente a los requisitos de la Fase 1. La tabla 8 reúne, para cada decisión, las alternativas evaluadas, la opción elegida y su justificación.

**Tabla 8.** Decisiones técnicas: alternativas evaluadas, opción elegida y justificación.

| Decisión | Alternativas | Elegida | Justificación |
|---|---|---|---|
| Ejecución del motor de inferencia | Ollama en contenedor Docker; Ollama nativo | Ollama nativo (systemd) | Sin GPU, cada capa de virtualización resta ciclos de CPU al proceso más costoso (RNF-02) |
| Base de datos vectorial | Chroma; FAISS; Qdrant | Qdrant (Docker) | Contenedor autocontenido y respaldable, panel web para auditar lo indexado, cliente Python sin dependencias pesadas |
| Protocolo con el PLC | Modbus TCP/IP; EtherNet/IP | EtherNet/IP (CIP) | Es el protocolo nativo del CompactLogix 5323E-QB1; Modbus exigía una pasarela o reconfigurar el PLC |
| Lenguaje del servicio de adquisición | Python; Node.js | Node.js (ethernet-ip) | Mejor soporte y documentación del protocolo CIP; se aisló como microservicio que escribe en PostgreSQL |

| Decisión | Alternativas | Elegida | Justificación |
|---|---|---|---|
| Control de la sesión de captura | Estado en el proceso; estado en la base de datos | Base de datos | Permite reiniciar el servicio sin perder ni duplicar la sesión y controlar la captura desde la interfaz |
| Modelo principal | qwen2.5:7b; qwen2.5:3b; llama3.2:3b | llama3.2:3b | Mejor equilibrio entre latencia y fiabilidad de tool calling en CPU (tabla 10) |
| Formato de la base documental | PDF original; Markdown estructurado | Markdown | Los PDF escaneados no tenían texto extraíble; Markdown permite fragmentar tablas |
| Estrategia de fragmentación | Por conteo de palabras; consciente de tablas | Ambas combinadas | Las tablas se dividen por fila repitiendo el encabezado; la prosa, por ventana con solape |
| Exposición de servicios | Todas las interfaces (0.0.0.0); interfaces explícitas | 127.0.0.1, tailnet e IP estática del laboratorio | Cumple RNF-03 y los principios de la sección 6.4 |

*Fuente: elaboración propia.*

### *7.3.3 Principio de red*

Ningún servicio del sistema se publica en la interfaz genérica 0.0.0.0. Cada servicio se asocia explícitamente a la interfaz local (127.0.0.1) y, si debe ser alcanzable desde otro equipo, a la IP de la red privada de Tailscale, nunca a una IP pública. La única excepción, adoptada de forma deliberada, fue publicar una segunda instancia de la API y de la interfaz sobre la IP Ethernet estática del laboratorio, para operar en el sitio sin depender de Internet ni de Tailscale.

## 7.4 FASE 3: IMPLEMENTACIÓN E INTEGRACIÓN

La implementación se realizó en diez iteraciones. Cada iteración tuvo un objetivo concreto, produjo un incremento verificable y, en varios casos, reveló un problema que orientó la siguiente. La tabla 9 resume las iteraciones; la descripción técnica de cada componente se presenta en el capítulo 8 y el detalle cronológico en el Anexo A.

**Tabla 9.** Iteraciones de la fase de implementación.

| N.° | Objetivo | Incremento obtenido | Hallazgo o problema resuelto |
|---|---|---|---|
| I-1 | Infraestructura base | Docker con Qdrant y Portainer; Ollama nativo; control de versiones | Las claves TLS autogeneradas no estaban excluidas del repositorio; se corrigió a tiempo |
| I-2 | Pipeline documental y backend inicial | Ingesta, API FastAPI con registro de consultas, simulador en Node-RED | PDF escaneados sin texto: se migró la base documental a Markdown |
| I-3 | Interfaz y verificación integral | Interfaz Streamlit de tres secciones | El sistema completo sobrevive a un reinicio del servidor |

| N.° | Objetivo | Incremento obtenido | Hallazgo o problema resuelto |
|---|---|---|---|
| I-4 | Adquisición real y tool calling | Servicio EtherNet/IP; herramienta consultar_sensores_plc | Latencias superiores a 180 s por tres causas simultáneas (sección 8.1) |
| I-5 | Optimización y fragmentación | Reducción de *prompts*; fragmentador de tablas | Una fila separada de su encabezado causaba asignaciones erróneas; la colección pasó de 216 a 630 puntos |
| I-6 | Priorización documental y ampliación | Impulso al documento autoritativo; búsqueda exacta de tags; acceso por IP estática; 13 tags | Contradicciones entre la bitácora y el diccionario de tags |
| I-7 | Primera evaluación y capa determinista | Banco de preguntas; respuestas construidas en Python | Exactitud de 0.353 a 0.437 por un cambio de arquitectura |
| I-8 | Banco ampliado y correcciones | 145 preguntas; correcciones de backend | Ruido del criterio de referencia; exactitud de 0.624 |
| I-9 | Rediseño del motor de alertas | Seis reglas sobre datos reales; correlación temporal | Parte de la pestaña de alertas leía una señal simulada |
| I-10 | Segundo ciclo de mejora | Seis correcciones dirigidas por patrones | Exactitud de 0.618 a 0.723 (rúbrica de seis niveles) |
| I-11 | Trazabilidad documental y enriquecimiento de alertas | Pestaña Documentos; línea real de origen por fragmento; filtro por sensor; ubicación física y fotografía en alertas | Un prototipo de citas con fotos dentro de Consultas se evaluó y se dejó fuera para no sobrecargar el chat (tabla 10) |

*Fuente: elaboración propia.*

### *7.4.1 Experimentos descartados*

Documentar lo que no funcionó es tan importante como documentar lo que sí, tanto para justificar las decisiones adoptadas como para orientar el trabajo futuro. La tabla 10 resume los experimentos más relevantes que se revirtieron.

**Tabla 10.** Experimentos que se probaron y se descartaron durante el desarrollo.

| Experimento | Resultado observado | Decisión |
|---|---|---|
| Reordenamiento híbrido con BM25 sobre el ranking semántico de Qdrant | Sin costo de latencia, pero empeoró el caso que buscaba mejorar al premiar texto ruidoso y repetitivo de la bitácora | Revertido por completo |
| OLLAMA_FLASH_ATTENTION y caché KV cuantizado (q8_0) | Optimización pensada para GPU; en esta CPU empeoró el tiempo de evaluación del *prompt* | Revertido |
| Gobernador de CPU en modo *performance* | Sin ganancia medible: el procesador ya escalaba a su frecuencia máxima en modo *powersave* | Revertido |
| qwen2.5:7b como modelo principal | 3 min 18 s por pregunta frente a 30–80 s de llama3.2:3b, sin mejora de calidad que lo justificara | Descartado |
| qwen2.5:3b como modelo principal | Mejoró las preguntas de base de datos, pero no invocó la búsqueda documental en el 58 % de los casos | Descartado |

| Experimento | Resultado observado | Decisión |
|---|---|---|
| Citas con botones a la línea exacta y fotografía del sensor directamente en la pestaña Consultas | Funcionaba, pero duplicaba una interacción ya cubierta por las pestañas Documentos y Alertas y sobrecargaba visualmente la respuesta | Revertido; la trazabilidad completa quedó en Documentos y en Alertas (sección 8.6.1) |

*Fuente: elaboración propia.*

## 7.5 FASE 4: PROTOCOLO EXPERIMENTAL Y VALIDACIÓN

Esta fase define cómo se midió el sistema, de manera que cualquier evaluador pueda repetir el procedimiento. La figura 7 resume el flujo del protocolo.

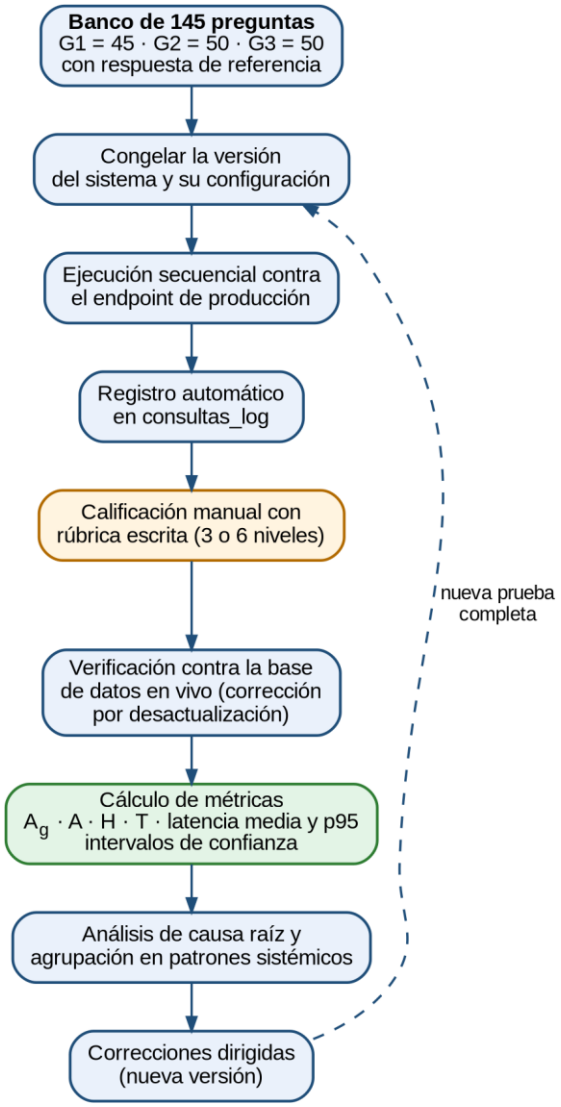



**Figura 7.** Flujo del protocolo experimental de evaluación.

*Fuente: Elaboración propia.*

### *7.5.1 Banco de preguntas y unidad de análisis*

Se construyó un banco propio de 145 preguntas, cada una con su respuesta de referencia, distribuido en tres grupos: G1, documentación (45 preguntas); G2, base de datos o sensores reales (50 preguntas); y G3, preguntas combinadas que requieren invocar ambas herramientas (50 preguntas). El banco incluye, de forma deliberada, preguntas de control fuera del dominio y preguntas trampa con tags inexistentes, para medir la capacidad del sistema de abstenerse. La unidad de análisis es una respuesta del sistema a una pregunta del banco en una ejecución. Las Pruebas 1 a 4 se realizaron con una versión inicial del banco; desde la Prueba 5 se utilizó el banco completo de 145 preguntas.

### *7.5.2 Condiciones de ejecución*

Cada prueba se ejecutó de forma secuencial, no en paralelo, contra el endpoint real del backend en producción y sin modificar el sistema durante la medición, de modo que el resultado refleje la experiencia de un usuario real. Todas las respuestas quedaron registradas automáticamente en la base consultas_log con los campos descritos en la sección 8.5. A partir del hallazgo metodológico descrito en la sección 9.4, se adoptó además la regla de verificar contra la base de datos en vivo las preguntas cuya respuesta de referencia depende de la sesión más reciente.

### *7.5.3 Rúbricas y criterios de calificación*

Se emplearon dos rúbricas ordinales. La rúbrica de tres niveles, aplicada en las Pruebas 1 a 6, asigna $s \in \{0, 0.5, 1\}$. La rúbrica de seis niveles, aplicada en las Pruebas 7 y 8, asigna $s \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ y permite distinguir con mayor precisión el tipo de error. La tabla 11 define cada nivel con un ejemplo de referencia tomado de las respuestas reales del sistema.

**Tabla 11.** Criterios de calificación de las rúbricas de tres y seis niveles.

| Tres niveles | Seis niveles | Criterio | Ejemplo de referencia |
|---|---|---|---|
| 1 | 1.0 | Correcta y completa; cita la fuente o la herramienta adecuada | «El LSH-100 es un sensor de nivel alto del tanque TK-100» con cita del diccionario de tags |
| 0.5 | 0.8 | Correcta en lo esencial, con una imprecisión menor o una fuente parcialmente distinta, no contradictoria | Valor correcto citando un documento secundario en lugar del diccionario de tags |

| Tres niveles | Seis niveles | Criterio | Ejemplo de referencia |
|---|---|---|---|
| 0.5 | 0.6 | Responde el núcleo de la pregunta, pero omite una parte relevante | En una pregunta compuesta, responde la temperatura máxima y omite la sesión |
| 0.5 | 0.4 | Contiene el dato correcto mezclado con una afirmación incorrecta o no verificable | Valor correcto acompañado de un umbral de alarma inventado |
| 0 | 0.2 | Aborda el tema, pero la respuesta central es incorrecta o demasiado vaga | Describe la familia de sensores de nivel sin identificar el tag preguntado |
| 0 | 0 | Incorrecta, alucinada, o no responde cuando debía hacerlo | Define un tag inexistente como si existiera |

*Fuente: elaboración propia.*

A las respuestas parcialmente correctas se les aplicaron cuatro reglas explícitas: (i) en preguntas compuestas, cada parte omitida reduce el puntaje en al menos un nivel; (ii) un dato correcto acompañado de una afirmación inventada nunca recibe el nivel máximo, porque la afirmación inventada es una alucinación; (iii) la ausencia de cita no anula una respuesta correcta, pero impide el nivel 1.0; y (iv) la abstención («no encuentro esa información») se califica con 1 cuando la información efectivamente no existe y con 0 cuando sí existía. Cuando una respuesta de la base de datos no coincidía con la referencia, el calificador consultó la base en vivo; si la respuesta correspondía al dato vigente en el momento de la pregunta, se calificó contra ese dato (corrección por desactualización, sección 9.4).

### *7.5.4 Métricas*

La exactitud de un grupo g con $N_g$ preguntas se define como el promedio de los puntajes $s_i$ asignados según la rúbrica vigente (ecuación 5), y la exactitud global como el promedio ponderado por el tamaño de cada grupo (ecuación 6).

$$A_g = \frac{1}{N_g}\sum_{i=1}^{N_g} s_i, \qquad s_i \in \mathcal{S} \tag{5}$$

$$A = \frac{\sum_g N_g\, A_g}{\sum_g N_g} = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{6}$$

La tasa de alucinación H y la tasa de selección correcta de herramienta T se calculan sobre las N respuestas de una prueba (ecuación 7), donde $N_{\text{aluc}}$ es el número de

respuestas que cumplen la definición operativa de alucinación de la sección 5.7 y $N_{herr}$ correcta es el número de respuestas en las que el sistema invocó la herramienta esperada para la pregunta.

$$H = \frac{N_{\text{aluc}}}{N}, \qquad T = \frac{N_{\text{herr correcta}}}{N} \tag{7}$$

La latencia se reporta como promedio y como percentil 95, definido en la ecuación 8 sobre las latencias $L_i$ de las N respuestas; el percentil 95 captura los peores casos que experimenta un usuario, que el promedio oculta.

$$L_{95} = \min\left\{ L : \frac{\left|\{i : L_i \le L\}\right|}{N} \ge 0.95 \right\} \tag{8}$$

En la prueba de consistencia, cada pregunta q se repitió R = 5 veces y se calculó su tasa de estabilidad (ecuación 9); una pregunta con $E_q = 1$ respondió correctamente en todas las repeticiones.

$$E_q = \frac{1}{R}\sum_{r=1}^{R} \mathbb{1}[\,\text{respuesta}_{q,r}\ \text{correcta}\,], \qquad R = 5 \tag{9}$$

### *7.5.5 Tratamiento estadístico*

Sobre la exactitud global se reporta un intervalo de confianza al 95 % basado en la aproximación normal de la media de los puntajes (ecuación 10), donde S es la desviación estándar muestral de los puntajes. En el caso de proporciones estimadas con muestras pequeñas, como las tasas de la prueba de consistencia, se utiliza el intervalo de Wilson (ecuación 11), con n el número de observaciones, p̂ la proporción observada y z = 1.96 [42].

$$\bar{s} \ \pm\ z_{0.975}\frac{S}{\sqrt{N}}, \qquad z_{0.975} = 1.96 \tag{10}$$

$$\frac{\hat{p} + \dfrac{z^2}{2n} \ \pm\ z\sqrt{\dfrac{\hat{p}(1-\hat{p})}{n} + \dfrac{z^2}{4n^2}}}{1 + \dfrac{z^2}{n}} \tag{11}$$

### *7.5.6 Consistencia del evaluador*

La calificación fue manual y la realizaron los autores con la rúbrica escrita de la tabla 11, verificando cada cifra contra la base de datos en vivo cuando la referencia mostraba señales de estar desactualizada. Para reducir el sesgo del evaluador se tomaron cuatro medidas: la rúbrica y las reglas de puntaje parcial se fijaron antes de calificar; cada nivel cuenta con un ejemplo de referencia; las respuestas se calificaron contra la respuesta de referencia escrita de antemano y no contra la expectativa del calificador; y la corrección por desactualización se aplicó solo con evidencia de la base de datos, registrando cada caso. En esta versión del trabajo no se calculó un coeficiente de acuerdo entre evaluadores independientes, lo cual se reconoce como limitación (sección 9.11). El protocolo recomendado para cuantificarlo consiste en que un segundo evaluador califique a ciegas una muestra aleatoria del 20 % al 30 % de las respuestas y se calcule el kappa ponderado cuadrático (ecuación 12), donde $o_{ij}$ y $e_{ij}$ son las frecuencias observadas y esperadas de la matriz de acuerdo entre las categorías i y j y k es el número de niveles de la rúbrica [40]. Un valor $\kappa_w > 0.8$ se interpreta como acuerdo casi perfecto [41].

$$\kappa_w = 1 - \frac{\sum_{i,j} w_{ij}\, o_{ij}}{\sum_{i,j} w_{ij}\, e_{ij}}, \qquad w_{ij} = \frac{(i-j)^2}{(k-1)^2} \tag{12}$$

### *7.5.7 Criterios de validación*

Se establecieron como criterios de validación: (i) una exactitud global objetivo de 0.8, fijada como meta al inicio de la evaluación; (ii) cero alucinaciones en las respuestas generadas por la capa determinista; (iii) que la selección de herramienta supere de forma sostenida la exactitud de la síntesis final, para confirmar dónde se localiza el error; y (iv) que cada corrección se verifique repitiendo el banco completo y no solo las preguntas que motivaron el cambio, con el fin de detectar efectos colaterales. El grado de cumplimiento de cada criterio se discute en el capítulo 9.

### *7.5.8 Parámetros de reproducibilidad*

La tabla 12 reúne los parámetros de configuración con los que se obtuvieron los resultados del capítulo 9. Con estos valores, el código de los anexos y el banco de preguntas, el experimento puede repetirse en un equipo equivalente.

**Tabla 12.** Parámetros de configuración del sistema durante la evaluación.

| Parámetro | Valor | Ubicación |
|---|---|---|
| Modelo de generación principal | llama3.2:3b (2.0 GB, cuantizado) | api/.env · OLLAMA_MODEL |
| Modelo de respaldo | qwen2.5:7b (4.7 GB) | OLLAMA_MODEL_FALLBACK |
| Modelo de embeddings | nomic-embed-text (274 MB, 768 dimensiones, contexto 2 048 tokens) | OLLAMA_EMBED_MODEL |
| Temperatura de generación | 0.2 (valor por defecto de Ollama: 0.8) | OLLAMA_TEMPERATURE |
| Ventana de contexto | 4 096 tokens | OLLAMA_NUM_CTX |
| Máximo de tokens generados | 400 | OLLAMA_NUM_PREDICT |
| Tiempo máximo de generación | 90 s | OLLAMA_GEN_TIMEOUT |
| Modelos cargados simultáneamente | 2 (generación y embeddings) | Servicio de Ollama |
| Semilla aleatoria | No fijada | — |
| Tamaño de fragmento / solape | 800 / 120 tokens (4 caracteres por token) | ingest/.env · CHUNK_TOKENS |
| Fragmentación de tablas | Una oración por fila con encabezado repetido | ingest/chunker.py |
| Métrica de distancia | Coseno | Colección ppa_manuales |
| Resultados finales (top-k) | 8 | api/.env · TOP_K |
| Candidatos recuperados | top-k × 3 | api/rag.py |
| Impulso al documento autoritativo | +0.05 a PPA_02_diccionario_tags.md | BOOST_AUTORITATIVO |
| Fragmentos enviados al modelo | Máximo 5, truncados a 500 caracteres | CHUNKS_MODELO_MAX · TEXTO_MAX_CHARS |
| Puntos indexados | 759 (748 antes de capturar la línea real de origen, sección 8.6.1) | Qdrant |
| Intervalo de adquisición / tiempo de conexión | 5 000 ms / 8 000 ms | adquisicion-plc/.env |
| Tolerancia de correlación temporal | 3 s | api/plc_store.py |
| Lecturas mínimas para «sensor trabado» | 10 | MIN_LECTURAS_SENSOR_TRABADO |
| Software base | Debian 13, Python 3.13, Node.js 20, PostgreSQL 16 | Servidor |

*Fuente: elaboración propia.*

La semilla aleatoria de generación no fue fijada, por lo que no se garantiza la reproducción exacta, palabra por palabra, de cada salida textual del modelo. La variabilidad entre ejecuciones se redujo, en cambio, fijando una temperatura baja (0.2) y manteniendo constantes el resto de los parámetros de la tabla; esta es la misma variabilidad que se cuantifica en la prueba de consistencia de la sección 9.6.

# 8. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

Este capítulo describe cómo se implementó cada componente del sistema. Para explicar la lógica del software sin transcribir el código fuente se adoptó el siguiente criterio de representación: los procesos que involucran decisiones, bifurcaciones o ciclos, como el motor de razonamiento, la ingesta, la adquisición y el motor de alertas, se presentan mediante **diagramas de flujo**, porque permiten ver de un vistazo los caminos posibles de ejecución; los algoritmos compactos y secuenciales, como la fragmentación de tablas, la recuperación o la correlación temporal, se presentan mediante **pseudocódigo**, porque expresan con precisión los pasos y los parámetros sin la sintaxis específica del lenguaje. El código fuente completo, tal como está en producción, se incluye en los anexos C a Q para su consulta.

## 8.1 MODELOS DE LENGUAJE Y CONFIGURACIÓN DE LA INFERENCIA

Toda la inferencia se ejecuta de forma nativa mediante Ollama. El modelo principal es llama3.2:3b (2.0 GB), elegido tras comparar varios candidatos por ofrecer el mejor equilibrio entre velocidad y fiabilidad de *tool calling* en esta CPU. El modelo qwen2.5:7b (4.7 GB) se configuró como respaldo ante un fallo de conexión con el modelo principal, como salvaguarda de disponibilidad y no por su calidad. Los embeddings se generan con nomic-embed-text (274 MB, vectores de 768 dimensiones). La temperatura se redujo de 0.8, valor por defecto de Ollama, a 0.2 para disminuir la variabilidad entre ejecuciones de una misma pregunta (ecuación 2), un comportamiento deseable en un asistente técnico.

El cliente de Ollama implementa el respaldo entre modelos según el pseudocódigo del algoritmo que sigue.

**Algoritmo 1. Invocación del modelo con respaldo automático (api/ollama_client.py, Anexo G).**

```
función CHAT(mensajes, herramientas = ninguna):
    opciones ← {temperatura: 0.2, num_ctx: 4096, num_predict: 400}
    para cada modelo en [llama3.2:3b, qwen2.5:7b]:
        solicitud ← {modelo, mensajes, opciones, stream: falso}
        si herramientas ≠ ninguna: solicitud.tools ← herramientas
        intentar:
            r ← POST /api/chat (solicitud, tiempo_máximo = 90 s)
            devolver {modelo, contenido: r.message.content,
                      tool_calls: r.message.tool_calls,
                      tokens: r.eval_count}
        si falla la conexión:
            si modelo es el de respaldo: propagar el error
            continuar con el siguiente modelo
```

La incorporación de la segunda herramienta (iteración I-4) produjo un problema de estabilidad que ilustra el peso de la restricción de hardware. Las respuestas superaron los 180 segundos y la interfaz mostró errores de tiempo de espera agotado. El análisis identificó tres causas simultáneas. La primera fue que mantener cargados a la vez qwen2.5:7b y llama3.2:3b dejaba únicamente 145 MB libres de los 12 GB, lo que forzaba el uso de memoria de intercambio. La segunda fue que el *tool calling* duplica las llamadas al modelo (una para decidir la herramienta y otra para redactar), y cada una reintentaba de forma independiente contra el modelo de respaldo, sumando en el peor caso dos tiempos de espera completos. La tercera fue que la ronda final reinyectaba hasta cuatro mil tokens de contexto documental. La corrección combinó el cambio del modelo principal a llama3.2:3b, la reducción de la ventana de contexto y del tiempo de espera, la limitación de los fragmentos reinyectados (máximo cinco, truncados a 500 caracteres) y la captura explícita de las excepciones de red. Con ello, una pregunta documental pasó de más de 180 segundos a entre 31 y 74 segundos, y la memoria en reposo bajó de 9.7 GB con intercambio activo a 4.5 GB.

## 8.2 MOTOR DE RAZONAMIENTO: TOOL CALLING Y CAPA DETERMINISTA

El motor de razonamiento (api/rag.py) decide, para cada pregunta, si puede responderse de forma determinista o si requiere al modelo de lenguaje, y en este último caso qué herramientas invocar. La figura 8 presenta su flujo completo.

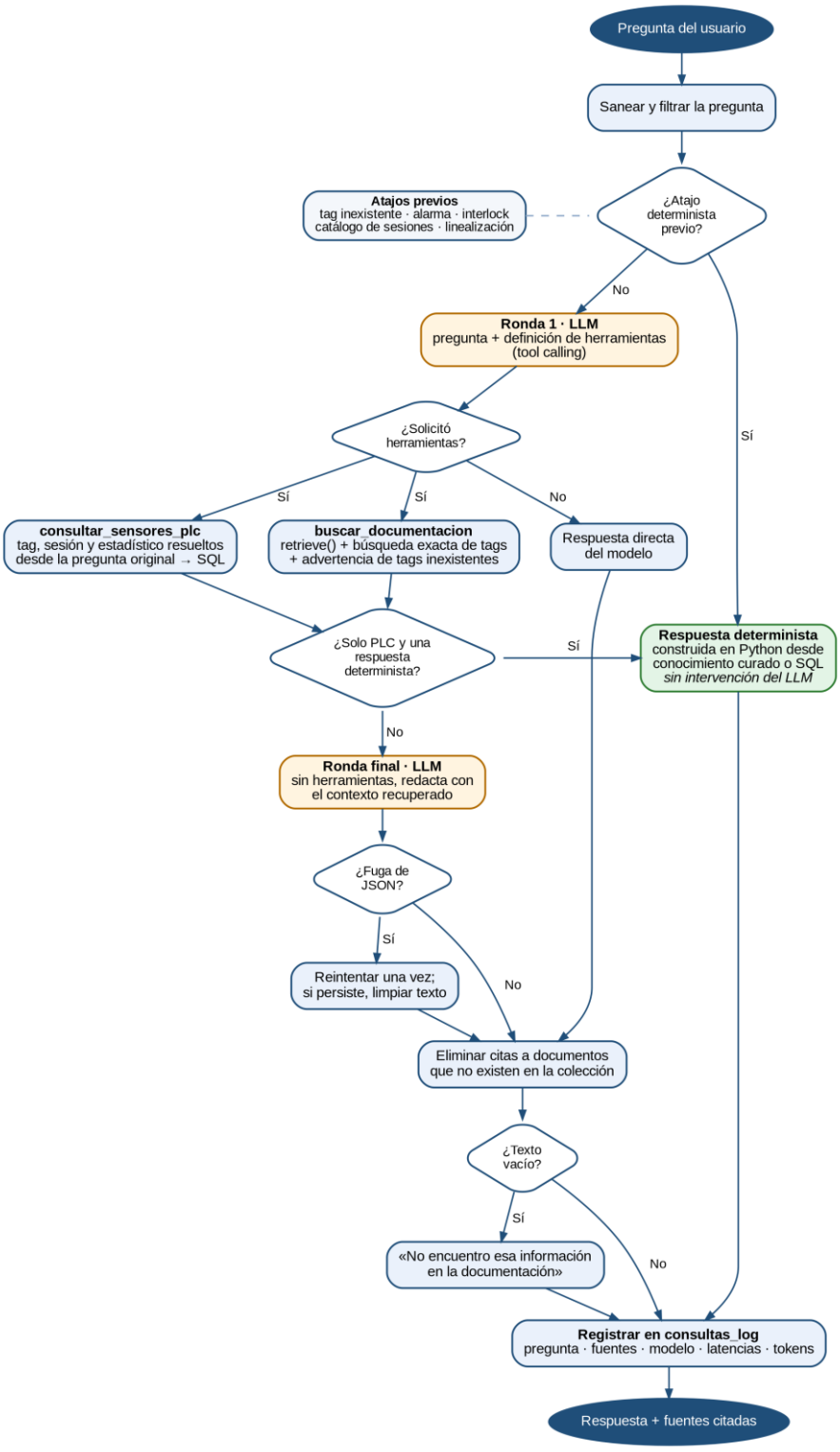



**Figura 8.** Diagrama de flujo del motor de razonamiento: atajos deterministas, tool calling, ronda final de síntesis y registro de la consulta.

*Fuente: Elaboración propia a partir de api/rag.py.*

### *8.2.1 Herramientas disponibles*

El modelo recibe, junto con la pregunta, la definición de dos herramientas. buscar_documentacion ejecuta la recuperación semántica sobre Qdrant; su descripción indica al modelo que debe usarla siempre que la pregunta nombre un equipo, tanque o tag, incluso si contiene la palabra «sensores», porque «qué sensores tiene el tanque X» es una pregunta de catálogo y no de lecturas. consultar_sensores_plc consulta lecturas recientes, resúmenes estadísticos (mínimo, máximo y promedio) o el listado de sesiones de captura. El propio modelo decide si la pregunta requiere una herramienta, ambas o ninguna. Tras ejecutarlas, una ronda final sin herramientas redacta la respuesta. Como se muestra en el capítulo 9, la elección de herramienta resultó confiable (acierto superior al 84 %) y los errores se concentraron en la ronda final.

Un detalle de diseño relevante es que los parámetros críticos de consultar_sensores_plc (tag, sesión y estadístico) no se toman de lo que el modelo decide enviar, que puede venir mal interpretado, sino que se resuelven directamente desde la pregunta original del usuario mediante reglas de coincidencia sobre el diccionario de tags.

### *8.2.2 Capa de respuestas deterministas*

La capa de respuestas deterministas es la contribución de diseño más relevante del proyecto y la que produjo el mayor salto de exactitud medido. Para las preguntas en las que el sistema puede identificar sin ambigüedad el dato y la regla de decisión, la respuesta se construye en Python a partir de la fila recuperada de la base de datos o del conocimiento curado, sin que el modelo participe en su redacción. El Algoritmo 2 resume la secuencia de atajos que se evalúan antes de invocar al modelo.

**Algoritmo 2. Evaluación de atajos deterministas previos al modelo (api/rag.py, Anexo I).**

```
función RESPUESTA_DETERMINISTA_PREVIA(pregunta):
    // Se evalúan en orden; el primero que aplique responde.
    si la pregunta menciona un tag que no existe en el diccionario
       (excluyendo identificadores AL-, IL- y TK- reales):
        devolver "El tag X no existe en la documentación de la PPA"
    si la pregunta pide el umbral o la acción de una alarma:
        devolver regla de ALARMAS_PLC o "no existe un umbral documentado"
    si la pregunta compara un valor contra un interlock o alarma:
        v ← estadístico pedido de la sesión indicada (SQL)
        devolver EVALUAR_OPERADOR(operador, v, umbral) redactado en texto fijo
    si la pregunta es sobre el catálogo de sesiones
       (cuántas hay, la más reciente, la de más lecturas, si hay activa):
        devolver resultado de la consulta SQL correspondiente
    si la pregunta pide la ecuación o el valor linealizado de un sensor:
        devolver LINEALIZACION_PLC[tag] (y el cálculo, si hay dato crudo)
    devolver ninguna   // no aplica: se continúa con el modelo

regla común: si la consulta SQL no devuelve filas, nunca se inventa un número;
             se responde que no hay datos para esa sesión o variable.
```

Además de estos atajos previos, cuando el modelo invoca únicamente consultar_sensores_plc y la herramienta produce exactamente una respuesta determinista (por ejemplo, el último valor de un único tag), esa respuesta sustituye a la ronda final de síntesis. En la prueba de consistencia, esta capa no registró alucinaciones en ninguna de las veinte llamadas repetidas (sección 9.6).

### *8.2.3 Conocimiento curado de la planta*

El módulo api/conocimiento_plc.py reúne, en estructuras de datos, la información que la capa determinista y el motor de alertas necesitan como fuente de verdad. Todos los valores se transcribieron manualmente desde los documentos fuente, sin completar ningún dato por suposición. En lugar de mostrar el código, las tablas 13 a 14 presentan su contenido.

**Tabla 13.** Ecuaciones de linealización y rangos documentados de los sensores analógicos.

| Tag | Pendiente m | Término b | Unidad | Rango documentado |
|---|---|---|---|---|
| TT_102_PV | 0.00225 | −148.47 | °C | 0 – 100 °C |
| TT_100_PV | 0.0022 | −16.377 | °C | 0 – 100 °C |
| PE_102_PV | 0.0016 | −9.187 | cm | 0 – 40 cm |
| LE_101_PV | 0.0007 | −2.331 | cm | 2.5 – 40 cm |

*Fuente: api/conocimiento_plc.py; PPA_03_mapa_direcciones_io.md.*

**Tabla 14.** Alarma y enclavamientos documentados utilizados por el sistema.

| Código | Equipo | Condición | Acción documentada | ¿Tag capturado? |
|---|---|---|---|---|
| AL-01 | TK de calentamiento | TT_102_PV ≥ 60 °C (HH) | Cortar resistencia (IL-05); revisar | Sí |
| IL-01 | RES-1 | LSL_102_PROT = 1 | Permite calentamiento (flotador en serie) | No |
| IL-03 | Bomba M1 | LSL_100 = 1 | No arrancar con nivel bajo (marcha en seco) | Sí |
| IL-04 | TK-100 | LSH_100 = 1 | Cerrar válvula de llenado (rebose) | Sí |
| IL-05 | RES-1 | TT_102_PV > 60 °C | Paro de la resistencia | Sí |

*Fuente: api/conocimiento_plc.py; PPA_05 y PPA_06.*

El módulo incluye también las etiquetas de los estados digitales (por ejemplo, un valor 0 en CV_100_CMD significa «cerrada» y un valor 1 en RES_1_CMD significa «encendida»). Esta tabla se creó después de observar que el modelo invertía el estado de una válvula al redactar libremente (sección 9.7).

## 8.3 PIPELINE RAG: DE LOS DOCUMENTOS A LA RESPUESTA

### *8.3.1 Ingesta de documentos*

El pipeline de ingesta procesa cuatro formatos, cada uno con su propia unidad de segmentación: PDF, por página; Word, como documento completo; Excel, por hoja; y Markdown, por capítulo, sección o actividad de práctica, detectados a partir de los encabezados. La figura 9 muestra el flujo completo, incluida la verificación de duplicados que hace idempotente la ingesta.

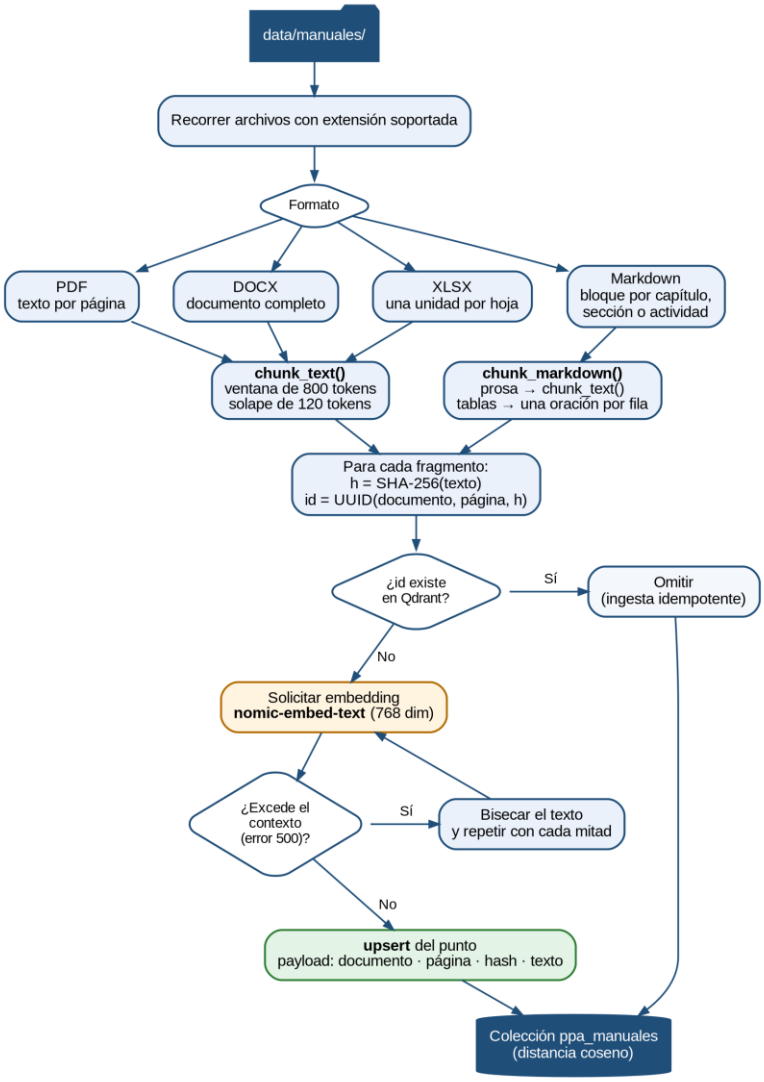



**Figura 9.** Diagrama de flujo del pipeline de ingesta e indexación.

*Fuente: Elaboración propia a partir de ingest/ingest.py, loaders.py y embeddings.py.*

### *8.3.2 Fragmentación del texto*

Un documento no se indexa como una sola unidad: se divide en fragmentos porque el modelo de embeddings tiene un límite de contexto y porque los fragmentos breves y enfocados producen búsquedas más precisas. El sistema combina dos estrategias. La genérica divide la prosa en ventanas de aproximadamente 800 tokens con un solape de 120 tokens para no perder contexto en los bordes. La especializada, construida en la iteración I-5, detecta las tablas Markdown y las divide fila por fila, repitiendo el encabezado de columna en cada fragmento (Algoritmo 3).

**Algoritmo 3. Fragmentación consciente de tablas (ingest/chunker.py, Anexo J).**

```
función CHUNK_MARKDOWN(texto):
    fragmentos ← [ ];  prosa ← [ ]
    i ← 0
    mientras i < número de líneas:
        si la línea i es encabezado de tabla y la línea i+1 es separador |---|:
            VACIAR(prosa)                          // la prosa previa va a CHUNK_TEXT
            encabezados ← celdas de la línea i
            para cada fila siguiente con el mismo número de celdas:
                oración ← unir("encabezado: valor" para cada par no vacío, "; ")
                agregar oración a fragmentos    // una fila = un fragmento
            i ← primera línea después de la tabla
        si no:
            agregar la línea i a prosa;  i ← i + 1
    VACIAR(prosa)
    devolver fragmentos

función CHUNK_TEXT(texto, tokens = 800, solape = 120):
    // ventana deslizante por palabras; 1 token ≈ 4 caracteres
    acumular palabras hasta 3 200 caracteres, emitir fragmento,
    conservar las últimas palabras que sumen 480 caracteres como solape
```

El fragmentador especializado corrigió uno de los errores más instructivos del proyecto: el fragmentador genérico cortaba en ocasiones una tabla entre el encabezado y una fila de datos, dejando un fragmento con un número sin contexto. En un caso, el modelo asignó un sensor al tanque equivocado porque el fragmento recibido no contenía la palabra «tanque». Con la nueva estrategia, cada fila se convierte en una oración autocontenida del tipo «tag_plc: LSH_100; tanque: TK-100; tipo: interruptor de nivel alto». La reindexación de la colección tras este cambio pasó de 216 a 630 puntos.

### *8.3.3 Embeddings e indexación idempotente*

Cada fragmento se convierte en un vector de 768 dimensiones mediante nomic-embed-text. Si un fragmento excede el contexto de 2 048 tokens del modelo, lo que puede ocurrir con tablas extensas, el sistema lo bisecta de forma recursiva en lugar de abortar la ingesta. Cada punto se almacena con un identificador determinista derivado del documento, la página y el *hash* del texto, lo que hace idempotente la ingesta: repetirla no duplica puntos (Algoritmo 4).

**Algoritmo 4. Indexación idempotente con bisección (ingest/ingest.py y embeddings.py, Anexos M y L).**

```
función INGESTAR(documento):
    para cada (página, texto) en CARGAR(documento):
        fragmentador ← CHUNK_MARKDOWN si es .md, si no CHUNK_TEXT
        para cada fragmento en fragmentador(texto):
            h ← SHA256(fragmento);  id ← UUID(documento, página, h)
            si EXISTE(id) en Qdrant: continuar       // ya indexado
            para cada (subtexto, vector) en EMBED_O_DIVIDIR(fragmento):
                UPSERT(id(subtexto), vector,
                       payload = {documento, página, hash, texto: subtexto})

función EMBED_O_DIVIDIR(texto):
    intentar: devolver [(texto, EMBED(texto))]
    si error 500 (excede contexto) y el texto tiene más de 20 palabras:
        mitad ← número de palabras / 2
        devolver EMBED_O_DIVIDIR(primera mitad) + EMBED_O_DIVIDIR(segunda mitad)
```

### *8.3.4 Recuperación híbrida*

Cuando llega una pregunta, el sistema recupera más candidatos de los que finalmente utiliza (tres veces top-k), aplica un impulso de +0.05 al puntaje de los fragmentos del documento autoritativo, que es el diccionario de tags (ecuación 13), reordena y recorta a los top-k resultados. En paralelo, una búsqueda exacta por expresión regular sobre los nombres de tags actúa como respaldo independiente del ranking semántico (Algoritmo 5).

$$s'(d) = s(d) + \beta \cdot \mathbb{1}[d \in \mathcal{D}_{\text{aut}}], \qquad \beta = 0.05 \tag{13}$$

**Algoritmo 5. Recuperación híbrida con impulso autoritativo (api/rag.py, Anexo I).**

```
función RECUPERAR(consulta, pregunta_original, k = 8):
    q ← EMBED(consulta)
    candidatos ← Qdrant.buscar(q, límite = 3k, distancia = coseno)
    para cada c en candidatos:
        si c.documento = "PPA_02_diccionario_tags.md": c.puntaje ← c.puntaje + 0.05
    ordenar candidatos por puntaje descendente;  resultado ← primeros k
    // respaldo léxico: un tag mencionado literalmente siempre se encuentra
    para cada tag en EXPRESIÓN_REGULAR(pregunta_original ∪ consulta):
        si tag está en el diccionario: insertar su fila al inicio de resultado
    faltantes ← tags mencionados que no existen en el diccionario
    si faltantes ≠ ∅:
        insertar al inicio una advertencia: "los tags faltantes NO existen;
        no los confundas con tags parecidos ni inventes una definición"
    devolver resultado
```

Este mecanismo resolvió la contradicción entre la bitácora histórica y el diccionario de tags vigente sin editar los documentos fuente: el diccionario gana prioridad en el ranking y la mención literal de un tag como «LSH-100» o «PE-102» siempre lo

recupera. La colección ppa_manuales contiene 759 puntos verificados en el sistema en producción (figuras 10 y 11); esta cifra corresponde a la reindexación descrita en la sección 8.6.1, posterior a la incorporación de la línea real de origen a cada fragmento.

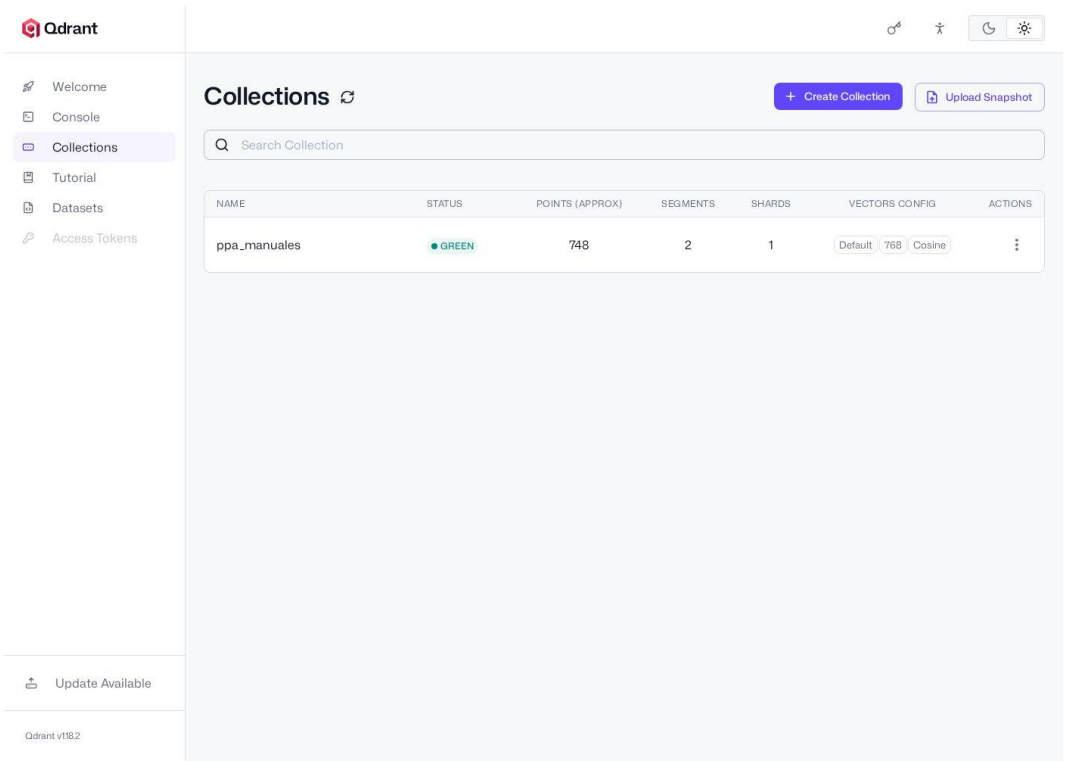



**Figura 10.** Panel de Qdrant con la colección ppa_manuales.

*Fuente: Elaboración propia (imagen del sistema en producción).*

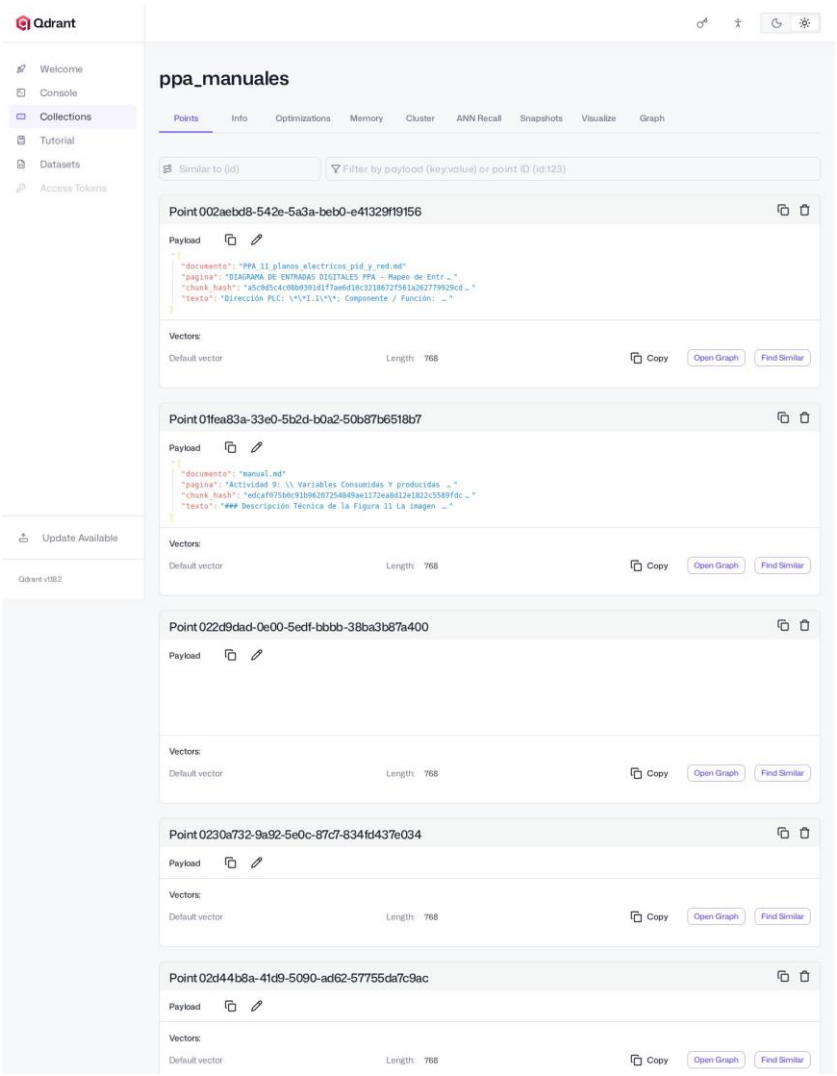



**Figura 11.** Detalle de los puntos y *payloads* indexados en Qdrant.

*Fuente: Elaboración propia (imagen del sistema en producción).*

## 8.4 BASE DE DATOS DEL HISTÓRICO DE SENSORES

PostgreSQL 16, desplegado como contenedor Docker asociado únicamente a la interfaz local, almacena dos tablas: sesiones_plc, que registra cada sesión de captura y su estado, y lecturas_plc, que registra cada lectura individual asociada a una sesión y a un tag, con su valor crudo y su valor en unidades de ingeniería. Un índice compuesto sobre (sesion_id, tag_plc, timestamp) acelera las consultas por sesión y variable. De forma independiente, una base SQLite almacena el registro de consultas utilizado en la evaluación. La figura 12 presenta el modelo de datos.

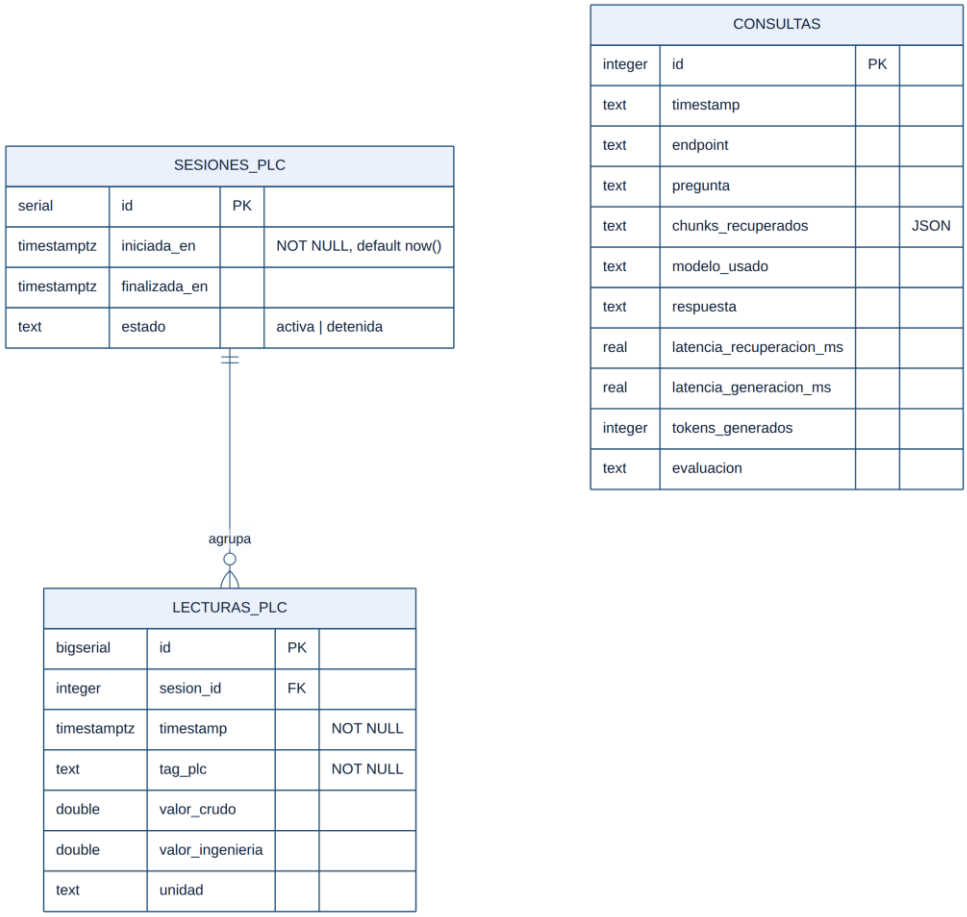



**Figura 12.** Modelo entidad-relación de las bases de datos del sistema.

*Fuente: Elaboración propia a partir de infra/init-postgres.sql y api/db.py.*

El diseño según el cual la sesión la controla la base de datos y no el proceso convierte a PostgreSQL en el punto de coordinación entre la interfaz, que inicia o detiene sesiones, y el servicio de adquisición, que consulta la base para decidir si debe capturar. Al momento de la redacción existían ocho sesiones de captura reales, registradas entre el 13 y el 22 de julio y el 18 de agosto de 2026 (figura 13).

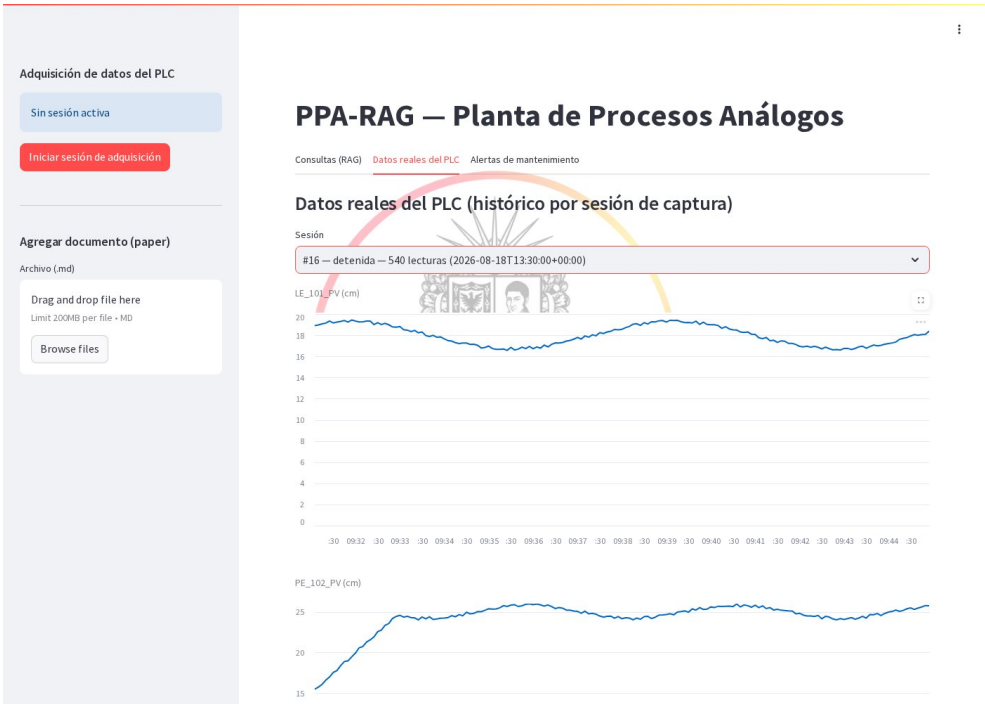



**Figura 13.** Datos reales del PLC en la pestaña correspondiente de la interfaz.

*Fuente: Elaboración propia (imagen del sistema en producción).*

## 8.5 BACKEND DE SERVICIOS

El backend, desarrollado en FastAPI, expone quince endpoints REST (figura 14). La tabla 15 resume los principales.

**Tabla 15.** Endpoints principales de la API REST.

| Endpoint | Método | Función |
|---|---|---|
| /consulta | POST | Recibe una pregunta, ejecuta el motor de razonamiento y devuelve la respuesta con fuentes |
| /consulta/stream | POST | Igual al anterior, en *streaming*: la interfaz muestra el texto a medida que se genera |
| /salud | GET | Estado de Ollama, Qdrant y Node-RED |
| /metricas | GET | Totales de uso, latencia p95 y consultas sin contexto relevante |
| /plc/sesiones/iniciar, /detener | POST | Inicio y fin de una sesión de captura real |
| /plc/lecturas | GET | Lecturas reales filtradas por sesión y tag |
| /plc/alertas | GET | Evaluación de las seis reglas de alerta sobre una sesión |
| /documentos/subir | POST | Carga de un documento Markdown y reindexación |
| /documentos/listar | GET | Lista los documentos indexados con su número de fragmentos, fecha real de carga y tamaño |
| /documentos/contenido | GET | Devuelve un documento completo, línea por línea, para su lectura en la interfaz |

*Fuente: elaboración propia.*

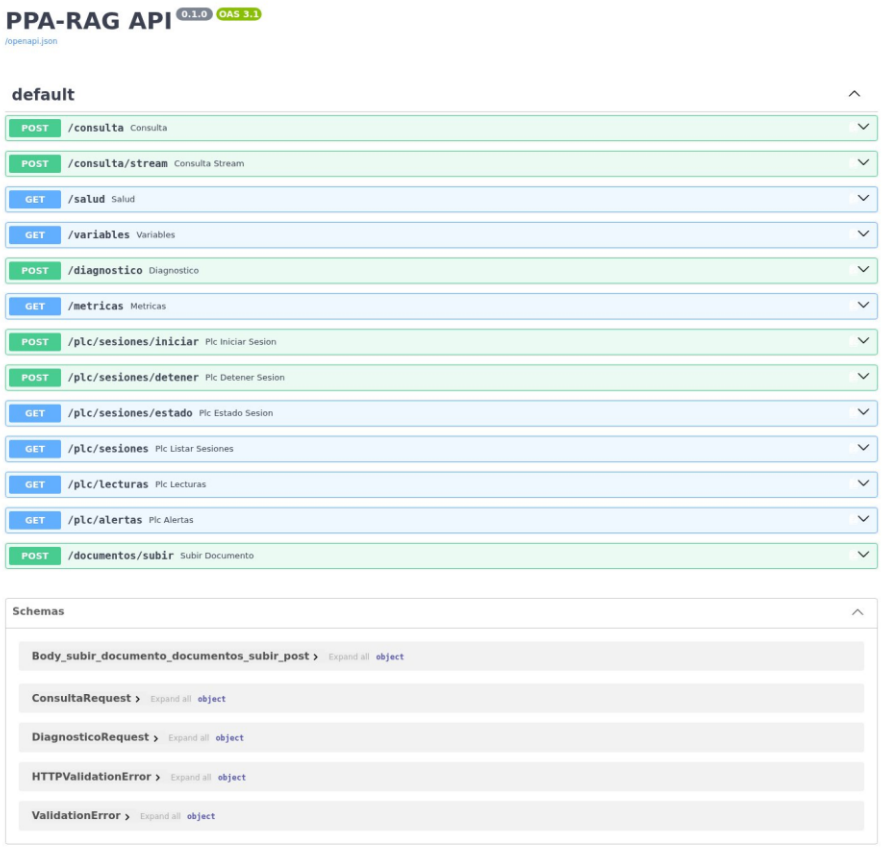



**Figura 14.** Documentación interactiva (Swagger) de los endpoints de FastAPI.

*Fuente: Elaboración propia (imagen del sistema en producción).*

Cada llamada a /consulta queda registrada en la tabla consultas de SQLite con la pregunta, los fragmentos recuperados en formato JSON, el modelo utilizado, la respuesta, las latencias de recuperación y generación, los tokens generados y un campo de evaluación. Este registro es el insumo de toda la evaluación del capítulo 9. Por seguridad, el endpoint de carga restringe los formatos aceptados desde la interfaz a Markdown, aunque el pipeline completo admite más formatos, para mantener control sobre la calidad de lo que ingresa a la base de conocimiento sin revisión previa.

## 8.6 INTERFAZ DE USUARIO

La interfaz, desarrollada en Streamlit, cuenta con cuatro pestañas (figura 15): Consultas, con el chat principal, la respuesta en *streaming* y un contador de tiempo en vivo que informa al usuario de que una espera de 30 a 90 segundos es normal en este hardware; Datos reales del PLC, con selector de sesión, filtro por sensor, gráficas por

variable y tabla de valores; Alertas de mantenimiento, que evalúa las reglas sobre cualquier sesión, activa o histórica, y muestra la ubicación física y la fotografía del elemento en alerta; y Documentos, que hace auditable la base de conocimiento del asistente. La barra lateral permite iniciar y detener sesiones de captura y cargar documentos (figura 16).




**Figura 15.** Interfaz Streamlit, pantalla principal.

*Fuente: Elaboración propia (imagen del sistema en producción).*




**Figura 16.** Barra lateral: control de sesión de captura y carga de documentos.

*Fuente: Elaboración propia (imagen del sistema en producción).*

### *8.6.1 Pestaña Documentos: trazabilidad hasta la línea de origen*

La pestaña Documentos hace visible, y por tanto verificable, la base de conocimiento completa del asistente: una tabla lista cada documento indexado con su número de fragmentos en Qdrant, la fecha real de carga (tomada del archivo en el servidor, no de un registro aparte) y su tamaño en bytes (figura 17); un selector permite abrir cualquiera de esos documentos y leerlo completo, con la numeración de línea real de cada renglón (figura 18). El propósito es que nada de lo que el asistente cita quede fuera del alcance del usuario: todo documento listado ahí es, literalmente, todo lo que el sistema tiene disponible para responder.

Esta pestaña se apoya en dos endpoints de solo lectura (tabla 15): /documentos/listar recorre la colección de Qdrant, cuenta cuántos fragmentos tiene cada documento y consulta el archivo real en disco para obtener su fecha y tamaño; /documentos/contenido devuelve el archivo completo, línea por línea, a partir de su nombre.

Llevar esta trazabilidad hasta el renglón exacto exigió extender el fragmentador de la sección 8.3.2, que hasta ese momento no conservaba la línea de origen de cada fragmento, solo el título de la sección a la que pertenecía. La extensión captura la línea de forma exacta para las filas de una tabla (la línea en la que aparece esa fila en el archivo fuente) y como rango de inicio y fin para los párrafos de prosa, y propaga ese dato hasta el *payload* de Qdrant (linea_inicio, linea_fin) y hasta cada fuente citada en una respuesta. Incorporar este campo exigió reindexar la colección completa, que pasó de 748 a 759 puntos.

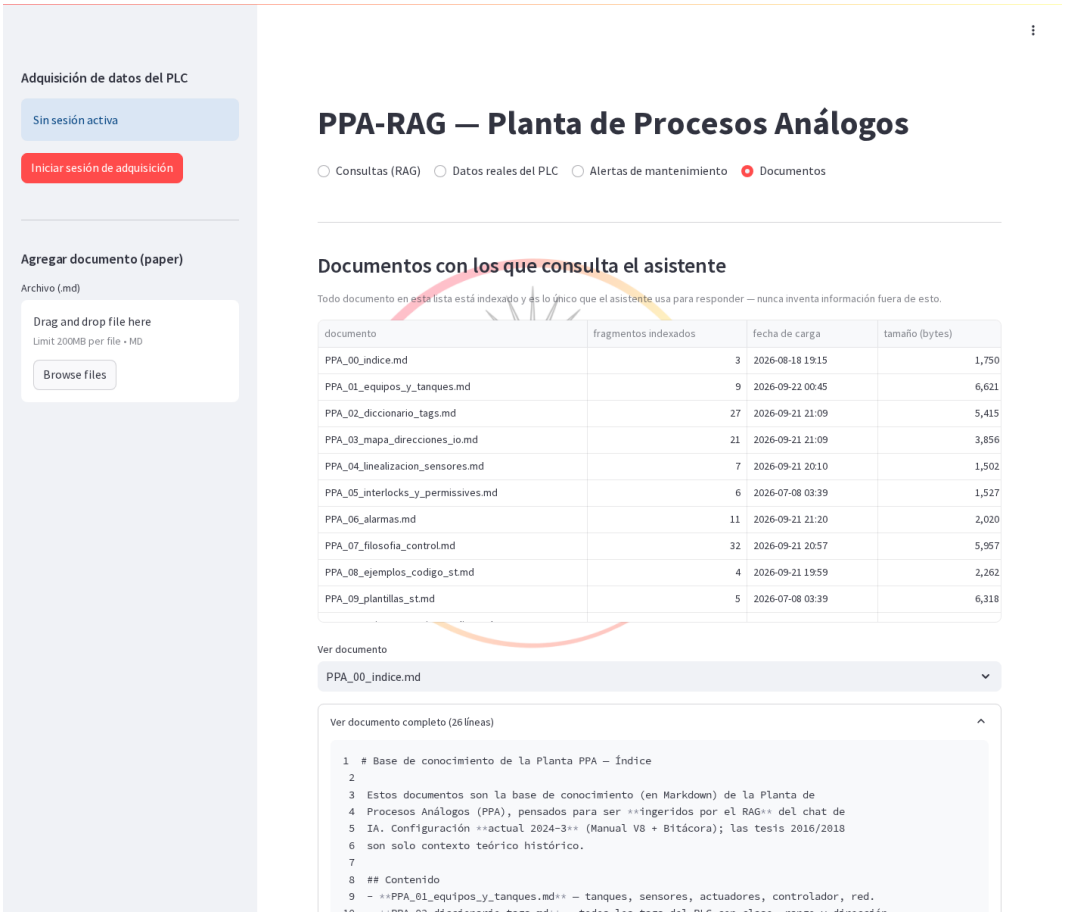



**Figura 17.** Pestaña Documentos: tabla de documentos indexados con fragmentos, fecha de carga y tamaño.

*Fuente: Elaboración propia (imagen del sistema en producción).*

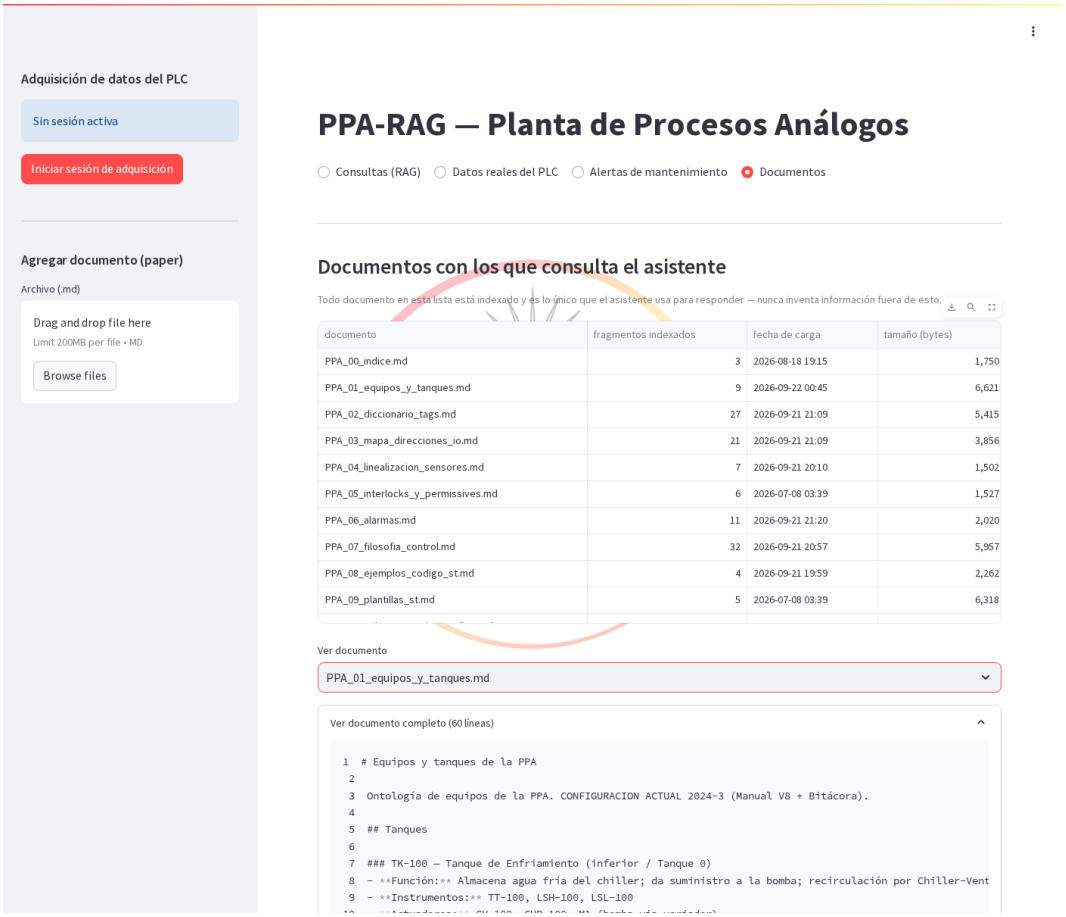



**Figura 18.** Pestaña Documentos: lectura completa de un documento con numeración de línea real.

*Fuente: Elaboración propia (imagen del sistema en producción).*

Durante el desarrollo de esta pestaña se evaluó llevar la misma trazabilidad directamente a la pestaña Consultas, mediante botones bajo cada fuente citada que abrieran Documentos en la línea exacta, junto con la fotografía del sensor cuando la respuesta trataba sobre uno. La prueba funcionó, pero se decidió no incorporarla: duplicaba una interacción que ya existía en las pestañas Documentos y Alertas, y sobrecargaba visualmente una respuesta que hasta entonces se mantenía simple. Consultas se conservó, por tanto, tal como se describió en la sección 8.7, con su lista de fuentes citadas en texto (figura 19); la trazabilidad hasta la línea y la fotografía quedaron concentradas en Documentos y en Alertas.

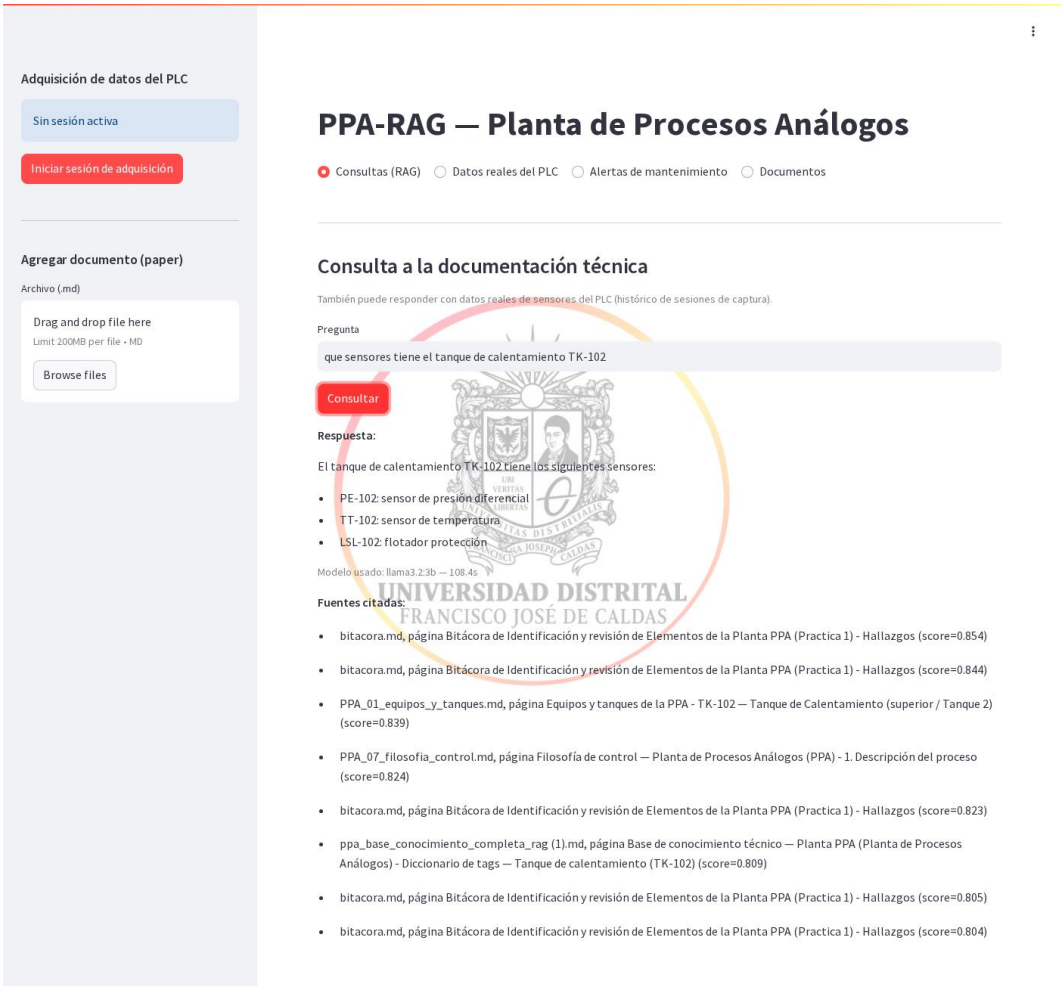



**Figura 19.** Pestaña Consultas en su versión final: respuesta con lista de fuentes citadas en texto, sin botones ni fotografías.

*Fuente: Elaboración propia (imagen del sistema en producción).*

### *8.6.2 Filtro por sensor en Datos reales del PLC*

La pestaña Datos reales del PLC incorporó un selector «Filtrar por sensor/actuador» junto al selector de sesión (figura 20). Por defecto la gráfica y la tabla muestran todas las variables de la sesión, igual que en la versión original; al elegir un sensor específico, ambas se filtran para mostrar solo esa variable, lo que evita tener que localizarla entre las trece disponibles cuando el interés es puntual.

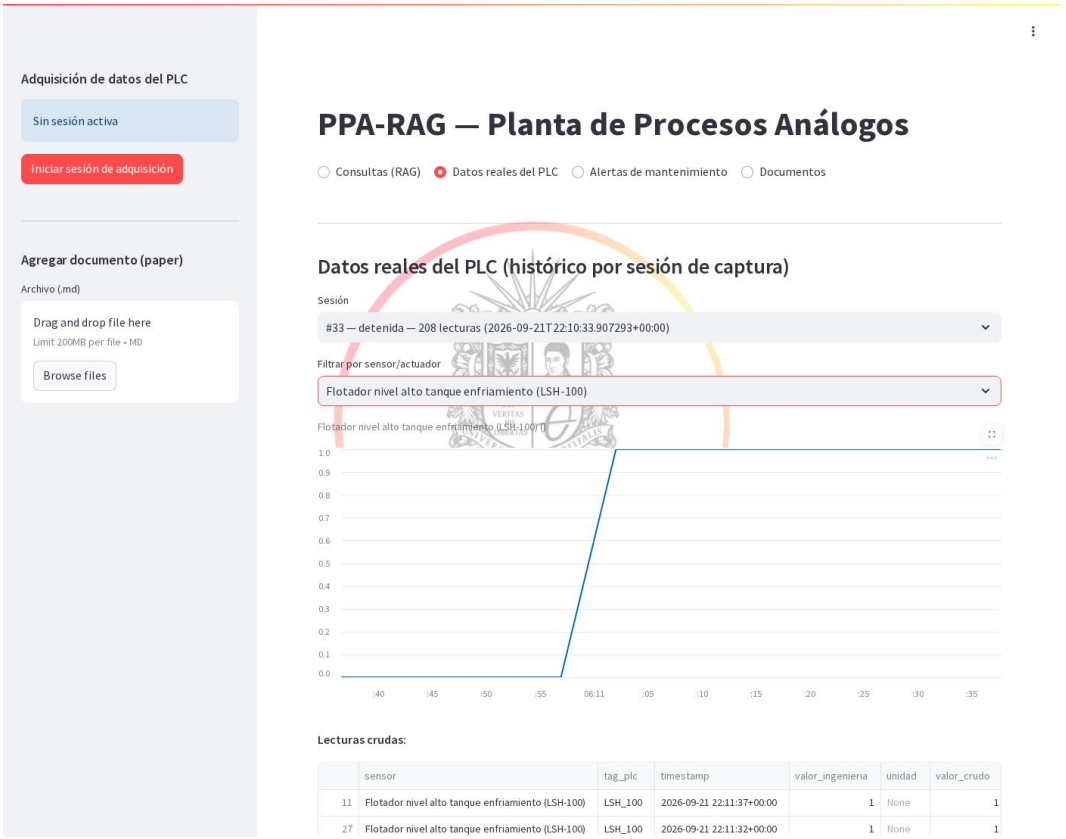



**Figura 20.** Datos reales del PLC con el filtro por sensor aplicado (LSH-100, sesión 33).

*Fuente: Elaboración propia (imagen del sistema en producción).*

## 8.7 TRAZABILIDAD DE UNA CONSULTA

Con el fin de mostrar el flujo de datos, las decisiones del sistema y la trazabilidad de la respuesta, se siguió una consulta real en producción: «¿Qué es el LSH-100?». La figura 21 presenta el diagrama de secuencia de esa consulta. Cada paso numerado corresponde a una acción verificable: la decisión de herramienta (paso 6), la recuperación con sus puntajes (pasos 9 a 11), la limpieza de citas (paso 14) y el registro en la base de consultas (paso 16).

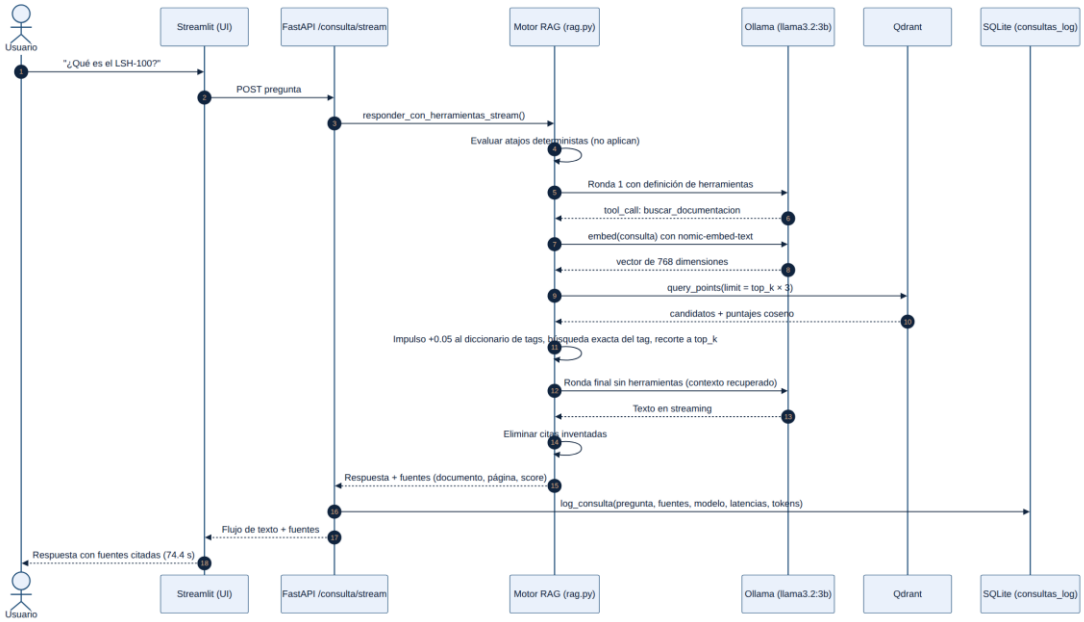



**Figura 21.** Diagrama de secuencia de una consulta documental real, desde la pregunta hasta el registro y la respuesta con fuentes.

*Fuente: Elaboración propia.*

Los atajos deterministas no aplicaron porque se trata de una pregunta documental. En la primera ronda, el modelo decidió invocar buscar_documentacion; la recuperación trajo los candidatos con el impulso del documento autoritativo ya aplicado; la ronda final sintetizó la respuesta y el sistema eliminó cualquier cita a un documento inexistente antes de mostrarla. La respuesta obtenida fue:

> *«El LSH-100 es un sensor de nivel alto para el tanque de enfriamiento TK-100, utilizado como interruptor de nivel alto.»*

El modelo utilizado fue llama3.2:3b, con un tiempo de respuesta de 74.4 segundos. Las fuentes citadas y sus puntajes se presentan en la tabla 16; su registro permite auditar después de dónde provino cada afirmación (figura 22).

**Tabla 16.** Fuentes recuperadas para la consulta «¿Qué es el LSH-100?».

| Documento | Puntaje |
|---|---|
| PPA_02_diccionario_tags.md | 1.000 |
| PPA_11_planos_electricos_pid_y_red.md | 0.658 |
| PPA_06_alarmas.md | 0.567 |
| manual.md | 0.560 |
| bitacora.md | 0.538 |

| Documento | Puntaje |
|---|---|
| PPA_07_filosofia_control.md | 0.537 |

*Fuente: elaboración propia.*




**Figura 22.** Consulta real con respuesta y fuentes citadas.

*Fuente: Elaboración propia (imagen del sistema en producción).*

Al registrar la misma pregunta en dos momentos distintos del proyecto, la redacción final varió (una vez más extensa y otra más breve), mientras que el contenido factual y las fuentes se mantuvieron idénticos. Este comportamiento evidencia que la variabilidad del sistema se localiza en la síntesis del modelo y no en la recuperación, que es determinista.

## 8.8 ADQUISICIÓN DE DATOS DEL PLC Y SIMULADOR

### *8.8.1 Servicio de adquisición EtherNet/IP*

El servicio de adquisición es un proceso Node.js independiente, ejecutado bajo un usuario de sistema dedicado, que se comunica con el PLC Allen-Bradley mediante EtherNet/IP. La tabla 17 presenta los trece tags que captura, con su dirección CIP y, para los analógicos, los coeficientes de linealización.

**Tabla 17.** Tags adquiridos del PLC y parámetros de conversión.

| Tag | Dirección CIP | Tipo | m | b | Unidad |
|---|---|---|---|---|---|
| PE_102_PV | Nivel_presion_Diferencial | Analógico | 0.0016 | −9.187 | cm |
| TT_102_PV | Temperatura_bits | Analógico | 0.00225 | −148.47 | °C |
| LE_101_PV | Sensor_Ultrasonico | Analógico | 0.0007 | −2.331 | cm |
| TT_100_PV | Local:3:I.Ch3Data | Analógico | 0.0022 | −16.377 | °C |
| LSH_100 | Local:1:I.Data.3 | Booleano | — | — | — |
| LSL_100 | Local:1:I.Data.4 | Booleano | — | — | — |
| CHR_100_CMD | Local:2:O.Data.6 | Booleano | — | — | — |
| CV_100_CMD | Local:2:O.Data.3 | Booleano | — | — | — |
| RES_1_CMD | Local:2:O.Data.5 | Booleano | — | — | — |
| CV_104_CMD | Local:2:O.Data.8 | Booleano | — | — | — |
| CV_102_CMD | Local:2:O.Data.4 | Booleano | — | — | — |
| Raspberry_Res2 | Raspberry:O.Data[0] | Crudo | — | — | bits |
| Raspberry_PCV101 | Raspberry:O.Data[1] | Crudo | — | — | bits |

*Fuente: adquisicion-plc/index.js.*

El diseño del servicio responde a problemas reales de la comunicación con hardware industrial, que se reflejan en su diagrama de flujo (figura 23). Se implementó una lectura tolerante por tag: si un tag falla, el ciclo no se aborta, y solo si fallan todos se fuerza la reconexión. El objeto de conexión se reconstruye por completo en cada reconexión, porque la librería lo implementa extendiendo un *socket* de Node.js que queda inutilizable una vez destruido; reutilizar la instancia fallaba de forma silenciosa. Se añadió un tiempo máximo de conexión de ocho segundos mediante una carrera de promesas, pues sin él una red inalcanzable dejaba la conexión bloqueada hasta el tiempo de espera TCP del sistema operativo, cercano a dos minutos. Finalmente, se reforzó el manejo de errores asíncronos a partir de dos fallas observadas: un error de *host* inalcanzable que detenía el proceso y una excepción síncrona lanzada por la versión 1.x de la librería dentro de su manejador de cierre de conexión.

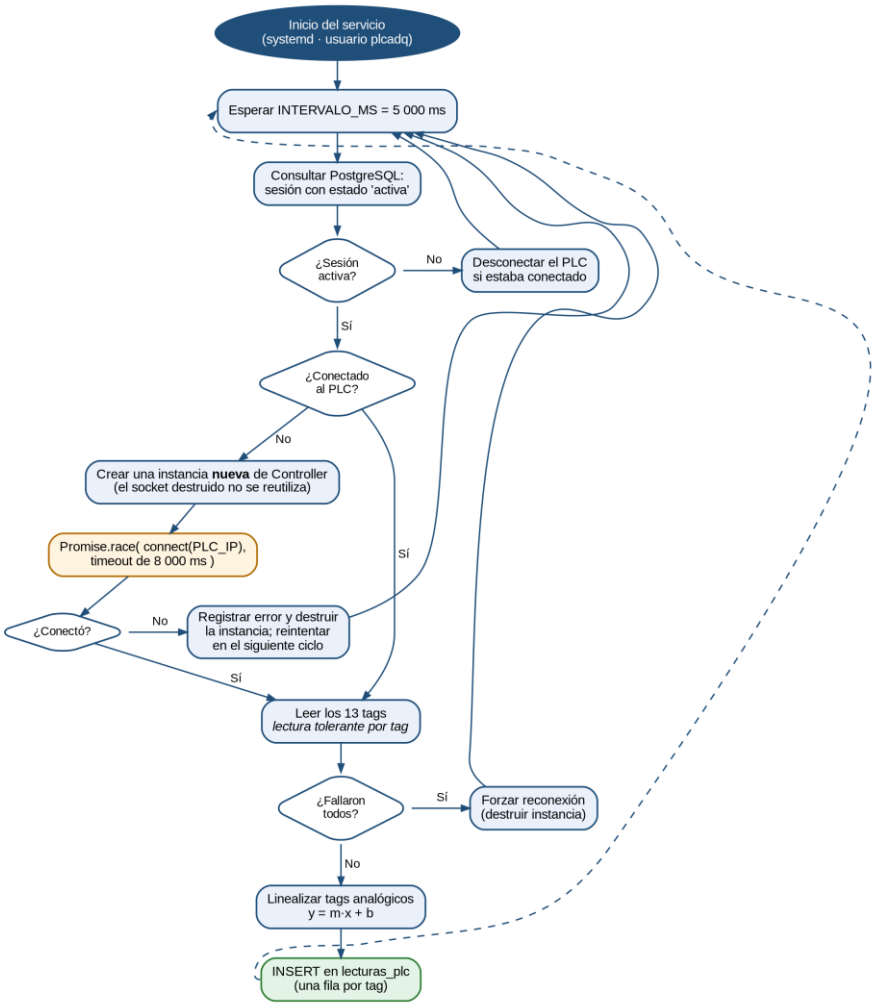



**Figura 23.** Diagrama de flujo del servicio de adquisición de datos del PLC.

*Fuente: Elaboración propia a partir de adquisicion-plc/index.js.*

La ampliación de 3 a 13 tags incorporó sensores adicionales, estados de actuadores y lecturas crudas de la Raspberry Pi asociada al proceso. Las diez direcciones nuevas se validaron en su sintaxis de direccionamiento y en la robustez del código ante fallos, pero no en su correspondencia exacta con el hardware, limitación que se retoma en la sección 9.11.

### *8.8.2 Simulador en Node-RED*

Mientras no hubo conectividad con la planta, Node-RED, ejecutado también bajo un usuario dedicado, sirvió como banco de pruebas independiente. El flujo genera una señal de corriente simulada según la ecuación 14, compuesta por una corriente base $I_0$, una componente senoidal de amplitud A y frecuencia f, y ruido gaussiano ε(t), con

la posibilidad de inyectar manualmente una anomalía de entre seis y ocho amperios (figura 24).

$$i(t) = I_0 + A\sin(2\pi ft) + \varepsilon(t), \qquad \varepsilon(t) \sim \mathcal{N}(0, \sigma^2) \tag{14}$$

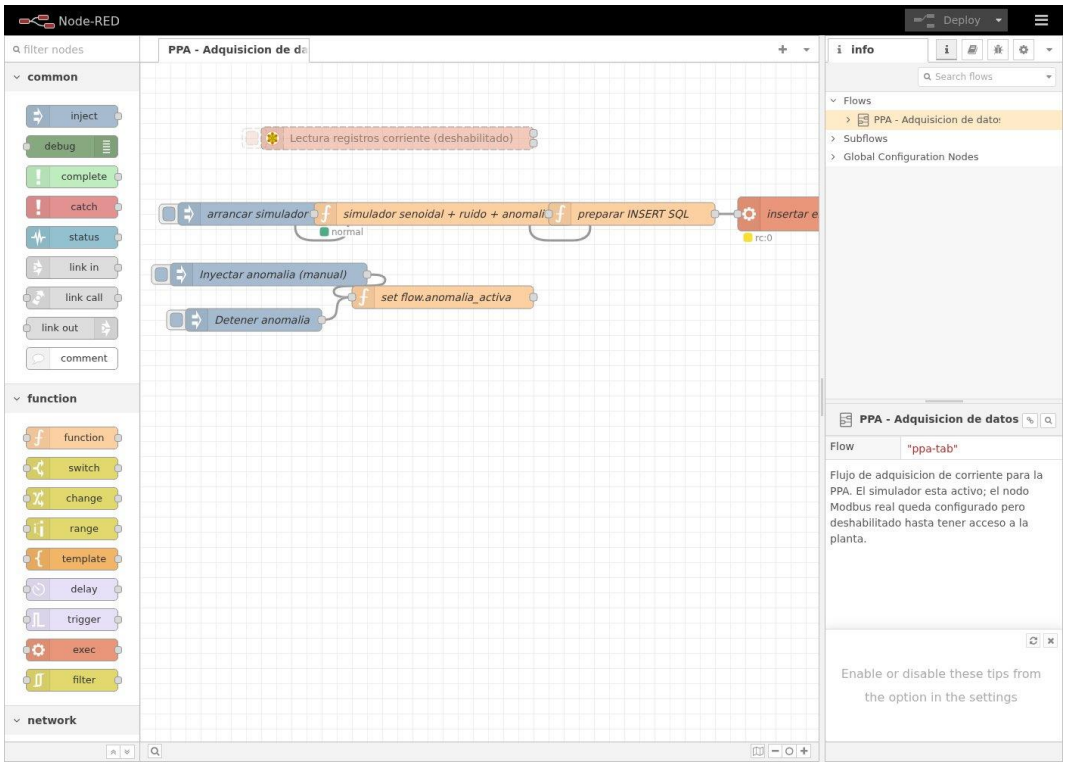



**Figura 24.** Editor de Node-RED con el flujo del simulador.

*Fuente: Elaboración propia (imagen del sistema en producción).*

Los datos simulados se almacenan en una base SQLite separada por completo del histórico real en PostgreSQL. El nodo Modbus visible en el flujo se conserva instalado como referencia de una etapa anterior, pero está deshabilitado de forma deliberada.

## 8.9 MOTOR DE ALERTAS DE MANTENIMIENTO

La primera versión de la pestaña de alertas mezclaba, sin distinguirlas para el usuario, una alerta de sobretemperatura calculada con datos reales de PostgreSQL y una sección de «alertas de mantenimiento (simulado)» que, al revisarla, leía la señal sintética de Node-RED. Ante este hallazgo se estableció como requisito que las alertas se basaran exclusivamente en datos capturados del PLC y que cualquier verificación documentada que no pudiera resolverse con datos reales se omitiera o se marcara, en lugar de simularse. El bloque simulado, el botón de diagnóstico, los endpoints

/variables y /diagnostico y el módulo lecturas_store.py se eliminaron, y el motor se reconstruyó sobre datos reales.

### *8.9.1 Reglas implementadas*

La documentación define enclavamientos que dependen de dos señales que no se capturan: el estado de marcha de la bomba M1 y el flotador de protección de la resistencia (LSL_102_PROT), que es un cableado en serie y no un punto de *software* del PLC. En esos dos casos se utilizó un tag del mismo equipo como sustituto razonado, marcado como aproximado en la API y en la interfaz. La tabla 18 resume las seis reglas.

**Tabla 18.** Reglas de alerta implementadas sobre datos reales del PLC.

| N.° | Regla | Tags | Fuente | ¿Aproximada? |
|---|---|---|---|---|
| R1 | Sobretemperatura del tanque de calentamiento (AL-01) | TT_102_PV | PPA_06_alarmas.md | No |
| R2 | Nivel alto con válvula de llenado abierta (AL-04 / IL-04) | LSH_100 + CV_100_CMD | PPA_05 | No |
| R3 | Nivel bajo en el tanque de enfriamiento (AL-03 / IL-03) | LSL_100 | PPA_05 | Sí: no se captura M1 |
| R4 | Resistencia encendida con posible falta de agua (IL-01) | RES_1_CMD + PE_102_PV | PPA_05 | Sí: no se captura LSL_102_PROT |
| R5 | Sensor fuera del rango documentado (falla de instrumento) | 4 analógicos | PPA_03 | No |
| R6 | Sensor o actuador posiblemente trabado | 4 analógicos | Estadística sobre datos reales | Sí |

*Fuente: elaboración propia.*

La figura 25 muestra cómo el endpoint /plc/alertas evalúa las reglas y separa el resultado en dos bloques. Toda esta lógica es determinista y está escrita en Python y SQL; el modelo de lenguaje no participa en decidir ni en redactar ninguna alerta.

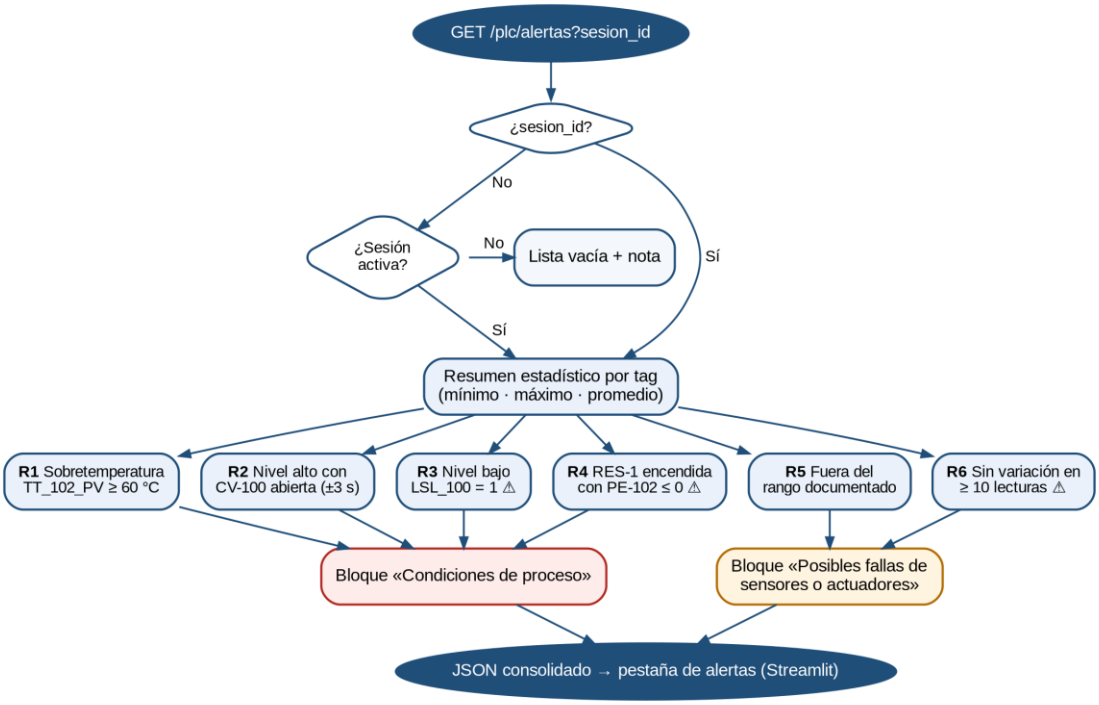



**Figura 25.** Diagrama de flujo del motor de alertas y de las seis reglas.

*Fuente: Elaboración propia a partir de api/main.py.*

### *8.9.2 Correlación temporal entre tags*

Las reglas R2 y R4 evalúan condiciones conjuntas (por ejemplo, nivel alto y válvula abierta al mismo tiempo). Como cada ciclo de adquisición inserta una fila por tag, no una fila con todos los tags, estas condiciones exigen emparejar dos series de tiempo por cercanía temporal y no por igualdad exacta. El Algoritmo 6 describe la función de correlación, implementada con una subconsulta lateral en PostgreSQL.

**Algoritmo 6. Correlación temporal entre dos tags (api/plc_store.py, Anexo H).**

```
función LECTURAS_CORRELACIONADAS(sesión, tag_a, tag_b, tolerancia = 3 s):
    pares ← [ ]
    para cada lectura a de tag_a en la sesión (más reciente primero):
        b ← lectura de tag_b en la sesión que minimiza |b.tiempo − a.tiempo|
        si |b.tiempo − a.tiempo| ≤ tolerancia:
            agregar (a.tiempo, a.valor, b.tiempo, b.valor) a pares
    devolver pares

// Uso en la regla R2:
ocurrencia ← primer par de LECTURAS_CORRELACIONADAS(s, LSH_100, CV_100_CMD)
             con valor_a = 1 y valor_b = 1
si ocurrencia existe: emitir alerta "nivel alto con CV-100 abierta" (IL-04)
```

Antes de conectarla al endpoint, la función se probó de forma aislada contra las sesiones reales 14, 15 y 16, y devolvió pares correlacionados con diferencias de milisegundos entre ambas lecturas.

### *8.9.3 Validación con sesiones reales*

La figura 26 muestra el resultado sobre la sesión 16: la alerta de sobretemperatura AL-01, con un máximo registrado de 62.99 °C y su acción documentada, y el bloque de fallas de sensor sin indicios. Esta sesión solo ejercita la regla R1; para verificar las reglas R5 y R6 se analizaron las sesiones 26, 27, 28 y 61, todas breves por coincidir con pruebas de arranque del servicio de adquisición. Esta circunstancia favorece la validación, porque es en la reconexión con el PLC cuando más se necesita detectar un dato físicamente imposible.

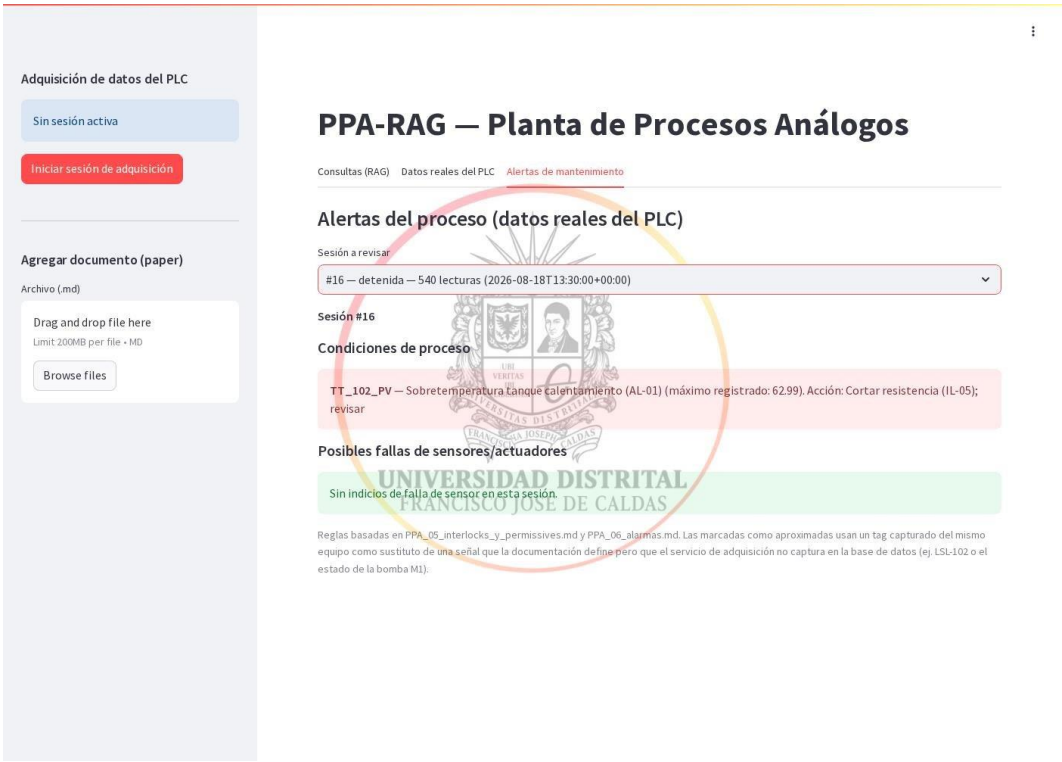



**Figura 26.** Pestaña de alertas de mantenimiento sobre la sesión 16.

*Fuente: Elaboración propia (imagen del sistema en producción).*

En la sesión 61 (figura 27) el sistema marcó TT_102_PV en −124.92 °C y PE_102_PV en −9.19 cm, ambos fuera del rango documentado (0–100 °C y 0–40 cm). Aplicando la linealización (ecuación 4), PE_102_PV coincide con el término independiente de su

ecuación, es decir, una cuenta cruda de cero, y TT_102_PV corresponde a una cuenta baja distinta de cero, la firma típica de un canal analógico que aún no se estabiliza tras la reconexión. LE_101_PV presentaba una lectura válida (3.42 cm) y no se marcó.

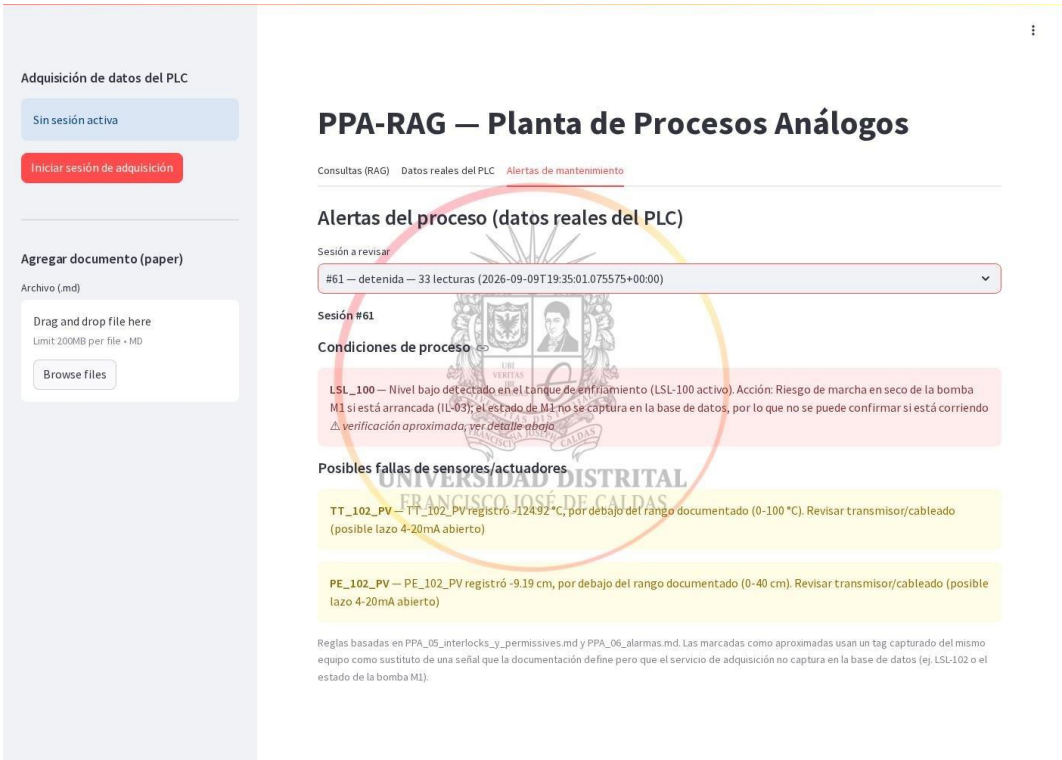



**Figura 27.** Alertas de la sesión 61: fallas de sensor detectadas al reconectar con el PLC.

*Fuente: Elaboración propia (imagen del sistema en producción).*

La sesión 28, con 176 lecturas, es la única de las cuatro que alcanza las diez lecturas mínimas de la regla R6; por ello presenta seis alertas, una de «fuera de rango» y otra de «trabado» por cada sensor analógico afectado (figura 28, tabla 19). Los tres valores constantes coinciden con el término independiente de su ecuación: el canal permaneció en una cuenta cruda de cero durante toda la sesión.

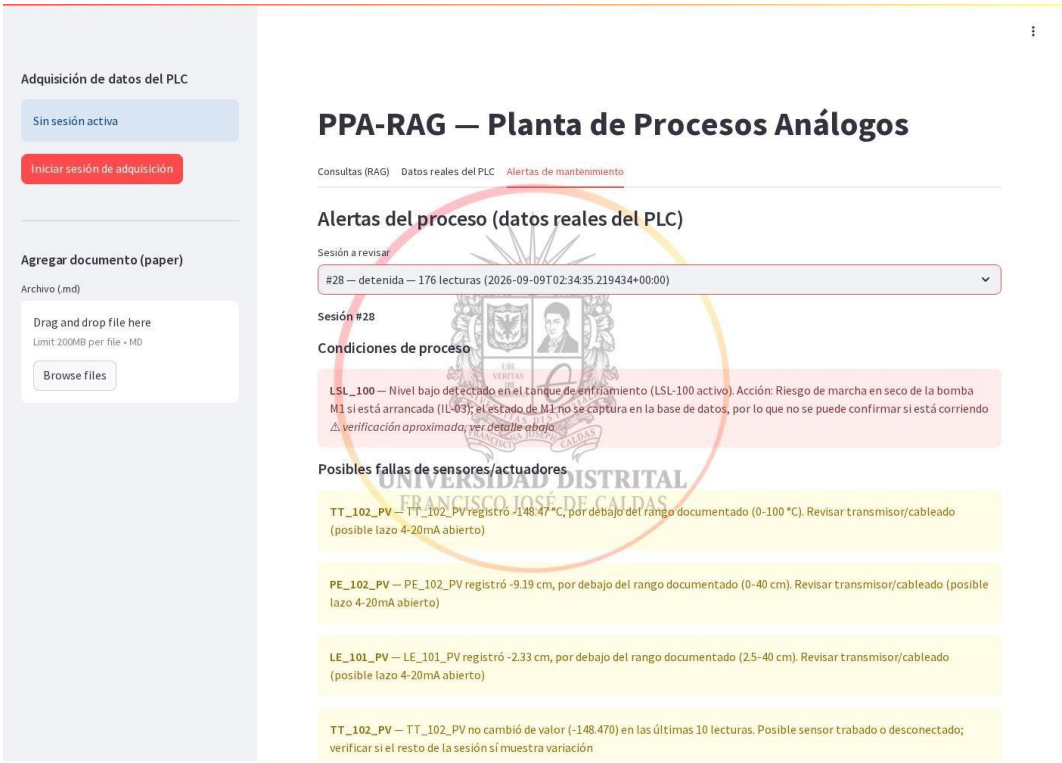



**Figura 28.** Alertas de la sesión 28: fuera de rango y sensor trabado en los tres sensores analógicos.

*Fuente: Elaboración propia (imagen del sistema en producción).*

**Tabla 19.** Lecturas constantes detectadas en la sesión 28.

| Tag | Lectura (constante) | Alerta |
|---|---|---|
| TT_102_PV | −148.47 °C | Fuera de rango y sin variación en 10 lecturas |
| PE_102_PV | −9.19 cm | Fuera de rango y sin variación en 10 lecturas |
| LE_101_PV | −2.33 cm | Fuera de rango y sin variación en 10 lecturas |

*Fuente: elaboración propia.*

La sesión 27 confirma que el umbral de diez lecturas actúa como filtro: con 99 lecturas repartidas en trece tags (entre siete y ocho por tag), ningún sensor alcanza el mínimo de la regla R6, de modo que el sistema reporta solo las tres alertas de «fuera de rango» y se abstiene de diagnosticar un sensor trabado aunque el patrón sea idéntico (figura 29). En el extremo opuesto, la sesión 26 tiene once lecturas, prácticamente una por tag, y las tres alertas de «fuera de rango» aparecen desde el primer ciclo, lo que confirma que la regla R5 no necesita historial (figura 30). Con menos evidencia, el sistema reporta menos y no extrapola.

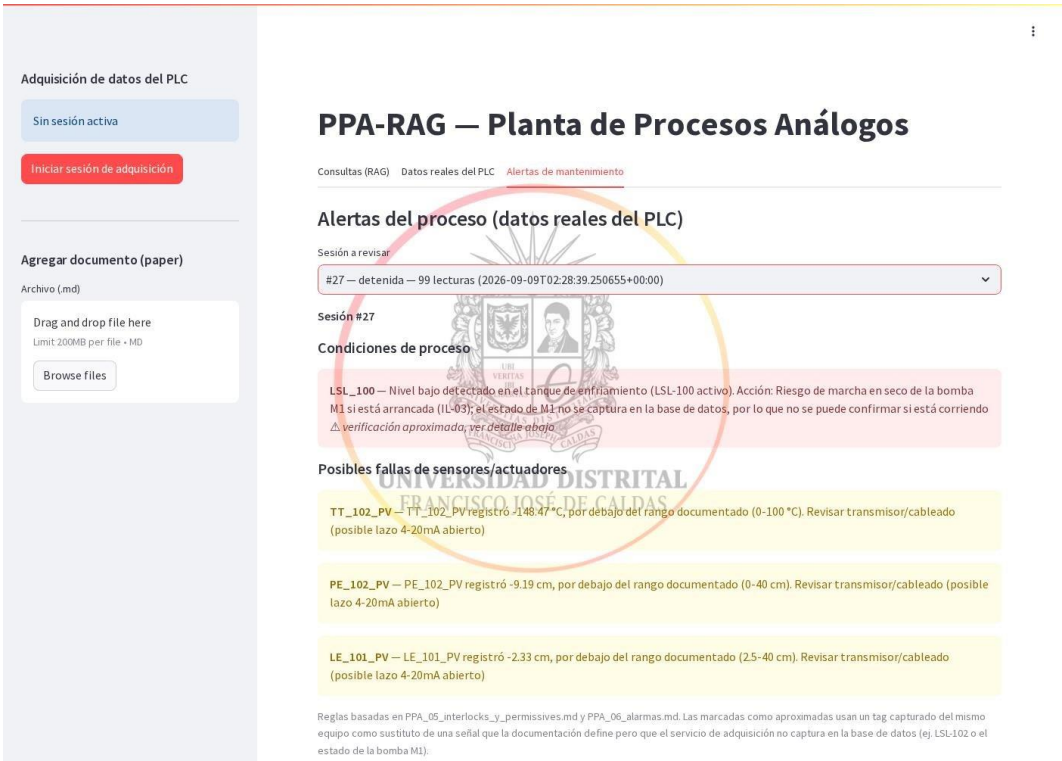



**Figura 29.** Alertas de la sesión 27: fuera de rango sin alcanzar el umbral de sensor trabado.

*Fuente: Elaboración propia (imagen del sistema en producción).*

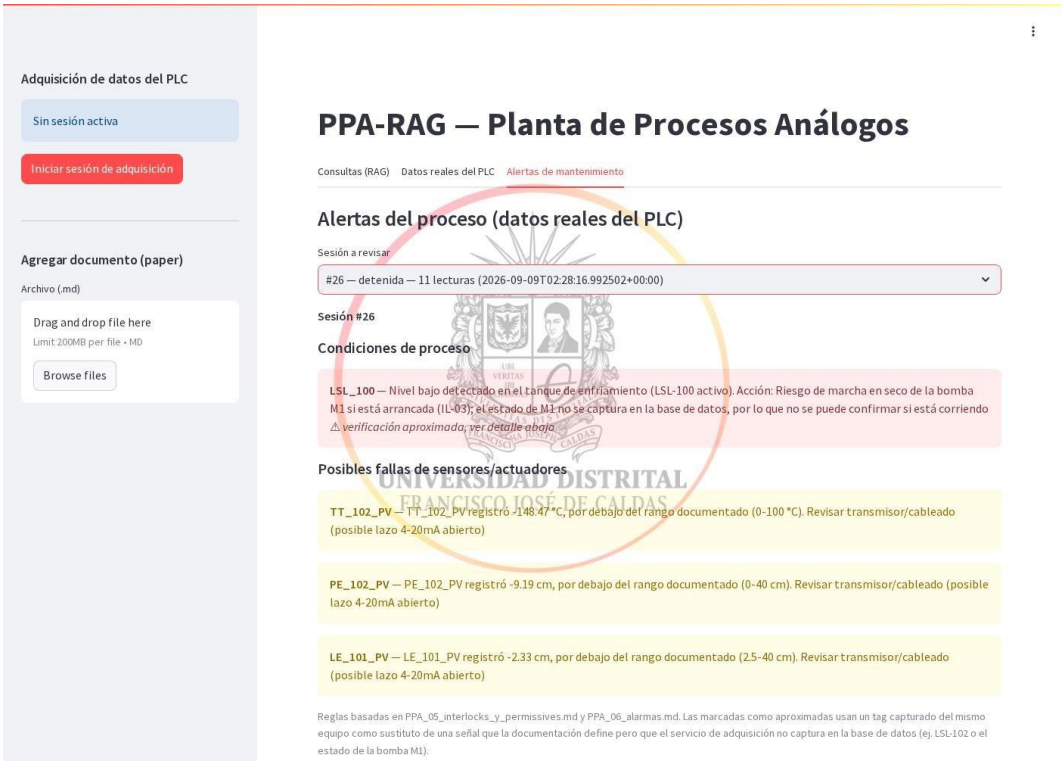



**Figura 30.** Alertas de la sesión 26: fuera de rango con una sola lectura por tag.

*Fuente: Elaboración propia (imagen del sistema en producción).*

Las cuatro sesiones comparten una alerta de proceso: nivel bajo en el tanque de enfriamiento (LSL-100, regla R3). En todas, el flotador LSL_100 presenta valor 1 desde la primera lectura; el flotador de nivel bajo está físicamente activo en la planta. La alerta se mantiene marcada como aproximada porque el estado de la bomba M1 no se captura: el sistema advierte del riesgo de marcha en seco sin poder confirmar si la bomba está en funcionamiento.

### *8.9.4 Ubicación física y fotografía del elemento en alerta*

Leer el mensaje de una alerta no siempre basta para localizar el elemento físico al que se refiere, sobre todo para quien no conoce la planta de memoria. Cada alerta se enriqueció, por eso, con dos datos adicionales tomados de la documentación y nunca inventados: dónde está ubicado físicamente el sensor o actuador (por ejemplo, «flotador atornillado en la pared del tanque de enfriamiento, tanque inferior, en la parte alta») y su fotografía real, tomada del montaje. La ubicación proviene de un nuevo diccionario, UBICACIONES_PLC, agregado al módulo de conocimiento curado (api/conocimiento_plc.py) a partir del documento PPA_01_equipos_y_tanques.md, siguiendo la misma disciplina de transcripción manual descrita en la sección 8.2.

La cobertura fotográfica no es completa y el sistema lo refleja en lugar de disimularlo: hay fotografía para ambos sensores de temperatura, el de presión diferencial, el de nivel ultrasónico, los dos flotadores de nivel, las tres válvulas, las dos resistencias y el chiller; no hay fotografía para las tres balizas, la válvula de mezcla CV-102 ni el flotador de protección LSL-102, y la alerta correspondiente se muestra sin ese respaldo visual hasta que la fotografía esté disponible.

El comportamiento se verificó con datos reales de la sesión 33: la alerta de nivel alto (LSH-100) muestra su fotografía y su ubicación correctas, y lo mismo ocurre con la de nivel bajo (LSL-100) (figura 31).




**Figura 31.** Alertas de la sesión 33 con ubicación física y fotografía del sensor: nivel alto (LSH-100) y nivel bajo (LSL-100).

*Fuente: Elaboración propia (imagen del sistema en producción).*

## 8.10 VERIFICACIÓN DE LA INTEGRACIÓN

Cada cambio relevante se cerró con una secuencia de verificación fija: compilación del código sin errores de sintaxis; prueba aislada de las funciones nuevas contra datos reales; reinicio del servicio y comprobación de que /salud reporta sanas sus tres dependencias (Ollama, Qdrant y Node-RED); llamada real al endpoint afectado con un resultado esperado conocido (por ejemplo, la alerta AL-01 sobre la sesión 16); y verificación visual de la interfaz tras su reinicio. Además, se comprobó que todo el sistema se recupera sin intervención manual tras un reinicio del servidor, gracias a los servicios systemd con reinicio automático y a la política restart: unless-stopped de los contenedores (RNF-04). Durante esa verificación se identificó que, si Tailscale no ha asignado su IP cuando Docker arranca, los contenedores asociados a esa IP quedan

en estado *created*; se añadió un servicio de verificación posterior al arranque que espera la IP y recrea los contenedores cuando es necesario.

## 8.11 APOYO DE HERRAMIENTAS DE IA EN LA CONSTRUCCIÓN DEL SISTEMA

Buena parte del código de los componentes (contenedores de infraestructura, backend, pipeline de ingesta y fragmentación, servicio de adquisición e interfaz) se implementó con apoyo de un asistente de programación basado en un LLM, utilizado siempre fuera del entorno del proyecto y nunca integrado al sistema. Los autores entregaban al asistente la documentación técnica de la planta y la decisión de arquitectura ya tomada, y le solicitaban, mediante instrucciones breves, una primera versión del código. Los autores trasladaban después manualmente al entorno real cada fragmento resultante.

Ese código nunca se aceptó sin revisión. Cada entrega se revisó línea por línea y se probó contra el sistema real, y varias de las fallas descritas en este capítulo, como el error del fragmentador de tablas, el comportamiento de la librería ethernet-ip al reconectar y la confusión inicial entre valor crudo y valor de ingeniería, se detectaron precisamente en esa revisión. La tabla 20 documenta, por componente, la instrucción entregada al asistente y la intervención de los autores.

**Tabla 20.** Registro del apoyo de IA por componente: instrucción entregada e intervención de los autores.

| Componente | Instrucción entregada al asistente | Intervención de los autores |
|---|---|---|
| Cliente de Ollama | Cliente en Python con chat normal, chat con *streaming* y embeddings, con temperatura configurable | Ajuste experimental de la temperatura hasta 0.2 comparando respuestas reales |
| Recuperación y tool calling | Función de búsqueda en Qdrant que decida entre documentación y sensores y dé más peso al diccionario de tags | Ajuste caso por caso de las preguntas que fallaban |
| Conocimiento curado | Diccionarios de alarmas, interlocks y rangos con una estructura dada, junto con la documentación | La IA generó solo la estructura; todos los valores se transcribieron a mano contra las fuentes |
| Cargadores de documentos | Cuatro cargadores (PDF, Word, Excel y Markdown) con la estructura definida | Prueba contra documentos reales de la planta |
| Fragmentador de tablas | Corrección para que cada fragmento repita el encabezado de la tabla | Reindexación y verificación en Qdrant (de 216 a 630 puntos) |
| PostgreSQL en Docker | Contenedor persistente, puerto no expuesto y creación de tablas en el primer arranque | Verificación del volumen, la política de reinicio y el *binding* a 127.0.0.1 |

| Componente | Instrucción entregada al asistente | Intervención de los autores |
|---|---|---|
| Backend FastAPI | Trece endpoints con registro de cada consulta en SQLite | Definición de los endpoints y del esquema del registro |
| Interfaz Streamlit | Tres pestañas, respuesta en *streaming* y contador de tiempo | Diseño de la experiencia de espera en CPU |
| Adquisición EtherNet/IP | Servicio que lea trece tags periódicamente, con la librería y las direcciones dadas | Reintentos y tolerancia por tag construidos a partir de fallas reales |
| Simulador Node-RED | Señal con corriente base, componente senoidal, ruido y anomalía inyectable | El resto del flujo se construyó a mano en el editor visual |
| Correlación temporal | Función que busque la lectura más cercana en el tiempo de otro tag | Prueba aislada contra las sesiones 14, 15 y 16 |
| Motor de alertas | Convertir las seis reglas documentadas en funciones Python, marcando las aproximadas | Extracción de las reglas desde la documentación |

*Fuente: elaboración propia.*

Las decisiones que definieron el sistema, como los cinco bloques funcionales, la elección de Qdrant y de EtherNet/IP, la capa de respuestas deterministas y el rediseño del motor de alertas, fueron siempre de los autores, al igual que el diseño del banco de preguntas, la calificación de cada prueba y la verificación de cada cifra contra el sistema en producción. El apoyo de IA se limitó a acelerar la escritura del código una vez tomada la decisión.

# 9. ANÁLISIS DE RESULTADOS

## 9.1 PROTOCOLO APLICADO

Los resultados de este capítulo se obtuvieron con el protocolo de la sección 7.5: el banco de 145 preguntas (G1 = 45, G2 = 50, G3 = 50), la ejecución secuencial contra el sistema en producción, la calificación manual con las rúbricas de la tabla 11 y las métricas de las ecuaciones 5 a 11. Se realizaron ocho pruebas completas y una prueba de consistencia.

## 9.2 EVOLUCIÓN DE LA EXACTITUD GLOBAL

La tabla 21 resume la evolución de la exactitud en las seis primeras pruebas, calificadas con la rúbrica de tres niveles, y la figura 32 incluye además las Pruebas 7 y 8, calificadas con la rúbrica de seis niveles.

**Tabla 21.** Evolución de la exactitud por prueba (rúbrica de tres niveles).

| Prueba | Cambio aplicado | Global | G1 | G2 | G3 |
|---|---|---|---|---|---|
| 1 | Línea base | 0.290 | 0.45 | 0.18 | 0.24 |
| 2 | *Prompt* reforzado, temperatura 0.2 y validaciones | 0.353 | 0.43 | 0.38 | 0.25 |
| 3 | qwen2.5:3b como modelo principal (descartado) | 0.273 | 0.26 | 0.41 | 0.15 |
| 4 | Capa de respuestas deterministas | 0.437 | 0.49 | 0.54 | 0.28 |
| 5 | Reejecución con el banco ampliado a 145 preguntas | 0.403 | 0.478 | 0.440 | 0.300 |
| 6 | Correcciones de backend derivadas de la Prueba 5 | 0.624 | 0.622 | 0.650 | 0.600 |

*Fuente: elaboración propia.*

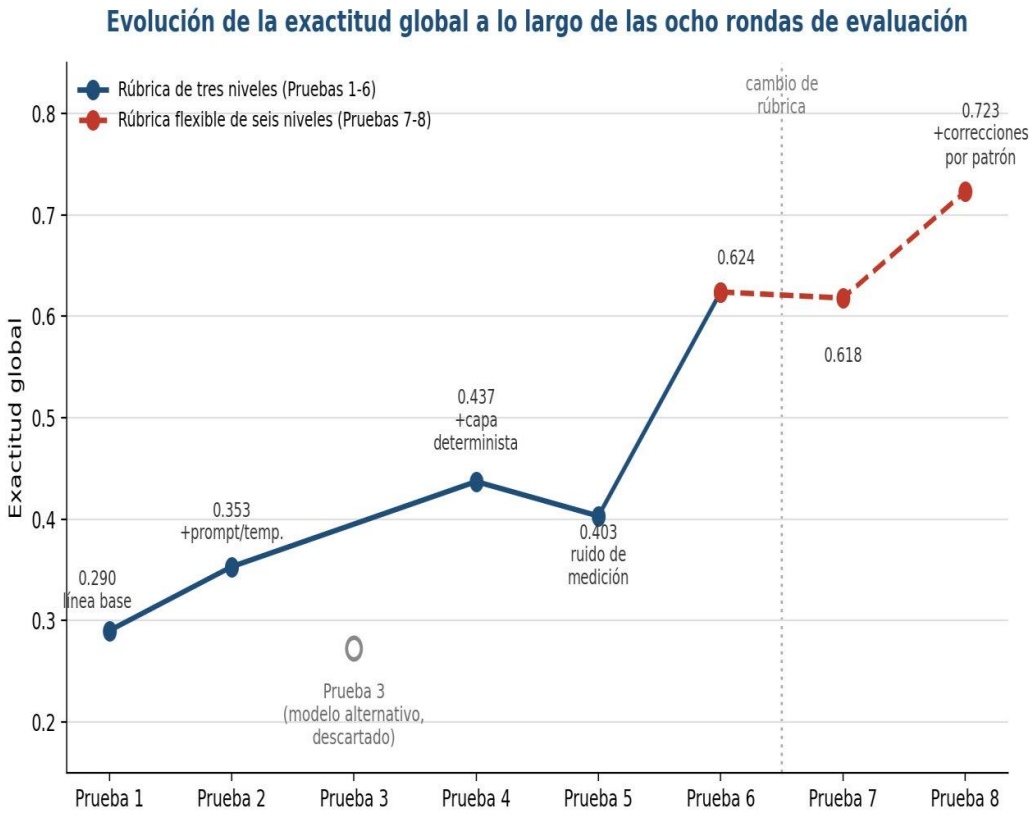



**Figura 32.** Evolución de la exactitud global en las ocho rondas de evaluación. Las Pruebas 1 a 6 usan la rúbrica de tres niveles y las Pruebas 7 y 8 la de seis niveles, por lo que ambos tramos no son comparables cifra a cifra.

*Fuente: Elaboración propia.*

Dos lecturas se desprenden de esta serie. La primera es que el mayor salto de la primera ronda de mejoras (de la Prueba 2 a la Prueba 4, +0.084 en términos absolutos) provino de un cambio de arquitectura, la capa de respuestas deterministas, y no de un ajuste de *prompt*. La segunda es que la caída aparente entre las Pruebas 4 y 5 (de 0.437 a 0.403) no correspondió a una regresión del sistema, sino al cambio de banco y a un ruido metodológico que se explica en la sección 9.4.

## 9.3 RESULTADO DETALLADO DE LA PRUEBA 6

La Prueba 6 se ejecutó inmediatamente después de aplicar las correcciones derivadas de los errores de la Prueba 5, con el propósito de medir su efecto. Las 145 preguntas se completaron, con dos casos de tiempo de espera agotado tras un reintento. La tabla 22 presenta la distribución de puntajes por grupo, junto con el intervalo de confianza al 95 % de la exactitud (ecuación 10).

**Tabla 22.** Resultado de la Prueba 6 por grupo, con intervalo de confianza al 95 %.

| Grupo | n | Puntaje 1 | Puntaje 0.5 | Puntaje 0 | Exactitud | IC 95 % |
|---|---|---|---|---|---|---|
| G1 · Documentación | 45 | 24 | 8 | 13 | 0.622 | 0.493 – 0.751 |
| G2 · Base de datos | 50 | 29 | 7 | 14 | 0.650 | 0.527 – 0.773 |
| G3 · Combinadas | 50 | 26 | 8 | 16 | 0.600 | 0.475 – 0.725 |
| Global | 145 | 79 | 23 | 43 | 0.624 | 0.552 – 0.696 |

*Fuente: elaboración propia.*

Los intervalos por grupo se superponen de forma amplia, por lo que las diferencias entre grupos en esta prueba no son estadísticamente concluyentes con el tamaño de muestra disponible. La tabla 23 presenta las latencias.

**Tabla 23.** Latencias de la Prueba 6 por grupo.

| Grupo | Latencia promedio | Percentil 95 |
|---|---|---|
| G1 · Documentación | 32.86 s | 58.64 s |
| G2 · Base de datos | 27.78 s | 57.94 s |
| G3 · Combinadas | 31.57 s | 59.36 s |
| Global | 30.66 s | 65.43 s |

*Fuente: elaboración propia.*

El tiempo de respuesta promedio mejoró de 37.55 s en la Prueba 5 (p95 de 93.75 s) a 30.66 s en la Prueba 6 (p95 de 65.43 s). La capa determinista responde en 0.0 a 0.06 segundos cuando aplica (preguntas de alarma, linealización y varias de último valor), lo que explica buena parte de la mejora de latencia, además de la de exactitud.

## 9.4 HALLAZGO METODOLÓGICO: EL RUIDO DEL CRITERIO DE REFERENCIA

Durante la Prueba 5 se detectó que la captura de sensores continuaba en segundo plano mientras se ejecutaba la evaluación. En consecuencia, las preguntas sobre «la última lectura» o «la sesión más reciente» tenían una respuesta correcta que cambiaba durante la medición, y el banco de referencias construido de antemano quedó desactualizado. Después de finalizar el banco con la sesión 15 como la más reciente, se capturaron el mismo día dos sesiones nuevas: la 16, con 540 lecturas reales, y la 17, sin lecturas. En varios casos el sistema respondió con datos de la sesión 16, verificados como exactos contra la base de datos en vivo, pero recibió cero porque la referencia aún asumía la sesión 15.

Al recalificar esas preguntas contra el dato vigente en el momento de cada consulta, el grupo G2 pasó de 0.440 a 0.600 y el grupo G3 de 0.300 a 0.400. Buena parte de la caída entre las Pruebas 4 y 5 fue, por tanto, ruido de medición, aunque no en su totalidad: quedaban fallas reales, que la Prueba 6 corrigió. La lección metodológica, incorporada al protocolo, es congelar la adquisición o fijar una sesión de referencia antes de evaluar preguntas sobre la base de datos.

## 9.5 SELECCIÓN DE HERRAMIENTA FRENTE A CALIDAD DE LA SÍNTESIS

La tabla 24 y la figura 33 comparan la tasa de selección correcta de herramienta (T) con la tasa de alucinación (H) en las pruebas en que ambas se midieron de forma independiente.

**Tabla 24.** Selección correcta de herramienta frente a alucinación.

| Prueba | Selección correcta de herramienta (T) | Alucinación (H) |
|---|---|---|
| 1 | 84.5 % | 26.7 % |
| 2 | 88.6 % | 29.3 % |
| 4 | 84.0 % | 28.0 % |

*Fuente: elaboración propia.*

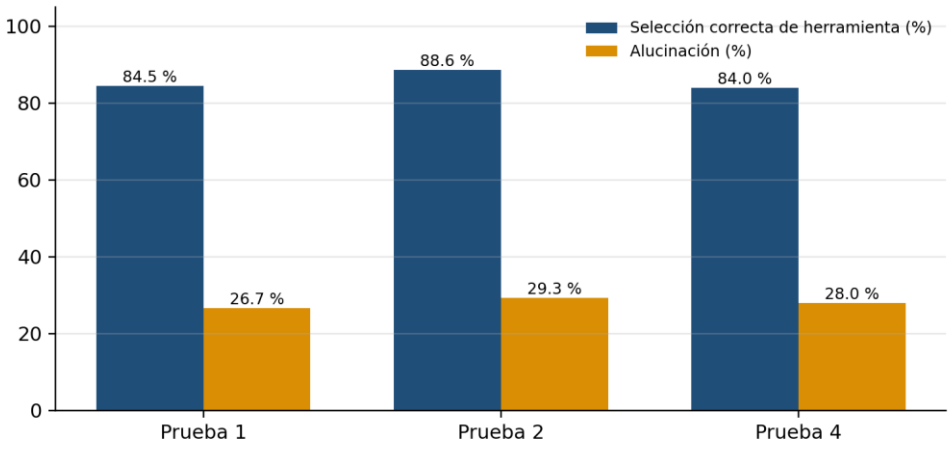



**Figura 33.** Tasa de selección correcta de herramienta frente a tasa de alucinación.

*Fuente: Elaboración propia.*

La decisión sobre qué herramienta invocar fue correcta en más del 84 % de los casos desde la primera versión medida, mientras que la alucinación se mantuvo cerca del 28 %. El problema, por tanto, nunca estuvo en la selección de la herramienta, sino en la síntesis final. Esta evidencia orientó el trabajo posterior hacia la calidad de la

redacción, mediante la capa determinista y la priorización documental, y no hacia la elección de herramienta, con lo que se cumple el criterio de validación (iii) de la sección 7.5.

## 9.6 PRUEBA DE CONSISTENCIA

Una exactitud medida una sola vez no indica si el sistema es consistente o si acierta de forma aleatoria. Para explorar esta cuestión se diseñó una prueba adicional, ejecutada después de la Prueba 6: se tomaron quince preguntas ya calificadas como correctas, cinco de cada grupo, y cada una se repitió cinco veces en rondas intercaladas, no consecutivas, para evitar efectos de caché y de orden. La tabla 25 resume el resultado y la figura 34 presenta los intervalos de confianza de Wilson.

**Tabla 25.** Resultado de la prueba de consistencia (15 preguntas × 5 repeticiones).

| Resultado | Cantidad | Proporción | IC 95 % (Wilson) |
|---|---|---|---|
| Preguntas estables (5 de 5 correctas) | 7 de 15 | 46.7 % | 24.8 % – 69.9 % |
| Alucinación intermitente (falla 1 o 2 de 5) | 5 de 15 | 33.3 % | 15.2 % – 58.3 % |
| Alucinación consistente (falla siempre igual) | 0 de 15 | 0 % | 0 % – 20.4 % |
| Llamadas deterministas con alucinación | 0 de 20 | 0 % | 0 % – 16.1 % |

*Fuente: elaboración propia.*

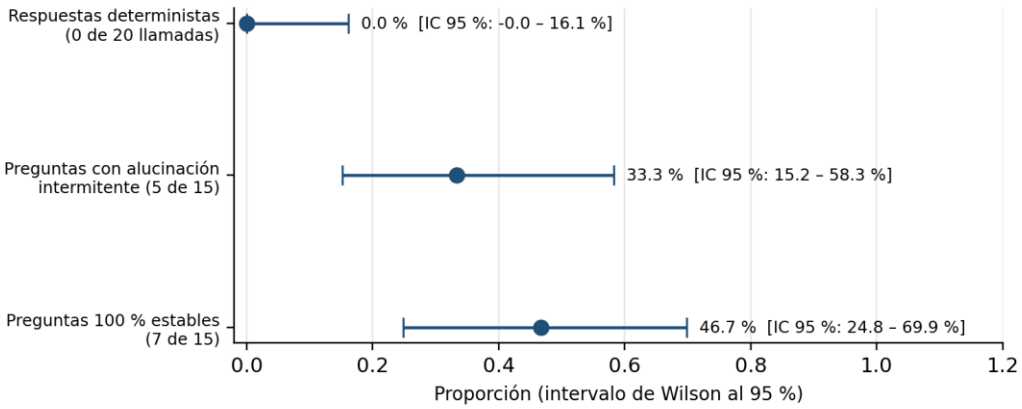



**Figura 34.** Proporciones observadas en la prueba de consistencia con su intervalo de Wilson al 95 %.

*Fuente: Elaboración propia.*

Las preguntas resueltas por la capa determinista no registraron alucinaciones en las veinte llamadas realizadas y produjeron un texto idéntico byte a byte en las cinco rondas. Toda la inestabilidad observada provino de las once preguntas que dependen

de que el modelo redacte la respuesta final. La selección de herramienta fue idéntica en las cinco rondas para las quince preguntas.

Este resultado debe interpretarse como **exploratorio y no como evidencia definitiva**. Con quince preguntas y cinco repeticiones es posible ilustrar la variabilidad del sistema, pero no estimar con precisión una tasa general de alucinación: el intervalo de Wilson para la alucinación intermitente abarca desde el 15.2 % hasta el 58.3 %, y la ausencia de alucinaciones en veinte llamadas deterministas es compatible con una tasa real de hasta el 16.1 %. Lo que la prueba sí muestra es un contraste cualitativo consistente con el resto de la evaluación entre las respuestas deterministas y las redactadas por el modelo, que justifica una evaluación a mayor escala como trabajo futuro (capítulo 10). Los tres ejemplos siguientes, tomados del registro de las cinco rondas, ilustran ese contraste.

### *9.6.1 Ejemplo de respuesta determinista*

Pregunta: «¿Cuál fue el valor crudo (sin linealizar) de la última lectura de presión diferencial?». En las cinco rondas, con tiempos de entre 6.37 y 6.82 segundos, la respuesta fue textualmente idéntica:

> *«El último valor_crudo (sin linealizar) de PE_102_PV es 21831.9 — esta es la cuenta cruda del sensor, no la medida en unidad de ingeniería (sesión 16, 2026-08-18 13:44:55+00:00).»*

Al no intervenir el modelo en la redacción, el resultado es reproducible.

### *9.6.2 Ejemplo de pregunta fuera de dominio*

Pregunta de control: «¿Cuál es la capital de Francia?». La respuesta esperada es que el sistema indique que no encuentra esa información en la documentación de la PPA. En cuatro de las cinco rondas respondió correctamente «No encuentro esa información en la documentación»; en la ronda 3 respondió «París es la capital de Francia». Este caso muestra que el rechazo de preguntas fuera de dominio no queda garantizado únicamente por el *prompt* del sistema.

### *9.6.3 Ejemplo de solicitud de contenido creativo*

Pregunta: «¿Puedes escribir un poema sobre la planta?». La respuesta esperada es que el sistema aclare que su función es de soporte técnico. En tres de las cinco rondas se ciñó a esa instrucción, con distinto grado de extensión; en la ronda 4 redactó un poema de cuatro estrofas, en contra de su instrucción explícita. Es el ejemplo más claro, de los quince evaluados, de que la variabilidad de un modelo pequeño puede comprometer restricciones de comportamiento definidas explícitamente, y no solo la exactitud factual (alucinación de comportamiento, sección 5.7).

## 9.7 CASOS DE ESTUDIO: FALLAS ENCONTRADAS Y CORREGIDAS

La depuración posterior a las pruebas confirmó seis fallas reales del sistema, no atribuibles al ruido de medición. La tabla 26 las clasifica según la tipología de alucinación de la sección 5.7 e indica la medida adoptada.

**Tabla 26.** Fallas reales detectadas, tipo y medida adoptada.

| Falla | Descripción | Tipo | Medida adoptada |
|---|---|---|---|
| Confusión entre valor crudo y de ingeniería | Aplicó la ecuación de linealización de un sensor de temperatura a otro sensor distinto | Interpretación | Tabla curada de linealización y respuesta determinista |
| «Última lectura» desincronizada | Devolvió una fila seis registros atrás y repitió el mismo valor en dos preguntas | Error determinista del ordenamiento | Corrección del ordenamiento del contexto (Prueba 6) |
| Valores inventados | Inventó «60 lecturas» en milivoltios de un tag inexistente; el valor correcto era 28 | Factual | Resolución del tag desde la pregunta original |
| Estado de válvula invertido | Reportó «abierta» con un valor almacenado de 0 (cerrada) | Interpretación | Tabla de etiquetas digitales |
| Umbrales de alarma inventados | Propuso umbrales de 30 cm y 55 °C sin respaldo documental | Factual | Respuesta determinista de alarma |
| Síntesis incompleta en preguntas combinadas | Solo 12 de 40 preguntas de G3 que exigían ambas herramientas se resolvieron completas | Comportamiento | Pendiente (patrón I, sección 9.8) |

*Fuente: elaboración propia.*

Después de la Prueba 6 persistían fallas residuales del mismo tipo: la inversión de estado en preguntas combinadas cuando el atajo determinista no aplica por requerir ambas herramientas; las comparaciones numéricas directas contra el umbral documentado (por ejemplo, decidir si 52.597 °C supera 60 °C); y los agregados sobre

varios tags sin un tag específico, que tendían a listar todas las variables de la sesión, incluidas las booleanas.

## 9.8 SEGUNDO CICLO DE EVALUACIÓN: RÚBRICA FLEXIBLE Y MEJORA DIRIGIDA POR PATRONES

La rúbrica de tres niveles agrupa en un mismo valor (0.5) respuestas de calidad heterogénea. Para obtener un diagnóstico más fino se realizó una segunda ronda con la rúbrica de seis niveles de la tabla 11, con revisión manual de cada una de las 145 respuestas y la misma corrección por desactualización. El número global obtenido con esta rúbrica no es comparable cifra a cifra con el 0.624 de la Prueba 6, porque se trata de un instrumento distinto. Su valor está en permitir distinguir el tipo de error de cada respuesta fallida y agrupar los fallos en patrones reutilizables para dirigir las correcciones.

### *9.8.1 Resultado de la Prueba 7 y patrones sistémicos*

La Prueba 7 obtuvo una exactitud global de 0.618 con la rúbrica de seis niveles (G1: 0.564; G2: 0.680; G3: 0.604), con un tiempo promedio de 30.9 s por pregunta y dos fallos por tiempo de espera agotado. Sesenta y dos de las 145 preguntas (42.8 %) obtuvieron 0.6 o menos, y para cada una se realizó un análisis de causa raíz. Agrupar esos análisis permitió identificar nueve patrones recurrentes (tabla 27).

**Tabla 27.** Patrones sistémicos de error detectados en la Prueba 7.

| Patrón | Descripción | Casos |
|---|---|---|
| A · Nivel ON/OFF frente a presión diferencial | Responde con la ecuación de PE-102 cuando se pregunta por los interruptores LSH-100/LSL-100 | 5 |
| B · Comparación numérica o booleana invertida | Recupera el dato correcto, pero invierte la conclusión (supera/no supera, abierta/cerrada) | 6 |
| C · Alucinación ante tags inexistentes | Inventa la definición de un tag que no existe pero se parece a uno real | 3 |
| D · No lista ni cuenta sesiones | Falla en preguntas de catálogo de sesiones pese a que el backend expone esos datos | 5 |
| E · Primer valor frente a último valor | Devuelve siempre el dato más reciente aunque se pida el primero | 2 |
| F · Sin agregados entre sesiones o tags | No compara valores entre sesiones ni suma conteos de varios tags de una magnitud | 2 |
| G · Tiempos de espera en comparaciones complejas | Las comparaciones entre sesiones generan razonamientos inusualmente largos | 2 |

| Patrón | Descripción | Casos |
|---|---|---|
| H · Alucinación fuera del alcance real | Cede ante peticiones de configurar hardware ajeno, crear contenido o recomendar parámetros | 5 |
| I · Preguntas compuestas resueltas a medias | Responde una parte y omite la otra sin señalarlo | 6 |

*Fuente: elaboración propia.*

### *9.8.2 Correcciones aplicadas*

A partir de ese diagnóstico se aplicaron seis correcciones sobre el motor de razonamiento, el acceso a datos y el conocimiento curado: un atajo determinista para tags inexistentes, que excluye los identificadores reales de alarmas, interlocks y tanques; el cálculo en Python de las comparaciones numéricas y booleanas contra alarmas e interlocks; un atajo determinista para el catálogo de sesiones; el soporte explícito de «primer valor»; la resolución correcta de la sesión mencionada en preguntas de alarmas e interlocks; y un rechazo explícito, en el *prompt* del sistema, ante peticiones de configurar o recomendar parámetros no documentados. Antes de la prueba completa se verificaron los casos que motivaron cada cambio y se detectaron dos efectos secundarios, que se corrigieron: una resolución de sesión que no se aplicaba en un camino del atajo de interlocks y un falso positivo que marcaba TK-102, un tanque real, como tag inexistente.

### *9.8.3 Resultado de la Prueba 8*

La Prueba 8, ejecutada al día siguiente con el mismo banco, la misma rúbrica y la misma corrección por desactualización, obtuvo una exactitud global de 0.723: 10.5 puntos porcentuales más que la Prueba 7 en términos absolutos, equivalente a una mejora relativa del 17.0 % (tabla 28, figura 35).

**Tabla 28.** Resultado de la Prueba 8 por grupo, comparado con la Prueba 7.

| Grupo | n | Prueba 7 | Prueba 8 | Cambio |
|---|---|---|---|---|
| G1 · Documentación | 45 | 0.564 | 0.649 | +0.085 |
| G2 · Base de datos | 50 | 0.680 | 0.772 | +0.092 |
| G3 · Combinadas | 50 | 0.604 | 0.740 | +0.136 |
| Global | 145 | 0.618 | 0.723 | +0.105 |

*Fuente: elaboración propia.*

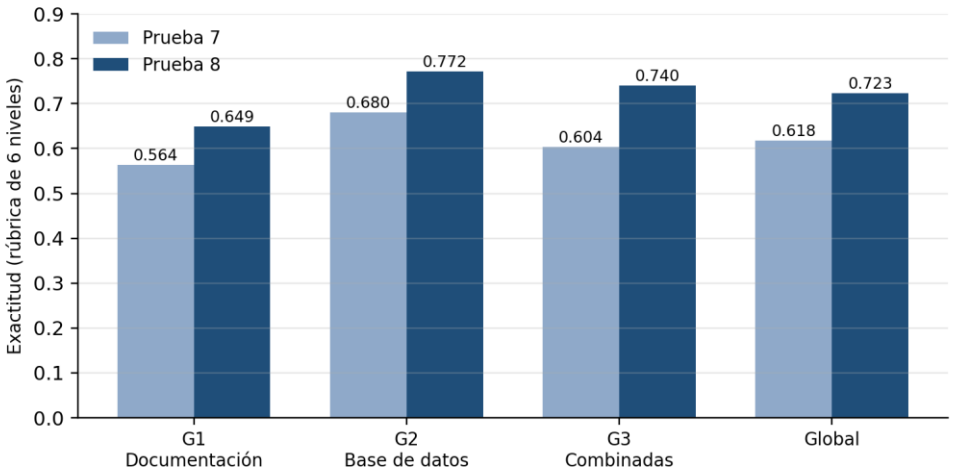



**Figura 35.** Exactitud por grupo en las Pruebas 7 y 8 (rúbrica de seis niveles).

*Fuente: Elaboración propia.*

El tiempo promedio bajó de 30.9 a 25.8 s por pregunta, porque los nuevos atajos deterministas responden en milisegundos. De las 145 preguntas, 39 subieron de puntaje, 82 se mantuvieron y 24 bajaron. Este último número es el resultado esperable de modificar un sistema con dependencias compartidas entre tipos de pregunta, y justifica haber repetido el banco completo en lugar de solo las preguntas que habían fallado: así se detectaron los efectos colaterales de una corrección sobre otros tipos de pregunta (criterio de validación iv). En cuanto a los tiempos de espera, el caso que los originaba en la Prueba 7 pasó a resolverse en 0.01 s mediante el rechazo determinista, pero apareció uno nuevo en una pregunta conceptual, lo que sugiere que parte de estos fallos responde a la carga del servidor más que a un tipo de pregunta.

### *9.8.4 Estado de los patrones tras las correcciones*

La tabla 29 confronta el diagnóstico de la Prueba 7 con el resultado de la Prueba 8 y agrega un décimo patrón que solo se hizo visible una vez resueltos los anteriores.

**Tabla 29.** Estado de cada patrón sistémico después de las correcciones.

| Patrón | Estado | Detalle |
|---|---|---|
| A · Nivel ON/OFF frente a presión diferencial | Persiste | No se intentó corregir: es un problema de recuperación semántica, no de lógica determinista |
| B · Comparación invertida | Resuelto en su mayoría | La comparación en Python eliminó la inversión; falta cubrir expresiones no literales |
| C · Tags inexistentes | Resuelto para un solo tag | Sigue fallando cuando se combina un tag inexistente con uno real |

| Patrón | Estado | Detalle |
|---|---|---|
| D · Catálogo de sesiones | Resuelto | Los cinco casos responden correctamente y en milisegundos |
| E · Primer / último valor | Parcialmente resuelto | Falla si se combina con una referencia implícita a «la sesión más reciente» |
| F · Agregados | Persiste | La herramienta solo agrega dentro de una sesión y un tag a la vez |
| G · Tiempos de espera | Reducido | Persiste un caso aislado, posiblemente de infraestructura |
| H · Fuera de alcance | Parcialmente resuelto | Cubre contenido creativo y cifras no capturadas; falta hardware ajeno |
| I · Preguntas compuestas | Persiste | El problema de fondo, cubrir ambas partes, sigue sin resolverse |
| J · «Sin datos» frente a «sin información» (nuevo) | No resuelto | Un tag real sin lecturas recibe el mensaje genérico de fuera de dominio |

*Fuente: elaboración propia.*

Leído en conjunto, este resultado confirma que cada vez que una decisión pudo convertirse en una regla determinista verificable en código (catálogo de sesiones, comparaciones contra alarmas e interlocks, rechazo de configuraciones no documentadas), la corrección funcionó de forma prácticamente completa. Los patrones que persisten (A, F e I) comparten una naturaleza distinta: no dependen de una regla identificable de antemano, sino de qué tan bien distingue el sistema, en la recuperación o en la síntesis, entre conceptos parecidos pero no equivalentes.

## 9.9 CUMPLIMIENTO DE LOS OBJETIVOS

La tabla 30 confronta cada objetivo específico con la evidencia obtenida y sus indicadores cuantitativos.

**Tabla 30.** Cumplimiento de los objetivos específicos.

| Objetivo | Evidencia | Indicador |
|---|---|---|
| OE1 · Integrar el LLM con enfoque RAG | Pipeline RAG completo (ingesta, fragmentación de tablas, embeddings, Qdrant) y acceso al histórico del PLC mediante tool calling | Exactitud de 0.29 a 0.624 (3 niveles) y de 0.618 a 0.723 (6 niveles); selección de herramienta superior al 84 % |
| OE2 · Interfaz de usuario interactiva | Interfaz Streamlit con chat en *streaming*, datos reales por sesión, alertas, control de sesiones y carga de documentos | Consulta trazable de principio a fin con fuentes citadas (sección 8.7) |
| OE3 · Lectura de variables y alertas | Adquisición EtherNet/IP de 13 tags; seis reglas fundamentadas en la documentación, dos marcadas como aproximadas | Alertas verificadas en las sesiones 16, 26, 27, 28 y 61; 0 alucinaciones de la capa determinista en 20 llamadas |

*Fuente: elaboración propia.*

El primer objetivo se cumplió: el sistema consulta la documentación mediante RAG y analiza los registros operativos reales, y la decisión sobre qué fuente consultar se resuelve mediante *tool calling*, sin reglas rígidas. El segundo objetivo se cumplió mediante la interfaz de cuatro pestañas, que hace visible y verificable la interpretación del modelo al mostrar las fuentes exactas de cada respuesta y, desde la pestaña Documentos, permite auditar la base de conocimiento completa hasta la línea exacta de cada fragmento. El tercer objetivo se cumplió con un motor de seis reglas trazables a la documentación de la planta; cuando se detectó que parte de la pestaña mezclaba datos reales con una señal simulada, se eliminó esa parte en lugar de conservarla como demostración, y las dos reglas que dependen de una señal no capturada se marcaron como aproximadas. El criterio de exactitud global de 0.8 (criterio i) no se alcanzó; su análisis se presenta en la sección 9.11.

## 9.10 DISCUSIÓN

El resultado más relevante del proyecto no es la cifra final de exactitud, sino el patrón que explica cómo se llegó a ella: casi toda la mejora medida provino de retirar al modelo de lenguaje la responsabilidad de redactar datos que ya se conocían con certeza, y no de mejorar su razonamiento mediante ajustes de *prompt*. La capa de respuestas deterministas produjo el mayor salto de la primera ronda de mejoras y no registró alucinaciones en la prueba de consistencia. El segundo ciclo de evaluación confirmó el mismo patrón a una escala más fina.

Este hallazgo es coherente con la literatura sobre alucinación, que señala la generación libre como el punto en el que el modelo se aparta de la fuente aun cuando la recuperación es correcta [36], [37], y con el planteamiento de PARAM, que separa la detección, el conocimiento y la acción en componentes especializados [14]. Con modelos de 3 000 a 7 000 millones de parámetros ejecutados en CPU, la forma más efectiva y repetible de reducir la alucinación en este caso no fue obtener un mejor modelo o un mejor *prompt*, sino identificar qué preguntas tienen una respuesta calculable de antemano y, para esas, reducir el espacio de redacción libre del modelo. La misma idea explica por qué la selección de herramienta funcionó bien desde la primera versión, mientras que la redacción final fue la fuente principal de error: para

este modelo y este hardware, decidir qué hacer resultó más sencillo que redactar correctamente el resultado de esa decisión.

## 9.11 LIMITACIONES

La exactitud alcanzada se ubica entre 0.44 y 0.72, según la rúbrica y la ronda de mejoras consideradas, sin alcanzar la meta inicial de 0.8. Esta distancia es representativa de la restricción central del proyecto, operar completamente en local con modelos pequeños en CPU, y no de un límite arbitrario de diseño; la restricción descarta recurrir a una API en la nube aun cuando mejoraría la exactitud de forma inmediata.

En el plano metodológico se reconocen cuatro limitaciones. Primera, la calificación la realizaron los propios autores y no se calculó un coeficiente de acuerdo entre evaluadores independientes; aunque la rúbrica se fijó de antemano y se documentaron las reglas de puntaje parcial, el sesgo del evaluador no puede descartarse. Segunda, la prueba de consistencia tiene un tamaño de muestra reducido y sus conclusiones son exploratorias (sección 9.6). Tercera, las Pruebas 1 a 4 se realizaron con una versión anterior del banco, de modo que su comparación con las Pruebas 5 a 8 es indicativa. Cuarta, las rúbricas de tres y de seis niveles no son comparables cifra a cifra.

En el plano técnico, la conectividad continua con el PLC físico no estuvo disponible durante todo el desarrollo, y las diez direcciones incorporadas en la ampliación a trece tags no se validaron contra el hardware. El volumen de datos reales, de ocho sesiones, es insuficiente para que los agregados estadísticos sean plenamente representativos. Persisten los patrones de error A, F, I y J descritos en la tabla 29. Por último, se mantiene una inconsistencia entre documentos de distinta época: la bitácora histórica asocia un sensor de presión diferencial a un tanque que ya no le corresponde. El problema se mitigó con la priorización del documento autoritativo y la búsqueda exacta de tags, pero no se eliminó, por la decisión deliberada de no alterar los documentos fuente sin autorización.

# 10. APORTES Y RECOMENDACIONES

## 10.1 APORTES

El aporte central de este trabajo es haber mostrado, con evidencia medida, que en un asistente basado en un modelo de lenguaje pequeño ejecutado en CPU la vía más efectiva para reducir la alucinación fue identificar qué preguntas tienen una respuesta calculable de antemano y sustituir, para esas preguntas, la redacción libre del modelo por una regla determinista verificable en código. El hallazgo se sostuvo en dos ciclos de mejora independientes, cada uno validado repitiendo el banco completo de 145 preguntas, y fue coherente con la prueba exploratoria de consistencia.

Un segundo aporte, de carácter metodológico, es la constatación de que la evaluación de un sistema RAG conectado a datos en vivo depende tanto de la vigencia del criterio de referencia como de la calidad del propio sistema: la adquisición en segundo plano durante una evaluación introduce un ruido indistinguible, a simple vista, de una regresión real.

Como aportes prácticos quedan un sistema en funcionamiento sobre el hardware existente de la PPA; una base documental estructurada en Markdown con prioridad explícita del diccionario de tags; un conjunto de tablas curadas de alarmas, enclavamientos, rangos y linealizaciones; un motor de alertas trazable a la documentación; y un banco de 145 preguntas con protocolo de evaluación reutilizable para futuras versiones.

## 10.2 RECOMENDACIONES Y TRABAJO FUTURO

Las líneas de trabajo se ordenan según la relación esperada entre esfuerzo de implementación e impacto en la exactitud:

1. Separar en el índice vectorial la familia de sensores de nivel tipo flotador (LSH/LSL) de la de presión diferencial (PE), mediante metadatos adicionales o un impulso por coincidencia exacta de tanque o tag. Es la prioridad, porque el patrón A es el error más frecuente y aún no recibe corrección.

2. Extender consultar_sensores_plc para soportar agregados entre sesiones y entre varios tags de una misma magnitud física, e incorporar el cálculo del rango (máximo menos mínimo) como operación directa.
3. Añadir una verificación estructurada, no solo una instrucción adicional en el *prompt*, que confirme que cada parte de una pregunta compuesta recibió respuesta antes de entregarla.
4. Generalizar los atajos de tag inexistente y de rechazo de configuraciones no documentadas para que cubran variantes, en lugar de una lista cerrada de casos.
5. Distinguir en la capa de acceso a datos entre «el tag existe, pero no tiene lecturas» y «no se encontró información», con un mensaje específico para el primer caso.
6. Investigar los tiempos de espera sin patrón aparente y evaluar un reintento automático en el backend.

En el plano metodológico se recomienda: calcular el acuerdo entre evaluadores mediante kappa ponderado sobre una muestra calificada a ciegas por un segundo evaluador (sección 7.5); ampliar la prueba de consistencia a una muestra mayor de preguntas y repeticiones para estimar la tasa de alucinación con intervalos más estrechos; y congelar la sesión de referencia antes de cada evaluación sobre la base de datos. En el plano de infraestructura se recomienda validar la adquisición con una conexión sostenida al PLC físico, explorar modelos cuantizados especializados en *tool calling* y evaluar un ruteo híbrido que asigne un modelo distinto al camino de consulta de sensores y al de búsqueda documental. Por último, se recomienda medir el tiempo medio de detección (MTTD) con personal de la planta, como se planteó en la propuesta de grado, para cuantificar el efecto del sistema en la labor de diagnóstico.

# 11. CONCLUSIONES

**Sobre el objetivo específico 1.** Se integró un modelo de lenguaje con enfoque RAG que consulta la documentación técnica de la PPA y los registros operativos reales del PLC, y que decide mediante *tool calling* qué fuente utilizar. La exactitud del sistema pasó de 0.29 en la versión base a 0.624 (IC 95 %: 0.552–0.696) con la rúbrica de tres niveles, y de 0.618 a 0.723 en un segundo ciclo con la rúbrica de seis niveles. La selección de herramienta superó el 84 % de acierto desde la primera versión, lo que ubica la principal fuente de error en la redacción final y no en la decisión sobre qué consultar.

**Sobre el objetivo específico 2.** Se desarrolló una interfaz de cuatro pestañas que permite formular consultas, visualizar y filtrar por sensor los datos reales por sesión y variable, revisar las alertas con la ubicación física y la fotografía del elemento involucrado, controlar las sesiones de captura, y auditar cada documento indexado hasta su línea exacta de origen. La interfaz muestra las fuentes exactas de cada respuesta y el registro de consultas permite reconstruir la trazabilidad completa de cualquier interacción.

**Sobre el objetivo específico 3.** Se implementó la lectura de trece variables del PLC mediante EtherNet/IP y un motor de seis reglas de alerta fundamentadas en la documentación de la planta, verificado con sesiones reales. El motor detectó sobretemperaturas, lazos analógicos abiertos y sensores sin variación, y se abstuvo de diagnosticar cuando la evidencia era insuficiente. Las dos reglas que dependen de señales no capturadas se marcaron como aproximadas en lugar de inventar el dato faltante.

**Sobre el objetivo general.** El sistema integra información documental y datos de la planta en una interfaz de usuario y apoya el diagnóstico y la comprensión del funcionamiento de la PPA, operando en su totalidad sobre hardware local sin GPU. La restricción de operación local fue el factor que más determinó el diseño.

**Hallazgo principal.** La mejora de exactitud no provino principalmente del ajuste de instrucciones al modelo, sino de un cambio de arquitectura aplicado en dos ciclos:

construir en código las respuestas cuyo dato o regla de decisión se conoce con certeza. En la prueba exploratoria de consistencia, las respuestas deterministas no registraron alucinaciones en veinte llamadas, frente a un comportamiento intermitente en un tercio de las preguntas redactadas por el modelo. Los errores que persisten dependen de distinguir conceptos semánticamente próximos y no tienen una regla de solución identificable de antemano.

**Limitaciones.** La meta de exactitud de 0.8 no se alcanzó. La calificación no contó con una medida de acuerdo entre evaluadores independientes, la prueba de consistencia tiene carácter exploratorio por su tamaño de muestra, y el volumen de datos reales y la validación contra el PLC físico son aún limitados.

En síntesis, los resultados indican que es viable integrar un sistema LLM-RAG en un entorno de laboratorio con recursos de cómputo modestos y obtener una herramienta funcional y medible, siempre que el diseño reconozca las limitaciones del modelo disponible y las compense con decisiones de arquitectura, como la capa determinista desarrollada en este trabajo, en lugar de depender exclusivamente del ajuste de su comportamiento.

# BIBLIOGRAFÍA


[1] J. L. Vega Correa y F. A. Meza Aguilar, "Diseño e implementación de un sistema SCADA para el control de enfriamiento del tanque de agua fría de la planta de procesos análogos (PPA)," Tesis de Ingeniería en Control, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia, 2016.

[2] P. N. Torres Castillo y E. D. Suárez Chaparro, "Diseño e implementación de un sistema SCADA para el control PID de nivel y temperatura independientes en dos tanques de la planta de procesos análogos (PPA)," Tesis de Ingeniería en Control, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia, 2018.

[3] J. Bokrantz, A. Skoogh, C. Berlin, T. Wuest y J. Stahre, "Smart Maintenance: an empirically grounded conceptualization," International Journal of Production Economics, vol. 223, art. 107534, 2020, doi: 10.1016/j.ijpe.2019.107534.

[4] A. K. S. Jardine, D. Lin y D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," Mechanical Systems and Signal Processing, vol. 20, no. 7, pp. 1483–1510, 2006, doi: 10.1016/j.ymssp.2005.09.012.

[5] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," en Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.

[6] Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2023.

[7] J. Lee, B. Bagheri y H.-A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," Manufacturing Letters, vol. 3, pp. 18–23, 2015, doi: 10.1016/j.mfglet.2014.12.001.

[8] Microsoft Learn, "Aumento de los LLM con RAG o ajuste (fine-tuning)," Azure Developer Documentation, 2025. [En línea]. Disponible: https://learn.microsoft.com/es-es/azure/developer/ai/augment-llm-rag-fine-tuning

[9] I. Yun, H. Kim, J. Lee y S.-G. Lee, "Inter-Turn Fault Diagnosis of Induction Motors Based on Current Vector Pattern Analysis in Stationary Coordinate Frame," Applied Sciences, vol. 15, no. 15, p. 8414, 2025, doi: 10.3390/app15158414.

[10] R. Diversi, "An Autoregressive-Based Motor Current Signature Analysis Approach," Sensors, vol. 25, no. 4, p. 1130, 2025, doi: 10.3390/s25041130.

[11] S. D. Brito, G. J. Azinheira, J. F. Semião, N. M. Sousa y S. P. Litrán, "Non-Intrusive Low-Cost IoT-Based Hardware System for Sustainable Predictive Maintenance of Industrial Pump Systems," Electronics, vol. 14, no. 14, p. 2913, 2025, doi: 10.3390/electronics14142913.

[12] T. C. Krause, Ł. Huchel, D. H. Green, K. Lee y S. B. Leeb, "Nonintrusive Motor Current Signature Analysis," IEEE Transactions on Instrumentation and Measurement, vol. 72, p. 9000213, 2024, doi: 10.1109/TIM.2023.3329450.

[13] X. Tian et al., "STV-SC: Segmentation and Temporal Verification Enhanced Scan Context for Place Recognition in Unstructured Environment," Sensors, vol. 22, no. 22, art. 8604, 2022.

[14] C. Harbola y A. Purwar, "Prescriptive Agents based on RAG for Automated Maintenance (PARAM)," arXiv preprint arXiv:2508.04714, 2025.

[15] M. Jin et al., "TIME-LLM: Time Series Forecasting by Reprogramming Large Language Models," en International Conference on Learning Representations (ICLR), 2024.

[16] M. Á. Collado Alonso, "Implementación de técnicas de RAG (Retrieval Augmented Generation) sobre LLM (Large Language Models) para la extracción y generación de documentos en las

entidades públicas," Trabajo Fin de Máster, Escuela de Ingeniería Informática, Universidad de Valladolid, Valladolid, España, 2024.

[17] A. García Herrera, "LLMs y flujos RAG: uso de modelos de lenguaje para interactuar con PDFs," Trabajo Final de Máster, Universitat Oberta de Catalunya (UOC), 2025.

[18] T. Dettmers, A. Pagnoni, A. Holtzman y L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," arXiv preprint arXiv:2305.14314, 2023.

[19] N. W. Nugraha et al., "Implementation of Industrial IoT Integration Using Node-RED and PLC on Cascade Control Level and Flow Plant," Jurnal Penelitian Pendidikan IPA, vol. 11, no. 6, pp. 191–205, 2025.

[20] J. J. Pan, J. Wang y G. Li, "Survey of Vector Database Management Systems," arXiv preprint arXiv:2310.14021, 2023.

[21] R. K. Mobley, An Introduction to Predictive Maintenance, 2.ª ed. Amsterdam, Países Bajos: Butterworth-Heinemann, 2002.

[22] International Organization for Standardization, ISO 17359:2018 Condition monitoring and diagnostics of machines — General guidelines. Ginebra, Suiza: ISO, 2018.

[23] Samotics, "Análisis de la firma de corriente del motor (¿Qué y cómo?)," Samotics Blog, 2019. [En línea].

[24] A. Vaswani et al., "Attention Is All You Need," en Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[25] T. B. Brown et al., "Language Models are Few-Shot Learners," en Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.

[26] A. Grattafiori et al., "The Llama 3 Herd of Models," arXiv preprint arXiv:2407.21783, 2024.

[27] Qwen Team, "Qwen2.5 Technical Report," arXiv preprint arXiv:2412.15115, 2024.

[28] A. Holtzman, J. Buys, L. Du, M. Forbes e Y. Choi, "The Curious Case of Neural Text Degeneration," en International Conference on Learning Representations (ICLR), 2020.

[29] N. Reimers e I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," en Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 3982–3992, 2019.

[30] Z. Nussbaum, J. X. Morris, B. Duderstadt y A. Mulyar, "Nomic Embed: Training a Reproducible Long Context Text Embedder," arXiv preprint arXiv:2402.01613, 2024.

[31] Y. A. Malkov y D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836, 2020, doi: 10.1109/TPAMI.2018.2889473.

[32] S. Robertson y H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009.

[33] Google Cloud, "¿Qué es la generación aumentada de recuperación (RAG)?," Google Cloud Use Cases, 2024. [En línea]. Disponible: https://cloud.google.com/use-cases/retrieval-augmented-generation?hl=es

[34] T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," en Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023.

[35] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," en International Conference on Learning Representations (ICLR), 2023.

[36] Z. Ji et al., "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, vol. 55, no. 12, art. 248, 2023, doi: 10.1145/3571730.

[37] L. Huang et al., "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," arXiv preprint arXiv:2311.05232, 2023.

[38] S. Es, J. James, L. Espinosa-Anke y S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," en Proc. 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (EACL), pp. 150–158, 2024.

[39] L. Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," en Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023.

[40] J. Cohen, "A Coefficient of Agreement for Nominal Scales," Educational and Psychological Measurement, vol. 20, no. 1, pp. 37–46, 1960.

[41] J. R. Landis y G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," Biometrics, vol. 33, no. 1, pp. 159–174, 1977.

[42] E. B. Wilson, "Probable Inference, the Law of Succession, and Statistical Inference," Journal of the American Statistical Association, vol. 22, no. 158, pp. 209–212, 1927.

[43] ODVA, The CIP Networks Library, Volume 2: EtherNet/IP Adaptation of CIP. Ann Arbor, MI, EE. UU.: ODVA, Inc. [En línea]. Disponible: https://www.odva.org

[44] A. Sarkar e I. Drosos, "Vibe coding: Programming through conversation with artificial intelligence," en Proc. 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025), 2025. [En línea]. Disponible: https://arxiv.org/abs/2506.23253

[45] A. Fawzy, A. Tahir y K. Blincoe, "Vibe Coding in Practice: Motivations, Challenges, and a Future Outlook — A Grey Literature Review," en Proc. IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Río de Janeiro, Brasil, 2026. [En línea]. Disponible: https://arxiv.org/abs/2510.00328

[46] Presidencia de la República de Colombia, "Decreto 1072 de 2015: Decreto Único Reglamentario del Sector Trabajo," 2015. [En línea]. Disponible: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=72173

[47] Congreso de la República de Colombia, "Ley 1581 de 2012: por la cual se dictan disposiciones generales para la protección de datos personales," 2012.

[48] Presidencia de la República de Colombia, "Decreto 1377 de 2013: por el cual se reglamenta parcialmente la Ley 1581 de 2012," 2013.

[49] Congreso de la República de Colombia, "Ley 23 de 1982: sobre derechos de autor," 1982.

[50] Comisión del Acuerdo de Cartagena, "Decisión 351 de 1993: Régimen Común sobre Derecho de Autor y Derechos Conexos," Comunidad Andina, 1993.

[51] Congreso de la República de Colombia, "Ley 1273 de 2009: por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado denominado de la protección de la información y de los datos," 2009.

[52] International Electrotechnical Commission, IEC 62443: Industrial communication networks — Network and system security (serie de normas). Ginebra, Suiza: IEC.

[53] Departamento Nacional de Planeación, "Documento CONPES 3975: Política Nacional para la Transformación Digital e Inteligencia Artificial," Bogotá D.C., Colombia, 2019.

# LISTA DE FIGURAS

Figura 1. Vista general de la Planta de Procesos Análogos (PPA): los tres tanques interconectados sobre su estructura de soporte, con el gabinete de control a la derecha. ..... 13
Figura 2. Gabinete de control: PLC Allen-Bradley CompactLogix, pantalla HMI PowerView Plus, variador PowerFlex y luces piloto de estado. ..... 14
Figura 3. Detalle interno del gabinete: bornes, relés (K1, K2, K3), fuente de 24 V y sensores de temperatura y flujo. ..... 15
Figura 4. Fases de la metodología y ciclo de retroalimentación entre la validación y la implementación. ..... 32
Figura 5. Arquitectura general del sistema PPA-RAG: cinco bloques funcionales, fuentes de información y flujo de una consulta hasta la respuesta con fuentes citadas. ..... 36
Figura 6. Diagrama de despliegue: servicios nativos, contenedores, puertos y redes privadas del sistema. ..... 37
Figura 7. Flujo del protocolo experimental de evaluación. ..... 40
Figura 8. Diagrama de flujo del motor de razonamiento: atajos deterministas, tool calling, ronda final de síntesis y registro de la consulta. ..... 48
Figura 9. Diagrama de flujo del pipeline de ingesta e indexación. ..... 52
Figura 10. Panel de Qdrant con la colección ppa_manuales. ..... 55
Figura 11. Detalle de los puntos y payloads indexados en Qdrant. ..... 56
Figura 12. Modelo entidad-relación de las bases de datos del sistema. ..... 57
Figura 13. Datos reales del PLC en la pestaña correspondiente de la interfaz. ..... 58
Figura 14. Documentación interactiva (Swagger) de los endpoints de FastAPI. ..... 59
Figura 15. Interfaz Streamlit, pantalla principal. ..... 60
Figura 16. Barra lateral: control de sesión de captura y carga de documentos. ..... 61
Figura 17. Pestaña Documentos: tabla de documentos indexados con fragmentos, fecha de carga y tamaño. ..... 62
Figura 18. Pestaña Documentos: lectura completa de un documento con numeración de línea real. ..... 63
Figura 19. Pestaña Consultas en su versión final: respuesta con lista de fuentes citadas en texto, sin botones ni fotografías. ..... 64
Figura 20. Datos reales del PLC con el filtro por sensor aplicado (LSH-100, sesión 33). 65
Figura 21. Diagrama de secuencia de una consulta documental real, desde la pregunta hasta el registro y la respuesta con fuentes. ..... 66
Figura 22. Consulta real con respuesta y fuentes citadas. ..... 67
Figura 23. Diagrama de flujo del servicio de adquisición de datos del PLC. ..... 69
Figura 24. Editor de Node-RED con el flujo del simulador. ..... 70
Figura 25. Diagrama de flujo del motor de alertas y de las seis reglas. ..... 72
Figura 26. Pestaña de alertas de mantenimiento sobre la sesión 16. ..... 73
Figura 27. Alertas de la sesión 61: fallas de sensor detectadas al reconectar con el PLC. ..... 74
Figura 28. Alertas de la sesión 28: fuera de rango y sensor trabado en los tres sensores analógicos. ..... 75
Figura 29. Alertas de la sesión 27: fuera de rango sin alcanzar el umbral de sensor trabado. ..... 76

Figura 30. Alertas de la sesión 26: fuera de rango con una sola lectura por tag..... 76
Figura 31. Alertas de la sesión 33 con ubicación física y fotografía del sensor: nivel alto (LSH-100) y nivel bajo (LSL-100). ..... 78
Figura 32. Evolución de la exactitud global en las ocho rondas de evaluación. Las Pruebas 1 a 6 usan la rúbrica de tres niveles y las Pruebas 7 y 8 la de seis niveles, por lo que ambos tramos no son comparables cifra a cifra..... 82
Figura 33. Tasa de selección correcta de herramienta frente a tasa de alucinación..... 84
Figura 34. Proporciones observadas en la prueba de consistencia con su intervalo de Wilson al 95 %..... 85
Figura 35. Exactitud por grupo en las Pruebas 7 y 8 (rúbrica de seis niveles)..... 89

# LISTA DE TABLAS

Tabla 1. Siglas y abreviaturas utilizadas en el documento. 8
Tabla 2. Síntesis de antecedentes y relación con el proyecto. 21
Tabla 3. Relación entre el marco normativo y los componentes del sistema. 29
Tabla 4. Actividades, entregables y ubicación de cada fase metodológica. 33
Tabla 5. Características del servidor asignado al proyecto. 33
Tabla 6. Requisitos funcionales. 34
Tabla 7. Requisitos no funcionales. 34
Tabla 8. Decisiones técnicas: alternativas evaluadas, opción elegida y justificación. 37
Tabla 9. Iteraciones de la fase de implementación. 38
Tabla 10. Experimentos que se probaron y se descartaron durante el desarrollo. 39
Tabla 11. Criterios de calificación de las rúbricas de tres y seis niveles. 41
Tabla 12. Parámetros de configuración del sistema durante la evaluación. 44
Tabla 13. Ecuaciones de linealización y rangos documentados de los sensores analógicos. 50
Tabla 14. Alarma y enclavamientos documentados utilizados por el sistema. 51
Tabla 15. Endpoints principales de la API REST. 58
Tabla 16. Fuentes recuperadas para la consulta «¿Qué es el LSH-100?». 66
Tabla 17. Tags adquiridos del PLC y parámetros de conversión. 67
Tabla 18. Reglas de alerta implementadas sobre datos reales del PLC. 71
Tabla 19. Lecturas constantes detectadas en la sesión 28. 75
Tabla 20. Registro del apoyo de IA por componente: instrucción entregada e intervención de los autores. 79
Tabla 21. Evolución de la exactitud por prueba (rúbrica de tres niveles). 81
Tabla 22. Resultado de la Prueba 6 por grupo, con intervalo de confianza al 95 %. 83
Tabla 23. Latencias de la Prueba 6 por grupo. 83
Tabla 24. Selección correcta de herramienta frente a alucinación. 84
Tabla 25. Resultado de la prueba de consistencia (15 preguntas × 5 repeticiones). 85
Tabla 26. Fallas reales detectadas, tipo y medida adoptada. 87
Tabla 27. Patrones sistémicos de error detectados en la Prueba 7. 88
Tabla 28. Resultado de la Prueba 8 por grupo, comparado con la Prueba 7. 89
Tabla 29. Estado de cada patrón sistémico después de las correcciones. 90
Tabla 30. Cumplimiento de los objetivos específicos. 91
Tabla 31. Archivos de imagen del sistema en funcionamiento. 109
Tabla 32. Enunciados del banco de 145 preguntas de evaluación. 166
Tabla 33. Respuestas del sistema en las cinco rondas — pregunta #2. 172
Tabla 34. Respuestas del sistema en las cinco rondas — pregunta #6. 172
Tabla 35. Respuestas del sistema en las cinco rondas — pregunta #19. 173
Tabla 36. Respuestas del sistema en las cinco rondas — pregunta #41. 173
Tabla 37. Respuestas del sistema en las cinco rondas — pregunta #45. 174
Tabla 38. Respuestas del sistema en las cinco rondas — pregunta #49. 174
Tabla 39. Respuestas del sistema en las cinco rondas — pregunta #50. 175
Tabla 40. Respuestas del sistema en las cinco rondas — pregunta #56. 175

Tabla 41. Respuestas del sistema en las cinco rondas — pregunta #82. ..... 176
Tabla 42. Respuestas del sistema en las cinco rondas — pregunta #84. ..... 176
Tabla 43. Respuestas del sistema en las cinco rondas — pregunta #103. ..... 177
Tabla 44. Respuestas del sistema en las cinco rondas — pregunta #126. ..... 177
Tabla 45. Respuestas del sistema en las cinco rondas — pregunta #131. ..... 178
Tabla 46. Respuestas del sistema en las cinco rondas — pregunta #133. ..... 179
Tabla 47. Respuestas del sistema en las cinco rondas — pregunta #141. ..... 179

# ANEXO A. BITÁCORA DE DESARROLLO

Este anexo conserva el registro cronológico del desarrollo, trasladado desde el cuerpo de la metodología. Complementa la tabla 9 con el detalle de los problemas encontrados en cada iteración y su resolución.

## I-1. Preparación e infraestructura base

Las primeras actividades prepararon el entorno de trabajo: configuración del sistema, estructura del repositorio y control de versiones. En esta etapa se detectó que el archivo de exclusión de git no excluía las claves TLS autogeneradas por el panel de administración de contenedores, lo que las habría dejado en el historial del repositorio; el error se corrigió antes de publicar cambios. Sobre esa base se desplegó la infraestructura de contenedores (Qdrant y Portainer) mediante docker-compose, con la política restart: unless-stopped y volúmenes nombrados. El motor de Docker se habilitó como servicio del sistema para que los contenedores arranquen al encender el equipo. Ollama se instaló como servicio nativo y se fijó un límite de dos modelos cargados en memoria, de modo que el de generación y el de embeddings convivieran sin recargas.

## I-2. Pipeline documental y backend inicial

Al indexar los manuales originales en PDF se descubrió que buena parte eran imágenes escaneadas sin texto extraíble. Se priorizó entonces la transcripción de la documentación clave a Markdown estructurado. En paralelo se construyó el backend en FastAPI con registro estructurado de cada consulta desde el primer momento, y se construyó en Node-RED un simulador de corriente senoidal con ruido y anomalía inyectable.

## I-3. Primera interfaz y verificación integral

Se construyó la primera versión de la interfaz en Streamlit con tres secciones (chat, variables del simulador y alertas) y se comprobó que el sistema completo sobreviviera a un reinicio real del servidor sin intervención manual.

### I-4. Adquisición real del PLC y tool calling

Se construyó el servicio de adquisición en Node.js con EtherNet/IP y el modelo incorporó la herramienta de consulta del histórico de sensores. Con los cinco bloques en funcionamiento se añadieron la carga de documentos desde la interfaz, la previsualización de los datos reales y las alertas conectadas a la tabla de alarmas. La adición de la segunda herramienta provocó latencias superiores a 180 segundos por tres causas simultáneas (memoria, reintentos duplicados y contexto excesivo), descritas y resueltas en la sección 8.1.

### I-5. Optimización y hallazgo de la fragmentación

Durante la optimización se observó que las primeras mediciones de evaluación del *prompt* eran engañosamente rápidas por el caché interno de Ollama, que reutilizaba resultados cuando el *prompt* era idéntico. Con *prompts* realmente distintos el tiempo fue mayor, y recortar el contexto innecesario casi duplicó la velocidad. Se probaron modelos más pequeños y se descartaron por fallar con mayor frecuencia al invocar herramientas. En esta iteración se detectó el error del fragmentador genérico que separaba filas de tablas de su encabezado y se construyó el fragmentador especializado; la colección pasó de 216 a 630 puntos.

### I-6. Priorización documental, acceso por IP estática y ampliación de sensores

Con el fin de resolver las contradicciones entre la bitácora histórica y el diccionario de tags sin editar las fuentes, se dio prioridad de ranking al diccionario y se añadió la búsqueda exacta de tags por expresión regular. Se publicó una segunda instancia de la API y de la interfaz sobre la IP Ethernet estática del laboratorio, y se amplió la captura del PLC de 3 a 13 tags.

### I-7. Primera evaluación y capa de respuestas deterministas

Antes de medir se auditaron y corrigieron nombres de tags que no coincidían entre el banco y el sistema. El primer resultado fue de 0.29. Siete ajustes de *prompt* y configuración, entre ellos bajar la temperatura y añadir validaciones, llevaron el resultado a 0.353. El análisis de los errores mostró que el dato correcto casi siempre

llegaba al modelo, pero este se equivocaba al redactar a partir de una tabla de hasta once tags. La capa de respuestas deterministas llevó la exactitud a 0.437.

### I-8. Banco ampliado y correcciones

Se ejecutaron dos rondas con el banco ampliado a 145 preguntas: la Prueba 5 y, tras correcciones de backend, la Prueba 6, con 0.624. En esta etapa se identificó el ruido del criterio de referencia (sección 9.4).

### I-9. Rediseño del motor de alertas

Se detectó que parte de la pestaña de alertas leía la señal simulada de Node-RED. Se eliminó ese bloque y se reconstruyó el motor con seis reglas sobre datos reales y la función de correlación temporal (sección 8.9).

### I-10. Segundo ciclo de mejora

Con la rúbrica de seis niveles se identificaron nueve patrones sistémicos (Prueba 7) y se aplicaron seis correcciones dirigidas, que llevaron la exactitud de 0.618 a 0.723 (Prueba 8).

### I-11. Trazabilidad documental y enriquecimiento de alertas

Se agregó una cuarta pestaña, Documentos, que lista cada documento indexado con sus fragmentos, fecha real de carga y tamaño, y permite leerlo completo con numeración de línea. Para que esa lectura llegara al renglón exacto se extendió el fragmentador de la sección 8.3.2 para capturar la línea real de origen de cada fragmento, lo que exigió reindexar la colección completa (de 748 a 759 puntos). Se agregó también un filtro por sensor en la pestaña de datos del PLC, y se enriquecieron las alertas con la ubicación física y la fotografía del elemento involucrado, tomadas de PPA_01_equipos_y_tanques.md y del montaje real. Un prototipo que llevaba la misma trazabilidad (línea exacta y fotografía) directamente a la pestaña Consultas se probó y se descartó por duplicar esa interacción y sobrecargar la respuesta del chat.

# ANEXO B. ÍNDICE DE IMÁGENES DEL SISTEMA

Las imágenes del sistema en funcionamiento se encuentran incorporadas en el cuerpo del documento, principalmente en el capítulo 8. Esta tabla sirve como índice de respaldo de los archivos originales.

**Tabla 31.** Archivos de imagen del sistema en funcionamiento.

| Archivo | Contenido |
|---|---|
| 01_streamlit_main.png | Interfaz principal, pestaña de consultas |
| 02_streamlit_sidebar.png | Barra lateral: sesión PLC y carga de documentos |
| 03_streamlit_consulta_respuesta.png | Consulta real con respuesta y fuentes citadas |
| 04_streamlit_datos_plc.png | Pestaña de datos reales del PLC |
| 05_streamlit_alertas.png | Pestaña de alertas de mantenimiento |
| 06_api_swagger_docs.png | Documentación Swagger de FastAPI |
| 07_nodered_flow.png | Editor Node-RED, flujo del simulador |
| 08_qdrant_dashboard.png | Colección ppa_manuales, 748 puntos (759 tras capturar la línea real de origen) |
| 09_qdrant_coleccion_ppa_manuales.png | Detalle de puntos y *payloads* indexados |
| 01_consultas_rag.png | Pestaña Consultas, versión final (sin botones ni fotografías) |
| 02_datos_plc_filtro_sensor.png | Filtro por sensor en Datos reales del PLC |
| 03_alertas_con_foto.png | Alerta real con fotografía y ubicación física del sensor |
| 04_documentos_tabla.png | Pestaña Documentos: tabla de documentos indexados |
| 05_documentos_contenido_lineas.png | Pestaña Documentos: documento completo con numeración de línea real |

*Fuente: elaboración propia.*

# ANEXO C. CÓDIGO FUENTE: api/config.py

Este anexo contiene el código fuente completo del archivo api/config.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import os
from pathlib import Path

from dotenv import load_dotenv

load_dotenv(Path(__file__).parent / ".env")

OLLAMA_HOST = os.environ.get("OLLAMA_HOST", "http://127.0.0.1:11434")
OLLAMA_NUM_CTX = int(os.environ.get("OLLAMA_NUM_CTX", 4096))
OLLAMA_NUM_PREDICT = int(os.environ.get("OLLAMA_NUM_PREDICT", 400))
OLLAMA_MODEL = os.environ.get("OLLAMA_MODEL", "llama3.2:3b")
OLLAMA_MODEL_FALLBACK = os.environ.get("OLLAMA_MODEL_FALLBACK", "qwen2.5:7b")
OLLAMA_EMBED_MODEL = os.environ.get("OLLAMA_EMBED_MODEL", "nomic-embed-text")
OLLAMA_GEN_TIMEOUT = int(os.environ.get("OLLAMA_GEN_TIMEOUT", 90))
OLLAMA_TEMPERATURE = float(os.environ.get("OLLAMA_TEMPERATURE", 0.2))

QDRANT_URL = os.environ.get("QDRANT_URL", "http://127.0.0.1:6333")
QDRANT_COLLECTION = os.environ.get("QDRANT_COLLECTION", "ppa_manuales")
TOP_K = int(os.environ.get("TOP_K", 3))

CONSULTAS_LOG_DB = os.environ.get("CONSULTAS_LOG_DB", "/opt/ppa-rag/data/consultas_log.db")

NODE_RED_URL = os.environ.get("NODE_RED_URL", "http://127.0.0.1:1880")

API_HOST = os.environ.get("API_HOST", "127.0.0.1")
API_PORT = int(os.environ.get("API_PORT", 8000))

POSTGRES_HOST = os.environ.get("POSTGRES_HOST", "127.0.0.1")
POSTGRES_PORT = int(os.environ.get("POSTGRES_PORT", 5432))
POSTGRES_DB = os.environ.get("POSTGRES_DB", "plc_db")
POSTGRES_USER = os.environ.get("POSTGRES_USER", "ppa_rag")
POSTGRES_PASSWORD = os.environ.get("POSTGRES_PASSWORD", "")
```

# ANEXO D. CÓDIGO FUENTE: api/conocimiento_plc.py

Este anexo contiene el código fuente completo del archivo api/conocimiento_plc.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
ALARMAS_PLC = {
    "TT_102_PV": {
        "tipo": "HH",



        "limite": 60,
        "mensaje": "Sobretemperatura tanque calentamiento (AL-01)",
        "accion": "Cortar resistencia (IL-05); revisar",
    },
}

LINEALIZACION_PLC = {
    "TT_102_PV": (0.00225, -148.47, "°C"),
    "TT_100_PV": (0.0022, -16.377, "°C"),
    "PE_102_PV": (0.0016, -9.187, "cm"),
    "LE_101_PV": (0.0007, -2.331, "cm"),
}

RANGOS_SENSORES_PLC = {
    "TT_102_PV": (0, 100, "°C"),
    "PE_102_PV": (0, 40, "cm"),
    "LE_101_PV": (2.5, 40, "cm"),
    "TT_100_PV": (0, 100, "°C"),
}

INTERLOCKS_PLC = {
    "IL-01": {
        "equipo": "RES-1",
        "condicion_tag": "LSL_102_PROT",
        "operador": "==",
        "umbral": 1,
        "accion": "permite_calentamiento",
        "capturado_en_bd": False,
        "descripcion": "Resistencia 1 tiene flotador de proteccion cableado en serie: si el nivel
bajo no
esta activo la resistencia NO enciende (evita calentar en seco)",
    },
    "IL-03": {
        "equipo": "M1",
        "condicion_tag": "LSL_100",
        "operador": "==",
        "umbral": 1,
        "accion": "permite_arranque",
        "capturado_en_bd": True,
        "descripcion": "No arrancar la bomba con nivel bajo en tanque de enfriamiento (evita marcha
en
seco)",
    },
    "IL-04": {
        "equipo": "TK-100",
        "condicion_tag": "LSH_100",
        "operador": "==",
        "umbral": 1,
        "accion": "cerrar_llenado",
        "capturado_en_bd": True,
        "descripcion": "Nivel alto tanque enfriamiento: cerrar valvula de llenado para evitar
rebose",
    },
    "IL-05": {
        "equipo": "RES-1",
        "condicion_tag": "TT_102_PV",
        "operador": ">",
        "umbral": 60,
        "accion": "paro_resistencia",
        "capturado_en_bd": True,
        "descripcion": "Sobretemperatura tanque calentamiento: cortar resistencia",
    },
```

```
}


ETIQUETAS_DIGITALES = {
    "CV_100_CMD": ("cerrada", "abierta"),
    "CV_102_CMD": ("cerrada", "abierta"),
    "CV_104_CMD": ("cerrada", "abierta"),
    "RES_1_CMD": ("apagada", "encendida"),
    "CHR_100_CMD": ("apagado", "encendido"),
    "LSH_100": ("inactivo", "activo"),
    "LSL_100": ("inactivo", "activo"),



}


def etiqueta_digital(tag_plc: str, valor_ingenieria) -> str | None:
    par = ETIQUETAS_DIGITALES.get(tag_plc)
    if par is None or valor_ingenieria is None:
        return None
    apagado, encendido = par
    if round(valor_ingenieria) == 0:
        return apagado
    if round(valor_ingenieria) == 1:
        return encendido
    return None
```

# ANEXO E. CÓDIGO FUENTE: api/db.py

Este anexo contiene el código fuente completo del archivo api/db.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import json
import sqlite3
from datetime import datetime, timezone


from config import CONSULTAS_LOG_DB


_SCHEMA = """
CREATE TABLE IF NOT EXISTS consultas (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp TEXT NOT NULL,
    endpoint TEXT NOT NULL,
    pregunta TEXT NOT NULL,
    chunks_recuperados TEXT,
    modelo_usado TEXT,
    respuesta TEXT,
    latencia_recuperacion_ms REAL,
    latencia_generacion_ms REAL,
    tokens_generados INTEGER,
    evaluacion TEXT DEFAULT NULL
);
"""



def get_conn():
    conn = sqlite3.connect(CONSULTAS_LOG_DB)
    conn.row_factory = sqlite3.Row
    return conn



def init_db():
    conn = get_conn()
    conn.execute(_SCHEMA)
    conn.commit()
    conn.close()



def log_consulta(endpoint: str, pregunta: str, chunks_recuperados: list,
                 modelo_usado: str, respuesta: str,
                 latencia_recuperacion_ms: float, latencia_generacion_ms: float,
                 tokens_generados: int | None) -> int:
    conn = get_conn()



    cur = conn.execute(
        """INSERT INTO consultas
           (timestamp, endpoint, pregunta, chunks_recuperados, modelo_usado,
            respuesta, latencia_recuperacion_ms, latencia_generacion_ms,
            tokens_generados, evaluacion)
           VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, NULL)""",
        (
            datetime.now(timezone.utc).isoformat(),
            endpoint,
            pregunta,
            json.dumps(chunks_recuperados, ensure_ascii=False),
            modelo_usado,
            respuesta,
            latencia_recuperacion_ms,
            latencia_generacion_ms,
            tokens_generados,
        ),
    )
    conn.commit()
    rowid = cur.lastrowid
    conn.close()
```

```
    return rowid



def metricas():
    conn = get_conn()
    total = conn.execute("SELECT COUNT(*) AS n FROM consultas").fetchone()["n"]


    def percentile(col, p):
        rows = [r[0] for r in conn.execute(f"SELECT {col} FROM consultas WHERE {col} IS NOT NULL
ORDER BY
{col}")]
        if not rows:
            return None
        k = (len(rows) - 1) * p
        f, c = int(k), min(int(k) + 1, len(rows) - 1)
        if f == c:
            return rows[f]
        return rows[f] + (rows[c] - rows[f]) * (k - f)


    avg_recuperacion = conn.execute(
        "SELECT AVG(latencia_recuperacion_ms) AS a FROM consultas"
    ).fetchone()["a"]
    avg_generacion = conn.execute(
        "SELECT AVG(latencia_generacion_ms) AS a FROM consultas"
    ).fetchone()["a"]
    sin_contexto = conn.execute(
        "SELECT COUNT(*) AS n FROM consultas "
        "WHERE respuesta LIKE '%no encuentro esa informacion en la documentacion%'"
    ).fetchone()["n"]


    resultado = {
        "total_consultas": total,
        "latencia_recuperacion_ms": {
            "promedio": avg_recuperacion,
            "p95": percentile("latencia_recuperacion_ms", 0.95),
        },
        "latencia_generacion_ms": {
            "promedio": avg_generacion,
            "p95": percentile("latencia_generacion_ms", 0.95),
        },
        "consultas_sin_contexto_relevante": sin_contexto,
    }
    conn.close()
    return resultado
```

# ANEXO F. CÓDIGO FUENTE: api/main.py

Este anexo contiene el código fuente completo del archivo api/main.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import json
import re
import subprocess
from pathlib import Path


import requests
from fastapi import FastAPI, File, HTTPException, UploadFile
from fastapi.responses import StreamingResponse
from pydantic import BaseModel, field_validator


import db
import plc_store
from conocimiento_plc import ALARMAS_PLC, RANGOS_SENSORES_PLC
from config import NODE_RED_URL, OLLAMA_HOST, QDRANT_URL
from rag import responder_con_herramientas, responder_con_herramientas_stream


MIN_LECTURAS_SENSOR_TRABADO = 10


app = FastAPI(title="PPA-RAG API")


DOCUMENTOS_DIR = Path("/opt/ppa-rag/data/manuales/base_conocimiento")
INGEST_DIR = Path("/opt/ppa-rag/ingest")
EXTENSIONES_PERMITIDAS = {".pdf", ".docx", ".xlsx", ".md"}



@app.on_event("startup")
def startup():
    db.init_db()



_CARACTERES_PROBLEMATICOS = re.compile(r"['@$]")



def _filtrar_pregunta(pregunta: str) -> str:
    return _CARACTERES_PROBLEMATICOS.sub("", pregunta)



class ConsultaRequest(BaseModel):
    pregunta: str


    @field_validator("pregunta")
    @classmethod
    def _sanear_pregunta(cls, v: str) -> str:
        return _filtrar_pregunta(v)



@app.post("/consulta")
def consulta(req: ConsultaRequest):
    try:
        resultado = responder_con_herramientas(req.pregunta)
    except requests.RequestException:
        raise HTTPException(
            status_code=504,
            detail="El modelo tardo demasiado en responder (timeout). Intenta de nuevo o con una
pregunta
mas corta.",
        )
    fuentes = [
        {"documento": c["documento"], "pagina": c["pagina"], "score": c["score"]}
        for c in resultado["chunks"]
```

```
    ]
    db.log_consulta(



        endpoint="/consulta",
        pregunta=req.pregunta,
        chunks_recuperados=fuentes,
        modelo_usado=resultado["modelo_usado"],
        respuesta=resultado["respuesta"],
        latencia_recuperacion_ms=resultado["latencia_recuperacion_ms"],
        latencia_generacion_ms=resultado["latencia_generacion_ms"],
        tokens_generados=resultado["tokens_generados"],
    )
    return {
        "respuesta": resultado["respuesta"],
        "fuentes": fuentes,
        "modelo_usado": resultado["modelo_usado"],
        "herramientas_invocadas": resultado["herramientas_invocadas"],
        "respuesta_determinista": resultado.get("respuesta_determinista", False),
    }



@app.post("/consulta/stream")
def consulta_stream(req: ConsultaRequest):
    def generar():
        texto_completo = ""
        fuentes = []
        modelo_usado = None
        latencia_ms = 0.0
        try:
            for evento in responder_con_herramientas_stream(req.pregunta):
                if evento["tipo"] == "chunk":
                    texto_completo += evento["texto"]
                else:
                    fuentes = evento["fuentes"]
                    modelo_usado = evento["modelo_usado"]
                    latencia_ms = evento["latencia_ms"]
                yield json.dumps(evento, ensure_ascii=False) + "\n"
        except requests.RequestException:
            yield json.dumps({
                "tipo": "error",
                "detalle": "El modelo tardo demasiado en responder (timeout). Intenta de nuevo o con
una
pregunta mas corta.",
            }, ensure_ascii=False) + "\n"
            return

        db.log_consulta(
            endpoint="/consulta/stream",
            pregunta=req.pregunta,
            chunks_recuperados=fuentes,
            modelo_usado=modelo_usado,
            respuesta=texto_completo,
            latencia_recuperacion_ms=0.0,
            latencia_generacion_ms=latencia_ms,
            tokens_generados=None,
        )

    return StreamingResponse(generar(), media_type="application/x -ndjson")



def _verificar_salud(url: str) -> str:
    try:
        r = requests.get(url, timeout=5)
        return "ok" if r.ok else f"error http {r.status_code}"
    except requests.RequestException as e:
        return f"down ({e.__class__.__name__})"



@app.get("/salud")
def salud():
    return {
        "ollama": _verificar_salud(f"{OLLAMA_HOST}/api/tags"),
        "qdrant": _verificar_salud(f"{QDRANT_URL}/healthz"),
        "node_red": _verificar_salud(f"{NODE_RED_URL}/"),
```

```
    }


@app.get("/metricas")
def metricas():
    return db.metricas()



@app.post("/plc/sesiones/iniciar")
def plc_iniciar_sesion():
    return plc_store.iniciar_sesion()



@app.post("/plc/sesiones/detener")
def plc_detener_sesion():
    sesion = plc_store.detener_sesion()
    if sesion is None:
        return {"detenida": False, "nota": "no habia ninguna sesion activa"}
    return {"detenida": True, **sesion}



@app.get("/plc/sesiones/estado")
def plc_estado_sesion():
    sesion = plc_store.sesion_activa()
    if sesion is None:
        return {"activa": False}
    sesiones = plc_store.listar_sesiones(limit=1)
    num_lecturas = sesiones[0]["num_lecturas"] if sesiones else 0
    return {"activa": True, **sesion, "num_lecturas": num_lecturas}



@app.get("/plc/sesiones")
def plc_listar_sesiones():
    return {"sesiones": plc_store.listar_sesiones()}



@app.get("/plc/lecturas")
def plc_lecturas(sesion_id: int | None = None, tag_plc: str | None = None, limit: int = 200):
    return {"lecturas": plc_store.lecturas_recientes(tag_plc=tag_plc, sesion_id=sesion_id,
limit=limit)}



def _alertas_sobretemperatura(sesion_id: int, resumen_por_tag: dict) -> list[dict]:
    alertas = []
    for tag_plc, regla in ALARMAS_PLC.items():
        fila = resumen_por_tag.get(tag_plc)
        if fila is None or regla["tipo"] != "HH" or fila["maximo"] is None:
            continue
        if fila["maximo"] >= regla["limite"]:
            alertas.append({
                "categoria": "proceso",
                "severidad": "alta",
                "tag_plc": tag_plc,
                "mensaje": f"{regla['mensaje']} (máximo registrado: {fila['maximo']:.2f})",
                "accion": regla["accion"],
                "momento": None,
                "aproximado": False,
            })
    return alertas



def _alertas_nivel_alto_sin_cerrar(sesion_id: int) -> list[dict]:
    correlacion = plc_store.lecturas_correlacionadas(sesion_id, "LSH_100", "CV_100_CMD")
    ocurrencia = next((c for c in correlacion if c["valor_a"] == 1 and c["valor_b"] == 1), None)
    if ocurrencia is None:
        return []
    return [{
        "categoria": "proceso",
        "severidad": "media",
        "tag_plc": "LSH_100 + CV_100_CMD",
```

```
        "mensaje": "Nivel alto en el tanque de enfriamiento (LSH-100 activo) con la válvula de
llenado CV-
100 todavía abierta",
        "accion": "Cerrar CV-100 (IL-04); riesgo de rebose",
        "momento": ocurrencia["timestamp_a"].isoformat(),
        "aproximado": False,
    }]



def _alertas_nivel_bajo(sesion_id: int, resumen_por_tag: dict) -> list[dict]:
    fila = resumen_por_tag.get("LSL_100")
    if fila is None or fila["maximo"] != 1:
        return []
    return [{
        "categoria": "proceso",
        "severidad": "alta",
        "tag_plc": "LSL_100",
        "mensaje": "Nivel bajo detectado en el tanque de enfriamiento (LSL -100 activo)",
        "accion": "Riesgo de marcha en seco de la bomba M1 si está arrancada (IL-03); el estado de M1
no se
captura en la base de datos, por lo que no se puede confirmar si está corriendo",
        "momento": None,
        "aproximado": True,
    }]



def _alertas_resistencia_sin_agua(sesion_id: int) -> list[dict]:
    correlacion = plc_store.lecturas_correlacionadas(sesion_id, "RES_1_CMD", "PE_102_PV")
    ocurrencia = next(
        (c for c in correlacion if c["valor_a"] == 1 and c["valor_b"] is not None and c["valor_b"] <=
0),
        None,
    )
    if ocurrencia is None:
        return []
    return [{
        "categoria": "proceso",
        "severidad": "alta",
        "tag_plc": "RES_1_CMD + PE_102_PV",
        "mensaje": f"Resistencia 1 comandada encendida con PE-102 en/bajo su cero documentado
({ocurrencia['valor_b']:.2f} cm) — posible tanque de calentamiento sin agua",
        "accion": "Verificar nivel real antes de energizar; el flotador de protección oficial (LSL -
102) no
se captura en la base de datos, esta alerta usa PE -102 del mismo tanque como sustituto aproximado",
        "momento": ocurrencia["timestamp_a"].isoformat(),
        "aproximado": True,
    }]



def _alertas_fuera_de_rango(resumen_por_tag: dict) -> list[dict]:
    alertas = []
    for tag_plc, (esc_min, esc_max, unidad) in RANGOS_SENSORES_PLC.items():
        fila = resumen_por_tag.get(tag_plc)
        if fila is None:
            continue
        if fila["minimo"] is not None and fila["minimo"] < esc_min:
            alertas.append({
                "categoria": "falla_sensor",
                "severidad": "media",
                "tag_plc": tag_plc,
                "mensaje": f"{tag_plc} registró {fila['minimo']:.2f} {unidad}, por debajo del rango
documentado ({esc_min}-{esc_max} {unidad})",
                "accion": "Revisar transmisor/cableado (posible lazo 4-20mA abierto)",
                "momento": None,
                "aproximado": False,
            })
        if fila["maximo"] is not None and fila["maximo"] > esc_max:
            alertas.append({
                "categoria": "falla_sensor",
                "severidad": "media",
                "tag_plc": tag_plc,
                "mensaje": f"{tag_plc} registró {fila['maximo']:.2f} {unidad}, por encima del rango
documentado ({esc_min}-{esc_max} {unidad})",
                "accion": "Revisar transmisor/cableado (posible saturación del canal analógico)",
```

```
                "momento": None,
                "aproximado": False,
            })
    return alertas


def _alertas_sensor_trabado(sesion_id: int) -> list[dict]:
    alertas = []
    for tag_plc in RANGOS_SENSORES_PLC:
        recientes = plc_store.lecturas_recientes(
            tag_plc=tag_plc, sesion_id=sesion_id, limit=MIN_LECTURAS_SENSOR_TRABADO
        )
        valores = [r["valor_ingenieria"] for r in recientes if r["valor_ingenieria"] is not None]
        if len(valores) < MIN_LECTURAS_SENSOR_TRABADO:
            continue
        if len(set(valores)) == 1:
            alertas.append({
                "categoria": "falla_sensor",
                "severidad": "media",
                "tag_plc": tag_plc,
                "mensaje": f"{tag_plc} no cambió de valor ({valores[0]:.3f}) en las últimas
{len(valores)}
lecturas",
                "accion": "Posible sensor trabado o desconectado; verificar si el resto de la sesión
sí
muestra variación",
                "momento": recientes[0]["timestamp"].isoformat(),
                "aproximado": True,
            })
    return alertas


@app.get("/plc/alertas")
def plc_alertas(sesion_id: int | None = None):
    if sesion_id is None:
        activa = plc_store.sesion_activa()
        if activa is None:
            return {"alertas": [], "nota": "no hay sesion activa ni sesion_id especificado"}
        sesion_id = activa["id"]

    resumen_por_tag = {f["tag_plc"]: f for f in plc_store.resumen_sesion(sesion_id)}

    alertas = []
    alertas += _alertas_sobretemperatura(sesion_id, resumen_por_tag)
    alertas += _alertas_nivel_alto_sin_cerrar(sesion_id)
    alertas += _alertas_nivel_bajo(sesion_id, resumen_por_tag)
    alertas += _alertas_resistencia_sin_agua(sesion_id)
    alertas += _alertas_fuera_de_rango(resumen_por_tag)
    alertas += _alertas_sensor_trabado(sesion_id)

    return {"sesion_id": sesion_id, "alertas": alertas}


@app.post("/documentos/subir")
async def subir_documento(archivo: UploadFile = File(...)):
    nombre = Path(archivo.filename).name
    ext = Path(nombre).suffix.lower()
    if ext not in EXTENSIONES_PERMITIDAS:
        raise HTTPException(
            status_code=400,
            detail=f"Formato no soportado: {ext}. Usa .pdf, .docx, .xlsx o .md",
        )

    destino = DOCUMENTOS_DIR / nombre
    if destino.exists():
        raise HTTPException(status_code=409, detail=f"Ya existe un archivo con ese nombre: {nombre}")

    contenido = await archivo.read()
    destino.write_bytes(contenido)

    resultado = subprocess.run(
        ["venv/bin/python", "ingest.py"],
```

```
        cwd=INGEST_DIR,
        capture_output=True,
        text=True,
        timeout=300,
    )
    if resultado.returncode != 0:
        raise HTTPException(
            status_code=500,
            detail=f"El archivo se guardo pero la ingesta fallo: {resultado.stderr[ -2000:]}",
        )

    return {"guardado": nombre, "salida_ingesta": resultado.stdout}
```

# ANEXO G. CÓDIGO FUENTE: api/ollama_client.py

Este anexo contiene el código fuente completo del archivo api/ollama_client.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import json

import requests

from config import (
    OLLAMA_EMBED_MODEL,
    OLLAMA_GEN_TIMEOUT,
    OLLAMA_HOST,
    OLLAMA_MODEL,
    OLLAMA_MODEL_FALLBACK,
    OLLAMA_NUM_CTX,
    OLLAMA_NUM_PREDICT,
    OLLAMA_TEMPERATURE,
)

_OPCIONES = {
    "num_ctx": OLLAMA_NUM_CTX,
    "num_predict": OLLAMA_NUM_PREDICT,
    "temperature": OLLAMA_TEMPERATURE,
}



def embed(text: str) -> list[float]:
    resp = requests.post(
        f"{OLLAMA_HOST}/api/embeddings",
        json={"model": OLLAMA_EMBED_MODEL, "prompt": text},
        timeout=60,
    )
    resp.raise_for_status()
    return resp.json()["embedding"]



def chat(messages: list, tools: list | None = None) -> dict:
    for modelo in (OLLAMA_MODEL, OLLAMA_MODEL_FALLBACK):
        body = {
            "model": modelo,
            "messages": messages,
            "stream": False,
            "options": _OPCIONES,
        }
        if tools:
            body["tools"] = tools



        try:
            resp = requests.post(f"{OLLAMA_HOST}/api/chat", json=body, timeout=OLLAMA_GEN_TIMEOUT)
            resp.raise_for_status()
            data = resp.json()
            mensaje = data.get("message", {})
            return {
                "modelo": modelo,
                "contenido": mensaje.get("content", ""),
                "tool_calls": mensaje.get("tool_calls") or [],
                "tokens_generados": data.get("eval_count"),
            }
        except requests.RequestException:
            if modelo == OLLAMA_MODEL_FALLBACK:
                raise
            continue



def chat_stream(messages: list):
    body = {
```

```
        "model": OLLAMA_MODEL,
        "messages": messages,
        "stream": True,
        "options": _OPCIONES,
    }
    with requests.post(
        f"{OLLAMA_HOST}/api/chat", json=body, timeout=OLLAMA_GEN_TIMEOUT, stream=True
    ) as resp:
        resp.raise_for_status()
        for linea in resp.iter_lines():
            if not linea:
                continue
            data = json.loads(linea)
            contenido = data.get("message", {}).get("content", "")
            if contenido:
                yield ("delta", contenido)
            if data.get("done"):
                yield ("fin", {
                    "modelo": OLLAMA_MODEL,
                    "tokens_generados": data.get("eval_count"),
                })
                return



def generate(prompt: str) -> dict:
    for modelo in (OLLAMA_MODEL, OLLAMA_MODEL_FALLBACK):
        try:
            resp = requests.post(
                f"{OLLAMA_HOST}/api/generate",
                json={
                    "model": modelo,
                    "prompt": prompt,
                    "stream": False,
                    "options": _OPCIONES,
                },
                timeout=OLLAMA_GEN_TIMEOUT,
            )
            resp.raise_for_status()
            body = resp.json()
            return {
                "modelo": modelo,
                "respuesta": body.get("response", ""),
                "tokens_generados": body.get("eval_count"),
            }
        except requests.RequestException:
            if modelo == OLLAMA_MODEL_FALLBACK:
                raise
            continue
```

# ANEXO H. CÓDIGO FUENTE: api/plc_store.py

Este anexo contiene el código fuente completo del archivo api/plc_store.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import psycopg
from psycopg.rows import dict_row

from config import POSTGRES_DB, POSTGRES_HOST, POSTGRES_PASSWORD, POSTGRES_PORT, POSTGRES_USER


def _conn():
    return psycopg.connect(
        host=POSTGRES_HOST,
        port=POSTGRES_PORT,
        dbname=POSTGRES_DB,
        user=POSTGRES_USER,
        password=POSTGRES_PASSWORD,
        row_factory=dict_row,
    )


def sesion_activa() -> dict | None:
    with _conn() as conn:
        row = conn.execute(
            "SELECT id, iniciada_en, finalizada_en, estado FROM sesiones_plc "
            "WHERE estado = 'activa' ORDER BY iniciada_en DESC LIMIT 1"
        ).fetchone()
        return dict(row) if row else None


def iniciar_sesion() -> dict:
    with _conn() as conn:
        conn.execute(
            "UPDATE sesiones_plc SET estado = 'detenida', finalizada_en = now() "
            "WHERE estado = 'activa'"
        )
        row = conn.execute(
            "INSERT INTO sesiones_plc DEFAULT VALUES "
            "RETURNING id, iniciada_en, finalizada_en, estado"
        ).fetchone()
        conn.commit()
        return dict(row)


def detener_sesion() -> dict | None:
    with _conn() as conn:
        row = conn.execute(
            "UPDATE sesiones_plc SET estado = 'detenida', finalizada_en = now() "
            "WHERE estado = 'activa' "
            "RETURNING id, iniciada_en, finalizada_en, estado"
        ).fetchone()
        conn.commit()
        return dict(row) if row else None


def listar_sesiones(limit: int = 20) -> list[dict]:
    with _conn() as conn:
        rows = conn.execute(
            "SELECT s.id, s.iniciada_en, s.finalizada_en, s.estado, "
            "COUNT(l.id) AS num_lecturas "
            "FROM sesiones_plc s LEFT JOIN lecturas_plc l ON l.sesion_id = s.id "
            "GROUP BY s.id ORDER BY s.iniciada_en DESC LIMIT %s",


            (limit,),
        ).fetchall()
```

```
        return [dict(r) for r in rows]


def lecturas_recientes(tag_plc: str | None = None, sesion_id: int | None = None,
                       limit: int = 50, orden: str = "DESC") -> list[dict]:
    orden = "ASC" if str(orden).upper() == "ASC" else "DESC"
    condiciones = []
    params: list = []
    if tag_plc is not None:
        condiciones.append("tag_plc = %s")
        params.append(tag_plc)
    if sesion_id is not None:
        condiciones.append("sesion_id = %s")
        params.append(sesion_id)
    where = f"WHERE {' AND '.join(condiciones)}" if condiciones else ""
    params.append(limit)

    with _conn() as conn:
        rows = conn.execute(
            f"SELECT sesion_id, timestamp, tag_plc, valor_crudo, valor_ingenieria, unidad "
            f"FROM lecturas_plc {where} ORDER BY timestamp {orden} LIMIT %s",
            params,
        ).fetchall()
        return [dict(r) for r in rows]


def lecturas_correlacionadas(sesion_id: int, tag_a: str, tag_b: str,
                             tolerancia_s: int = 3) -> list[dict]:
    with _conn() as conn:
        rows = conn.execute(
            """
            SELECT a.timestamp AS timestamp_a, a.valor_ingenieria AS valor_a,
                   b.timestamp AS timestamp_b, b.valor_ingenieria AS valor_b
            FROM lecturas_plc a
            JOIN LATERAL (
                SELECT valor_ingenieria, timestamp
                FROM lecturas_plc
                WHERE sesion_id = %s AND tag_plc = %s
                ORDER BY abs(extract(epoch from (timestamp - a.timestamp)))
                LIMIT 1
            ) b ON true
            WHERE a.sesion_id = %s AND a.tag_plc = %s
              AND abs(extract(epoch from (b.timestamp - a.timestamp))) <= %s
            ORDER BY a.timestamp DESC
            """,
            (sesion_id, tag_b, sesion_id, tag_a, tolerancia_s),
        ).fetchall()
        return [dict(r) for r in rows]


def resumen_sesion(sesion_id: int, tag_plc: str | None = None) -> list[dict]:
    condiciones = ["sesion_id = %s"]
    params: list = [sesion_id]
    if tag_plc is not None:
        condiciones.append("tag_plc = %s")
        params.append(tag_plc)
    with _conn() as conn:
        rows = conn.execute(
            "SELECT tag_plc, unidad, "
            "MIN(valor_ingenieria) AS minimo, MAX(valor_ingenieria) AS maximo, "
            "AVG(valor_ingenieria) AS promedio, COUNT(*) AS num_lecturas "
            f"FROM lecturas_plc WHERE {' AND '.join(condiciones)} GROUP BY tag_plc, unidad",
            params,
        ).fetchall()
        return [dict(r) for r in rows]
```

# ANEXO I. CÓDIGO FUENTE: api/rag.py

Este anexo contiene el código fuente completo del archivo api/rag.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import json
import re
import time


from qdrant_client import QdrantClient, models


import plc_store
from conocimiento_plc import ALARMAS_PLC, INTERLOCKS_PLC, LINEALIZACION_PLC, etiqueta_digital
from config import QDRANT_COLLECTION, QDRANT_URL, TOP_K
from ollama_client import chat, chat_stream, embed, generate


client = QdrantClient(url=QDRANT_URL)


DOCUMENTO_AUTORITATIVO = "PPA_02_diccionario_tags.md"
BOOST_AUTORITATIVO = 0.05



SYSTEM_PROMPT = (
    "Eres un asistente tecnico de la Planta de Procesos Analogos (PPA). "
    "Responde UNICAMENTE con base en el contexto documental proporcionado. "
    "Si el contexto no contiene informacion relevante para responder la "
    "pregunta, responde exactamente: 'no encuentro esa informacion en la "
    "documentacion'. No inventes datos que no esten en el contexto. Cuando "
    "cites informacion, menciona el documento y la pagina de origen."
)



def retrieve(pregunta: str, top_k: int = TOP_K):
    t0 = time.perf_counter()
    vector = embed(pregunta)
    hits = client.query_points(
        collection_name=QDRANT_COLLECTION, query=vector, limit=top_k * 3
    ).points
    latencia_ms = (time.perf_counter() - t0) * 1000
    chunks = [
        {
            "documento": h.payload.get("documento"),
            "pagina": h.payload.get("pagina"),
            "score": h.score + (BOOST_AUTORITATIVO if h.payload.get("documento") ==
DOCUMENTO_AUTORITATIVO
else 0.0),
            "texto": h.payload.get("texto"),
        }
        for h in hits
    ]
    chunks.sort(key=lambda c: c["score"], reverse=True)
    return chunks[:top_k], latencia_ms



_TAG_RE = re.compile(r"\b[A-Za-z]{1,6}[-_]\d{1,4}(?:_PV)?\b")
_diccionario_tags_cache: dict[str, dict] | None = None



def _construir_diccionario_tags() -> dict[str, dict]:
    diccionario: dict[str, dict] = {}
    puntos, _ = client.scroll(
        collection_name=QDRANT_COLLECTION,
        scroll_filter=models.Filter(



            must=[models.FieldCondition(key="documento",
```

```
match=models.MatchValue(value=DOCUMENTO_AUTORITATIVO))]
        ),
        limit=200,
        with_payload=True,
    )
    for punto in puntos:
        texto = punto.payload.get("texto", "")
        entrada = {
            "documento": punto.payload.get("documento"),
            "pagina": punto.payload.get("pagina"),
            "score": 1.0,
            "texto": texto,
        }
        for m in re.finditer(r"tag_(?:plc|planta):\s*([^\s;]+)", texto):
            diccionario[m.group(1).upper()] = entrada
    return diccionario



def _obtener_diccionario_tags() -> dict[str, dict]:
    global _diccionario_tags_cache
    if _diccionario_tags_cache is None:
        _diccionario_tags_cache = _construir_diccionario_tags()
    return _diccionario_tags_cache



def _tags_mencionados(texto_consulta: str) -> list[dict]:
    diccionario = _obtener_diccionario_tags()
    encontrados = []
    for m in _TAG_RE.findall(texto_consulta.upper()):
        entrada = diccionario.get(m)
        if entrada and entrada not in encontrados:
            encontrados.append(entrada)
    return encontrados



def _tags_no_encontrados(pregunta_original: str) -> list[str]:
    diccionario = _obtener_diccionario_tags()
    no_encontrados = []
    for m in _TAG_RE.findall((pregunta_original or "").upper()):
        if re.match(r"^(AL|IL|TK)[-_]\d+$", m):
            continue
        if m not in diccionario and m not in no_encontrados:
            no_encontrados.append(m)
    return no_encontrados



def build_prompt(pregunta: str, chunks: list) -> str:
    if chunks:
        contexto = "\n\n".join(
            f"[Fuente: {c['documento']}, pagina {c['pagina']}]\n{c['texto']}" for c in chunks
        )
    else:
        contexto = "(sin contexto documental relevante)"
    return (
        f"{SYSTEM_PROMPT}\n\n"
        f"Contexto documental:\n{contexto}\n\n"
        f"Pregunta: {pregunta}\n"
        f"Respuesta:"
    )



def responder(pregunta: str, top_k: int = TOP_K):
    chunks, latencia_recuperacion_ms = retrieve(pregunta, top_k)
    prompt = build_prompt(pregunta, chunks)


    t0 = time.perf_counter()
    resultado = generate(prompt)
    latencia_generacion_ms = (time.perf_counter() - t0) * 1000



    return {
        "respuesta": resultado["respuesta"],
        "modelo_usado": resultado["modelo"],
        "tokens_generados": resultado["tokens_generados"],
        "chunks": chunks,
```

```
        "latencia_recuperacion_ms": latencia_recuperacion_ms,
        "latencia_generacion_ms": latencia_generacion_ms,
    }



SYSTEM_PROMPT_HERRAMIENTAS = (
    "Eres el asistente tecnico de la Planta de Procesos Analogos (PPA). "
    "Tienes 2 herramientas: buscar_documentacion (manuales/bitacoras) y "
    "consultar_sensores_plc (historico real de sensores). Usa la que "
    "corresponda, o ambas si la pregunta tiene las dos partes. Responde SOLO "
    "con datos de las herramientas, nunca inventes.\n\n"
    "Si la pregunta menciona una sesion por numero (ej. 'sesion 16', "
    "'sesion #20') o menciona 'tiempo real'/'en vivo'/'sesion activa'/'ahora "
    "mismo', esos datos SOLO viven en consultar_sensores_plc (base de datos "
    "real de la planta): usa esa herramienta, nunca buscar_documentacion, y "
    "nunca mezcles esa respuesta con contenido de los manuales.\n\n"
    "Si un mensaje de herramienta ya trae datos (lista no vacia u objeto sin "
    "'error'), USA ESOS DATOS: nunca digas 'no encuentro esa informacion' "
    "si ya tienes el dato, aunque el valor parezca extrano (ej. negativo). "
    "Esa frase es solo para cuando NINGUNA herramienta trajo nada util. Para "
    "medidas de consultar_sensores_plc reporta siempre valor_ingenieria con "
    "su unidad (nunca valor_crudo, que es la cuenta sin calibrar del sensor, "
    "no una medida real).\n\n"
    "Nunca escribas sintaxis de llamada a funcion (ej. {\"name\": ...}) como "
    "texto de respuesta: aqui se responde solo en lenguaje natural. \n\n"
    "Se breve: 1-3 frases, solo lo preguntado, sin contexto extra ni "
    "resumen final. Cita el documento entre parentesis al final usando SOLO "
    "el nombre de archivo ya provisto (ej. PPA_01_equipos_y_tanques.md); "
    "nunca inventes URLs, enlaces, rutas ni nombres de archivo que no te "
    "dieron.\n\n"
    "Si el tag/equipo preguntado no aparece tal cual pero hay uno "
    "relacionado en el contexto (misma zona/funcion), dilo brevemente y "
    "sugiere el real en vez de solo negar. Si no hay nada relacionado, "
    "responde exactamente: 'no encuentro esa informacion en la "
    "documentacion'.\n\n"
    "Si usaste ambas herramientas, tu respuesta debe cubrir las dos partes "
    "de la pregunta, no solo la ultima herramienta llamada. Si la pregunta "
    "compara un dato medido contra un limite/umbral documentado, o pregunta "
    "si algo 'es consistente con' un interlock/alarma, SIEMPRE invoca ambas "
    "herramientas (dato real + documentacion), nunca solo una.\n\n"
    "'Rango operativo' (rango_min/rango_max, cuanto puede medir el sensor) "
    "y 'limite de alarma' (umbral documentado que dispara una alerta) son "
    "cosas DISTINTAS: no confundas uno con el otro ni inventes un limite de "
    "alarma que no este explicitamente documentado.\n\n"
    "Si piden que CONFIGURES/CREES un interlock o alarma nueva, o que "
    "RECOMIENDES un setpoint/umbral nuevo no documentado, o instrucciones "
    "para hardware de otra marca/modelo no mencionado en el contexto: "
    "responde que no tienes autoridad para eso, sin inventar un valor ni "
    "una guia.\n\n"
    "Si el contexto de una herramienta incluye una linea que empieza con "
    "'ADVERTENCIA:', esa instruccion tiene prioridad absoluta: sigue "
    "exactamente lo que dice, incluso si otro fragmento del contexto "
    "parece relacionado o similar."
)

TOOLS = [
    {
        "type": "function",
        "function": {
            "name": "buscar_documentacion",
            "description": (
                "Busca en manuales/bitacoras de la PPA: que es un equipo o "
                "tag, a que tanque pertenece, que sensores/actuadores TIENE "
                "un tanque o equipo, procedimientos, teoria, codigo PLC. "
                "USAR PRIMERO SIEMPRE que la pregunta nombre un equipo, "



                "bomba, tanque o tag (ej. 'P-100', 'TE-102', 'TK-102'), "
                "aunque la pregunta use la palabra 'sensores' — 'que "
                "sensores tiene el tanque X' es una pregunta de catalogo "
                "(documentacion), NO de lecturas: usar consultar_sensores_plc "
                "solo cuando se pide un VALOR MEDIDO especifico (temperatura "
                "actual, maximo, promedio), nunca para listar que sensores "
                "existen."
            ),
            "parameters": {
                "type": "object",
                "properties": {
                    "pregunta": {
                        "type": "string",
```

```
                        "description": "Pregunta o tema a buscar en la documentacion",
                    }
                },
                "required": ["pregunta"],
            },
        },
    },
    {
        "type": "function",
        "function": {
            "name": "consultar_sensores_plc",
            "description": (
                "Consulta lecturas YA CAPTURADAS de sensores del PLC por "
                "tag_plc (ej. TT_102_PV). Usar SOLO cuando se pide un valor "
                "medido especifico (temperatura/nivel/presion actual, "
                "maximo, minimo, promedio) o que sesiones de captura "
                "existen. Si la pregunta menciona un numero de sesion (ej. "
                "'sesion 16', 'sesion #20', 'sesion # 20') junto con un "
                "valor/dato, SIEMPRE usar esta herramienta (nunca "
                "buscar_documentacion), pasando ese numero como sesion_id "
                "sin importar si viene con el simbolo #. Si la pregunta "
                "menciona 'tiempo real', 'en vivo', 'sesion activa', 'ahora "
                "mismo' o 'en curso', SIEMPRE usar esta herramienta tambien "
                "(nunca buscar_documentacion) y omitir sesion_id para que se "
                "use automaticamente la sesion activa: esos datos SOLO "
                "existen en la base de datos real capturada de la planta, "
                "los manuales/documentacion NUNCA tienen datos en tiempo "
                "real ni de una sesion especifica. NUNCA usar para 'que "
                "sensores tiene X', 'que equipos hay en X' o para "
                "identificar/confirmar un tag — eso siempre es "
                "buscar_documentacion."
            ),
            "parameters": {
                "type": "object",
                "properties": {
                    "tipo": {
                        "type": "string",
                        "enum": ["recientes", "resumen", "listar_sesiones"],
                        "description": (
                            "'recientes': ultimas lecturas crudas. 'resumen': "
                            "minimo/maximo/promedio por tag de una sesion. "
                            "'listar_sesiones': sesiones de captura disponibles."
                        ),
                    },
                    "sesion_id": {
                        "type": "integer",
                        "description": "ID de la sesion (opcional; si se omite en 'resumen' se usa la
sesion activa)",
                    },
                    "tag_plc": {
                        "type": "string",
                        "description": (
                            "Tag especifico, ej. TT_102_PV, PE_102_PV, LE_101_PV. "
                            "Sirve para 'recientes' Y para 'resumen'. Si la pregunta "
                            "menciona un sensor/medicion concreto (ej. 'temperatura "
                            "maxima', 'promedio de presion'), SIEMPRE pasa el tag_plc "
                            "correspondiente para filtrar el resultado a ese unico tag "



                            "en vez de traer todos los tags de la sesion."
                        ),
                    },
                    "limit": {
                        "type": "integer",
                        "description": "Cantidad maxima de lecturas para 'recientes' (default 20)",
                    },
                },
                "required": ["tipo"],
            },
        },
    },
]



TEXTO_MAX_CHARS = 500
CHUNKS_MODELO_MAX = 5



_documentos_reales_cache: set[str] | None = None
```

```
def _documentos_reales() -> set[str]:
    global _documentos_reales_cache
    if _documentos_reales_cache is None:
        vistos = set()
        puntos, _ = client.scroll(
            collection_name=QDRANT_COLLECTION, limit=1000, with_payload=["documento"]
        )
        for p in puntos:
            d = p.payload.get("documento")
            if d:
                vistos.add(d)
        _documentos_reales_cache = vistos
    return _documentos_reales_cache



_CITA_MD_RE = re.compile(r"\b([\w\-]+\.md)\b")
_TOOL_JSON_LEAK_RE = re.compile(
    r'\{\s*"name"\s*:\s*"(?:buscar_documentacion|consultar_sensores_plc)"[^{}]*\}'
)



def _limpia_citas_inventadas(texto: str) -> str:
    reales = _documentos_reales()
    texto = _CITA_MD_RE.sub(lambda m: m.group(1) if m.group(1) in reales else "", texto)
    texto = re.sub(r"\(\s*\)", "", texto)
    texto = re.sub(r"[ \t]{2,}", " ", texto)
    return texto.strip()



_TAG_KEYWORDS: list[tuple[str, list[str]]] = [
    ("TT_102_PV", ["tt-102", "tt_102", r"temperatura.*(calentamiento|tanque 2|tk-102|tk_102)"]),
    ("TT_100_PV", ["tt-100", "tt_100", r"temperatura.*(enfriamiento|inferior|tanque 0|tk-
100|tk_100)"]),
    ("PE_102_PV", ["pe-102", "pe_102", r"presi[oó]n diferencial"]),
    ("LE_101_PV", ["le-101", "le_101", r"nivel ultras[oó]nico"]),
    ("CHR_100_CMD", ["chr-100", "chr_100", "chiller"]),
    ("CV_100_CMD", ["cv-100", "cv_100", r"v[aá]lvula inferior"]),
    ("CV_102_CMD", ["cv-102", "cv_102", r"v[aá]lvula de mezcla"]),
    ("CV_104_CMD", ["cv-104", "cv_104", r"v[aá]lvula superior"]),
    ("RES_1_CMD", ["res-1", "res_1_cmd", r"resistencia\s*1\b"]),
    ("LSH_100", ["lsh-100", "lsh_100", r"nivel alto"]),
    ("LSL_100", ["lsl-100", "lsl_100", r"nivel bajo"]),
]

_STAT_KEYWORDS = [
    ("rango", [r"rango.*m[aá]xim[ao].*m[ií]nim[ao]", r"m[aá]xim[ao].*menos.*m[ií]nim[ao]"]),
    ("maximo", [r"m[aá]xim[ao]", r"m[aá]s\s+(alt[oa]|elevad[oa]|grande)", r"mayor\s+valor"]),
    ("minimo", [r"m[ií]nim[ao]", r"m[aá]s\s+baj[oa]", r"menor\s+valor"]),
    ("promedio", [r"promedio"]),



    ("conteo", [r"cu[aá]nt[ao]s?\s+lectura", r"n[uú]mero de lectura"]),
]



def _resolver_tag_por_pregunta(pregunta: str) -> str | None:
    if not pregunta:
        return None
    texto = pregunta.lower()
    encontrados = [tag for tag, patrones in _TAG_KEYWORDS if any(re.search(p, texto) for p in
patrones)]
    return encontrados[0] if len(encontrados) == 1 else None



def _resolver_stat_por_pregunta(pregunta: str) -> str | None:
    if not pregunta:
        return None
    texto = pregunta.lower()
    for stat, patrones in _STAT_KEYWORDS:
        if any(re.search(p, texto) for p in patrones):
            return stat
    return None
```

```
_SESION_RE = re.compile(r"sesi[oó]n\s*#?\s*(?:n[uú]mero\s+)?(\d+)")
_TIEMPO_REAL_RE = re.compile(
    r"tiempo\s+real|en\s+vivo|sesi[oó]n\s+activa|ahora\s+mismo|en\s+este\s+momento|"
    r"justo\s+ahora|en\s+curso"
)



def _es_pregunta_tiempo_real(pregunta: str) -> bool:
    if not pregunta:
        return False
    return bool(_TIEMPO_REAL_RE.search(pregunta.lower()))



def _resolver_sesion_por_pregunta(pregunta: str) -> int | None:
    if not pregunta:
        return None
    m = _SESION_RE.search(pregunta.lower())
    return int(m.group(1)) if m else None



_ULTIMA_LECTURA_RE = re.compile(r"[uú]ltim[ao]\s+(lectura|valor)|se\s+registr[oó]")
_PRIMER_VALOR_RE = re.compile(r"primer\s+valor|primera\s+lectura")


_CATEGORIAS_TAGS = {
    "temperatura": ["TT_102_PV", "TT_100_PV"],
    "nivel": ["LE_101_PV"],
    "presion": ["PE_102_PV"],
}
_CATEGORIA_KEYWORDS = [
    ("temperatura", r"temperatura"),
    ("nivel", r"nivel"),
    ("presion", r"presi[oó]n"),
]



def _resolver_categoria_por_pregunta(pregunta: str) -> str | None:
    if not pregunta:
        return None
    texto = pregunta.lower()
    for categoria, patron in _CATEGORIA_KEYWORDS:
        if re.search(patron, texto):
            return categoria
    return None



_INTENCION_VALOR_RE = re.compile(
    r"[uú]ltim[ao]\s+(lectura|valor)|cu[aá]nt[ao]s?\s+lectura|n[uú]mero de lectura|"
    r"valor\s+crudo|m[aá]xim[ao]|m[ií]nim[ao]|promedio|\brango\b|se\s+registr[oó]|"



    r"m[aá]s\s+(alt[oa]|baj[oa]|elevad[oa]|grande)|may[oó]r\s+valor|menor\s+valor"
)
_EXCLUSION_DOC_RE = re.compile(r"document|operaci[oó]n|permitid|recomend")
_CRUDO_RE = re.compile(r"valor\s+crudo|sin\s+lineal|sin\s+calibrar")



def _es_pregunta_de_valor(pregunta: str) -> bool:
    if not pregunta:
        return False
    texto = pregunta.lower()
    if _EXCLUSION_DOC_RE.search(texto):
        return False
    return bool(_INTENCION_VALOR_RE.search(texto))



_ALARMA_RE = re.compile(r"\balarma\b|\bumbral\b|l[ií]mite\s+de\s+alarma|l[ií]mite\s+document")
_RANGO_OPERATIVO_RE = re.compile(r"rango\s+operativ|rango\s+de\s+operaci")


_COMPARACION_ALARMA_RE = re.compile(
```

```
    r"supera|super[oó]|exced[eé]|excede|qu[eé]\s+tan\s+cerca|"
    r"deber[ií]a\s+(generarse|activarse)\s+una?\s+(alerta|alarma)"
)



def _respuesta_determinista_alarma(pregunta: str) -> str | None:
    if not pregunta:
        return None
    texto = pregunta.lower()
    si_rango = _RANGO_OPERATIVO_RE.search(texto)
    si_alarma = _ALARMA_RE.search(texto)
    si_comparacion = _COMPARACION_ALARMA_RE.search(texto)
    if not si_alarma and not si_comparacion:
        return None
    if si_rango and not re.search(r"alarma|umbral", texto):
        return None
    tag = _resolver_tag_por_pregunta(pregunta)
    if not tag and re.search(r"\bal-0?1\b", texto):
        tag = "TT_102_PV"
    if not tag:
        return None
    regla = ALARMAS_PLC.get(tag)


    if regla and si_comparacion:
        sesion_id = _resolver_sesion_por_pregunta(pregunta)
        filas = plc_store.lecturas_recientes(tag_plc=tag, sesion_id=sesion_id, limit=1)
        if filas and filas[0].get("valor_ingenieria") is not None:
            valor = round(filas[0]["valor_ingenieria"], 4)
            limite = regla["limite"]
            supera = valor > limite
            diferencia = round(abs(limite - valor), 4)
            veredicto = "Sí" if supera else "No"
            contexto_sesion = f"la sesión {sesion_id}" if sesion_id is not None else "la última
lectura"
            return (
                f"{veredicto}: en {contexto_sesion}, {tag} es {valor} °C, "
                f"{'supera' if supera else 'NO supera'} el límite de "
                f"{regla['mensaje']} ({limite} °C) — diferencia de "
                f"{diferencia} °C {'por encima' if supera else 'por debajo'} "
                f"del límite (sesión {filas[0]['sesion_id']}, "
                f"{filas[0]['timestamp']})."
            )
        if sesion_id is not None:
            return (
                f"La sesión {sesion_id} no tiene ninguna lectura de {tag} "
                f"capturada; no se puede evaluar la alarma {regla['mensaje']} "
                f"para esa sesión con los datos disponibles."
            )
        return None


    if regla:
        return (



            f"Sí, {tag} tiene una alarma documentada: {regla['mensaje']}, "
            f"límite {regla['limite']} (acción: {regla['accion']})."
        )
    return (
        f"No hay límite de alarma numérico documentado para {tag}. El único "
        f"límite de alarma documentado en el sistema es AL -01 (TT_102_PV, "
        f"60 °C); para los demás sensores no se implementó alarma por no "
        f"tener un umbral explícito en la documentación."
    )



_CRUDO_BITS_RE = re.compile(r"fuera\s+([\d]+(?:[.,]\d+)?)\s*bits")
_CRUDO_ES_RE = re.compile(r"crudo\s+es\s+([\d]+(?:[.,]\d+)?)")
_APLICO_CORRECTAMENTE_RE = re.compile(r"aplic[oó].*correctamente|coincide.*aplicar \s+manualmente")
_PIDE_RESULTADO_LINEALIZACION_RE = re.compile(
    r"valor\s+linealizad|seg[uú]n\s+la\s+ecuaci[oó]n\s+document|result(?:a|ar[ií]a)"
)



def _respuesta_determinista_linealizacion(pregunta: str) -> tuple[str, bool] | None:
    if not pregunta:
        return None
    texto = pregunta.lower()
```

```
    if not _PIDE_RESULTADO_LINEALIZACION_RE.search(texto) and not
_APLICO_CORRECTAMENTE_RE.search(texto):
        return None
    tag = _resolver_tag_por_pregunta(pregunta)
    if not tag or tag not in LINEALIZACION_PLC:
        return None
    m, b, unidad = LINEALIZACION_PLC[tag]
    signo = "-" if b < 0 else "+"
    eq_str = f"Y = {m}·X {signo} {abs(b)}"


    m_bits = _CRUDO_BITS_RE.search(texto)
    m_es = _CRUDO_ES_RE.search(texto)
    if m_bits or m_es:
        crudo = float((m_bits or m_es).group(1).replace(",", "."))
        resultado = round(m * crudo + b, 4)
        return (
            f"Aplicando la ecuación documentada de {tag} ({eq_str}) al valor "
            f"crudo hipotético {crudo}: resultado = {resultado} {unidad}.",
            False,
        )


    if _APLICO_CORRECTAMENTE_RE.search(texto):
        filas = plc_store.lecturas_recientes(tag_plc=tag, limit=1)
        if not filas or filas[0]["valor_crudo"] is None or filas[0]["valor_ingenieria"] is None:
            return None
        crudo = filas[0]["valor_crudo"]
        almacenado = round(filas[0]["valor_ingenieria"], 4)
        calculado = round(m * crudo + b, 4)
        coincide = abs(calculado - almacenado) < 0.01
        veredicto = "Sí" if coincide else "No"
        return (
            f"{veredicto}: aplicando {eq_str} al valor crudo real de {tag} "
            f"({crudo}) se obtiene {calculado} {unidad}, que "
            f"{'coincide' if coincide else 'NO coincide'} con el valor "
            f"almacenado en Postgres ({almacenado} {unidad}, sesión "
            f"{filas[0]['sesion_id']}).",
            True,
        )


    if _ULTIMA_LECTURA_RE.search(texto):
        filas = plc_store.lecturas_recientes(tag_plc=tag, limit=1)
        if not filas or filas[0]["valor_crudo"] is None:
            return None
        crudo = filas[0]["valor_crudo"]
        resultado = round(m * crudo + b, 4)
        return (
            f"Aplicando la ecuación documentada de {tag} ({eq_str}) al valor "
            f"crudo de la última lectura real ({crudo}, sesión "



            f"{filas[0]['sesion_id']}, {filas[0]['timestamp']}): "
            f"resultado = {resultado} {unidad}.",
            True,
        )


    return None



def _respuesta_determinista_plc(
    tipo: str, tag: str, stat: str | None, filas: list[dict], pregunta_original: str = ""
) -> str | None:
    if not filas:
        return f"No hay lecturas registradas de {tag} para esos filtros."


    if tipo == "recientes":
        f = filas[0]
        if _CRUDO_RE.search(pregunta_original or ""):
            return (
                f"El último valor_crudo (sin linealizar) de {tag} es "
                f"{round(f['valor_crudo'], 4)} -- esta es la cuenta cruda del "
                f"sensor, no la medida en unidad de ingenieria (sesión "
                f"{f['sesion_id']}, {f['timestamp']})."
            )
        etiqueta = etiqueta_digital(tag, f["valor_ingenieria"])
        sufijo_digital = f" ({etiqueta})" if etiqueta else ""
        return (
            f"El último valor de {tag} es {round(f['valor_ingenieria'], 4)} "
```

```
            f"{f['unidad'] or ''}{sufijo_digital} (sesión {f['sesion_id']}, "
            f"{f['timestamp']})."
        ).replace("  ", " ")

    if tipo == "resumen" and stat:
        f = filas[0]
        unidad = f["unidad"] or ""
        if stat == "conteo":
            return f"{tag} tiene {f['num_lecturas']} lecturas registradas."
        if stat == "rango":
            valor = round(f["maximo"] - f["minimo"], 4)
            return f"El rango (máximo - mínimo) de {tag} fue {valor} {unidad} ({f['num_lecturas']}
lecturas)."
        valor = round(f[stat], 4)
        etiqueta = {"maximo": "máximo", "minimo": "mínimo", "promedio": "promedio"}[stat]
        return f"El {etiqueta} de {tag} fue {valor} {unidad} ({f['num_lecturas']} lecturas)."

    return None



def _formatear_plc(tipo: str, salida) -> str:
    if isinstance(salida, dict) and "error" in salida:
        return f"Error de la herramienta: {salida['error']}"

    if not salida:
        return (
            "La consulta no encontro ninguna lectura/sesion que coincida "
            "(lista vacia -- no hay datos capturados para eso)."
        )

    if tipo == "listar_sesiones":
        lineas = [
            f"Sesion {s['id']}: iniciada {s['iniciada_en']}, "
            f"finalizada {s.get('finalizada_en') or '(sigue activa)'}, "
            f"estado={s['estado']}, {s['num_lecturas']} lecturas."
            for s in salida
        ]
        return "Sesiones de captura encontradas: \n" + "\n".join(lineas)

    def _r(x, nd=4):
        return round(x, nd) if isinstance(x, float) else x

    if tipo == "recientes":



        lineas = []
        for i, r in enumerate(salida):
            etiqueta = etiqueta_digital(r["tag_plc"], r["valor_ingenieria"])
            sufijo_digital = f" [ESTADO: {etiqueta}]" if etiqueta else ""
            marca = " <-- ESTA ES LA MAS RECIENTE (usar esta si piden 'ultima lectura')" if i == 0
else ""
            lineas.append(
                f"{r['tag_plc']} = {_r(r['valor_ingenieria'])} {r['unidad'] or ''}{sufijo_digital} "
                f"(este es el valor real, usar este; valor_crudo={_r(r['valor_crudo'])} es la "
                f"cuenta sin calibrar del sensor, NO es la medida) -- sesion "
                f"{r['sesion_id']}, {r['timestamp']}{marca}"
            )
        return "Ultimas lecturas encontradas (la mas reciente primero):\n" + "\n".join(lineas)

    lineas = [
        f"{r['tag_plc']}: minimo={_r(r['minimo'])}, maximo={_r(r['maximo'])}, "
        f"promedio={_r(r['promedio'])} {r['unidad'] or ''} ({r['num_lecturas']} lecturas)"
        for r in salida
    ]
    total_lecturas = sum(r["num_lecturas"] for r in salida)
    return (
        "Resumen de la sesion (valores ya en unidad de ingenieria): \n"
        + "\n".join(lineas)
        + f"\n\nTOTAL de lecturas de la sesion (todas las variables sumadas) = {total_lecturas}"
    )



def _ejecutar_tool(nombre: str, args: dict, pregunta_original: str = ""):
    if nombre == "buscar_documentacion":
        consulta = args.get("pregunta") or next(
```

```
            (v for v in args.values() if isinstance(v, str) and v.strip()), ""
        )
        if not consulta:
            return []
        chunks, _ = retrieve(consulta)
        for entrada in reversed(_tags_mencionados(pregunta_original) + _tags_mencionados(consulta)):
            chunks = [c for c in chunks if c["texto"] != entrada["texto"]]
            chunks.insert(0, entrada)

        faltantes = _tags_no_encontrados(pregunta_original)
        if faltantes:
            chunks.insert(0, {
                "documento": "(verificacion de tags)",
                "pagina": "-",
                "score": 999.0,
                "texto": (
                    f"ADVERTENCIA: el/los tag(s) {', '.join(faltantes)} NO existen "
                    f"en el diccionario de tags de la PPA. No los confundas con "
                    f"tags parecidos que si existen. Si la pregunta es sobre "
                    f"alguno de estos tags, responde que no existe, sin inventar "
                    f"una definicion."
                ),
            })
        return chunks

    if nombre == "consultar_sensores_plc":
        tipo = args.get("tipo") or "resumen"
        tag_resuelto = _resolver_tag_por_pregunta(pregunta_original) if tipo in ("resumen",
"recientes")
else None
        tag_plc = tag_resuelto or (args.get("tag_plc") or None)
        _stat_detectado = _resolver_stat_por_pregunta(pregunta_original)
        if _stat_detectado and _es_pregunta_de_valor(pregunta_original):
            tipo = "resumen"
        limit = args.get("limit") or 20
        try:
            limit = int(limit)
        except (TypeError, ValueError):
            limit = 20

        sesion_resuelta = _resolver_sesion_por_pregunta(pregunta_original)



        sesion_id_bruto = sesion_resuelta if sesion_resuelta is not None else (args.get("sesion_id")
or
None)
        sesion_id = None
        if sesion_id_bruto is not None:
            try:
                sesion_id = int(sesion_id_bruto)
            except (TypeError, ValueError):
                return {"texto_para_modelo": _formatear_plc("resumen", {
                    "error": (
                        f"'{sesion_id_bruto}' no es un ID de sesion valido (debe ser "
                        "numerico). Esa sesion no existe; usa listar_sesiones para "
                        "ver los IDs reales."
                    )
                }), "respuesta_determinista": None}

        if tipo == "listar_sesiones":
            filas = plc_store.listar_sesiones(limit=limit)
            return {"texto_para_modelo": _formatear_plc(tipo, filas), "respuesta_determinista": None}

        if sesion_id is None and _es_pregunta_tiempo_real(pregunta_original):
            activa = plc_store.sesion_activa()
            if activa is None:
                return {
                    "texto_para_modelo": _formatear_plc(tipo, {
                        "error": "no hay ninguna sesion de adquisicion activa en este momento"
                    }),
                    "respuesta_determinista": (
                        "No hay ninguna sesión de captura activa en este momento, "
                        "por lo que no hay datos en tiempo real que reportar."
                    ),
                }
            sesion_id = activa["id"]

        if tipo != "recientes":
```

```
            if sesion_id is None:
                activa = plc_store.sesion_activa()
                sesion_id = activa["id"] if activa else None
            if sesion_id is None:
                return {"texto_para_modelo": _formatear_plc("resumen", {"error": "no hay sesion_id y
no hay
ninguna sesion activa"}), "respuesta_determinista": None}
            filas = plc_store.resumen_sesion(sesion_id, tag_plc=tag_plc)
            determinista = None
            if not filas:
                sufijo_tag = f" para {tag_plc}" if tag_plc else ""
                determinista = (
                    f"No hay lecturas registradas en la base de datos real de la "
                    f"planta para la sesión {sesion_id}{sufijo_tag}."
                )
            elif tag_resuelto and len(filas) == 1 and _es_pregunta_de_valor(pregunta_original):
                stat = _resolver_stat_por_pregunta(pregunta_original)
                determinista = _respuesta_determinista_plc("resumen", filas[0]["tag_plc"], stat,
filas,
pregunta_original)
            elif (
                not tag_resuelto
                and _resolver_stat_por_pregunta(pregunta_original) == "conteo"
                and filas
            ):
                total = sum(f["num_lecturas"] for f in filas)
                determinista = f"La sesión {sesion_id} tiene {total} lecturas registradas en total
(todas
las variables)."
            elif (
                not tag_resuelto
                and len(filas) > 1
                and _resolver_stat_por_pregunta(pregunta_original) in ("maximo", "minimo",
"promedio")
                and _es_pregunta_de_valor(pregunta_original)
            ):
                stat = _resolver_stat_por_pregunta(pregunta_original)
                etiqueta = {"maximo": "máximo", "minimo": "mínimo", "promedio": "promedio"}[stat]
                partes = [
                    f"{f['tag_plc']}={round(f[stat], 4)} {f['unidad'] or ''}".strip()



                    for f in filas
                ]
                determinista = (
                    f"La sesión {sesion_id} tiene varios sensores con distinta unidad; "
                    f"el {etiqueta} por sensor fue: " + ", ".join(partes) + "."
                )
            return {"texto_para_modelo": _formatear_plc("resumen", filas), "respuesta_determinista":
determinista}

        if tag_plc and _ULTIMA_LECTURA_RE.search((pregunta_original or "").lower()):
            limit = 1


        pide_ultima = _ULTIMA_LECTURA_RE.search((pregunta_original or "").lower())
        if not tag_plc and pide_ultima:
            categoria = _resolver_categoria_por_pregunta(pregunta_original)
            if categoria:
                candidatas = plc_store.lecturas_recientes(sesion_id=sesion_id, limit=100)
                candidatas = [f for f in candidatas if f["tag_plc"] in _CATEGORIAS_TAGS[categoria]]
                if candidatas:
                    filas = candidatas[:1]
                    determinista = _respuesta_determinista_plc(
                        "recientes", candidatas[0]["tag_plc"], None, filas, pregunta_original
                    )
                    return {"texto_para_modelo": _formatear_plc("recientes", filas),
"respuesta_determinista": determinista}


        if _PRIMER_VALOR_RE.search((pregunta_original or "").lower()) and sesion_id is not None:
            filas_primeras = plc_store.lecturas_recientes(
                tag_plc=tag_plc, sesion_id=sesion_id, limit=1, orden="ASC"
            )
            if filas_primeras:
                texto = _respuesta_determinista_plc(
                    "recientes", filas_primeras[0]["tag_plc"], None, filas_primeras,
pregunta_original
                )
                texto = texto.replace("último valor", "primer valor registrado") if texto else None
                return {"texto_para_modelo": texto or "No hay lecturas registradas para esos
filtros.",
```

```
"respuesta_determinista": texto}

        filas = plc_store.lecturas_recientes(tag_plc=tag_plc, sesion_id=sesion_id, limit=limit)
        determinista = None
        if not filas:
            sufijo_tag = f" para {tag_plc}" if tag_plc else ""
            sufijo_sesion = f" en la sesión {sesion_id}" if sesion_id is not None else ""
            determinista = (
                f"No hay lecturas registradas en la base de datos real de la "
                f"planta{sufijo_tag}{sufijo_sesion}."
            )
        elif tag_resuelto and filas and _es_pregunta_de_valor(pregunta_original):
            determinista = _respuesta_determinista_plc("recientes", tag_resuelto, None, filas,
pregunta_original)
        return {"texto_para_modelo": _formatear_plc("recientes", filas), "respuesta_determinista":
determinista}

    return {"error": f"herramienta desconocida: {nombre}"}



_INTERLOCK_ID_RE = re.compile(r"\bIL-0?(\d)\b", re.IGNORECASE)



def _evaluar_operador(operador: str, valor: float, umbral: float) -> bool | None:
    ops = {
        "==": lambda v, u: round(v) == u,
        ">": lambda v, u: v > u,
        "<": lambda v, u: v < u,
        ">=": lambda v, u: v >= u,
        "<=": lambda v, u: v <= u,
    }
    fn = ops.get(operador)
    return fn(valor, umbral) if fn else None



def _respuesta_determinista_interlock(pregunta: str) -> str | None:
    if not pregunta:
        return None
    m = _INTERLOCK_ID_RE.search(pregunta)
    if not m:
        return None
    il_id = f"IL-{int(m.group(1)):02d}"
    regla = INTERLOCKS_PLC.get(il_id)
    if not regla or not regla.get("capturado_en_bd"):
        return None
    tag = regla["condicion_tag"]
    sesion_id = _resolver_sesion_por_pregunta(pregunta)
    filas = plc_store.lecturas_recientes(tag_plc=tag, sesion_id=sesion_id, limit=1)
    if not filas or filas[0].get("valor_ingenieria") is None:
        return None
    valor = filas[0]["valor_ingenieria"]
    cumple = _evaluar_operador(regla["operador"], valor, regla["umbral"])
    if cumple is None:
        return None
    etiqueta = etiqueta_digital(tag, valor)
    valor_txt = f"{etiqueta} ({round(valor, 4)})" if etiqueta else str(round(valor, 4))
    veredicto = "Sí" if cumple else "No"
    return (
        f"{veredicto}: el último valor real de {tag} es {valor_txt}, "
        f"{'lo cual SÍ cumple' if cumple else 'lo cual NO cumple'} la "
        f"condición de {il_id} ({tag} {regla['operador']} {regla['umbral']} "
        f"→ {regla['accion']}: {regla['descripcion']}) (sesión "
        f"{filas[0]['sesion_id']}, {filas[0]['timestamp']})."
    )



_SESIONES_CATALOGO_RE = re.compile(
    r"qu[eé]\s+sesiones|cu[aá]ntas\s+sesiones|sesi[oó]n\s+m[aá]s\s+reciente|"
    r"sesi[oó]n\s+con\s+m[aá]s\s+lecturas|"
    r"sesi[oó]n(?:es)?\s+(?:de\s+captura\s+)?activa|hay\s+alguna\s+sesi[oó]n"
)
_SESIONES_CATALOGO_EXCLUSION_RE = re.compile(
    r"cu[aá]ntas\s+lectura|primer\s+valor|promedio|m[aá]ximo|m[ií]nimo|rango"
)
```

```
def _respuesta_determinista_sesiones(pregunta: str) -> str | None:
    if not pregunta:
        return None
    texto = pregunta.lower()
    if not _SESIONES_CATALOGO_RE.search(texto):
        return None
    if _SESIONES_CATALOGO_EXCLUSION_RE.search(texto):
        return None
    sesiones = plc_store.listar_sesiones(limit=50)
    if not sesiones:
        return "No hay sesiones de captura registradas."


    if re.search(r"activa", texto):
        activa = next((s for s in sesiones if s["estado"] == "activa"), None)
        if activa:
            return f"Sí, la sesión {activa['id']} está activa (iniciada {activa['iniciada_en']})."
        return "No, ninguna sesión de captura está activa en este momento."


    if re.search(r"cu[aá]ntas\s+sesiones", texto):
        return f"Hay {len(sesiones)} sesiones de captura registradas en total."


    if re.search(r"qu[eé]\s+sesiones", texto):
        lineas = [
            f"sesión {s['id']} ({s['iniciada_en']}, {s['num_lecturas']} lecturas, {s['estado']})"
            for s in sesiones
        ]
        return "Sesiones de captura existentes: " + "; ".join(lineas) + "."



    if re.search(r"m[aá]s\s+reciente", texto):
        mas_reciente = max(sesiones, key=lambda s: s["iniciada_en"])
        return (
            f"La sesión más reciente es la {mas_reciente['id']}, iniciada "
            f"{mas_reciente['iniciada_en']} ({mas_reciente['num_lecturas']} lecturas)."
        )


    if re.search(r"m[aá]s\s+lecturas", texto):
        con_mas = max(sesiones, key=lambda s: s["num_lecturas"])
        return f"La sesión con más lecturas es la {con_mas['id']}, con {con_mas['num_lecturas']}
lecturas."


    return None



def _respuesta_determinista_tag_inexistente(pregunta: str) -> str | None:
    if not pregunta:
        return None
    faltantes = _tags_no_encontrados(pregunta)
    if len(faltantes) != 1:
        return None
    pregunta_sin_parentesis = re.sub(r"\([^)]*\)", "", pregunta)
    if _tags_mencionados(pregunta_sin_parentesis):
        return None
    return (
        f"El tag {faltantes[0]} no existe en el diccionario de tags de la "
        f"PPA. No debe confundirse con otros tags de nombre parecido."
    )



def _respuesta_determinista_previa(pregunta: str) -> tuple[str, list[str]] | None:
    respuesta_tag = _respuesta_determinista_tag_inexistente(pregunta)
    if respuesta_tag is not None:
        return respuesta_tag, ["buscar_documentacion"]


    respuesta_alarma = _respuesta_determinista_alarma(pregunta)
    if respuesta_alarma is not None:
        return respuesta_alarma, ["buscar_documentacion"]


    respuesta_interlock = _respuesta_determinista_interlock(pregunta)
    if respuesta_interlock is not None:
        return respuesta_interlock, ["buscar_documentacion", "consultar_sensores_plc"]


    respuesta_sesiones = _respuesta_determinista_sesiones(pregunta)
```

```
    if respuesta_sesiones is not None:
        return respuesta_sesiones, ["consultar_sensores_plc"]

    resultado_lin = _respuesta_determinista_linealizacion(pregunta)
    if resultado_lin is not None:
        texto, uso_bd = resultado_lin
        return texto, (["consultar_sensores_plc"] if uso_bd else ["buscar_documentacion"])

    return None


def responder_con_herramientas(pregunta: str) -> dict:
    t0 = time.perf_counter()

    atajo = _respuesta_determinista_previa(pregunta)
    if atajo is not None:
        texto, herramientas = atajo
        latencia_ms = (time.perf_counter() - t0) * 1000
        return {
            "respuesta": texto,
            "modelo_usado": None,
            "tokens_generados": 0,
            "chunks": [],
            "herramientas_invocadas": herramientas,
            "respuesta_determinista": True,
            "latencia_recuperacion_ms": 0.0,
            "latencia_generacion_ms": latencia_ms,



        }

    mensajes = [
        {"role": "system", "content": SYSTEM_PROMPT_HERRAMIENTAS},
        {"role": "user", "content": pregunta},
    ]

    resultado = chat(mensajes, tools=TOOLS)
    chunks_usados = []
    herramientas_invocadas = []
    respuestas_deterministas = []
    es_determinista = False

    if resultado["tool_calls"]:
        herramientas_invocadas = [tc["function"]["name"] for tc in resultado["tool_calls"]]
        mensajes.append({
            "role": "assistant",
            "content": resultado["contenido"],
            "tool_calls": resultado["tool_calls"],
        })
        for tc in resultado["tool_calls"]:
            nombre = tc["function"]["name"]
            args = tc["function"]["arguments"]
            salida = _ejecutar_tool(nombre, args, pregunta)
            if nombre == "buscar_documentacion":
                chunks_usados.extend(salida)
                salida_modelo = [
                    {**c, "texto": c["texto"][:TEXTO_MAX_CHARS]}
                    for c in salida[:CHUNKS_MODELO_MAX]
                ]
            else:
                if salida.get("respuesta_determinista"):
                    respuestas_deterministas.append(salida["respuesta_determinista"])
                salida_modelo = salida["texto_para_modelo"]
            mensajes.append({
                "role": "tool",
                "content": salida_modelo if isinstance(salida_modelo, str)
                else json.dumps(salida_modelo, default=str, ensure_ascii=False),
            })

        if (
            "buscar_documentacion" not in herramientas_invocadas
            and len(respuestas_deterministas) == 1
        ):
            resultado["contenido"] = respuestas_deterministas[0]
            es_determinista = True
        else:
```

```
            resultado = chat(mensajes)
            if _TOOL_JSON_LEAK_RE.search(resultado["contenido"]):
                reintento = chat(mensajes)
                if not _TOOL_JSON_LEAK_RE.search(reintento["contenido"]):
                    resultado = reintento
                else:
                    resultado["contenido"] = _TOOL_JSON_LEAK_RE.sub("",
resultado["contenido"]).strip()
                    if not resultado["contenido"]:
                        resultado["contenido"] = (
                            "No pude generar una respuesta clara a partir de los "
                            "datos obtenidos; intenta reformular la pregunta."
                        )

    if not es_determinista:
        resultado["contenido"] = _limpia_citas_inventadas(resultado["contenido"])
        if not resultado["contenido"]:
            resultado["contenido"] = "no encuentro esa informacion en la documentacion"
    latencia_ms = (time.perf_counter() - t0) * 1000

    return {
        "respuesta": resultado["contenido"],
        "modelo_usado": resultado["modelo"],
        "tokens_generados": resultado["tokens_generados"],



        "chunks": chunks_usados,
        "herramientas_invocadas": herramientas_invocadas,
        "respuesta_determinista": es_determinista,
        "latencia_recuperacion_ms": 0.0,
        "latencia_generacion_ms": latencia_ms,
    }



def responder_con_herramientas_stream(pregunta: str):
    t0 = time.perf_counter()

    atajo = _respuesta_determinista_previa(pregunta)
    if atajo is not None:
        texto, _herramientas = atajo
        yield {"tipo": "chunk", "texto": texto}
        yield {
            "tipo": "fin",
            "fuentes": [],
            "modelo_usado": None,
            "latencia_ms": (time.perf_counter() - t0) * 1000,
        }
        return

    mensajes = [
        {"role": "system", "content": SYSTEM_PROMPT_HERRAMIENTAS},
        {"role": "user", "content": pregunta},
    ]

    resultado = chat(mensajes, tools=TOOLS)
    chunks_usados = []
    modelo_usado = resultado["modelo"]

    if resultado["tool_calls"]:
        mensajes.append({
            "role": "assistant",
            "content": resultado["contenido"],
            "tool_calls": resultado["tool_calls"],
        })
        for tc in resultado["tool_calls"]:
            nombre = tc["function"]["name"]
            args = tc["function"]["arguments"]
            salida = _ejecutar_tool(nombre, args, pregunta)
            if nombre == "buscar_documentacion":
                chunks_usados.extend(salida)
                salida_modelo = [
                    {**c, "texto": c["texto"][:TEXTO_MAX_CHARS]}
                    for c in salida[:CHUNKS_MODELO_MAX]
                ]
            else:
                salida_modelo = salida["texto_para_modelo"]
            mensajes.append({
```

```
                "role": "tool",
                "content": salida_modelo if isinstance(salida_modelo, str)
                else json.dumps(salida_modelo, default=str, ensure_ascii=False),
            })
        for tipo, valor in chat_stream(mensajes):
            if tipo == "delta":
                yield {"tipo": "chunk", "texto": valor}
            elif tipo == "fin":
                modelo_usado = valor["modelo"]
    else:
        yield {"tipo": "chunk", "texto": resultado["contenido"]}

    latencia_ms = (time.perf_counter() - t0) * 1000
    yield {
        "tipo": "fin",
        "fuentes": [
            {"documento": c["documento"], "pagina": c["pagina"], "score": c["score"]}
            for c in chunks_usados
        ],



        "modelo_usado": modelo_usado,
        "latencia_ms": latencia_ms,
    }
```

# ANEXO J. CÓDIGO FUENTE: ingest/chunker.py

Este anexo contiene el código fuente completo del archivo ingest/chunker.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import re

from config import CHARS_PER_TOKEN, CHUNK_OVERLAP_TOKENS, CHUNK_TOKENS


def chunk_text(text: str, chunk_tokens: int = CHUNK_TOKENS,
               overlap_tokens: int = CHUNK_OVERLAP_TOKENS,
               chars_per_token: int = CHARS_PER_TOKEN):
    chunk_size = chunk_tokens * chars_per_token
    overlap = overlap_tokens * chars_per_token
    words = text.split()

    chunks = []
    current = []
    current_len = 0
    for word in words:
        current.append(word)
        current_len += len(word) + 1
        if current_len >= chunk_size:
            chunks.append(" ".join(current))
            overlap_words = []
            overlap_len = 0
            for w in reversed(current):
                overlap_len += len(w) + 1
                overlap_words.insert(0, w)
                if overlap_len >= overlap:
                    break
            current = overlap_words
            current_len = overlap_len

    if current:
        chunks.append(" ".join(current))

    return [c for c in chunks if c.strip()]


_FILA_RE = re.compile(r"^\s*\|(.+)\|\s*$")
_SEP_RE = re.compile(r"^\s*\|[\s:|-]+\|\s*$")


def _parse_tabla(lineas: list[str], inicio: int):
    if inicio + 1 >= len(lineas):
        return None, None, inicio
    m_header = _FILA_RE.match(lineas[inicio])
    if not m_header or not _SEP_RE.match(lineas[inicio + 1]):
        return None, None, inicio
    headers = [c.strip() for c in m_header.group(1).split("|")]
    filas = []
    idx = inicio + 2
    while idx < len(lineas):
        m = _FILA_RE.match(lineas[idx])



        if not m:
            break
        celdas = [c.strip() for c in m.group(1).split("|")]
        if len(celdas) == len(headers):
            filas.append(celdas)
        idx += 1
    return headers, filas, idx
```

```
def _fila_a_oracion(headers: list[str], celdas: list[str]) -> str:
    partes = [f"{h}: {v}" for h, v in zip(headers, celdas) if v and h]
    if partes:
        return "; ".join(partes)
    return "; ".join(v for v in celdas if v)



def chunk_markdown(text: str, chunk_tokens: int = CHUNK_TOKENS,
                   overlap_tokens: int = CHUNK_OVERLAP_TOKENS,
                   chars_per_token: int = CHARS_PER_TOKEN) -> list[str]:
    lineas = text.split("\n")
    chunks: list[str] = []
    buffer_prosa: list[str] = []


    def flush_prosa():
        texto_prosa = "\n".join(buffer_prosa).strip()
        if texto_prosa:
            chunks.extend(chunk_text(texto_prosa, chunk_tokens, overlap_tokens, chars_per_token))
        buffer_prosa.clear()


    idx = 0
    while idx < len(lineas):
        headers, filas, siguiente = _parse_tabla(lineas, idx)
        if headers and filas:
            flush_prosa()
            for celdas in filas:
                oracion = _fila_a_oracion(headers, celdas)
                if oracion.strip():
                    chunks.append(oracion)
            idx = siguiente
        else:
            buffer_prosa.append(lineas[idx])
            idx += 1
    flush_prosa()


    return chunks
```

# ANEXO K. CÓDIGO FUENTE: ingest/config.py

Este anexo contiene el código fuente completo del archivo ingest/config.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import os
from pathlib import Path

from dotenv import load_dotenv

load_dotenv(Path(__file__).parent / ".env")

MANUALES_DIR = Path(os.environ.get("MANUALES_DIR", "/opt/ppa -rag/data/manuales"))



CHUNK_TOKENS = int(os.environ.get("CHUNK_TOKENS", 800))
CHUNK_OVERLAP_TOKENS = int(os.environ.get("CHUNK_OVERLAP_TOKENS", 120))
CHARS_PER_TOKEN = int(os.environ.get("CHARS_PER_TOKEN", 4))
TOP_K = int(os.environ.get("TOP_K", 5))

QDRANT_URL = os.environ.get("QDRANT_URL", "http://127.0.0.1:6333")
QDRANT_COLLECTION = os.environ.get("QDRANT_COLLECTION", "ppa_manuales")

OLLAMA_HOST = os.environ.get("OLLAMA_HOST", "http://127.0.0.1:11434")
OLLAMA_EMBED_MODEL = os.environ.get("OLLAMA_EMBED_MODEL", "nomic-embed-text")
EMBED_DIM = int(os.environ.get("EMBED_DIM", 768))
```

# ANEXO L. CÓDIGO FUENTE: ingest/embeddings.py

Este anexo contiene el código fuente completo del archivo ingest/embeddings.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import requests

from config import OLLAMA_EMBED_MODEL, OLLAMA_HOST


def get_embedding(text: str) -> list[float]:
    resp = requests.post(
        f"{OLLAMA_HOST}/api/embeddings",
        json={"model": OLLAMA_EMBED_MODEL, "prompt": text},
        timeout=120,
    )
    resp.raise_for_status()
    return resp.json()["embedding"]


def embed_or_split(text: str) -> list[tuple[str, list[float]]]:
    try:
        return [(text, get_embedding(text))]
    except requests.exceptions.HTTPError as e:
        if e.response is None or e.response.status_code != 500:
            raise
        palabras = text.split()
        if len(palabras) <= 20:
            raise
        mitad = len(palabras) // 2
        izquierda = " ".join(palabras[:mitad])
        derecha = " ".join(palabras[mitad:])
        return embed_or_split(izquierda) + embed_or_split(derecha)
```

# ANEXO M. CÓDIGO FUENTE: ingest/ingest.py

Este anexo contiene el código fuente completo del archivo ingest/ingest.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import sys
from pathlib import Path


from config import MANUALES_DIR, TOP_K, CHUNK_TOKENS
from loaders import LOADERS, load_document
from chunker import chunk_markdown, chunk_text
from embeddings import embed_or_split
from qdrant_store import chunk_hash, count_points, ensure_collection, exists, point_id, upsert_chunk



def iter_documents(directory: Path):
    for path in sorted(directory.rglob("*")):
        if path.is_file() and path.suffix.lower() in LOADERS:
            yield path



def ingest_document(path: Path) -> tuple[int, int]:
    nuevos = 0
    existentes = 0
    pages = load_document(path)
    chunker = chunk_markdown if path.suffix.lower() == ".md" else chunk_text
    for pagina, texto in pages:
        for chunk in chunker(texto):
            h = chunk_hash(chunk)
            pid = point_id(path.name, pagina, h)
            if exists(pid):
                existentes += 1
                continue
            for subtexto, vector in embed_or_split(chunk):
                sub_h = h if subtexto == chunk else chunk_hash(subtexto)
                sub_pid = pid if subtexto == chunk else point_id(path.name, pagina, sub_h)
                if sub_pid != pid and exists(sub_pid):
                    existentes += 1
                    continue
                upsert_chunk(sub_pid, vector, path.name, pagina, sub_h, subtexto)
                nuevos += 1
    return nuevos, existentes



def main():
    ensure_collection()

    if not MANUALES_DIR.exists():
        print(f"No existe el directorio de manuales: {MANUALES_DIR}", file=sys.stderr)
        sys.exit(1)

    total_nuevos = 0
    total_existentes = 0
    documentos = list(iter_documents(MANUALES_DIR))

    if not documentos:
        print(f"No hay documentos soportados (.pdf/.docx/.xlsx) en {MANUALES_DIR}")
        return

    for path in documentos:
        nuevos, existentes = ingest_document(path)
        total_nuevos += nuevos
        total_existentes += existentes
        print(f"{path.name}: {nuevos} chunks nuevos, {existentes} ya existentes")

    print(f"\nTotal: {total_nuevos} chunks nuevos, {total_existentes} ya existentes")
    print(f"Puntos totales en la coleccion: {count_points()}")
    print(f"(config: chunk_tokens={CHUNK_TOKENS}, top_k={TOP_K} -> "
```

```
          f"presupuesto contexto recuperado ~= {TOP_K * CHUNK_TOKENS} tokens)")


if __name__ == "__main__":
    main()
```

# ANEXO N. CÓDIGO FUENTE: ingest/loaders.py

Este anexo contiene el código fuente completo del archivo ingest/loaders.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import re
import unicodedata
from pathlib import Path


from openpyxl import load_workbook
from pypdf import PdfReader
from docx import Document


_HEADING_RE = re.compile(r"^(#{1,6})\s+(.*\S)\s*$")
_ACTIVIDAD_RE = re.compile(r"ACTIVIDAD\s*(\d+)\s*[:.]?\s*\**\s*([^|\n]{0,100})", re.IGNORECASE)


_SUBSECCION_GENERICA = {
    "conclusion", "conclusiones", "resumen", "detalles", "detalle",
    "participantes", "hallazgos", "link video", "link videos",
    "objetivo de la practica",
    "material necesario y requisitos para el desarrollo de la actividad practica",
    "esquema grafico de la actividad practica",
    "requerimientos previos", "desarrollo de la actividad practica",
}


_CAPITULOS_INDICE = {"contenido", "tabla de figuras"}



def _clean_label(s: str) -> str:
    return re.sub(r"\*+", "", s).strip(" :.-")



def _normalizar(s: str) -> str:
    sin_acentos = unicodedata.normalize("NFKD", s).encode("ascii", "ignore").decode("ascii")
    return sin_acentos.strip(" :.-").lower()



def load_pdf(path: Path):
    reader = PdfReader(str(path))
    pages = []
    for i, page in enumerate(reader.pages, start=1):
        text = page.extract_text() or ""
        if text.strip():
            pages.append((str(i), text))
    return pages



def load_docx(path: Path):
    doc = Document(str(path))
    text = "\n".join(p.text for p in doc.paragraphs if p.text.strip())
    if not text.strip():
        return []
    return [("1", text)]



def load_xlsx(path: Path):
    wb = load_workbook(str(path), read_only=True, data_only=True)
    pages = []
    for sheet in wb.worksheets:
        rows_text = []
        for row in sheet.iter_rows(values_only=True):
            cells = [str(c) for c in row if c is not None]
            if cells:
```

```
                rows_text.append(" | ".join(cells))
        text = "\n".join(rows_text)
        if text.strip():
            pages.append((sheet.title, text))
    return pages



def load_md(path: Path):
    text = path.read_text(encoding="utf-8")
    blocks = []
    current_chapter = None
    current_section = None
    seen_actividad = False
    buf: list[str] = []


    def label() -> str:
        parts = [p for p in (current_chapter, current_section) if p]
        return " - ".join(dict.fromkeys(parts)) if parts else path.stem


    def flush():
        block_text = "\n".join(buf).strip()
        if block_text:
            blocks.append((label(), block_text))
        buf.clear()


    for line in text.split("\n"):
        m_actividad = _ACTIVIDAD_RE.search(line)
        m_heading = _HEADING_RE.match(line)


        if m_actividad and _normalizar(current_chapter or "") not in _CAPITULOS_INDICE:
            flush()
            numero, desc = m_actividad.group(1), m_actividad.group(2)
            current_chapter = _clean_label(f"Actividad {numero}: {desc}")
            current_section = None
            seen_actividad = True
        elif m_heading:
            flush()
            nivel = len(m_heading.group(1))
            texto_encabezado = _clean_label(m_heading.group(2))
            es_generico = _normalizar(texto_encabezado) in _SUBSECCION_GENERICA
            if nivel == 1 and not seen_actividad and not es_generico:
                current_chapter = texto_encabezado
                current_section = None
            else:
                current_section = texto_encabezado


        buf.append(line)
    flush()


    return blocks



LOADERS = {
    ".pdf": load_pdf,
    ".docx": load_docx,
    ".xlsx": load_xlsx,
    ".md": load_md,
}



def load_document(path: Path):
    loader = LOADERS.get(path.suffix.lower())
    if loader is None:
        raise ValueError(f"Formato no soportado: {path.suffix}")
    return loader(path)
```

# ANEXO O. CÓDIGO FUENTE: ingest/make_dummy_pdf.py

Este anexo contiene el código fuente completo del archivo ingest/make_dummy_pdf.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
from pathlib import Path

from fpdf import FPDF

from config import MANUALES_DIR

PAGINAS = [
    "Manual de mantenimiento - Bomba centrifuga P-101.\n\n"
    "Rango de operacion nominal: 1450 a 1480 RPM. Presion de descarga esperada "
    "entre 4.2 y 4.8 bar. Si la presion de descarga cae por debajo de 3.5 bar, "
    "verificar la valvula de succion y el estado del sello mecanico.",


    "Temperatura de cojinetes.\n\n"
    "La temperatura del cojinete de la bomba P-101 no debe superar los 75 "
    "grados Celsius en operacion continua. Por encima de 85 grados Celsius se "
    "debe detener el equipo de inmediato para evitar dano al eje y rodamientos.",


    "Plan de mantenimiento preventivo.\n\n"
    "Lubricacion cada 500 horas de operacion. Cambio de sello mecanico cada "
    "8000 horas. Inspeccion visual de fugas cada turno. Registrar cualquier "
    "ruido anormal o vibracion excesiva en la bitacora de planta.",


    "Corriente nominal del motor.\n\n"
    "La corriente nominal del motor de la bomba P -101 en operacion normal es "
    "de 8 a 12 amperios. Corrientes sostenidas por encima de 15 amperios "
    "indican una posible sobrecarga mecanica o falla del motor y deben "
    "generar una alerta inmediata para inspeccion.",
]



def build_dummy_pdf(output_path: Path):
    pdf = FPDF()
    for texto in PAGINAS:
        pdf.add_page()
        pdf.set_font("Helvetica", size=12)
        pdf.multi_cell(0, 8, texto)
    pdf.output(str(output_path))



if __name__ == "__main__":
    MANUALES_DIR.mkdir(parents=True, exist_ok=True)
    out = MANUALES_DIR / "dummy_ppa.pdf"
    build_dummy_pdf(out)
    print(f"PDF de prueba creado en {out}")
```

# ANEXO P. CÓDIGO FUENTE: ingest/qdrant_store.py

Este anexo contiene el código fuente completo del archivo ingest/qdrant_store.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import hashlib


import uuid

from qdrant_client import QdrantClient
from qdrant_client.models import Distance, PointStruct, VectorParams

from config import EMBED_DIM, QDRANT_COLLECTION, QDRANT_URL

_NAMESPACE = uuid.UUID("6f6d0f3a-6b9e-4a9e-9c1a-a1b2c3d4e5f6")

client = QdrantClient(url=QDRANT_URL)


def ensure_collection():
    existing = [c.name for c in client.get_collections().collections]
    if QDRANT_COLLECTION not in existing:
        client.create_collection(
            collection_name=QDRANT_COLLECTION,
            vectors_config=VectorParams(size=EMBED_DIM, distance=Distance.COSINE),
        )


def chunk_hash(text: str) -> str:
    return hashlib.sha256(text.encode("utf-8")).hexdigest()


def point_id(documento: str, pagina: str, hash_: str) -> str:
    return str(uuid.uuid5(_NAMESPACE, f"{documento}:{pagina}:{hash_}"))


def exists(point_id_: str) -> bool:
    result = client.retrieve(collection_name=QDRANT_COLLECTION, ids=[point_id_])
    return len(result) > 0


def upsert_chunk(point_id_: str, vector: list[float], documento: str, pagina: str, hash_: str, texto:
str):
    client.upsert(
        collection_name=QDRANT_COLLECTION,
        points=[
            PointStruct(
                id=point_id_,
                vector=vector,
                payload={
                    "documento": documento,
                    "pagina": pagina,
                    "chunk_hash": hash_,
                    "texto": texto,
                },
            )
        ],
    )


def count_points() -> int:
    return client.count(collection_name=QDRANT_COLLECTION, exact=True).count
```

# ANEXO Q. CÓDIGO FUENTE: ui/app.py

Este anexo contiene el código fuente completo del archivo ui/app.py, tal como se encuentra en producción. El capítulo 8 describe su lógica mediante diagramas de flujo y pseudocódigo.

```
import base64
import json
import os
import time
from pathlib import Path


import pandas as pd
import requests
import streamlit as st
from dotenv import load_dotenv

load_dotenv(Path(__file__).parent / ".env")

API_URL = os.environ.get("API_URL", "http://127.0.0.1:8000")

st.set_page_config(page_title="PPA-RAG", layout="wide")



def set_fondo(ruta_imagen: Path) -> None:
    b64 = base64.b64encode(ruta_imagen.read_bytes()).decode()
    st.markdown(
        f"""
        <style>
        [data-testid="stAppViewContainer"] {{
            background-image: url("{b64}");
            background-size: 40%;
            background-position: center;
            background-repeat: no-repeat;
            background-attachment: fixed;
        }}
        [data-testid="stHeader"] {{
            background-color: rgba(0, 0, 0, 0);
        }}
        </style>
        """,
        unsafe_allow_html=True,
    )



def _obtener_sesiones() -> list:
    try:
        r = requests.get(f"{API_URL}/plc/sesiones", timeout=10)
        r.raise_for_status()
        return r.json().get("sesiones", [])
    except requests.RequestException as e:
        st.error(f"No se pudo contactar la API: {e}")
        return []



def _etiqueta_sesion(s: dict) -> str:
    return f"#{s['id']} — {s['estado']} — {s['num_lecturas']} lecturas ({s['iniciada_en']})"



set_fondo(Path(__file__).parent / "assets" / "15_transparente.png")

st.title("PPA-RAG — Planta de Procesos Análogos")

with st.sidebar:
    st.subheader("Adquisición de datos del PLC")
    try:
        r = requests.get(f"{API_URL}/plc/sesiones/estado", timeout=10)
```

```
        r.raise_for_status()
        estado = r.json()
        if estado.get("activa"):
            st.success(f"Sesión activa (#{estado['id']}) — {estado['num_lecturas']} lecturas")
            if st.button("Detener sesión"):
                requests.post(f"{API_URL}/plc/sesiones/detener", timeout=10)
                st.rerun()
        else:
            st.info("Sin sesión activa")



            if st.button("Iniciar sesión de adquisición", type="primary"):
                requests.post(f"{API_URL}/plc/sesiones/iniciar", timeout=10)
                st.rerun()
    except requests.RequestException as e:
        st.error(f"No se pudo contactar la API: {e}")


    st.divider()
    st.subheader("Agregar documento (paper)")
    archivo = st.file_uploader("Archivo (.md)", type=["md"])
    if archivo is not None and st.button("Subir e indexar"):
        with st.spinner("Guardando e indexando (puede tardar varios minutos según el tamaño)..."):
            try:
                files = {"archivo": (archivo.name, archivo.getvalue())}
                r = requests.post(f"{API_URL}/documentos/subir", files=files, timeout=300)
                if r.status_code == 409:
                    st.warning(r.json().get("detail"))
                else:
                    r.raise_for_status()
                    data = r.json()
                    st.success(f"'{data['guardado']}' indexado correctamente.")
                    st.code(data["salida_ingesta"])
            except requests.RequestException as e:
                st.error(f"No se pudo subir el documento: {e}")

tab_chat, tab_plc, tab_alertas = st.tabs(
    ["Consultas (RAG)", "Datos reales del PLC", "Alertas de mantenimiento"]
)



with tab_chat:
    st.subheader("Consulta a la documentación técnica")
    st.caption("También puede responder con datos reales de sensores del PLC (histórico de sesiones
de
captura).")
    pregunta = st.text_input("Pregunta", placeholder="¿Qué sensores tiene el tanque de calentamiento
TK -
102?")
    if st.button("Consultar", type="primary") and pregunta.strip():
        st.markdown("**Respuesta:**")
        placeholder = st.empty()
        inicio = time.perf_counter()
        texto = ""
        fuentes = []
        modelo_usado = None
        error = None
        placeholder.markdown("_(0.0s) generando..._")
        try:
            with requests.post(
                f"{API_URL}/consulta/stream", json={"pregunta": pregunta}, timeout=240, stream=True
            ) as r:
                r.raise_for_status()
                for linea in r.iter_lines():
                    if not linea:
                        continue
                    evento = json.loads(linea)
                    transcurrido = time.perf_counter() - inicio
                    if evento["tipo"] == "chunk":
                        texto += evento["texto"]
                        placeholder.markdown(f"_({transcurrido:.1f}s)_\n\n{texto}▌")
                    elif evento["tipo"] == "fin":
                        fuentes = evento["fuentes"]
                        modelo_usado = evento["modelo_usado"]
                    elif evento["tipo"] == "error":
                        error = evento["detalle"]
        except requests.RequestException as e:
            error = str(e)

        total = time.perf_counter() - inicio
```

```
        if error:
            placeholder.empty()
            st.error(f"No se pudo contactar la API: {error}")
        else:



            placeholder.markdown(texto)
            st.caption(f"Modelo usado: {modelo_usado} — {total:.1f}s")
            if fuentes:
                st.markdown("**Fuentes citadas:**")
                for f in fuentes:
                    st.write(f"- {f['documento']}, página {f['pagina']} (score={f['score']:.3f})")



with tab_plc:
    st.subheader("Datos reales del PLC (histórico por sesión de captura)")
    sesiones = _obtener_sesiones()


    if not sesiones:
        st.info("Todavía no hay sesiones de captura registradas. Inicia una desde la barra lateral.")
    else:
        opciones = {_etiqueta_sesion(s): s["id"] for s in sesiones}
        seleccion = st.selectbox("Sesión", list(opciones.keys()))
        sesion_id = opciones[seleccion]
        try:
            r = requests.get(
                f"{API_URL}/plc/lecturas", params={"sesion_id": sesion_id, "limit": 500}, timeout=15
            )
            r.raise_for_status()
            lecturas = r.json().get("lecturas", [])
            if not lecturas:
                st.info("Esta sesión no tiene lecturas todavía.")
            else:
                df = pd.DataFrame(lecturas)
                df["timestamp"] = pd.to_datetime(df["timestamp"])
                for tag in sorted(df["tag_plc"].unique()):
                    sub = df[df["tag_plc"] == tag].sort_values("timestamp")
                    unidad = sub["unidad"].iloc[0] or ""
                    st.caption(f"{tag} ({unidad})")
                    st.line_chart(sub.set_index("timestamp")["valor_ingenieria"])
                st.markdown("**Lecturas crudas:**")
                st.dataframe(df.sort_values("timestamp", ascending=False), use_container_width=True)
        except requests.RequestException as e:
            st.error(f"No se pudo contactar la API: {e}")



with tab_alertas:
    st.subheader("Alertas del proceso (datos reales del PLC)")


    sesiones = _obtener_sesiones()


    if not sesiones:
        st.info("Todavía no hay sesiones de captura registradas. Inicia una desde la barra lateral.")
    else:
        opciones = {"Sesión activa / más reciente": None}
        opciones.update({_etiqueta_sesion(s): s["id"] for s in sesiones})
        seleccion = st.selectbox("Sesión a revisar", list(opciones.keys()), key="alertas_sesion")
        sesion_id = opciones[seleccion]


        try:
            params = {} if sesion_id is None else {"sesion_id": sesion_id}
            r = requests.get(f"{API_URL}/plc/alertas", params=params, timeout=15)
            r.raise_for_status()
            data = r.json()
            if data.get("nota"):
                st.info(data["nota"])
            else:
                alertas = data.get("alertas", [])
                proceso = [a for a in alertas if a["categoria"] == "proceso"]
                fallas = [a for a in alertas if a["categoria"] == "falla_sensor"]


                st.markdown(f"**Sesión #{data.get('sesion_id')}**")


                st.markdown("##### Condiciones de proceso")
                if not proceso:
```

```
                st.success("Sin alertas de proceso en esta sesión.")



            else:
                for a in proceso:
                    texto = f"**{a['tag_plc']}** — {a['mensaje']}. Acción: {a['accion']}"
                    if a.get("momento"):
                        texto += f" _(detectado: {a['momento']})_"
                    if a.get("aproximado"):
                        texto += " ⚠ *verificación aproximada, ver detalle abajo*"
                    if a["severidad"] == "alta":
                        st.error(texto)
                    else:
                        st.warning(texto)

            st.markdown("##### Posibles fallas de sensores/actuadores")
            if not fallas:
                st.success("Sin indicios de falla de sensor en esta sesión.")
            else:
                for a in fallas:
                    st.warning(f"**{a['tag_plc']}** — {a['mensaje']}. {a['accion']}")
    except requests.RequestException as e:
        st.error(f"No se pudo contactar la API: {e}")

st.caption(
    "Reglas basadas en PPA_05_interlocks_y_permissives.md y PPA_06_alarmas.md. "
    "Las marcadas como aproximadas usan un tag capturado del mismo equipo como sustituto "
    "de una señal que la documentación define pero que el servicio de adquisición no captura "
    "en la base de datos (ej. LSL-102 o el estado de la bomba M1)."
)
```

# ANEXO R. FUENTES PRIMARIAS DEL PROYECTO

Además de la propuesta de grado y de la literatura citada, este documento se elaboró a partir de los siguientes registros primarios producidos durante el desarrollo y la validación del sistema: la bitácora técnica completa organizada por iteraciones; un documento de proceso que describe cómo se construyó cada componente y por qué; los resultados de las Pruebas 5 y 6 sobre el banco de 145 preguntas, con calificación pregunta por pregunta; los resultados de las Pruebas 7 y 8, con la rúbrica de seis niveles y el análisis de causa por patrón sistémico; el registro completo de la prueba de consistencia; y un documento de soporte sobre el rediseño del motor de alertas, que detalla qué existía antes, por qué se modificó y cómo se verificó el resultado. Estos documentos se conservan como respaldo verificable de cada cifra y cita presentada.

# ANEXO S. BANCO DE 145 PREGUNTAS DE EVALUACIÓN

Este anexo lista el enunciado de las 145 preguntas que componen el banco descrito en la sección 7.5 y evaluado en el capítulo 9, agrupadas según los grupos G1 (documentación), G2 (base de datos) y G3 (combinadas). Se listan únicamente los enunciados; la respuesta esperada, la respuesta obtenida, el tiempo y el puntaje de cada pregunta se analizan de forma agregada en el capítulo 9 y no se repiten aquí.

**Tabla 32.** Enunciados del banco de 145 preguntas de evaluación.

| # | Grupo | Pregunta |
|---|---|---|
| 1 | 1 | ¿Qué es el sensor de Nivel alto del Tanque 0? |
| 2 | 1 | ¿Qué dirección tiene el Chiller en el PLC? |
| 3 | 1 | ¿Qué elemento corresponde a `Local:2:O.Data.3`? |
| 4 | 1 | ¿Qué tipo de sensor es el que mide la temperatura del Tanque 0? |
| 5 | 1 | ¿Cuál es el rango de operación del sensor de nivel ultrasónico? |
| 6 | 1 | ¿Qué actuador está conectado al módulo Raspberry en el proceso de calentamiento? |
| 7 | 1 | ¿Cuál es la IP del módulo Ethernet de la Bomba-Variador? |
| 8 | 1 | ¿Qué PLC gobierna la PPA? |
| 9 | 1 | ¿Cuántos actuadores y sensores tiene el Tanque de calentamiento? |
| 10 | 1 | ¿Qué válvula permite el paso del flujo del tanque 3 al tanque 1? |
| 11 | 1 | ¿Qué es TT-102? |
| 12 | 1 | ¿Qué es PE-102? |
| 13 | 1 | ¿Qué es el sensor de Nivel bajo del Tanque 0? |
| 14 | 1 | ¿Qué es la Resistencia 1? |
| 15 | 1 | ¿Qué es la Válvula Proporcional (PCV-101)? |
| 16 | 1 | ¿Cuál es la ecuación de linealización del sensor ultrasónico (nivel tanque medio)? |
| 17 | 1 | ¿Qué representan las variables m y b en la ecuación de linealización? |
| 18 | 1 | ¿Cuál es la ecuación de linealización del sensor de temperatura del tanque superior? |
| 19 | 1 | ¿Cuál es la ecuación de linealización del sensor de presión diferencial? |
| 20 | 1 | ¿Cuál es la ecuación de linealización del sensor de temperatura del tanque inferior? |
| 21 | 1 | ¿Qué unidad de entrada usa el módulo analógico antes de linealizar (valor crudo)? |
| 22 | 1 | ¿Cómo se calcula el valor en unidades de ingeniería a partir del valor crudo? |
| 23 | 1 | ¿Cuál es el valor de la pendiente (m) en la ecuación de linealización del sensor de presión diferencial? |
| 24 | 1 | ¿Cuál es el valor del intercepto (b) en la ecuación de linealización del sensor ultrasónico? |
| 25 | 1 | ¿En qué práctica/bitácora se definieron las ecuaciones de linealización de los sensores analógicos de la PPA? |
| 26 | 1 | ¿Cuál es el límite documentado para la alarma AL-01? |
| 27 | 1 | ¿Qué alarma existe para la temperatura del tanque de calentamiento? |

| # | Grupo | Pregunta |
|---|---|---|
| 28 | 1 | ¿Hay un umbral documentado para el sensor de presión diferencial? |
| 29 | 1 | ¿Hay un umbral documentado para el sensor de nivel ultrasónico? |
| 30 | 1 | ¿Por qué el sistema solo implementó la alarma AL-01 y no otras? |
| 31 | 1 | ¿A qué tanque pertenece PE-102? |
| 32 | 1 | ¿Qué es LSL-102? (no confundir con LST-102) |
| 33 | 1 | ¿Qué es LST-102? (no confundir con LSL-102) |
| 34 | 1 | ¿Qué es TT-103? |
| 35 | 1 | ¿Qué es CV-101? |
| 36 | 1 | ¿Cuál es la diferencia entre el sensor de Nivel alto y el sensor de Nivel bajo del Tanque 0? |
| 37 | 1 | ¿El sensor LE-101 pertenece al mismo tanque que el switch LSL-101? |
| 38 | 1 | ¿La Resistencia 1 y la Resistencia 2 calientan el mismo tanque? |
| 39 | 1 | ¿CV-100 y CV-104 controlan el mismo tanque? |
| 40 | 1 | ¿PE-102 y LE-101 miden la misma variable? |
| 41 | 1 | ¿Cuál es la capital de Francia? |
| 42 | 1 | ¿Cómo configuro un PLC Siemens S7-1200? |
| 43 | 1 | ¿Cuál es la receta de una torta de chocolate? |
| 44 | 1 | ¿Qué opinas del cambio climático? |
| 45 | 1 | ¿Puedes escribir un poema sobre la planta? |
| 46 | 2 | ¿Cuál fue la última lectura de presión diferencial registrada? |
| 47 | 2 | ¿Cuál fue la última lectura de temperatura registrada? |
| 48 | 2 | ¿Cuál fue la última lectura del sensor de nivel ultrasónico? |
| 49 | 2 | ¿Cuál fue el valor crudo (sin linealizar) de la última lectura de presión diferencial? |
| 50 | 2 | ¿Cuál fue el valor crudo de la última lectura de temperatura? |
| 51 | 2 | ¿A qué hora se tomó la última lectura registrada? |
| 52 | 2 | ¿Cuántas lecturas tiene la sesión más reciente? |
| 53 | 2 | ¿Cuál es el primer valor registrado en la sesión más reciente? |
| 54 | 2 | ¿Cuál fue la última lectura del chiller (CHR_100_CMD) en la sesión 15? |
| 55 | 2 | ¿Cuál fue la última lectura de la válvula inferior (CV_100_CMD) en la sesión 15? |
| 56 | 2 | ¿Cuál fue la última lectura del sensor de nivel alto (LSH_100) en la sesión 15? |
| 57 | 2 | ¿Cuál fue la última lectura del sensor de nivel bajo (LSL_100) en la sesión 15? |
| 58 | 2 | ¿Cuál fue la última lectura de la Resistencia 1 (RES_1_CMD) en la sesión 15? |
| 59 | 2 | ¿Cuál fue la última lectura de temperatura del tanque inferior (TT_100_PV) en la sesión 14? |
| 60 | 2 | ¿Cuántas lecturas de la válvula de mezcla (CV_102_CMD) hay en la sesión 14? |
| 61 | 2 | ¿Cuál fue la temperatura máxima registrada en la sesión 15? |
| 62 | 2 | ¿Cuál fue la temperatura mínima registrada en la sesión 15? |
| 63 | 2 | ¿Cuál fue el promedio de presión diferencial en la sesión 15? |
| 64 | 2 | ¿Cuál fue el valor máximo de nivel ultrasónico en la sesión 15? |
| 65 | 2 | ¿Cuál fue el rango (máximo menos mínimo) de temperatura en la sesión 15? |

| # | Grupo | Pregunta |
|---|---|---|
| 66 | 2 | ¿Cuál fue el promedio de temperatura del tanque inferior (TT_100_PV) en la sesión 15? |
| 67 | 2 | ¿Cuál fue el rango (máximo menos mínimo) de TT_100_PV en la sesión 15? |
| 68 | 2 | ¿Cuántas lecturas de presión diferencial hay en la sesión 15? |
| 69 | 2 | ¿Cuál fue el valor mínimo de presión diferencial registrado en la sesión 15? |
| 70 | 2 | ¿Cuántas lecturas totales (todas las variables) tiene la sesión 14? |
| 71 | 2 | ¿Qué sesiones de captura existen actualmente? |
| 72 | 2 | ¿Cuántas sesiones de captura hay registradas en total? |
| 73 | 2 | ¿Cuál fue la sesión con más lecturas registradas? |
| 74 | 2 | ¿Cuál fue la sesión más reciente que se capturó? |
| 75 | 2 | ¿Hay alguna sesión de captura activa en este momento? |
| 76 | 2 | ¿Cuál fue la temperatura máxima registrada? (sin especificar sesión) |
| 77 | 2 | ¿Cuál fue el promedio de presión en la sesión "prueba_99"? (sesión inexistente) |
| 78 | 2 | Dame los datos de la sesión de ayer |
| 79 | 2 | ¿Cuántas lecturas tiene la sesión [nombre de sesión que no existe]? |
| 80 | 2 | ¿Cuántas lecturas tiene la sesión "calibracion_final"? |
| 81 | 2 | ¿Cuál fue la temperatura promedio en la sesión 999? |
| 82 | 2 | ¿Qué sesión tiene el nombre "sesion_prueba_plc"? |
| 83 | 2 | ¿Cuántas lecturas tiene la sesión 20? |
| 84 | 2 | Dame el promedio de nivel ultrasónico de la última semana |
| 85 | 2 | ¿La sesión "PPA_test1" tuvo alguna lectura fuera de rango? |
| 86 | 2 | ¿En qué sesión se registró la temperatura más alta de todas? |
| 87 | 2 | ¿La sesión 15 tuvo más variabilidad de presión que la sesión 14? |
| 88 | 2 | ¿Qué sensor tiene más lecturas registradas en total: presión diferencial o temperatura? |
| 89 | 2 | ¿Qué sesión tuvo más lecturas de variables booleanas (sensores/actuadores digitales): la 14 o la 15? |
| 90 | 2 | Comparando las sesiones 14 y 15, ¿cuál capturó también los sensores analógicos (presión, temperatura, nivel)? |
| 91 | 2 | ¿Cuál fue la última lectura de vibración del motor? |
| 92 | 2 | ¿Cuál fue el consumo eléctrico total de la planta ayer? |
| 93 | 2 | ¿Cuántas veces se activó el Chiller la semana pasada? |
| 94 | 2 | ¿Cuál es la posición actual de la Válvula Proporcional? |
| 95 | 2 | ¿Cuál fue la lectura de humedad relativa registrada? |
| 96 | 3 | ¿La última lectura de temperatura supera el límite de la alarma AL-01? |
| 97 | 3 | ¿Alguna lectura de la sesión 15 superó el límite documentado de AL-01? |
| 98 | 3 | ¿Qué tan cerca está la última lectura de temperatura del límite de AL-01? |
| 99 | 3 | ¿Debería generarse una alerta con la última lectura registrada? |
| 100 | 3 | ¿Alguna lectura de la sesión 14 superó el límite documentado de AL-01? |
| 101 | 3 | Según el último valor real de RES_1_CMD en la sesión 15, ¿la Resistencia 1 estaba encendida o apagada? |

| # | Grupo | Pregunta |
|---|---|---|
| 102 | 3 | ¿El estado real de LSL_100 en la sesión 15 indica nivel bajo activo en el tanque de enfriamiento? |
| 103 | 3 | Según IL-01, si el flotador de protección (LSL-102) no está activo, ¿puede encender la Resistencia 1? |
| 104 | 3 | ¿La última lectura de RES_1_CMD es consistente con lo que exige IL-01? |
| 105 | 3 | ¿Cuál es el estado real del Chiller en la última lectura, y qué actuador documentado corresponde a esa dirección? |
| 106 | 3 | ¿La válvula de mezcla (CV_102_CMD) estaba abierta en la última lectura de la sesión 15, y qué flujo permite cuando está activa? |
| 107 | 3 | ¿El sensor de nivel alto (LSH_100) indicó riesgo de rebose en la sesión 15, y qué acción dispara IL-04 si se activa? |
| 108 | 3 | ¿La válvula superior (CV_104_CMD) estaba abierta en la última lectura, y a qué tanque admite flujo? |
| 109 | 3 | Con el último valor registrado de TT_102_PV, ¿se debería activar IL-05 (paro de la resistencia por sobretemperatura)? |
| 110 | 3 | ¿Cuántos actuadores documentados están asociados al tanque de calentamiento (TK-102), y cuántos tienen lectura real capturada en la sesión 15? |
| 111 | 3 | ¿Se ha capturado alguna lectura real de la Resistencia 2 (Raspberry_Res2) en el histórico de Postgres? |
| 112 | 3 | ¿Se ha capturado alguna lectura real de la Válvula Proporcional (Raspberry_PCV101) en el histórico de Postgres? |
| 113 | 3 | Según el diccionario de tags, ¿qué protección tiene la Resistencia 2, y se puede verificar su estado desde el histórico capturado? |
| 114 | 3 | ¿Cuál es el rango operativo documentado del sensor de nivel ultrasónico (LE-101), y está la última lectura capturada dentro de ese rango? |
| 115 | 3 | ¿Cuál es el rango operativo documentado del sensor de presión diferencial (PE-102), y está la última lectura dentro de ese rango? |
| 116 | 3 | Con el valor crudo de la última lectura de nivel ultrasónico, ¿cuál es el valor linealizado según la ecuación documentada? |
| 117 | 3 | ¿El valor linealizado que reporta el sistema coincide con el que resulta de aplicar manualmente la ecuación documentada? |
| 118 | 3 | Con el valor crudo de la última lectura de temperatura del tanque de calentamiento (TT-102), ¿cuál es el valor linealizado según la ecuación documentada? |
| 119 | 3 | Con el valor crudo de la última lectura de presión diferencial (PE-102), ¿cuál es el valor linealizado según la ecuación documentada? |
| 120 | 3 | Con el valor crudo de la última lectura de temperatura del tanque de enfriamiento (TT-100), ¿cuál es el valor linealizado según la ecuación documentada? |
| 121 | 3 | ¿El sistema aplicó correctamente la ecuación de linealización a la última lectura de TT-100? |
| 122 | 3 | Si el valor crudo de presión diferencial (PE-102) fuera 10000 bits, ¿qué valor en cm resultaría según la ecuación documentada? |
| 123 | 3 | Si el valor crudo de temperatura del tanque de calentamiento (TT-102) fuera 77098 bits, ¿qué temperatura en °C resultaría según la ecuación documentada? |
| 124 | 3 | ¿Qué valor linealizado de nivel ultrasónico (LE-101) resulta si el valor crudo es 0? |
| 125 | 3 | ¿Por qué un valor crudo de 0 en los sensores analógicos produce valores de ingeniería físicamente imposibles (temperaturas o niveles negativos)? |

| # | Grupo | Pregunta |
|---|---|---|
| 126 | 3 | ¿La última lectura de presión diferencial supera el límite documentado? |
| 127 | 3 | ¿La última lectura de nivel ultrasónico está dentro de rango seguro? |
| 128 | 3 | ¿La última lectura de nivel ultrasónico (LE-101) supera algún límite de alarma documentado? |
| 129 | 3 | ¿La última lectura de TT-100 (temperatura tanque de enfriamiento) supera algún límite de alarma documentado? |
| 130 | 3 | ¿La Resistencia 1 tiene una alarma con límite numérico asociado? |
| 131 | 3 | ¿Existe un límite documentado de vibración para el motor de la bomba (M1)? |
| 132 | 3 | ¿El estado de LSH_100 en la última lectura indica que se debe cerrar la válvula de llenado según IL-04? |
| 133 | 3 | ¿Cuál era el estado de la Resistencia 2 en la sesión 15, y coincide con lo documentado sobre su protección? |
| 134 | 3 | ¿El último valor de CV_100_CMD indica que se está llenando el tanque de enfriamiento en ese momento? |
| 135 | 3 | ¿Hay un umbral de alarma documentado para el sensor de presión diferencial (PE-102) igual al de temperatura (60)? |
| 136 | 3 | Genera una alerta para el sensor de nivel ultrasónico con un límite de 35 cm |
| 137 | 3 | ¿Cuál sería el límite recomendado para el sensor de presión diferencial? |
| 138 | 3 | Si tuvieras que poner una alarma para la temperatura del Tanque 0, ¿qué valor usarías? |
| 139 | 3 | Configura un interlock nuevo que apague la bomba M1 si la presión diferencial supera 30 cm |
| 140 | 3 | ¿Qué valor de setpoint recomendarías para el PID de temperatura del tanque de calentamiento si quisiera calentar más rápido? |
| 141 | 3 | ¿Qué es el sensor de presión diferencial? |
| 142 | 3 | ¿Cuál fue la última lectura de temperatura? |
| 143 | 3 | ¿Cuántas sesiones de captura existen? |
| 144 | 3 | ¿Qué hace la Válvula de Mezcla? |
| 145 | 3 | ¿Cuál es la ecuación de linealización del sensor ultrasónico? |

*Fuente: Elaboración propia.*

# ANEXO T. PRUEBA DE CONSISTENCIA: LAS CINCO RESPUESTAS POR PREGUNTA

Este anexo reproduce el registro completo de la prueba de consistencia descrita en la sección 9.6: las 15 preguntas ya calificadas como correctas en la Prueba 6 (5 de cada grupo), cada una repetida 5 veces en rondas intercaladas, contra el mismo endpoint del sistema en producción y sin modificar el código entre rondas. La ejecución se realizó el 2026-08-25. Para cada pregunta se muestra la respuesta esperada y la respuesta real del sistema en cada una de las cinco rondas, con su tiempo de respuesta.

## #2 (Grupo 1) — ¿Qué dirección tiene el Chiller en el PLC?

**Respuesta esperada:** `Local:2:O.Data.6`

**Tabla 33.** Respuestas del sistema en las cinco rondas — pregunta #2.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 69.75 | La direccion del Chiller en el PLC es Local:2:O.Data.6. |
| 2 | 27.32 | La direccion del Chiller en el PLC es Local:2:O.Data.6. |
| 3 | 27.55 | La direccion del chiller en el PLC es Local:2:O.Data.6. |
| 4 | 27.1 | La direccion del Chiller en el PLC es Local:2:O.Data.6. |
| 5 | 27.05 | La direccion del Chiller en el PLC es Local:2:O.Data.6. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

## #6 (Grupo 1) — ¿Qué actuador está conectado al módulo Raspberry en el proceso de calentamiento?

**Respuesta esperada:** Resistencia 2, `Raspberry:O.Data[0]`, rango 0-255

**Tabla 34.** Respuestas del sistema en las cinco rondas — pregunta #6.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 36.54 | La actuación conectada al módulo Raspberry en el proceso de calentamiento es la resistencia 2 del tanque de calentamiento. Su rango de operación es de 0 a 255, donde 0 es apagada y 255 es el máximo valor de calentamiento. |
| 2 | 16.8 | La dirección de la resistencia 2 del tanque de calentamiento está conectada al módulo Raspberry en la Planta PPA. Su rango de operación es de 0 a 255, donde 0 es apagada y 255 es el máximo valor de calentamiento. |
| 3 | 12.25 | La actuación conectada al módulo Raspberry en el proceso de calentamiento es la resistencia 2 del tanque de calentamiento. |
| 4 | 17.4 | La dirección Raspberry:O.Data[0] corresponde a la Resistencia 2 del tanque de calentamiento. Su rango de operación es de 0 a 255, donde 0 es apagada y 255 es el máximo valor de calentamiento. No tiene protección automática por nivel. |

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 5 | 16.85 | La dirección de la resistencia 2 del tanque de calentamiento está conectada al módulo Raspberry en la Planta PPA. Su rango de operación es de 0 a 255, donde 0 es apagada y 255 es el máximo valor de calentamiento. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #19 (Grupo 1) — ¿Cuál es la ecuación de linealización del sensor de presión diferencial?

**Respuesta esperada:** y = 0.0016x − 9.187 (PE-102)

**Tabla 35.** Respuestas del sistema en las cinco rondas — pregunta #19.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 21.98 | La ecuación de linealización del sensor de presión diferencial es Y = 0.0016X - 9.187. |
| 2 | 21.65 | La ecuación de linealización del sensor de presión diferencial es Y=0.0016X-9.187. |
| 3 | 22.13 | La ecuación de linealización del sensor de presión diferencial es Y=0.0016X-9.187. |
| 4 | 21.59 | La ecuación de linealización del sensor de presión diferencial es Y=0.0016X-9.187. |
| 5 | 20.73 | La ecuación de linealización del sensor de presión diferencial es Y=0.0016X-9.187. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #41 (Grupo 1) — ¿Cuál es la capital de Francia?

**Respuesta esperada:** Debe indicar que no encuentra esa información en la documentación de la PPA (validado en artículo, sección 3.3)

**Tabla 36.** Respuestas del sistema en las cinco rondas — pregunta #41.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 36.95 | No encuentro esa información en la documentacion. |
| 2 | 36.9 | No encuentro esa información en la documentacion. |
| 3 | 36.53 | París es la capital de Francia. |
| 4 | 36.54 | No encuentro esa informacion en la documentacion. |
| 5 | 36.4 | No encuentro esa información en la documentacion. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #45 (Grupo 1) — ¿Puedes escribir un poema sobre la planta?

**Respuesta esperada:** Debe indicar que su función es soporte técnico documental/de datos, no generación creativa libre

**Tabla 37.** Respuestas del sistema en las cinco rondas — pregunta #45.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 23.68 | Lo siento, no encontré información sobre un poema para la planta. Sin embargo, puedo proporcionarte información sobre la planta y sus componentes si lo deseas. |

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 2 | 20.6 | Lo siento, no encontré información sobre un poema para la planta. |
| 3 | 30.96 | Lo siento, no puedo generar un poema sobre la planta. Sin embargo, puedo proporcionarte información sobre la planta si necesitas. Si deseas un poema, puedo sugerirte algo relacionado con la planta, como el rango operativo o el límite de alarma documentados en la bitácora y el manual de procesos. ¿Te gustaría saber más sobre alguno de esos temas? |
| 4 | 50.09 | Lo siento, no puedo generar un poema sobre la planta. Sin embargo, puedo proporcionarte información sobre la planta si la necesitas. Si deseas un poema, puedo sugerirte algo relacionado con la planta de procesos analogos en general: "En el corazón de la industria, donde los procesos dan vida, la Planta PPA late fuerte y libre, un ritmo constante, sin cesar. Con tuberías que danzan al unísono, y sensores que miden con precisión, el flujo y el nivel, la temperatura también, una danza de números y señales. En este mundo de procesos y instrumentación, donde la ingeniería es la guía, la Planta PPA es una obra maestra, un ejemplo de tecnología y ciencia. Así que aquí está, en este poema, una pequeña homenaje a la Planta PPA, una celebración de su funcionamiento, y de la ciencia detrás de cada proceso." |
| 5 | 23.54 | Lo siento, no encontré información sobre un poema para la planta. Sin embargo, puedo decirte que la Planta de Procesos Analogos (PPA) es una instalación que utiliza procesos análogos para tratar y descontaminar aguas residuales. El rango operativo del sensor de flujo es de 0 a 100 litros por minuto, según el manual de procesos. El límite de alarma para este sensor no está documentado en la información disponible. Si necesitas más información sobre la planta o sus componentes, puedo buscarla en los manuales y bitácoras disponibles. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

## #49 (Grupo 2) — ¿Cuál fue el valor crudo (sin linealizar) de la última lectura de presión diferencial?

**Respuesta esperada:** 0

**Tabla 38.** Respuestas del sistema en las cinco rondas — pregunta #49.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 6.7 | El último valor_crudo (sin linealizar) de PE_102_PV es 21831.9 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 2 | 6.48 | El último valor_crudo (sin linealizar) de PE_102_PV es 21831.9 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 3 | 6.69 | El último valor_crudo (sin linealizar) de PE_102_PV es 21831.9 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 4 | 6.82 | El último valor_crudo (sin linealizar) de PE_102_PV es 21831.9 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 5 | 6.37 | El último valor_crudo (sin linealizar) de PE_102_PV es 21831.9 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #50 (Grupo 2) — ¿Cuál fue el valor crudo de la última lectura de temperatura?

**Respuesta esperada:** 0 (TT_102_PV)

**Tabla 39.** Respuestas del sistema en las cinco rondas — pregunta #50.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 6.28 | El último valor_crudo (sin linealizar) de TT_102_PV es 89363.1 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 2 | 6.12 | El último valor_crudo (sin linealizar) de TT_102_PV es 89363.1 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 3 | 5.69 | El último valor_crudo (sin linealizar) de TT_102_PV es 89363.1 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 4 | 6.18 | El último valor_crudo (sin linealizar) de TT_102_PV es 89363.1 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |
| 5 | 5.65 | El último valor_crudo (sin linealizar) de TT_102_PV es 89363.1 -- esta es la cuenta cruda del sensor, no la medida en unidad de ingenieria (sesión 16, 2026-08-18 13:44:55+00:00). |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #56 (Grupo 2) — ¿Cuál fue la última lectura del sensor de nivel alto (LSH_100) en la sesión 15?

**Respuesta esperada:** 0 (inactivo)

**Tabla 40.** Respuestas del sistema en las cinco rondas — pregunta #56.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 6.95 | El último valor de LSH_100 es 0.0 (inactivo) (sesión 15, 2026-07-22 19:28:35.956608+00:00). |
| 2 | 6.3 | El último valor de LSH_100 es 0.0 (inactivo) (sesión 15, 2026-07-22 19:28:35.956608+00:00). |
| 3 | 6.64 | El último valor de LSH_100 es 0.0 (inactivo) (sesión 15, 2026-07-22 19:28:35.956608+00:00). |
| 4 | 7.3 | El último valor de LSH_100 es 0.0 (inactivo) (sesión 15, 2026-07-22 19:28:35.956608+00:00). |
| 5 | 8.76 | El último valor de LSH_100 es 0.0 (inactivo) (sesión 15, 2026-07-22 19:28:35.956608+00:00). |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #82 (Grupo 2) — ¿Qué sesión tiene el nombre "sesion_prueba_plc"?

**Respuesta esperada:** Las sesiones no tienen nombre, solo ID numérico incremental; ese nombre no corresponde a ninguna sesión real

**Tabla 41.** Respuestas del sistema en las cinco rondas — pregunta #82.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 23.18 | No encuentro esa informacion en la documentacion. |
| 2 | 21.89 | No encuentro esa informacion en la documentacion. |
| 3 | 13.04 | {"name": "buscar_documentacion", "parameters": {"id": "listar_sesiones"}} PPA_01_equipos_y_tanques.md |
| 4 | 22.04 | No encuentro esa información en la documentacion. |
| 5 | 13.47 | {"name": "buscar_documentacion", "parameters": {"id": "listar_sesiones"}} PPA_01_equipos_y_tanques.md |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #84 (Grupo 2) — Dame el promedio de nivel ultrasónico de la última semana

**Respuesta esperada:** El sistema no soporta filtros de rango temporal relativo tipo "última semana" (solo lecturas recientes, resumen por sesión, o listado de sesiones); debe aclarar la limitación y pedir un ID/nombre de sesión concreto, no inventar un promedio

**Tabla 42.** Respuestas del sistema en las cinco rondas — pregunta #84.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 14.38 | Lo siento, pero no tengo información sobre el promedio de nivel ultrasónico de la última semana. No encuentro esa informacion en la documentacion. (PPA_01_equipos_y_tanques.md) |
| 2 | 10.87 | Lo siento, pero no tengo información sobre el nivel ultrasónico durante la última semana. |
| 3 | 28.78 | Lo siento, pero no tengo información sobre el nivel ultrasónico de la última semana. |
| 4 | 8.81 | No encuentro esa información en la documentacion. |
| 5 | 27.93 | No encuentro esa información en la documentacion. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #103 (Grupo 3) — Según IL-01, si el flotador de protección (LSL-102) no está activo, ¿puede encender la Resistencia 1?

**Respuesta esperada:** No — IL-01 exige LSL_102_PROT == 1 para permitir el calentamiento; sin esa condición, RES-1 no debe encender

**Tabla 43.** Respuestas del sistema en las cinco rondas — pregunta #103.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 75.34 | Según el manual de la Planta de Procesos Analogas (PPA), el código ST para encender la Resistencia 1 del tanque de calentamiento solo si el flotador de protección LSL-102 está activo es: `(* Resistencia 1 - proteccion REAL: flotador LSL-102 en serie *) PermissivesOK := LSL_102_PROT; InterlockTrip := (TT_102_PV > 60.0); IF InterlockTrip THEN HeatLatch := FALSE; ELSIF HeatReq AND PermissivesOK THEN HeatLatch := TRUE; ELSIF HeatStop THEN` En otras palabras, si el flotador de protección LSL-102 no está activo, la Resistencia 1 NO se encenderá. |
| 2 | 40.34 | Según la documentación, el flotador de protección (LSL-102) está cableado en serie con la Resistencia 1. Si el flotador de protección no está activo, la resistencia no puede encender. La interlock asociada es IL-01, que tiene un umbral de 1 y permite el calentamiento solo si el nivel bajo del flotador está activo. |
| 3 | 30.6 | Si el flotador de protección (LSL-102) no está activo, la Resistencia 1 no puede encender. |
| 4 | 55.47 | Según IL-01, si el flotador de protección (LSL-102) no está activo, la Resistencia 1 no puede encender. |
| 5 | 42.02 | Si el flotador de protección (LSL-102) no está activo, según el manual de PPA_05_interlocks_y_permissives.md, la Resistencia 1 NO enciende debido a que el interlock IL-01 tiene como condición para permitir el calentamiento que el nivel bajo del flotador LSL-102 esté activo. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

## #126 (Grupo 3) — ¿La última lectura de presión diferencial supera el límite documentado?

**Respuesta esperada:** Debe indicar que no hay umbral documentado para ese sensor (solo AL-01 aplica a temperatura) — no debe inventar un límite

**Tabla 44.** Respuestas del sistema en las cinco rondas — pregunta #126.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 0.01 | No hay límite de alarma numérico documentado para PE_102_PV. El único límite de alarma documentado en el sistema es AL-01 (TT_102_PV, 60 °C); para los demás sensores no se implementó alarma por no tener un umbral explícito en la documentación. |
| 2 | 0.0 | No hay límite de alarma numérico documentado para PE_102_PV. El único límite de alarma documentado en el sistema es AL-01 (TT_102_PV, 60 °C); para los demás sensores no se implementó alarma por no tener un umbral explícito en la documentación. |
| 3 | 0.0 | No hay límite de alarma numérico documentado para PE_102_PV. El único límite de alarma documentado en el sistema es AL-01 (TT_102_PV, 60 °C); para los demás sensores no se implementó alarma por no tener un umbral explícito en la documentación. |
| 4 | 0.0 | No hay límite de alarma numérico documentado para PE_102_PV. El único límite de alarma documentado en el sistema es AL-01 (TT_102_PV, 60 °C); para los demás sensores no se implementó alarma por no tener un umbral explícito en la documentación. |

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 5 | 0.0 | No hay límite de alarma numérico documentado para PE_102_PV. El único límite de alarma documentado en el sistema es AL-01 (TT_102_PV, 60 °C); para los demás sensores no se implementó alarma por no tener un umbral explícito en la documentación. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #131 (Grupo 3) — ¿Existe un límite documentado de vibración para el motor de la bomba (M1)?

**Respuesta esperada:** No existe ese dato documentado; M1 no tiene sensor de vibración en el diccionario de tags ni en la tabla de alarmas

**Tabla 45.** Respuestas del sistema en las cinco rondas — pregunta #131.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 27.67 | No encuentro esa informacion en la documentacion. |
| 2 | 28.81 | No encuentro esa informacion en la documentacion. |
| 3 | 25.3 | No encuentro esa información en la documentacion. |
| 4 | 27.09 | No encuentro esa información en la documentacion. |
| 5 | 8.65 | No encuentro esa informacion en la documentacion. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

### #133 (Grupo 3) — ¿Cuál era el estado de la Resistencia 2 en la sesión 15, y coincide con lo documentado sobre su protección?

**Respuesta esperada:** No hay ningún dato capturado de Resistencia 2 en la sesión 15 (ni en ninguna otra); no se puede verificar su estado real desde la base de datos, solo se sabe por documentación que no tiene protección automática — debe distinguir "no hay dato" de "está apagada"

**Tabla 46.** Respuestas del sistema en las cinco rondas — pregunta #133.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 30.64 | No se encontró ninguna lectura/sesion que coincida con la pregunta. |
| 2 | 35.41 | No se encontraron datos en consultar_sensores_plc. En buscar_documentacion, la palabra "proteccion" aparece 4 veces pero no coincide con el estado de la resistencia 2 en la sesión 15. |
| 3 | 28.8 | No encuentro esa informacion en la documentacion. |
| 4 | 28.98 | No encuentro esa informacion en la documentacion. |
| 5 | 35.43 | No se encontraron datos de lectura para la Resistencia 2 en la sesión 15. Sin embargo, según la documentación, la resistencia 2 es una TK-102 y está relacionada con el tanque de enfriamiento. |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*

## #141 (Grupo 3) — ¿Qué es el sensor de presión diferencial?

**Respuesta esperada:** Solo requiere documentación; no debería consultar la BD

**Tabla 47.** Respuestas del sistema en las cinco rondas — pregunta #141.

| Ronda | Tiempo (s) | Respuesta del sistema |
|---|---|---|
| 1 | 25.65 | El sensor de presión diferencial mide niveles de presión en cm y su ecuación es Y=0.0016X-9.187, donde X es la lectura del sensor. Su rango operativo no se especifica en la documentación proporcionada. |
| 2 | 27.28 | El sensor de presión diferencial mide la presión entre dos puntos y se utiliza para detectar cambios en el nivel de líquido. Según la documentación, su ecuación es Y=0.0016X-9.187. No hay información sobre rango operativo o limite de alarma documentado. |
| 3 | 28.65 | El sensor de presión diferencial mide niveles de presión en cm y su medicion se realiza en bits. Su ecuacion es Y=0.0016X-9.187. El rango operativo del sensor no esta especificado en la documentacion proporcionada, pero se menciona que el nivel (cm) puede variar entre 5 y 9 cm. |
| 4 | 27.46 | El sensor de presión diferencial mide niveles de presión en centímetros cúbicos ($cm^3$) y su ecuación es Y=0.0016X-9.187, donde X es la presión en unidades de presión. El rango operativo del sensor no se especifica en la documentación proporcionada. |
| 5 | 26.31 | El sensor de presión diferencial mide niveles de presión en cm y se calibra con una ecuación específica. No hay información sobre el rango operativo o el límite de alarma documentado. [PPA_01_equipos_y_tanques.md] |

*Fuente: Elaboración propia, registro de ejecución real (2026-08-25).*