Práctica Profesional Supervisada

Paez, Marín Ariel

Desarrollo de Software para el Monitoreo de la Calidad del Agua en Cuencas Hídricas del Conurbano Bonaerense con Tecnologías IoT e IA

Instituto de Ingeniería y Agronomía

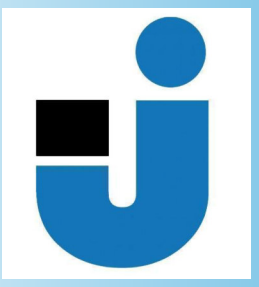
Carrera: Ingeniería en Informática

2025

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática



PRÁCTICA PROFESIONAL SUPERVISADA
Informe final

Desarrollo de Software para el Monitoreo de la Calidad del Agua en Cuencas Hídricas del Conurbano Bonaerense con Tecnologías IoT e IA

Martín Ariel Paez

Florencio Varela, Abril 2025

ESTUDIANTE
Apellido y Nombres: Paez Martín Ariel
Correo electrónico: mpaez.ok@gmail.com

ORGANIZACIÓN DONDE SE REALIZA LA PRÁCTICA PROFESIONAL SUPERVISADA
Nombre de la institución: Universidad Nacional Arturo Jauretche
Dirección: Av. Calchaquí 6200, Florencio Varela, (1888) Buenos Aires, Argentina
Teléfono: +54 11 4275-6100
Sector: Programa Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de interés social, Instituto de Ingeniería y Agronomía

TUTOR DE LA ORGANIZACIONAL
Apellido y Nombres: Prof. Mg. OSIO, Jorge
Correo electrónico: josio@unaj.edu.ar

DOCENTE SUPERVISOR
Apellido y Nombres: Prof. Dr, CAPPELLETTI, Marcelo
Correo electrónico: mcappelletti@unaj.edu.ar
Apellido y Nombres: Prof. Ing OLIVERA, Lucas
Correo electrónico: lolivera.unaj@gmail.com

COORDINADOR DE LA CARRERA DE INGENIERÍA INFORMÁTICA
Apellido y Nombres: Dr. Ing. Morales, Martin
Correo electrónico: martin.morales@unaj.edu.ar

Resumen

La creciente preocupación por el impacto ambiental en los recursos hídricos motiva el desarrollo de soluciones tecnológicas que permitan un monitoreo continuo y confiable. En este trabajo se presenta un sistema IoT orientado a la supervisión de parámetros en arroyos. La metodología adoptada emplea sondas interconectadas mediante tecnología LoRa y un protocolo de comunicación propietario desarrollado para este trabajo, capaz de establecer rutas dinámicas y garantizar la transmisión eficiente de datos en entornos de baja conectividad. La infraestructura se complementa con un servidor IoT que centraliza la recepción de la información, posibilita su almacenamiento y ofrece interfaces de monitoreo en tiempo real para los usuarios. Sobre esta plataforma se integraron modelos de aprendizaje automático, destinados a clasificar los usos del agua. Los resultados alcanzados demuestran la viabilidad de una solución integral que fortalece las herramientas disponibles para la preservación de los recursos hídricos y la toma de decisiones ambientales.

Palabras Clave: Internet de las cosas, inteligencia artificial, calidad del agua.

Abstract

The growing concern about the environmental impact on water resources drives the development of technological solutions that enable continuous and reliable monitoring. This work presents an IoT system for stream parameter supervision. The proposed methodology employs probes interconnected through LoRa technology and a proprietary communication protocol, capable of establishing dynamic routes and ensuring efficient data transmission in low-connectivity environments. The infrastructure is complemented by an IoT server that centralizes data reception, enables storage, and provides real-time monitoring interfaces for users. On this platform, machine learning models were integrated to classify critical conditions in water quality. The results demonstrate the feasibility of a comprehensive solution that strengthens the tools available for water resource preservation and environmental decision-making.

Keywords: Internet of Things, Artificial Intelligence, Water Quality.

Dedicatoria y agradecimiento

Gracias, Dios, por acompañarme en cada paso de la carrera. En momentos de confusión y cansancio, me diste claridad y discernimiento. Me ayudaste a cumplir con todas las metas que me propuse, relacionarme con quienes me rodeaban y descubrir oportunidades que me permitieron avanzar.

Dedico este trabajo a mi familia y amigos más cercanos, por su apoyo constante, comprensión y confianza. En particular, un afectuoso reconocimiento a mi mamá, cuyo ejemplo me impulsó a transitar la vida universitaria, y a mi futura esposa, quien me acercó nuevamente a Dios en un momento clave de mi carrera profesional.

Agradezco a la Universidad Nacional Arturo Jauretche y a la carrera de Ingeniería en Informática, no solo por la formación brindada, sino también por la calidad y compromiso de su cuerpo docente. Tuve la oportunidad de compartir experiencias fuera del aula con personas importantes para mí, dentro del ámbito académico y/o profesional. Realmente son muchas personas, así que me tendré que limitar a mencionar a aquellos que, en simultáneo, se vincularon directamente con la Práctica Profesional Supervisada (PPS), representaron a otros y por quienes guardo un profundo agradecimiento.

Un reconocimiento especial merece Jorge Osio y su equipo de investigación, cuya labor acompañó de manera continua mi formación, proporcionándome herramientas y perspectivas que enriquecieron mi desarrollo. Del mismo modo, hago mención a Marcelo Cappelletti y a los investigadores de su departamento con quienes compartí el trayecto de mi PPS.

Asimismo, agradezco al área de Vinculación, y en particular a Andrés Fernández, sobre todo por confiar en mí; pero además, por abrir muchas puertas. Entre todas ellas, la que permitió obtener fondos para comenzar un proyecto de vinculación, que hoy se extiende con mi PPS.

Finalmente, reconozco enormemente a mis compañeros y compañeras de cursada por el intercambio de experiencias, aprendizajes y consejos que enriquecieron este trayecto hacia la culminación de la carrera. Gracias a ellos, que compartieron vivencias similares, encontré la motivación y la constancia para proponerme metas cada vez más ambiciosas.

Índice General

Índice de imágenes

Introducción

El arroyo Las Conchitas constituye una cuenca hídrica ubicada en el sur del Conurbano Bonaerense, compartida por los partidos de Florencio Varela y Berazategui. Este curso de agua enfrenta desafíos ambientales significativos, entre ellos la contaminación de origen industrial y urbano, que deteriora la calidad del agua y afecta de manera directa al ecosistema. Esta problemática no es un caso aislado, sino representativa de la situación de numerosas cuencas hidrográficas en distintas regiones de Argentina.

Diversos organismos han desarrollado investigaciones sobre el estado ambiental del arroyo Las Conchitas, incluyendo el relevamiento de su flora y fauna. Sin embargo, en la mayoría de los casos —y particularmente en los estudios llevados a cabo por equipos de la Universidad Nacional Arturo Jauretche (UNAJ)— la recopilación de datos se realiza de forma manual. Este enfoque metodológico limita tanto la frecuencia como el volumen de datos recolectados, lo que dificulta la generación de análisis más profundos y sostenidos en el tiempo.

El presente proyecto tiene como objetivo investigar, diseñar, implementar e integrar los componentes de software requeridos para un sistema de monitoreo de la calidad del agua, cuyo hardware fue desarrollado en la UNAJ. Se propone un sistema que combina redes de sensores, comunicación IoT, persistencia de datos, visualización mediante dashboards interactivos y análisis avanzado a través de técnicas de inteligencia artificial. La solución se estructura en tres componentes principales: una red de sondas con comunicación multihop, un backend encargado de la persistencia y procesamiento de datos junto con la exposición de servicios, y un frontend para monitoreo y análisis visual.

La red de sondas se organiza en una topología lineal, en correspondencia con la disposición natural de los cauces hídricos. Para garantizar la conectividad y optimizar las rutas de transmisión, se emplean repetidores y gateways seleccionados dinámicamente. Con el fin de soportar esta arquitectura multihop se diseñó e implementó un protocolo de comunicación propietario, orientado a mantener la consistencia de los mensajes, controlar el flujo de datos y gestionar los procesos de reconexión.

Las siguientes secciones abordan los fundamentos teóricos del trabajo, la especificación e implementación del protocolo, la arquitectura integral del sistema IoT y la aplicación de técnicas de inteligencia artificial para el análisis y clasificación de los datos recolectados. Finalmente, en la sección de resultados se presentan las pruebas de campo y las evaluaciones de desempeño obtenidas.

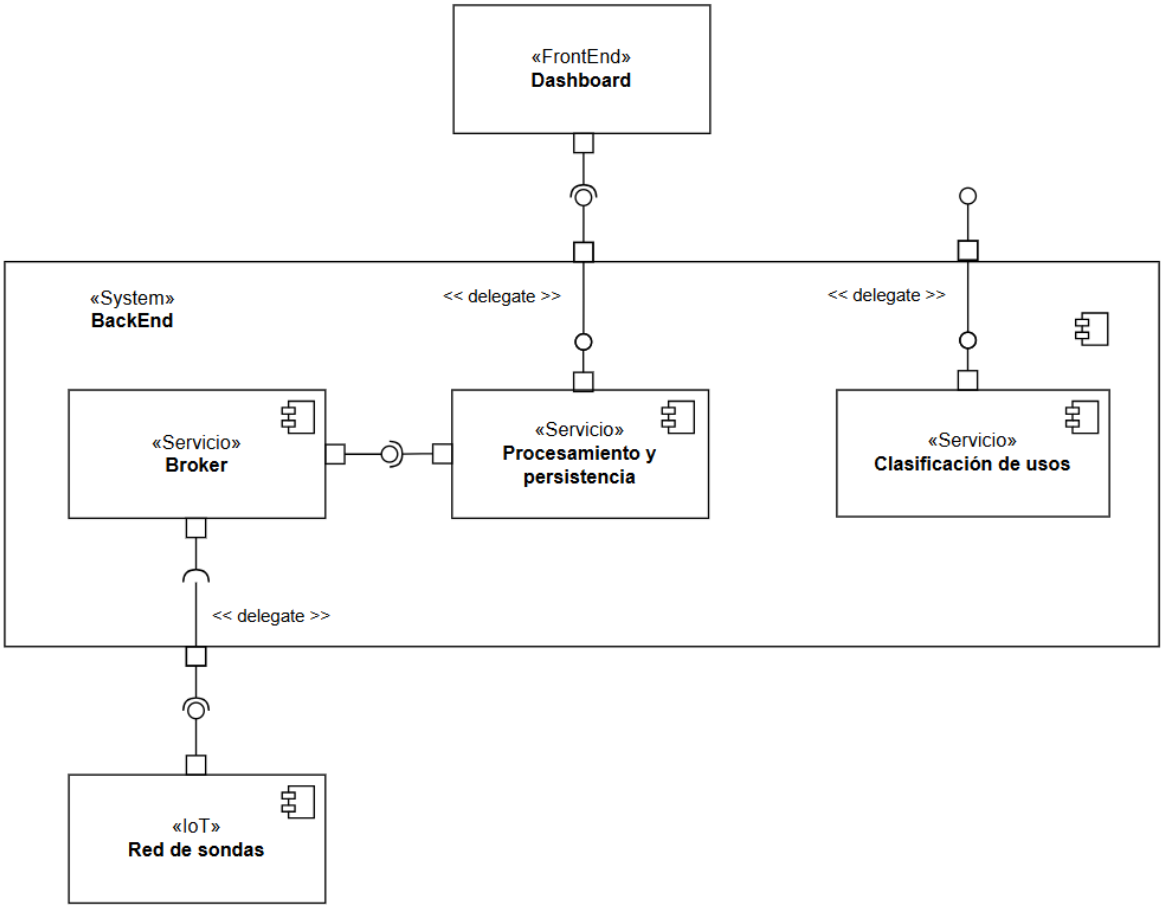


Imagen 1. Diagrama de componentes del sistema

Marco Conceptual y Contexto Ambiental

El monitoreo de la calidad del agua en cuencas hidrográficas urbanas es una tarea fundamental para la gestión ambiental, la protección de los ecosistemas y la salud pública. Las cuencas, como sistemas dinámicos e interconectados, cumplen un rol esencial en la regulación del ciclo hidrológico, el mantenimiento de la biodiversidad y el soporte de múltiples actividades humanas (UNESCO, 2020). Sin embargo, en contextos urbanos e

industriales como el del Conurbano Bonaerense, estos cuerpos de agua están sometidos a intensas presiones antrópicas que afectan su equilibrio ecológico.

Parámetros Físico-Químicos en la Evaluación de la Calidad del Agua

El pH, la temperatura, la conductividad eléctrica y la turbidez son parámetros físico-químicos clave para evaluar la calidad del agua (APHA, 2017). Su monitoreo continuo permite detectar alteraciones asociadas a descargas contaminantes, cambios en el uso del suelo y procesos naturales. La medición conjunta de estos parámetros resulta útil para estudiar la idoneidad del agua para diferentes usos, detectar tendencias de contaminación y generar alertas tempranas.

El pH indica la acidez o alcalinidad del agua. Valores fuera del rango 6.5–8.5 pueden afectar la solubilidad de nutrientes y metales, impactando negativamente en la biota acuática (Chapman, 1996).

La temperatura afecta la solubilidad del oxígeno, la velocidad de las reacciones químicas y el metabolismo de los organismos acuáticos. Aumentos anómalos pueden indicar contaminación térmica, asociada a efluentes industriales.

La conductividad eléctrica mide la capacidad del agua para conducir corriente eléctrica, relacionada con la concentración de sales disueltas. Altos valores pueden reflejar la presencia de contaminantes inorgánicos, como fertilizantes, residuos industriales o cloacales.

La turbidez se refiere a la cantidad de partículas suspendidas en el agua. Altos niveles afectan la fotosíntesis de organismos autótrofos y pueden asociarse a arrastres de suelos, descargas urbanas o sedimentación.

Usos del Agua y Relevancia del Monitoreo

El agua de las cuencas urbanas puede tener múltiples usos: recreativos, agrícolas, industriales, domésticos, o de preservación ambiental. La clasificación de estos usos depende directamente de la calidad del recurso y de su aptitud para cada destino, siguiendo normativas como las establecidas por el Código Alimentario Argentino y la Organización Mundial de la Salud (OMS, 2011).

En regiones densamente pobladas como el sur del Gran Buenos Aires, los cursos de agua suelen ser utilizados informalmente como receptores de efluentes cloacales o industriales, lo que genera un riesgo sanitario y compromete la posibilidad de usos legítimos. El monitoreo en tiempo real permite, en este contexto, brindar información crucial para la toma de decisiones en políticas públicas y en la planificación territorial.

Biodiversidad y Ecosistemas en Riesgo

Los ecosistemas acuáticos de cuenca, como el del arroyo Las Conchitas, albergan una biodiversidad significativa, compuesta por flora ribereña, invertebrados, peces, anfibios y aves. Diversos estudios (Tundisi & Tundisi, 2008; Bonetto, 1994) han documentado cómo la contaminación orgánica e inorgánica produce efectos en cascada sobre estas comunidades: desde la disminución de especies sensibles hasta la proliferación de organismos oportunistas, con impactos en las cadenas tróficas y la resiliencia del sistema.

En el caso particular del arroyo Las Conchitas, investigaciones desarrolladas por equipos de la Universidad Nacional Arturo Jauretche han documentado cambios en la composición florística y faunística asociados a la alteración de las condiciones ambientales. La falta de registros sistemáticos y continuos limita, sin embargo, la posibilidad de generar modelos predictivos y de evaluar adecuadamente el deterioro ecológico.

Tecnologías Emergentes para el Monitoreo Ambiental

La incorporación de tecnologías de Internet de las Cosas (IoT) e Inteligencia Artificial (IA) en el monitoreo de cuencas representa una oportunidad significativa para superar las limitaciones de los métodos tradicionales. El despliegue de sensores interconectados permite la recolección automática y georreferenciada de datos en tiempo real, mientras que los modelos de IA pueden asistir en la clasificación de los usos del agua, detección de patrones y generación de alertas tempranas ante eventos anómalos (Liu et al., 2018).

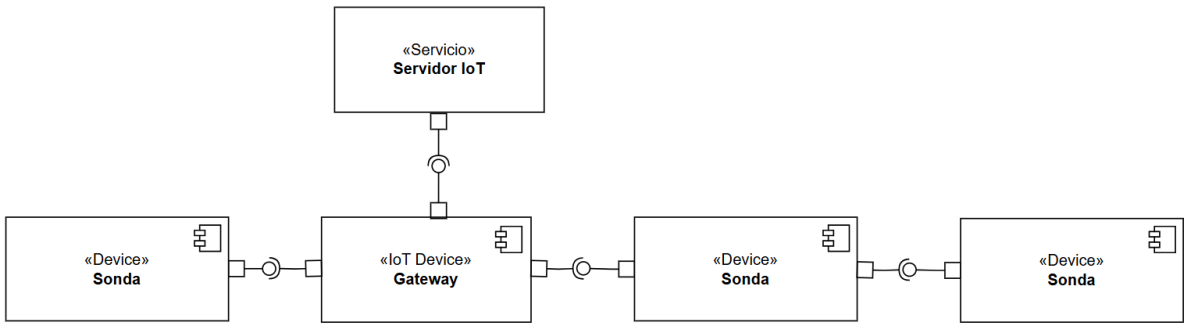
Estas tecnologías, aplicadas en proyectos de extensión e investigación como el de la UNAJ, no solo amplían las capacidades científicas, sino que también democratizan el acceso a

información ambiental, fomentando la participación ciudadana y la transparencia institucional.

Red de sondas y protocolo de comunicación propietario.

En este capítulo se presentan los fundamentos y la operación del sistema de comunicación implementado, estructurado en tres secciones: investigación de protocolos disponibles en el mercado, especificación del protocolo propio y su implementación. Las pruebas de campo se documentan en el capítulo de resultados.

La topología de la red se diseñó de manera lineal, siguiendo la disposición natural de los cauces de las cuencas hídricas, lo que facilita la cobertura de los puntos de medición. Los nodos intermedios actúan como repetidores y son seleccionados dinámicamente por el sistema para garantizar la conectividad hacia uno de los gateway, que son los únicos dispositivos IoT centralizados de la red.



Imágen 2. Diagrama de componentes de la red de sondas.

Análisis comparativo de protocolos de comunicación empleados en soluciones IoT

Aspectos clave en protocolos IoT

El análisis de soluciones de comunicación en entornos IoT requiere atender ciertos aspectos técnicos que condicionan su aplicabilidad. Esta sección delimita los criterios utilizados para comparar las soluciones existentes (Atzori, Iera, & Morabito, 2010).

Topologías multihop y redes en malla: permiten que los mensajes atraviesen varios nodos antes de alcanzar su destino. Cada nodo intermedio actúa como repetidor, lo que mejora la cobertura y la resiliencia sin depender de enlaces directos entre origen y destino (Cilfone et al., 2019).

Enrutamiento dinámico vs. estático: en redes dinámicas, las rutas no están predefinidas, sino que se descubren y actualizan automáticamente ante cambios como la aparición, desconexión o fallo de nodos. Esta capacidad es clave en redes móviles o sujetas a reconfiguración. (Al-Fuqaha et al., 2015).

LPWAN y eficiencia energética: las tecnologías de red de área amplia y bajo consumo (Low Power Wide Area Networks) están diseñadas para transmisiones de largo alcance y consumo reducido, pero suelen limitarse a topologías estrella o punto a punto, sin soporte nativo para multihop (Augustin, Yi, Clausen, & Townsley, 2016).

Separación por capas (modelo OSI): diferenciar qué nivel del modelo OSI aborda cada solución (capa física, enlace, red, red, etc.) permite establecer comparaciones justas y evaluar la posibilidad de integración o reemplazo de componentes.

Estos conceptos permiten contextualizar el rol, los alcances y las limitaciones de cada protocolo considerado a continuación.

Protocolos sobre LoRa para redes mesh

En esta categoría se ubican los principales competidores de nuestra solución, ya que están específicamente diseñados para implementar redes en malla utilizando tecnología LoRa (Augustin, Yi, Clausen, & Townsley, 2016). Su estudio es interesante como base para comenzar el desarrollo. No se trata solo de protocolos de comunicación, sino de stacks completos o bibliotecas que abordan el problema del multihop y el enrutamiento entre nodos distribuidos. Aquí es necesario identificar tanto las fortalezas de cada alternativa como las limitaciones que justifican el desarrollo de una solución propia.

LoRaMesher

Es una biblioteca de código abierto orientada a dispositivos con FreeRTOS (como el ESP32) que trabaja con LoRa, e implementa un protocolo de enrutamiento basado en vector de distancia. Utiliza RadioLib como capa de acceso a radio y requiere soporte nativo del sistema operativo para la gestión de tareas, colas y temporizadores. (LoRaMesher, s.f.)

- Ofrece una red mesh estática.
- Requiere FreeRTOS, lo cual limita su portabilidad.
- El mantenimiento de tablas de rutas puede presentar problemas de escalabilidad en redes extensas.

Meshtastic

Meshtastic es un ecosistema de comunicación orientado a usuarios finales. Se basa en LoRa y añade un protocolo de red mallada cifrado (AES-256), con soporte para auto-descubrimiento de nodos, reenvío de mensajes y herramientas gráficas de configuración. (Meshtastic, s.f.)

- No está orientado a una integración modular a nivel de protocolo.
- La lógica de enrutamiento está acoplada al firmware.
- Muy robusto para topologías dinámicas, pero innecesariamente complejo para topologías lineales o controladas.
- Aunque es open source, la arquitectura general está pensada para el uso final, no como base para otros desarrollos.

RadioHead + protocolo propio

RadioHead es una biblioteca ligera que implementa comunicación punto a punto con mecanismos básicos de capa 2: direccionamiento, ACKs, control de duplicados, etc.

- No implementa multihop ni enrutamiento dinámico. (AirSpayce, s.f.).
- Es útil como base para desarrollar un stack propio sobre LoRa.

Nuestro protocolo puede utilizar RadioHead como capa 2 y construir sobre ella una capa 3 personalizada con reenvío inteligente, detección de cambios de topología y acuses de recibo multihop.

MySensors

Framework modular orientado a aplicaciones de domótica y automatización casera. Admite múltiples tecnologías de transporte (incluyendo LoRa), y permite crear redes sencillas con enrutamiento básico. (MySensors, s.f.)

- No implementa mesh dinámico.
- Pensado para entornos pequeños (hobbista), con limitaciones en seguridad, descubrimiento y escalabilidad.
- Utiliza rutas fijas hacia el gateway, sin protocolos avanzados de reconfiguración o recuperación ante fallos.

Symphony Link

Protocolo propietario desarrollado por Link Labs, basado en LoRa, que ofrece funcionalidades de red avanzadas: repetidores, QoS, actualizaciones OTA, seguridad con infraestructura de clave pública, entre otras. (Link Labs, s.f.)

- No es un protocolo abierto.
- Requiere hardware y servicios de la empresa desarrolladora.
- Imposible de adaptar o extender libremente para escenarios específicos como los que se plantea nuestra solución.

Alternativas complementarias (no basadas en LoRa)

Además de los protocolos diseñados para operar sobre LoRa, existen otras soluciones de red en malla compatibles con ESP32 que utilizan distintas tecnologías de transmisión (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015). Estas alternativas pueden ser adecuadas en contextos donde el consumo energético, la cobertura o los requisitos de hardware sean diferentes. Si bien no resuelven directamente el problema de comunicación sobre enlaces LoRa, resultan relevantes en el análisis comparativo por su soporte para redes multihop y su nivel de madurez.

ESP-MESH (Wi-Fi Mesh de Espressif)

Implementación de malla desarrollada por Espressif sobre Wi-Fi. Permite que múltiples nodos ESP32 se comuniquen entre sí en forma distribuida, reenvíen datos y mantengan la conectividad incluso ante fallos parciales. (Espressif, s.f.)

- Aprovecha la infraestructura de Wi-Fi ya existente.
- Presenta un consumo energético elevado, lo que lo hace menos adecuado para nodos alimentados por batería.
- Requiere una planificación cuidadosa de la red para evitar saturación del canal.

ESP-BLE-MESH (Bluetooth Mesh para ESP32)

Implementación del estándar Bluetooth Mesh basada en BLE (Bluetooth Low Energy). Permite establecer redes malladas interoperables con otros dispositivos compatibles con la especificación Bluetooth Mesh. (Bluetooth SIG, 2023)

- Admite topologías malladas con múltiples roles por nodo.
- Alta complejidad de configuración, requiere definir modelos, elementos y publicaciones/suscripciones.
- Más eficiente energéticamente que Wi-Fi, pero con un alcance por nodo más limitado.

OpenThread

Protocolo de red en malla desarrollado por Google, basado en el estándar Thread (IEEE 802.15.4). Diseñado para IoT, con enfoque en bajo consumo, seguridad y autoorganización. (OpenThread, s.f.)

- No está soportado de forma nativa por ESP32, pero puede utilizarse mediante hardware adicional compatible con 802.15.4 (como módulos externos o co-procesadores).
- Es altamente escalable, soporta múltiples tipos de nodos (router, end-device) y cuenta con mecanismos de seguridad integrados.
- Ideal para entornos con requisitos estrictos de interoperabilidad y eficiencia.

Wirepas Mesh

Solución propietaria orientada a aplicaciones industriales de gran escala. Utiliza tecnologías sub-GHz o BLE, con una pila completa que gestiona enrutamiento, sincronización y reconfiguración automática. (Wirepas, s.f.)

- Requiere licencias y regalías por dispositivo.
- No es compatible con ESP32 directamente.
- Destinado a escenarios donde se prioriza la robustez de red y la administración centralizada, a costa de flexibilidad y apertura.

Protocolos de capa de red / enlace con soporte Mesh

A diferencia de soluciones específicas para LoRa o alternativas livianas sobre ESP32, los protocolos de esta sección trabajan en capas más altas (red o enlace) y resuelven de forma integral el enrutamiento dinámico y la gestión de topologías malladas. En general, implementan mecanismos estandarizados como RPL y están diseñados para operar sobre 6LoWPAN (IPv6 sobre IEEE 802.15.4) o variantes de IEEE 802.15.4, aportando robustez y escalabilidad a costa de mayor complejidad y consumo. (Winter et al., 2012; Thubert, 2012)

Estas soluciones son más formales, estandarizadas y robustas, y representan la elección natural para proyectos industriales complejos o entornos donde se requiera interoperabilidad con estándares IPv6. Sin embargo, requieren mayor capacidad de cómputo, memoria y soporte de hardware específico (como radios IEEE 802.15.4), por lo que no siempre resultan adecuadas para entornos de bajo costo, nodos ultra low-power o desarrollos donde se necesita un control detallado del stack. (Shelby et al., 2008)

Pueden considerarse competencia directa si el hardware del proyecto puede soportarlas, aunque no siempre están orientadas al nivel de control total que brinda una implementación propia.

Zigbee

Protocolo completo y maduro para redes IoT de corto alcance y bajo consumo. (Zigbee Alliance, s.f.)

- Basado en IEEE 802.15.4 (capas física y de enlace), con pila de red, seguridad y aplicación propia.
- Soporte Mesh: Multihop con formación automática de topología.
- Amplio ecosistema, interoperabilidad entre dispositivos certificados, seguridad incorporada.
- Stack cerrado en muchas implementaciones, menos control sobre la red, requiere hardware Zigbee dedicado.
- Orientado a Redes domésticas o comerciales, no es ideal para proyectos que requieran control detallado del protocolo.

Thread / OpenThread

Diseñado para IoT moderno con soporte IP nativo y topologías malladas. (OpenThread, s.f.)

- Basado en IEEE 802.15.4 + 6LoWPAN + IPv6.
- Soporte Mesh: Usa enrutamiento dinámico, roles jerárquicos y autoformación.
- Bajo consumo, seguro, escalable, con direccionamiento IP directo.
- No es nativo en ESP32; requiere transceptor 802.15.4 y configuración más compleja.
- Orientado a domótica y automatización industrial con soporte IP, ideal para soluciones interoperables.

Wi-SUN

Estándar abierto para redes malladas urbanas de largo alcance y alta confiabilidad. (Wi-SUN Alliance, s.f.)

- Basado en IEEE 802.15.4g + 6LoWPAN + RPL.
- Soporte Mesh: Totalmente integrado, cada nodo puede reenviar.
- Alcance extendido, tolerancia a fallos, ideal para infraestructura urbana y despliegues a gran escala.
- Hardware especializado, no orientado a ESP32; elevado costo y complejidad.
- Orientado a Smart Cities, redes industriales críticas, utilities.

RPL (Routing Protocol for Low-power and Lossy Networks)

Protocolo de enrutamiento estandarizado por IETF para redes IPv6 de bajo consumo. (Winter et al., 2012)

- Basado en operación sobre 6LoWPAN.
- Soporte Mesh: Multihop dinámico, optimizado para enlaces inestables.
- Alta robustez, soporte para jerarquías y distintas métricas de ruta.
- Alto consumo de memoria y CPU; no apto para microcontroladores simples ni directamente sobre LoRa.
- Orientado a aplicaciones con recursos medios-altos que requieran topologías flexibles bajo IPv6.

RPL se encuentra disponible en sistemas como Contiki-NG y RIOT OS, que actúan como base para nodos IP en entornos LLN (Low-power and Lossy Networks). De los dos, RIOT OS presenta mejor soporte para ESP32 y una estructura más profesional para desarrollos reales.

Protocolos de capa física y tecnologías subyacentes

Esta sección agrupa tecnologías que actúan en la capa física del modelo OSI y que, si bien no compiten directamente con el protocolo desarrollado (que opera en capas superiores), sí representan alternativas a LoRa en términos de medio de transmisión y características operativas. Estos protocolos no necesariamente compiten directamente con LoRa como solución de red, pero sí son relevantes como tecnología subyacente en protocolos que podrían competir con el enfoque propuesto.

BLE Long Range

Extensión de Bluetooth orientada a mayor alcance y menor consumo. Utiliza modulación GFSK con corrección de errores. Su simplicidad y eficiencia lo convierten en una opción válida para comunicaciones punto a punto o como base para redes en malla (BLE Mesh).
Compite con LoRa en escenarios de corto a mediano alcance con bajo consumo energético. (Bluetooth SIG, 2023).

NB-IoT / LTE-M

Tecnologías LPWAN soportadas por infraestructura celular. Ofrecen gran cobertura y buena penetración en interiores, aunque con mayor consumo y dependencia del operador.
Compite con LoRa en aplicaciones industriales o urbanas con infraestructura disponible.

Wi-Fi HaLow (IEEE 802.11ah)

Versión de Wi-Fi adaptada a dispositivos IoT. Opera en la banda sub-GHz para mejorar alcance y reducir consumo. (IEEE, s.f.).
Es un competidor emergente en el mismo segmento de LoRa, aunque aún con escasa adopción.

Sigfox

Red LPWAN propietaria y centralizada. Su diseño minimalista reduce al máximo el consumo y el ancho de banda, a cambio de una infraestructura cerrada y limitada flexibilidad. (Sigfox, s.f.).
Es conceptualmente similar a LoRa, pero con un modelo de operación más restrictivo.

IEEE 802.15.4

Estándar para comunicaciones inalámbricas de bajo consumo en las capas física y de enlace. Es la base de protocolos más complejos como Zigbee, Thread y 6LoWPAN, que agregan capacidades de red sobre este estándar. (IEEE, s.f.).

Protocolos IoT generalistas y populares

Muchos de los protocolos ampliamente utilizados en soluciones IoT resuelven aspectos clave de la comunicación entre dispositivos, pero no fueron concebidos para operar en topologías multihop. Por lo tanto, no compiten directamente con el protocolo desarrollado, aunque su conocimiento resulta fundamental para contextualizar la solución propuesta.

MQTT-SN

Versión simplificada de MQTT diseñada para dispositivos con recursos limitados. Utiliza un modelo de publicación/suscripción, pero depende de un broker centralizado, que actúa como único punto de coordinación. Esta arquitectura limita su aplicabilidad en redes distribuidas sin infraestructura persistente. (**MQTT-SN**, s.f.).

En conclusión, es popular, pero con dependencia de infraestructura central.

CoAP (Constrained Application Protocol)

Protocolo de capa de aplicación orientado a dispositivos de bajo consumo. Utiliza una arquitectura cliente-servidor, basada en UDP, y puede extenderse a un modelo de suscripción mediante el mecanismo Observe. Por sí solo no resuelve la comunicación multihop, aunque puede utilizarse junto a protocolos de enrutamiento como RPL en redes 6LoWPAN. (CoAP, s.f.).
En conclusión, es ligero y versátil, pero requiere soporte adicional para redes en malla. (Shelby et al., 2014).

Modbus RTU / Modbus TCP

Son protocolos ampliamente adoptados en entornos industriales. Modbus RTU está pensado para enlaces seriales como RS-485, en una arquitectura maestro-esclavo. Modbus TCP opera sobre redes IP. Ninguna variante incluye soporte para redes multihop ni tolera bien latencias elevadas, lo que limita su uso sobre tecnologías como LoRa sin adaptaciones específicas.
En conclusión, es eficiente en entornos cableados, pero no diseñados para redes distribuidas o inalámbricas. (Modbus Organization, s.f.).

BACnet (Building Automation and Control Networks)

Es un protocolo orientado a la automatización de edificios. Utiliza mecanismos como token passing sobre RS-485 (BACnet MS/TP) o redes IP (BACnet/IP). No contempla repetidores inteligentes ni multihop lógico. Implementaciones sobre tecnologías como LoRa presentan serias limitaciones en latencia y sincronización. (BACnet International, s.f.).
En conclusión, es sólido en contextos industriales o domóticos, pero no apto para escenarios distribuidos dinámicos.

Dash7 Alliance Protocol

Alternativa LPWAN pensada para dispositivos de bajo consumo. Define su propia capa física y no es compatible con LoRa. Su adopción es significativamente menor que otras soluciones como LoRaWAN, y su ecosistema está menos desarrollado. LoRaWAN (Dash7 Alliance, s.f.).
En conclusión, es interesante desde el punto de vista técnico, pero con escasa penetración.

Sistemas operativos y frameworks que implementan protocolos

Además de bibliotecas o protocolos independientes, existen sistemas operativos embebidos y SDKs que incorporan de manera nativa protocolos de red mallada. Estos entornos ofrecen una solución integral con funcionalidades de red, seguridad, administración de memoria y multitarea, lo que los vuelve especialmente atractivos para desarrollos profesionales o experimentales complejos.

En este apartado analizaremos tecnología mencionadas previamente, lo que distingue esta sección es el enfoque en el entorno completo (SO o framework) y no solo en el protocolo. Estos entornos no sólo resuelven la conectividad, sino que proporcionan un marco completo de desarrollo, con tareas concurrentes, manejo de recursos y herramientas para testeo y despliegue. Si bien pueden ser más pesados que una implementación directa sobre bare-metal o RTOS mínimo, ofrecen una base sólida para implementar protocolos complejos sin desarrollarlos desde cero.

Contiki-NG

Sistema operativo ligero para dispositivos embebidos, con fuerte orientación académica. (Contiki-NG, 2025)

- Soporte Mesh: Incorpora RPL sobre 6LoWPAN e IPv6.
- Ideal para simulaciones, prototipos de investigación y validación de conceptos.
- Portarlo a nuevos dispositivos puede ser complejo; no soporta ESP32 de forma oficial.
- Orientado a proyectos de investigación, pruebas de red, simulaciones en Cooja (Simulador de redes sensores Contiki). Para entornos académicos o validación de protocolos

RIOT OS

Sistema operativo moderno y modular, diseñado para IoT profesional. (RIOT OS, 2025)

- Soporte Mesh: Incluye RPL, soporte para múltiples interfaces de red (LoWPAN, Ethernet, LoRa, etc.).

- Mejor estructura que Contiki-NG, soporte oficial para ESP32, comunidad activa.
- Mayor complejidad de configuración y por tanto curva de aprendizaje.
- Orientado a desarrollo profesional sobre redes IP en sistemas embebidos, aplicaciones reales con conectividad avanzada.

ESP-IDF y SDKs de Espressif

Entorno de desarrollo oficial para el ESP32, con soporte directo a tecnologías malladas propias. (Espressif, 2025)

- Soporte Mesh: Compatible con ESP-MESH (WiFi) y BLE-MESH (Bluetooth).
- Integración nativa con el hardware ESP32, documentación oficial, soporte comercial.
- Mesh limitado a tecnologías específicas (no sobre LoRa) con topologías más simples comparadas con RPL.
- Orientado a aplicaciones comerciales o prototipos rápidos usando WiFi o Bluetooth, con soporte oficial y buena documentación. Puede ser usado para desarrollos reales sobre ESP32 con tecnologías de red comunes.

Justificación del desarrollo de un protocolo propio

Si bien hay múltiples opciones viables, se considera que de entre todos los protocolos analizados, Wi-SUN es la opción más sólida para una solución estandarizada, escalable y con multihop dinámico real.

Tiene las siguientes ventajas:

- Red mesh con RPL embebido.
- Robusta, confiable y pensada para despliegues grandes.
- Estándar abierto, con buen soporte de interoperabilidad.

Pero también implica costos y barreras importantes:

- Hardware especializado (no ESP32, ni ecosistema maker).
- Menor flexibilidad para control fino del stack.
- Desarrollo más costoso, menos adaptado a entornos mixtos o prototipos.

En conclusión, el desarrollo de un protocolo propietario representa una alternativa técnicamente sólida y estratégicamente justificada, orientada a contribuir con la comunidad en la búsqueda de soluciones IoT de bajo coste.

Si bien existen propuestas robustas como Wi-SUN, que ofrecen mayores garantías en entornos industriales y urbanos a gran escala, su adopción implica costos más elevados, requerimientos de hardware especializado y una menor flexibilidad para adaptar el comportamiento del stack de red a necesidades específicas.

Especificación del Protocolo

Objetivo y Alcance

El alcance del protocolo incluye la comunicación de datos desde nodos sensores en entornos no confiables y energéticamente restringidos, utilizando un esquema de retransmisión simple y robusto, apto para su implementación en microcontroladores de baja potencia sin sistema operativo. Este diseño está alineado con los objetivos de eficiencia, escalabilidad y simplicidad que demanda un sistema IoT orientado a tareas de monitoreo en campo.

El protocolo de comunicación desarrollado en este proyecto surge como una solución a las limitaciones inherentes a entornos de monitoreo ambiental distribuido, donde múltiples sondas deben transmitir datos hacia un único gateway IoT utilizando enlaces inalámbricos de largo alcance y bajo consumo energético. En particular, la intención es operar sobre la tecnología LoRa, aunque su definición se mantiene independiente de la capa física, permitiendo su adaptación futura a otros medios de transmisión con características similares.

El objetivo principal del protocolo es habilitar una topología en árbol compuesta por nodos sensores que actúan simultáneamente como retransmisores de mensajes, permitiendo así el enrutamiento multisalto de paquetes hacia el gateway. Esta estrategia resulta especialmente adecuada en escenarios donde la disposición geográfica y las barreras físicas impiden una comunicación directa entre todos los nodos y el receptor central.

Se priorizó una arquitectura simple, sin establecimiento de rutas previo ni mecanismos de sincronización complejos, de forma de reducir el tamaño de los paquetes transmitidos y

simplificar la lógica de cada nodo. La existencia de un único receptor por nodo evita la formación de bucles y elimina la necesidad de campos TTL en los encabezados, lo cual favorece aún más la eficiencia del protocolo bajo restricciones de ancho de banda y energía.

El protocolo contempla la posibilidad de pérdidas ocasionales de paquetes, dado que su finalidad es transmitir datos periódicos de sensores ambientales, donde la tolerancia a fallos puntuales es aceptable. Sin embargo, se incorporaron mecanismos de reconocimiento (ACK) y control de tiempo (timeout) para detectar caídas de enlace y reorganizar dinámicamente la red cuando sea necesario.

Arquitectura General del Protocolo

El protocolo desarrollado opera en las capas 2 (enlace) y 3 (red) del modelo OSI, resolviendo el direccionamiento, la retransmisión de paquetes y el control de enlaces sin depender de mecanismos externos ni de funcionalidades provistas por capas superiores. La definición se mantuvo intencionalmente desacoplada de la capa física, lo que garantiza su adaptabilidad a diferentes tecnologías de transmisión. No obstante, la arquitectura fue concebida con el objetivo de funcionar de manera eficiente sobre enlaces LoRa, el cual se caracteriza por su baja tasa de datos, largo alcance y bajo consumo energético.

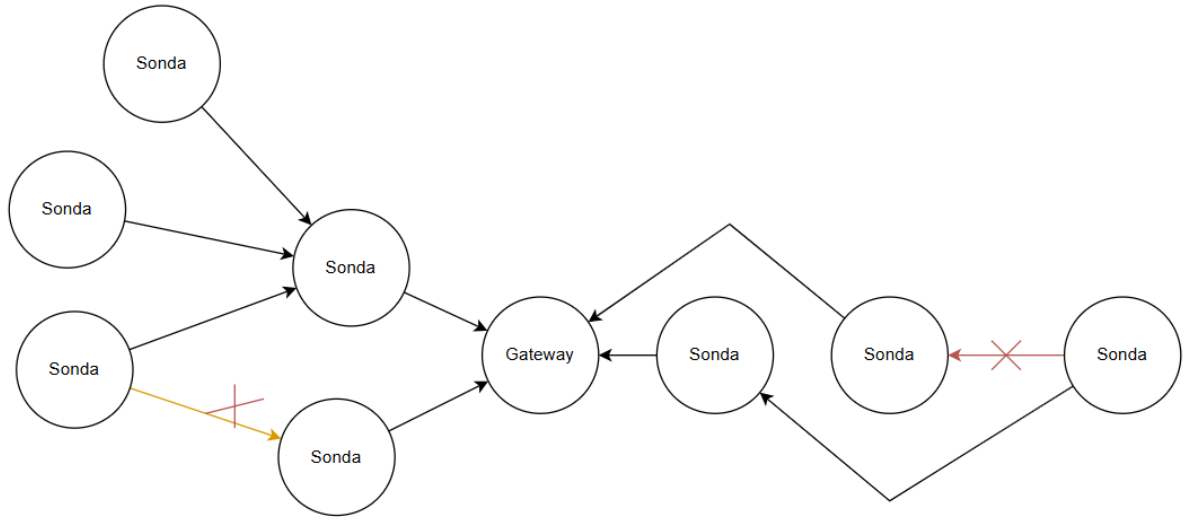
Si bien el protocolo resuelve funcionalidades típicas de las capas 2 y 3 del modelo OSI, no impone una separación estricta entre ambas. Por el contrario, promueve una implementación compacta y eficiente, especialmente en entornos IoT, donde es deseable fusionar responsabilidades entre capas para reducir el tamaño de los mensajes, la complejidad del software y el consumo energético.

Organización lógica de la red

Desde una perspectiva estructural, el protocolo asume una topología lógica en forma de árbol, en la cual cada nodo sensor está asociado a un único receptor configurado. Esta decisión permite que la red se mantenga libre de ciclos, evitando la necesidad de implementar mecanismos adicionales para la detección y supresión de bucles. A su vez, la ruta de reenvío hacia el gateway se define de forma implícita: cada nodo transmite sus mensajes al receptor preestablecido, quien repite el proceso hasta alcanzar la raíz del árbol.

El gateway, ubicado en la cima de esta jerarquía, representa el único nodo con conexión directa al exterior del sistema y actúa como colector final de todos los datos generados por la red. Dado que un mismo nodo puede ser receptor de múltiples otros, se facilita la construcción de árboles asimétricos que se ajustan a la distribución física y a las condiciones de propagación del entorno.

La arquitectura también contempla la posibilidad de reorganización dinámica en caso de fallos o degradación de los enlaces, permitiendo a los nodos identificar alternativas válidas para la retransmisión minimizando la necesidad de mensajes de reconocimiento al reutilizar todos los ACK enviados en la red y mediante configuración de timeouts o frecuencias de muestreo. Esta capacidad de autoconfiguración contribuye a la resiliencia del sistema frente a cambios en la topología o condiciones ambientales adversas.



Imágen 3. Red de sondas sin bucles.

Identificación de Nodos y Direccionamiento

Cada nodo de la red posee un identificador numérico fijo, único dentro del dominio de la red, que cumple simultáneamente las funciones de direccionamiento en capa de enlace (capa 2) y capa de red (capa 3). Esta decisión de unificar la identificación responde a un criterio de eficiencia, propio de entornos IoT con restricciones de ancho de banda y procesamiento, evitando la duplicación de información en los encabezados de los mensajes y simplificando la lógica de reenvío y verificación.

Desde el punto de vista del diseño, se optó por un esquema de direccionamiento plano y estático por nodo, sin jerarquías ni asignaciones dinámicas. No obstante, el protocolo contempla que los nodos puedan ser reubicados físicamente dentro de la red, y que en consecuencia deban establecer nuevos vínculos de retransmisión hacia el gateway. Esta flexibilidad se logra gracias al mecanismo de reconocimiento y a la evaluación dinámica de los ACK recibidos, que permiten seleccionar receptores más convenientes de forma distribuida.

Cabe destacar que la elección de un identificador simple también responde a criterios particulares del proyecto, orientado específicamente a redes de sensores donde las condiciones operativas —como el número de nodos, la frecuencia de muestreo y el tipo de datos transmitidos— permiten mantener este esquema sin comprometer la robustez ni la funcionalidad general del sistema.

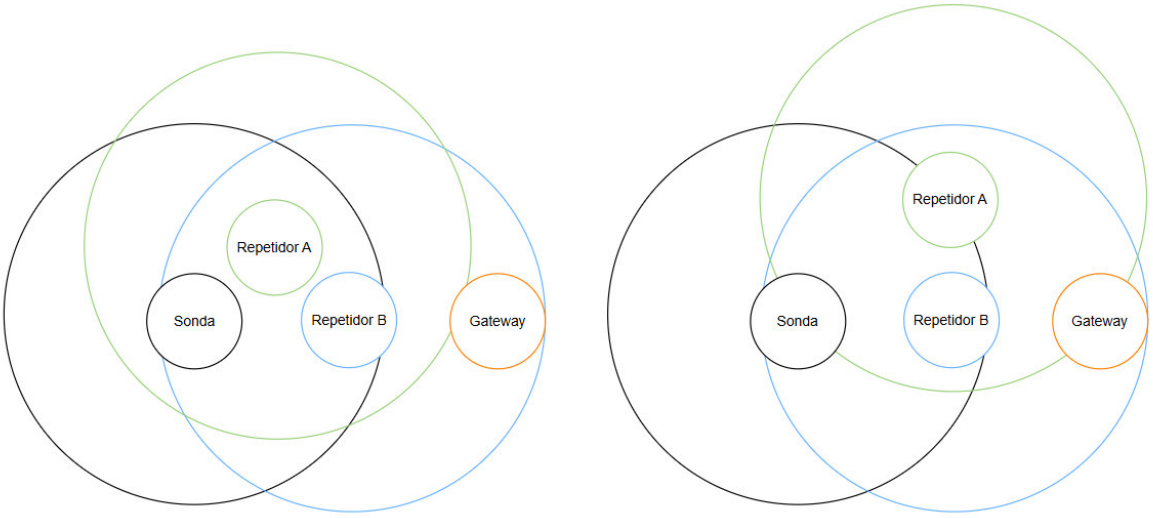
Mecanismos de Control y Selección de Rutas

El funcionamiento del protocolo se basa en una lógica descentralizada, donde cada nodo sensor participa activamente en la construcción y mantenimiento de su ruta hacia el gateway, sin requerir de un conocimiento global de la topología ni de una planificación previa. Este comportamiento emergente es posible gracias al intercambio periódico de mensajes de descubrimiento (RecognitionRequest) y confirmación (ConnectionStatus), junto con un criterio de selección local para definir el receptor ideal.

Cada nodo mantiene un único receptor configurado, lo cual asegura una estructura lógica en forma de árbol dirigida hacia el gateway. Este diseño evita la formación de bucles y simplifica el reenvío de mensajes, ya que todo paquete de datos sigue un único camino ascendente, nodo por nodo, hasta llegar a destino.

Cuando un nodo se enciende o pierde conectividad con su receptor, inicia un proceso de descubrimiento mediante el envío de un mensaje de tipo RecognitionRequest. Este mensaje, de tipo broadcast, puede ser recibido por múltiples nodos vecinos. Los que ya cuenten con conexión válida al gateway responden con un ConnectionStatus, conteniendo información clave como la distancia al gateway (en cantidad de saltos) y el RSSI hacia su propio receptor (el nodo al cual envía los mensajes destinados al gateway). A partir de estas respuestas, el

nodo destino selecciona su nuevo receptor, favoreciendo rutas cortas y con buena calidad de enlace. Además, la comunicación del mensaje en sí misma tiene asociada un RSSI, cuya obtención debe estar contemplada en la interfaz de la capa física y empleado en el algoritmo asociado a dicha selección. En caso de que el nodo que contesta sea el mismo gateway, los campos contendrán la distancia cero y el RSSI máximo ideal.



Imágen 4. Selección del receptor.

El proceso de selección es local e independiente, permitiendo que la red se reconfigure dinámicamente ante cambios en la topología, fallos o reubicaciones de nodos. No se requiere un proceso de reconstrucción global: basta con que el nodo desconectado vuelva a emitir un RecognitionRequest para restablecer su ruta y las de sus nodos descendientes en el árbol.

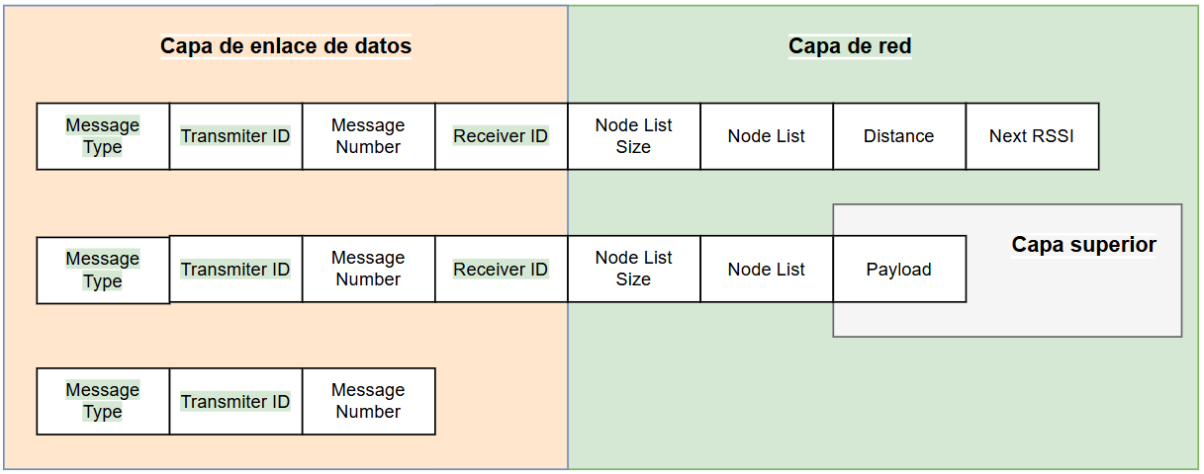
Una vez establecido un receptor, los mensajes de tipo DataRequest siguen la ruta hacia el gateway. En cada salto, el nodo receptor agrega su identificador a la lista de nodos por los que pasó el mensaje (NodeList), la cual será luego utilizada para retornar un mensaje de tipo ConnectionStatus a modo de acuse de recibo (ACK).

Los ACK son utilizados por todos los nodos dentro del alcance, independientemente de si son o no el destinatario, para evaluar, con base en el contenido del ACK, si debe actualizar su receptor actual. Ya que los ACK están presentes en la comunicaciones iniciadas con los demás tipos de mensajes, se decidió que la información asociada a dicha evaluación únicamente esté presente en estos, simplificando el procesamiento de los mensajes.

Tipos de Mensaje y Estructuras

El protocolo define tres tipos de mensajes principales, cada uno con una estructura específica y un propósito bien delimitado dentro del flujo de comunicación entre nodos:

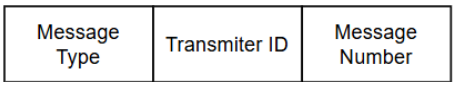
- RecognitionRequest: Iniciar un proceso de descubrimiento de ruta.
- DataRequest: Transmitir datos desde una sonda hacia el gateway.
- ConnectionStatus: Confirmar la recepción exitosa de un mensaje y facilitar la actualización de rutas.



Imágen 5. Estructura en bytes de los distintos tipos de paquetes.

RecognitionRequest

Este mensaje es emitido por un nodo que no tiene aún un receptor asignado, con el fin de detectar nodos cercanos que puedan actuar como retransmisores. Es de tipo broadcast y su estructura mínima garantiza una rápida propagación sin congestionar la red.



Imágen 6. Estructura en bytes del paquete de reconocimiento.

MsgType: Identifica el mensaje como RecognitionRequest.
TransmitterId: ID del nodo que emite el reconocimiento.
MsgNum: Número de secuencia generado por el emisor, usado para evitar duplicados.

Los mensajes de tipo RecognitionRequest no se reenvían. Al tratarse de mensajes de reconocimiento con propósito de descubrimiento de nodos vecinos, se transmiten como broadcast y sólo se espera una respuesta por parte de aquellos nodos que estén en condiciones de actuar como receptores. Este comportamiento evita congestionar la red con mensajes redundantes y garantiza que cada nodo pueda construir su relación de conexión a partir de evaluaciones locales, basadas en la calidad de los ACKs que recibe como respuesta.

DataRequest

Es el mensaje utilizado para transportar datos desde una sonda hasta el gateway.

Imágen 7. Estructura en bytes del paquete de datos.

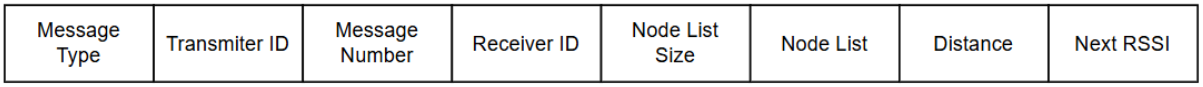
MsgType: Identifica el mensaje como DataRequest.
TransmitterId: Nodo que envió o retransmitió el mensaje.
MsgNum: Número secuencial para detección de duplicados.
ReceiverId: Nodo al que va dirigido el siguiente salto.
NodeListSize: Cantidad de nodos que retransmitieron el mensaje hasta el momento.
NodeList: Lista ordenada de nodos que participaron hasta el momento en la retransmisión, e irá incrementando hasta llegar al gateway.
Payload: Información a transmitir.

Los mensajes de tipo DataRequest son enviados desde las sondas hacia el gateway siguiendo el receptor asignado por cada nodo. Cada vez que un nodo recibe un DataRequest destinado a él, verifica su validez y, en caso de tener conexión con el gateway, procede a reenviarlo al siguiente salto definido por su propio receptor. En ese proceso, el nodo agrega su identificador a la NodeList, construyendo así el historial completo de retransmisiones necesarias para alcanzar el destino.

El algoritmo de reenvío no contempla redistribución dinámica del mensaje en tiempo real: si el receptor actual no es alcanzable, el mensaje se pierde. Este diseño privilegia la simplicidad y eficiencia, en lugar de asegurar confiabilidad total en la entrega.

ConnectionStatus (ACK)

Este mensaje actúa como confirmación de recepción de cualquiera de los otros mensajes. Además de validar la entrega, permite al destinatario de este mensaje decidir si debe actualizar su receptor según ciertos criterios de calidad de enlace.



Imágen 8. Estructura en bytes del Ack.

MsgType: Identificador del tipo ACK.
TransmitterId: Nodo que envía el ACK.
MsgNum: Número secuencial para detección de duplicados.
ReceiverId: Próximo nodo destino en el recorrido inverso.
NodeListSize: Longitud de la porción de ruta restante.
NodeList: Lista ordenada de nodos que participaron en la retransmisión y todavía no recibieron el ACK. Esta lista va disminuyendo su cardinalidad a medida que el mensaje se acerca al destino final.
Distance: Número de saltos restantes hasta el destino final.
OtherRssi: Valor RSSI medido entre el nodo que emite el ACK y su nodo receptor, o sea, el nodo al que enviaría un mensaje si quisiera comunicarse con el gateway.

A diferencia de lo que ocurre con los mensajes de datos, los ACKs no son aprovechados exclusivamente al nodo destinatario. Todos los nodos que los reciben, incluso si no son el próximo en la ruta inversa, pueden evaluarlos pasivamente y decidir si les conviene actualizar su receptor actual.

Manejo de Duplicados y Control de Flujo

Dado el carácter distribuido y asincrónico de la red, así como las posibles retransmisiones que pueden surgir por la lógica de reenvío, el protocolo implementa un mecanismo simple pero eficaz para la detección de duplicados y el control de flujo. Esta estrategia permite evitar procesamientos redundantes y mitigar efectos indeseados sobre la red, como bucles de reenvío o congestión.

Cada nodo mantiene una tabla interna que asocia un único identificador de mensaje (MsgNum) al ID del nodo emisor (TransmitterId). De este modo, al recibir un mensaje, el nodo receptor puede determinar si dicho mensaje ya fue procesado recientemente.

El número de mensaje (MsgNum) se incrementa secuencialmente en el emisor, y debido a que su tamaño está acotado, eventualmente puede producirse un desbordamiento (overflow). Para manejar esta situación, se aplica una lógica de ventana de tolerancia ante la pérdida de mensajes intermedios y los efectos del wrap-around.

El criterio para aceptar un mensaje recibido es el siguiente:

- Si el valor de MsgNum recibido es estrictamente mayor que el último almacenado, el mensaje se considera nuevo y es aceptado.
- Si es menor, se evalúa si puede tratarse de un caso de desbordamiento. Para ello, se define una ventana de aceptación (por ejemplo, 15 mensajes). Si la distancia entre el número almacenado y el valor máximo posible, sumada a la distancia desde cero hasta el nuevo número recibido, es menor o igual a esta ventana, entonces se considera que el mensaje es válido pese al overflow, y se acepta.
- En caso contrario, el mensaje es descartado por considerarse duplicado o inválido.

Este enfoque permite una tolerancia configurable a la pérdida de paquetes, asumiendo que ciertos mensajes pueden perderse sin comprometer el funcionamiento general de la red. Además, evita mantener historiales extensos de mensajes recientes, lo que resulta adecuado para dispositivos con recursos limitados.

Mantenimiento de Conectividad y Timeout

El protocolo contempla mecanismos sencillos para detectar la pérdida de conectividad con el gateway, lo cual resulta esencial en escenarios donde los nodos pueden ser reubicados, sufrir condiciones adversas, escasa conexión, etc.

Cada nodo espera recibir periódicamente confirmaciones de entrega (mensajes de tipo ConnectionStatus) en respuesta a los mensajes que emite. Estos ACKs permiten verificar que el camino hacia el gateway continúa activo. Si durante un intervalo de tiempo determinado el

nodo no recibe ACKs, se considera que ha perdido la conexión. En tal caso, el nodo desasigna su receptor actual y reinicia el proceso de descubrimiento de ruta mediante un nuevo mensaje de tipo RecognitionRequest.

El valor del timeout no es fijo, sino que debe ser ajustado según el contexto operativo. La frecuencia de muestreo —es decir, la periodicidad con la que el nodo genera mensajes— juega un rol central en esta configuración. En aplicaciones donde los datos se envían con frecuencia alta, un timeout más corto permite detectar rápidamente interrupciones en la conectividad. En cambio, en escenarios de baja frecuencia, un timeout más largo evita falsos positivos ante la ausencia temporal de tráfico.

Si bien el protocolo no impone un valor predeterminado, se recomienda que el timeout sea al menos varias veces mayor que el intervalo de muestreo esperado, permitiendo absorber retrasos por congestión o reintentos en la red. En aplicaciones más críticas o con requerimientos específicos, este parámetro puede ser configurado manualmente para adaptarse a las necesidades del sistema.

Este enfoque brinda flexibilidad operativa sin comprometer la simplicidad de la implementación, manteniendo el diseño adecuado para entornos IoT con recursos limitados y condiciones de red variables.

Limitaciones y Consideraciones

El protocolo fue diseñado con una orientación clara hacia aplicaciones IoT de bajo consumo, baja tasa de datos y tráfico predominantemente unidireccional desde nodos sensores hacia un gateway central. En ese contexto, se priorizó una arquitectura ligera, sin dependencia de estructuras de control persistentes ni mecanismos complejos de gestión de estado.

El protocolo asume ciertas condiciones operativas —como tráfico regular, pérdida ocasional aceptable, y topología relativamente estable— que delimitan su ámbito de aplicación. Estas limitaciones no derivan de deficiencias del diseño, sino de elecciones explícitas orientadas a lograr un equilibrio entre simplicidad, eficiencia y suficiencia funcional para el entorno en el que se desplegará.

Una limitación importante es que el protocolo no implementa un mecanismo explícito de reintento o retransmisión en caso de pérdida de paquetes, ni garantiza confirmación de entrega punto a punto. El ConnectionStatus (ACK) se emite como respuesta al mensaje recibido, pero si no alcanza su destino, no se considera una condición de error que dispare un reenvío. Este diseño es coherente con la naturaleza periódica y tolerante a fallos de las aplicaciones previstas: la pérdida de mensajes resulta aceptable siempre que se mantenga dentro del intervalo de tiempo configurado. La estrategia operativa consiste en enviar el próximo dato medido al completar el ciclo de muestreo, sin esperar confirmaciones. Además, el timeout utilizado para detectar la pérdida de conexión con el receptor es configurable por nodo, lo que permite ajustar su sensibilidad según la criticidad de la ubicación dentro de la topología.

Por la naturaleza del problema, y no por una cuestión del protocolo, cuando la desconexión ocurre en zonas cercanas al gateway el impacto puede ser significativamente mayor, ya que afecta a todas las sondas cuyas rutas dependen de ese nodo. Para mitigar este efecto, se recomienda aumentar la densidad de sondas en las proximidades del gateway, de modo que si un nodo falla, existan otras sondas dentro del mismo radio de alcance que puedan asumir su rol de retransmisión. Esto mismo puede entenderse como una disminución de la densidad al alejarse del gateway, lo cual contribuye a optimizar los recursos.

En cuanto a la dinámica de red, el protocolo no actualiza rutas existentes en tiempo real ni propaga cambios topológicos de manera explícita. Dado que no se almacenan tablas globales ni se mantiene una representación estructural de la red, la detección de cambios como la reubicación física de un nodo o la pérdida de conectividad se produce de forma implícita: si un nodo no recibe confirmaciones durante un período de tiempo predefinido, asume la pérdida de enlace con su receptor y reinicia el proceso de descubrimiento (RecognitionRequest). Esta estrategia permite tolerar modificaciones en la red, aunque no garantiza una transición suave entre topologías.

También es importante mencionar que, por simplicidad y adecuación a los requisitos de la aplicación de referencia, se optó por un modelo de direccionamiento simplificado, en el cual un único identificador numérico cumple el rol tanto de dirección física como lógica. Esta

decisión facilita la implementación y reduce el tamaño de los mensajes, aunque podría no ser adecuada en escenarios que requieran mayor granularidad o segmentación funcional.

Implementación del Protocolo

Arquitectura General del Sistema

La implementación del protocolo se organiza en una arquitectura modular en capas, inspirada en principios de la arquitectura hexagonal (Cockburn, 2005) y en ciertos aspectos estructurales de las APIs RESTful (Fielding, 2000). La principal diferencia radica en que los controladores de la capa de presentación son intercambiados en tiempo de ejercicio en función del estado interno del sistema, permitiendo así adaptar su comportamiento de forma dinámica dentro de un marco determinístico definido por el patrón de estados. Esta organización promueve una alta cohesión interna y un desacoplamiento efectivo entre componentes, facilitando la evolución del sistema, su testabilidad y su adaptación a diferentes entornos operativos.

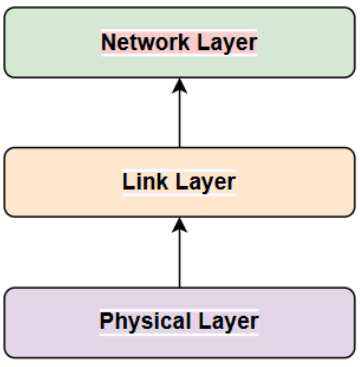
La solución hace uso extensivo del patrón State (Gamma, Helm, Johnson, & Vlissides, 1994) para modelar el comportamiento dinámico del nodo a nivel de capa 3 del modelo OSI. Además, define una interfaz que debe ser respetada por la capa física, de modo que esta pueda ser sustituida sin impacto sobre las capas superiores. Para ello, se implementa el patrón Adapter, que encapsula la interacción con la librería LoRa utilizada en esta implementación. Por su parte, la capa de enlace se apoya en el patrón Factory Method para instanciar dinámicamente los componentes requeridos, evitando el acoplamiento directo entre consumidores y dependencias. Esta fábrica que modela la capa de enlace, relacionada mediante composición con la capa física, es accedida por referencia desde la capa de red, permitiendo construir objetos de la capa de enlace en función de los servicios requeridos en cada situación operativa.

Desde el punto de vista estructural, la arquitectura puede representarse mediante un diagrama concéntrico. En el centro se ubica el Contexto; en capas sucesivas se organizan los estados, los servicios, los serializadores, los adaptadores físicos y, finalmente, el entorno de operación. Esta disposición enfatiza la dirección de las dependencias: las capas externas dependen de las internas, pero no a la inversa, lo que favorece la extensibilidad y la separación de

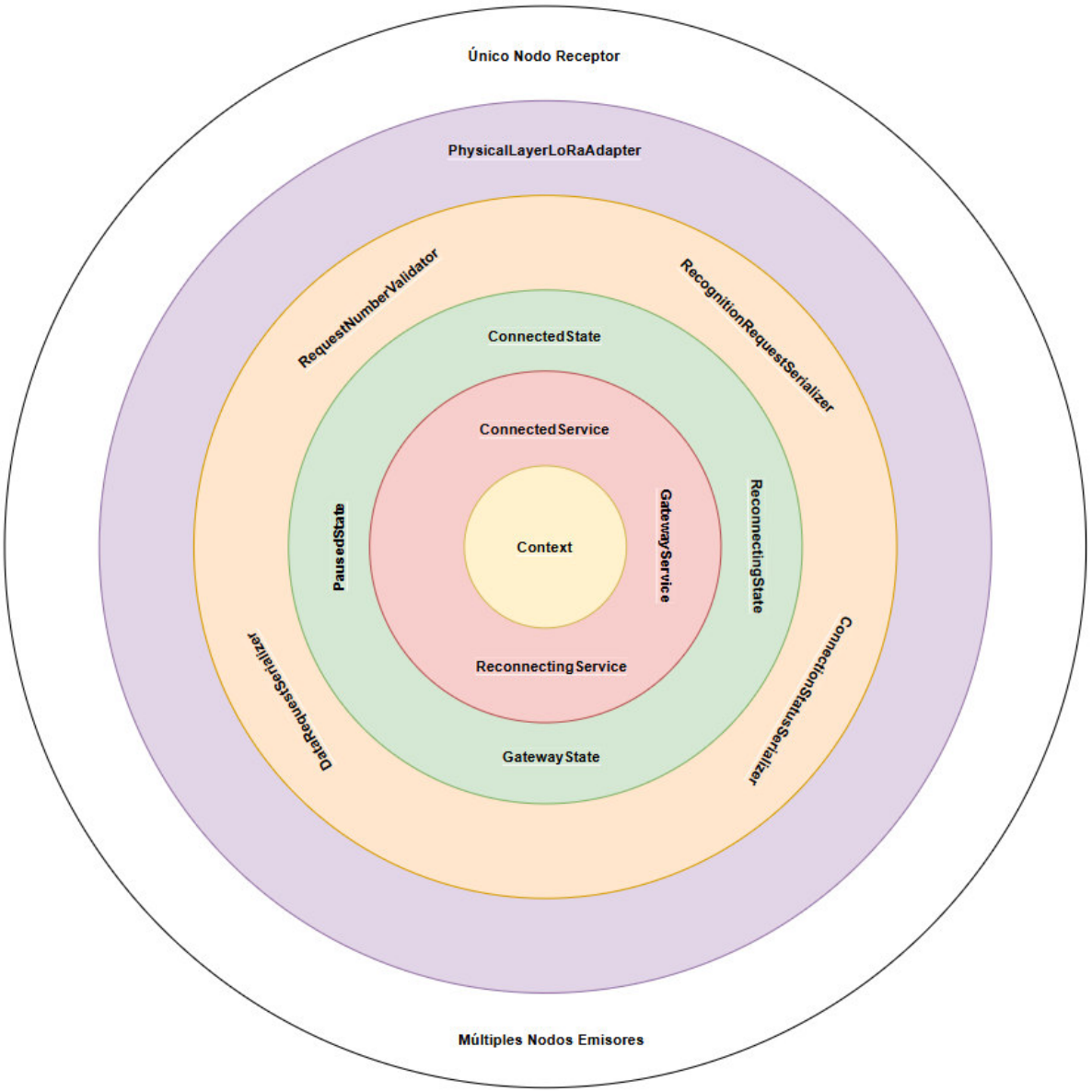
responsabilidades. Como se discutirá más adelante, el Contexto hace una excepción particular a esta regla.

Una característica distintiva de esta implementación es que el dominio —representado por el contexto— toma conciencia de la capa de presentación, lo cual se aleja del enfoque clásico de desacoplamiento estricto. Esta decisión responde a una necesidad funcional concreta: el comportamiento de los controladores (y de los servicios que estos utilizan) depende del estado actual del sistema. En otras palabras, la capa de presentación es dinámica, y su lógica de actuación está determinada por el contexto del protocolo.

El contexto cumple un doble rol. Por un lado, actúa como núcleo del dominio según los principios de la arquitectura hexagonal; por otro, implementa el componente Context del patrón State, actuando como portador del estado operativo del nodo. Este mantiene una referencia al estado actual del sistema, y dicho estado mantiene a su vez una referencia a su servicio asociado, encargado de ejecutar las transiciones de estado de acuerdo con las reglas del protocolo. La implementación del patrón separa claramente la lógica de comportamiento —encapsulada en los servicios— de la lógica de orquestación y vinculación con la capa de enlace —ubicada en los estados—. En consecuencia, el comportamiento del sistema es determinístico: cada transición de estado responde a decisiones explícitas tomadas en el estado anterior, mediante la lógica definida en su servicio asociado.



Imágen 9. Primeras tres capas del modelo OSI.



Imágen 10. Arquitectura de la implementación del protocolo.

Capa de Red: El Patrón State como base estructural

La capa de red se construye por un patrón de estados distribuida, en la que los estados representan controladores que encapsulan las decisiones de orquestación del protocolo; los servicios modularizan la lógica de comportamiento específica de cada estado; los serializadores actúan como vínculo operativo con la capa de enlace; y el contexto provee un espacio compartido de memoria y coordinación, pero sin contener lógica propia.

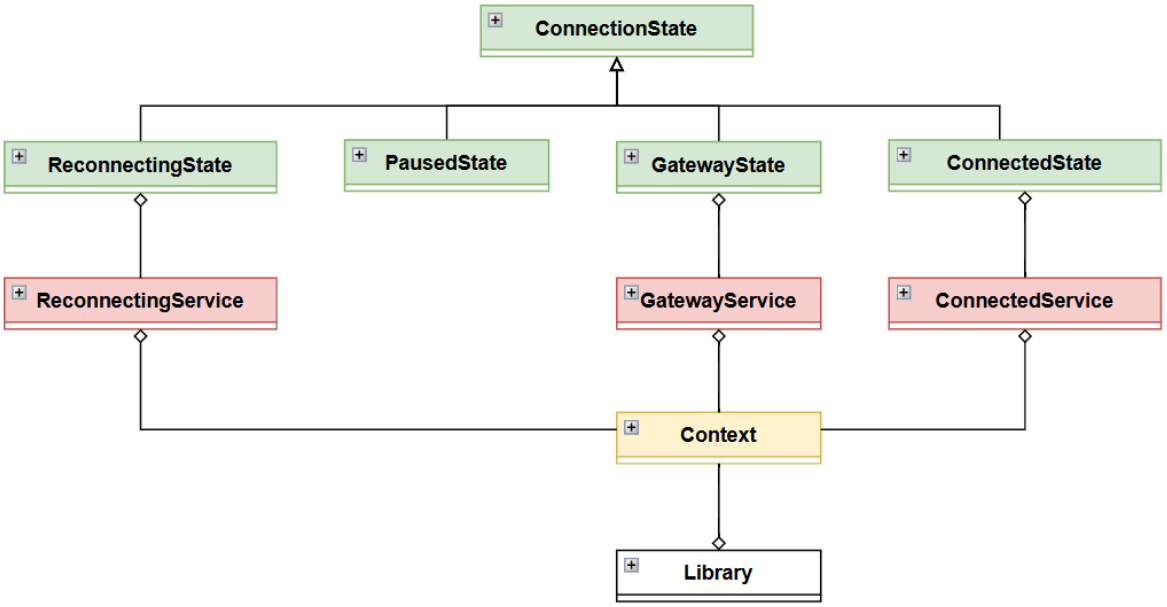
Se eligió el Patrón State como base estructural para representar esta lógica distribuida, como una forma de organizar los distintos controladores del sistema en clases independientes, coherente con la arquitectura modular planteada. Cada estado se implementa como una clase que define el comportamiento del nodo frente a distintos eventos, manteniendo alta cohesión y facilitando la evolución del sistema.

En este diseño, los estados no solo definen la respuesta a los mensajes recibidos, sino que también orquestan el flujo del protocolo según el estado operativo del nodo. El Contexto, en cambio, actúa como intermediario: mantiene referencias al estado activo y a otros componentes del sistema, y permite el cambio de estado cuando la lógica del protocolo (ubicada en los propios estados) así lo determina. No contiene lógica de orquestación propia, sino que delega completamente el control a los estados, quienes son el núcleo funcional de esta capa.

Estados y Servicios

Cada estado del protocolo modela una situación operativa concreta:

- ReconnectingState: estado inicial del nodo. Su objetivo es reestablecer conexión con el entorno de red. Maneja la detección de vecinos, la reconstrucción de la tabla de ruteo y la solicitud de parámetros operativos.
- ConnectedState: estado operativo principal. Permite enviar, reenviar y recibir paquetes, así como generar acuses de recibo. Coordina con la capa de enlace para gestionar transmisiones y verificar la validez de los encabezados.
- GatewayState: representa el rol del nodo cuando actúa como puerta de enlace. Permite recibir y redirigir mensajes de otros nodos hacia capas superiores o infraestructuras externas.
- PausedState: estado transitorio no contemplado dentro del patrón State clásico. En este estado, el nodo detiene temporalmente su actividad sin perder el historial de estado anterior, permitiendo retomar su operación desde ese punto.



Imágen 11. UML simplificado de las clases que componen la capa 3.

Cada estado mantiene una asociación directa con un objeto Service, que implementa la lógica de comportamiento modularizada del propio estado. Esta separación permite desacoplar la lógica de orquestación (propia del estado) de la lógica de procesamiento y toma de decisiones (contenida en el servicio), en un esquema análogo al que se da entre los controladores y la capa de aplicación en arquitecturas RESTful. Por esta razón, no se considera a los servicios como entidades independientes del estado, sino como extensiones funcionales del mismo.

Además, cada estado consume serializadores, que actúan como punto de enlace con la capa 2 del modelo OSI. Estos serializadores permiten interpretar, validar y descomponer los paquetes entrantes, así como generar las respuestas correspondientes. A partir de esta información, el estado toma decisiones y, mediante su servicio, determina las acciones a ejecutar.

El comportamiento del sistema ante la recepción de un mensaje sigue un patrón uniforme: el Contexto delega en el estado actual, que utiliza su servicio y los serializadores para procesar el mensaje, tomar decisiones, y eventualmente modificar el propio contexto.

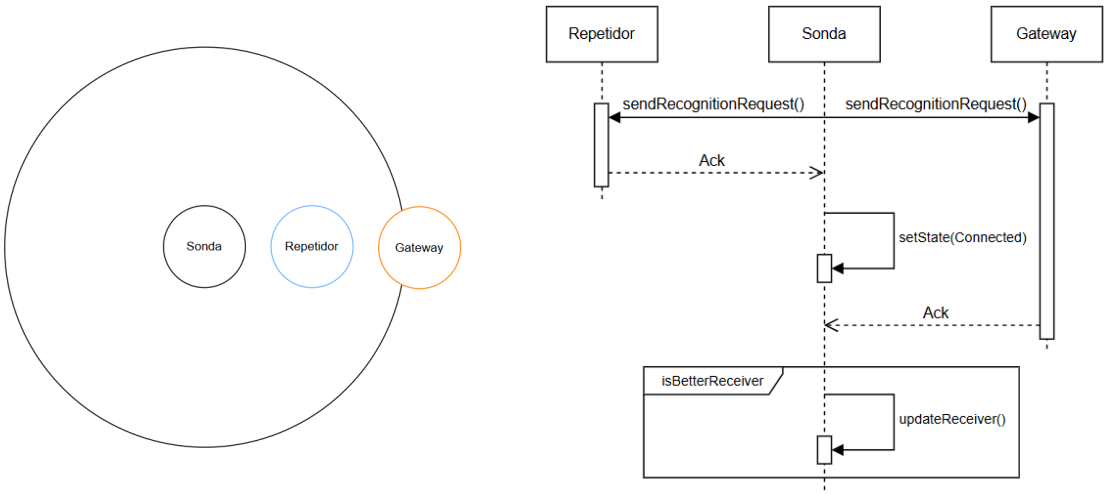
Interacción entre Contexto, Estados y Librería

El Contexto es también el punto de integración entre la lógica del protocolo y la librería de firmware. Esta última expone métodos sencillos para interactuar con el protocolo, encapsulando su complejidad interna. Para ello, mantiene una referencia al contexto, que le permite ejecutar comandos de forma directa.

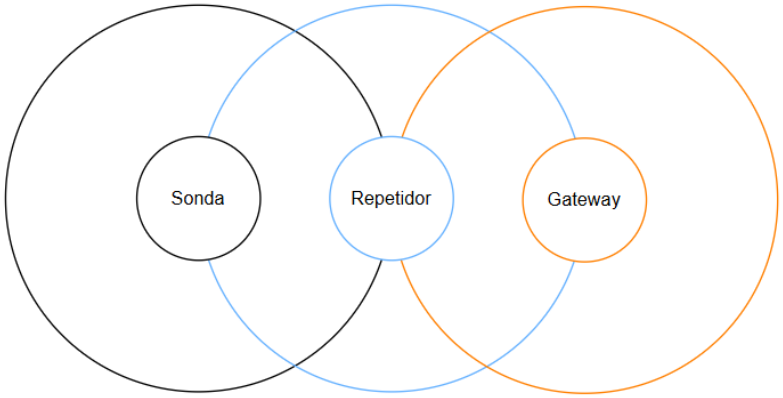
Este diseño permite que el firmware pueda utilizar funcionalidades del protocolo sin conocer su implementación interna, cumpliendo así con el principio de inversión de dependencias. Entre sus responsabilidades, la librería:

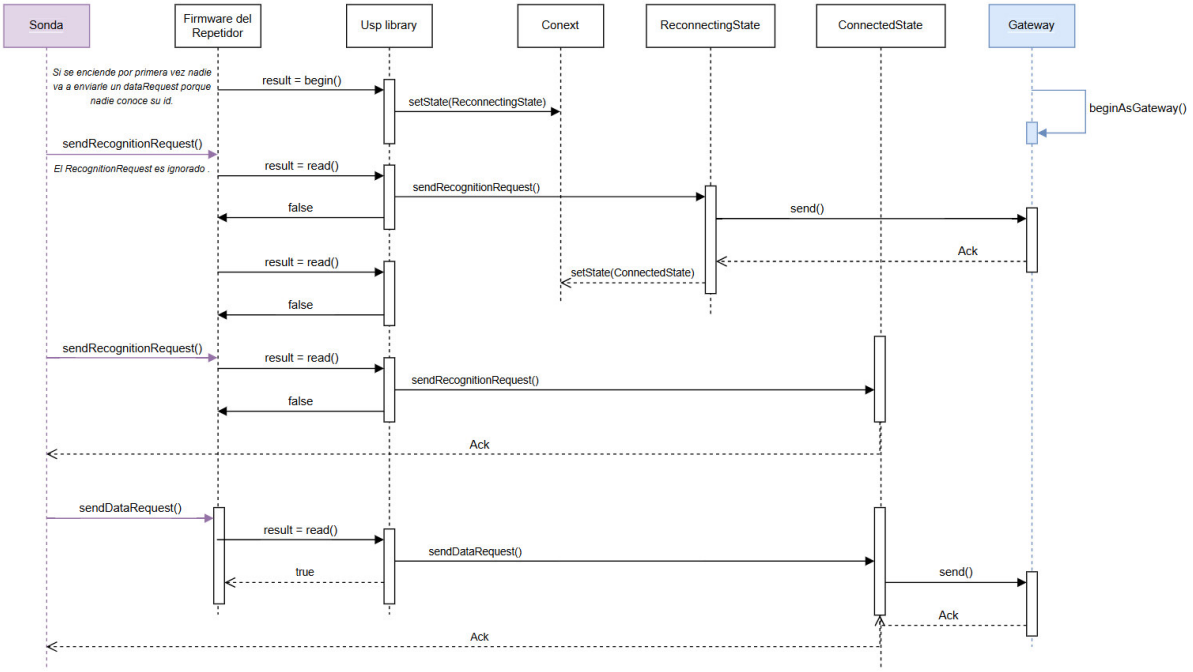
- Inicializa el contexto con el estado ReconnectingState al encender el nodo.
- Solicita servicios al contexto, que son delegados a los estados.
- Modifica el estado mediante llamadas explícitas a setState, por ejemplo, para pasar el nodo a estado PausedState frente a una orden del usuario.

Este último punto introduce una particularidad relevante respecto al patrón State clásico: en esta implementación, el cambio de estado no es exclusivo de la lógica interna del estado activo. El entorno externo (específicamente, la librería) puede forzar transiciones en función de requerimientos operativos. Este mecanismo resulta necesario para introducir capacidades de pausa y reanudación controladas por el usuario, que no podrían gestionarse únicamente desde el interior del protocolo sin violar su aislamiento conceptual.



Imágen 12. Diagrama de secuencia para la selección del receptor.





Imágen 13. Diagrama de secuencia para el camino feliz.

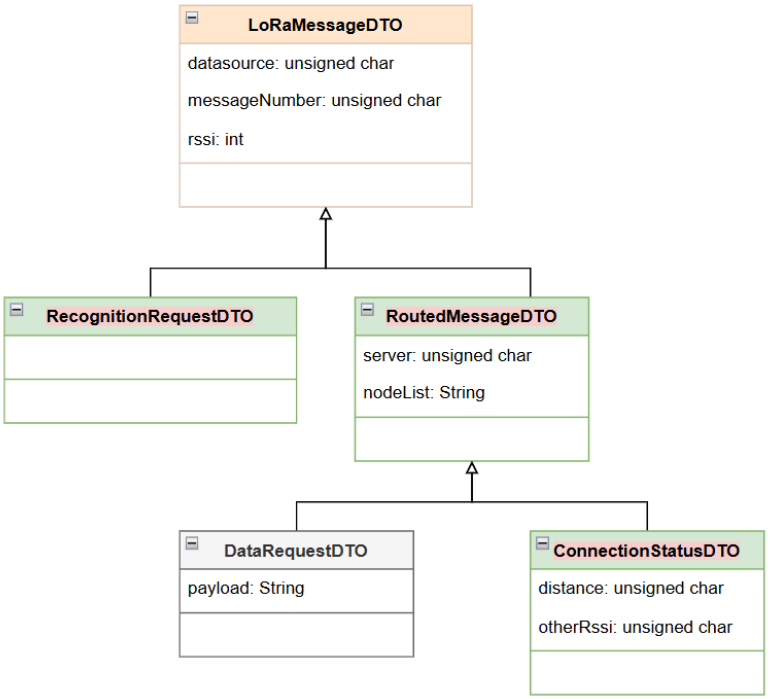
Capa de Enlace: Serialización y Validación

La capa de enlace se encuentra diseñada para encapsular toda la lógica vinculada a la transmisión, serialización y validación de mensajes. Su estructura permite mantener una separación clara entre la lógica de bajo nivel y las decisiones que se toman en la capa de red.

Jerarquía de DTOs

En el núcleo de esta capa se encuentra la clase base LoRaMessageDTO, que representa un paquete genérico intercambiado entre nodos. A partir de ella, se definen subclases especializadas que encapsulan distintos tipos de mensajes operativos:

- RecognitionRequestDTO: utilizado durante el proceso de descubrimiento de nodos.
- DataRequestDTO: usado para transportar mensajes de aplicación.
- ConnectionStatusDTO: representa el estado de conexión de un nodo.



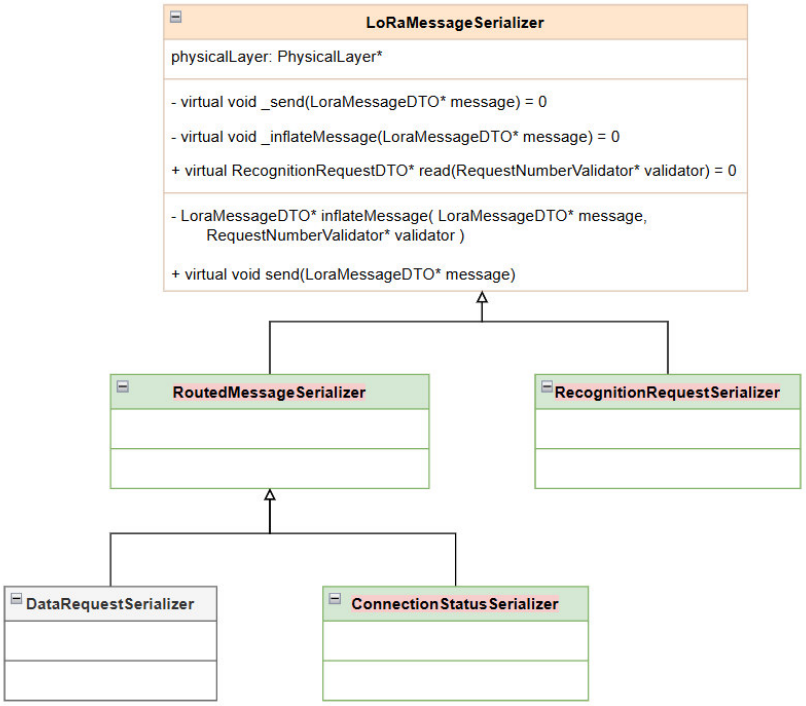
Imágen 14. UML de las clases que modelan los paquetes.

Estos DTOs permiten modelar los distintos tipos de mensaje de forma estructurada, desacoplando su definición del modo en que se transportan o manipulan.

Jerarquía de Serializadores

Cada tipo de mensaje cuenta con un serializador asociado que hereda de la clase padre LoRaMessageSerializer. Esta clase implementa el patrón Template definiendo las operaciones de serialización y deserialización necesarias para codificar y decodificar los mensajes en formato compatible con LoRa. Los hijos se encargan de implementar particularidades del DTO correspondiente que complementan el algoritmo principal.

Esta estructura permite abstraer completamente la lógica de transmisión del comportamiento del protocolo, facilitando la evolución del sistema, el testeo unitario y la incorporación de nuevos tipos de mensajes sin afectar la lógica de capa 3.



Imágen 15. UML simplificado de las clases que parsean y serializan paquetes.

Integración mediante Factory Method

La capa de enlace está compuesta por las clases responsables de gestionar la serialización, deserialización y validación de los mensajes intercambiados por los nodos. Entre ellas se encuentran:

- Los serializadores, que implementan la interfaz LoRaMessageSerializer y permite traducir entre objetos de transferencia (DTOs) y secuencias de bytes adecuadas para transmisión por LoRa.
- Validadores, como el encargado de detectar mensajes repetidos, que encapsulan lógicas de integridad propias de la comunicación punto a punto.

Estas clases no contienen lógica de enrutamiento ni toman decisiones operativas: su responsabilidad se limita al manejo correcto de la transmisión de bajo nivel, por lo que se ubican conceptualmente dentro de la capa 2 del modelo OSI.

Manteniendo la separación clara, utiliza el patrón Factory Method accesible desde las capas superiores —en particular desde los estados de la capa de red— que actúa como punto de creación de objetos de la capa de enlace. Su función es proporcionar instancias de serializadores, validadores u otros componentes de la capa de enlace según se requiera en

tiempo de ejecución. De esta manera, se evita el acoplamiento directo a clases concretas, se mantiene la flexibilidad de reemplazo por implementación y se refuerza la independencia entre capas.

Este enfoque permite a los elementos de la capa de red orquestar el protocolo haciendo uso de las funcionalidades de la capa de enlace de manera controlada y desacoplada, alineándose con los principios de diseño modular y responsabilidad única.

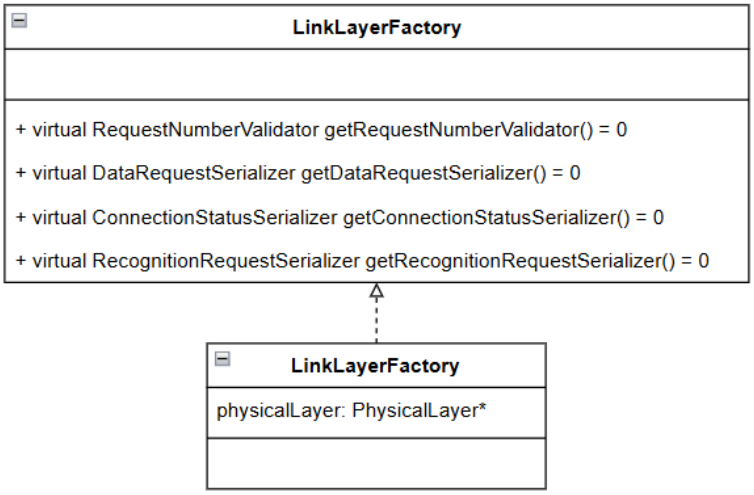


Imagen 16. UML simplificado del acceso a la capa de enlace de datos.

Capa Física: Adaptador LoRa

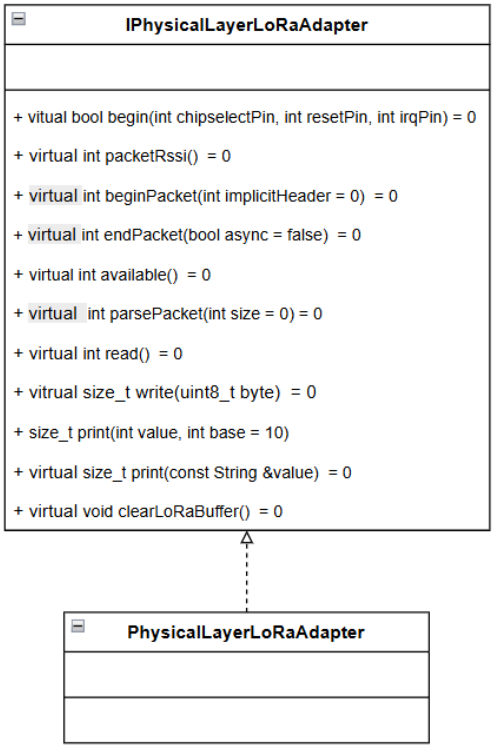
La capa física del sistema está representada por una interfaz genérica que abstrae las operaciones necesarias para enviar y recibir datos en el medio físico. Esta interfaz define los métodos mínimos que el protocolo necesita para interactuar con el hardware, sin depender de detalles específicos de implementación. La implementación se realizó en C++, habiendo que aclarar que el lenguaje no contempla interfaces como tales, aunque se pueden emplear clases abstractas.

En este contexto, PhysicalLayerLoRaAdapter actúa como un adaptador concreto encapsulando internamente el uso de la librería LoRa empleada por el firmware. Esta clase sigue el patrón de diseño Adapter, permitiendo traducir las llamadas genéricas del protocolo a las funciones específicas ofrecidas por la librería LoRa seleccionada. De esta manera, se evita

que el resto del sistema conozca o dependa directamente de dicha librería, manteniendo así la separación de responsabilidades.

El uso del patrón Adapter garantiza que el protocolo pueda operar sobre cualquier tecnología física siempre que se provea una implementación alternativa de la interfaz definida. Esto habilita, por ejemplo, reemplazar LoRa por otra tecnología de comunicación (como FSK o Zigbee) sin necesidad de modificar el código del protocolo, sino simplemente proveyendo un nuevo adaptador conforme a la misma interfaz.

Este diseño refuerza los principios de bajo acoplamiento y alta cohesión, y permite que la capa física sea sustituida, probada o extendida de manera independiente del resto del sistema, contribuyendo así a la portabilidad y mantenibilidad del código.



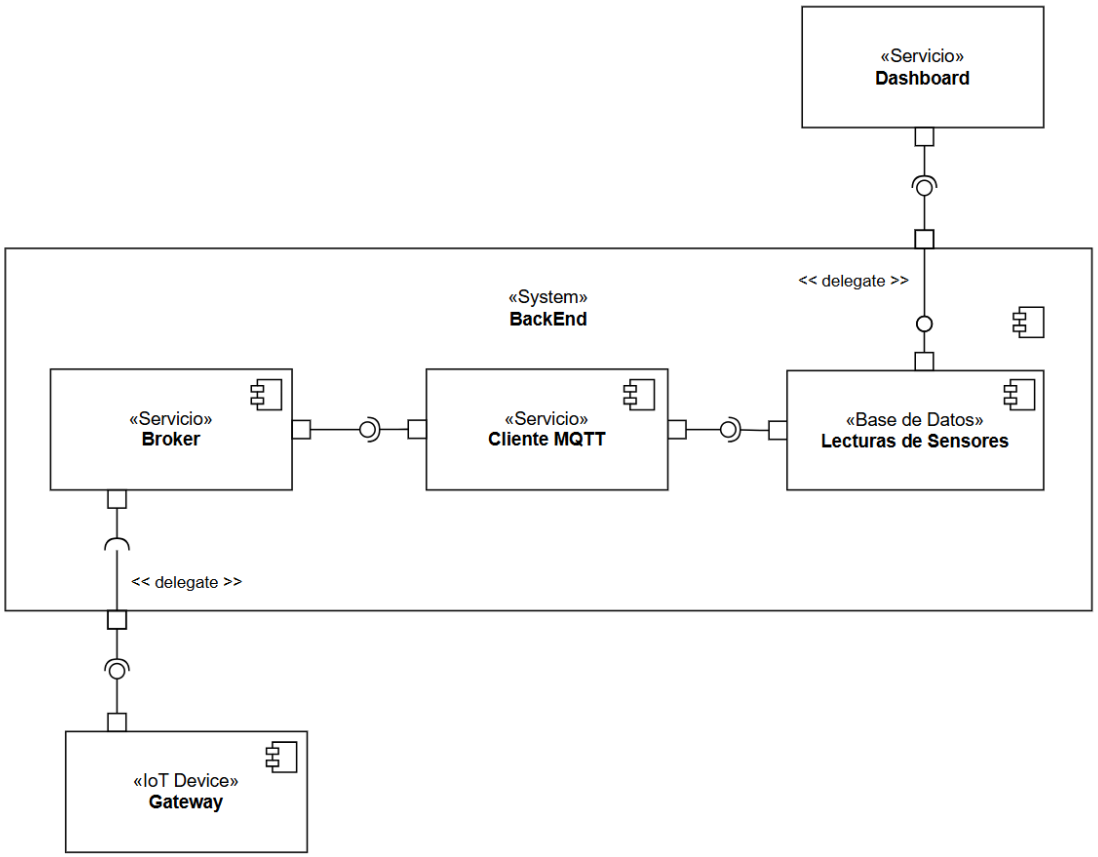
Imágen 17. UML simplificado de la capa física.

Sistema IoT

Arquitectura

El proyecto incluye una solución de monitoreo remoto de variables físicas en tiempo real que utiliza el protocolo MQTT (OASIS, 2019) para la transmisión eficiente de los parámetros sensados, con persistencia de las lecturas en una base de datos para su posterior visualización a través de dashboards interactivos.

La arquitectura propuesta se basa en la integración de diversos componentes desacoplados, lo que contribuye significativamente a la escalabilidad, mantenibilidad y adaptabilidad del sistema. La implementación se apoya en herramientas y servicios ampliamente utilizados en la industria, favoreciendo su adopción y evolución.



Imágen 18. Diagrama de componentes del Sistema IoT.

Por un lado tenemos los gateways envían la información de múltiples sondas desplegadas en las cuencas hídricas. La información es transmitida desde las sondas hacia los gateways utilizando el protocolo propietario implementado.

Se cuenta con un broker MQTT que cumple el rol central de intermediario en el modelo de comunicación publish-subscribe. Su función consiste en recibir los mensajes enviados por los dispositivos que actúan como publicadores, y distribuirlos a los componentes que se han suscrito a los topics correspondientes. De esta manera, el broker desacopla la comunicación entre emisores y receptores, permitiendo un sistema flexible y escalable.

El backend cuenta con un cliente MQTT que actúa como suscriptor a los mensajes publicados por los dispositivos IoT. Dado que el protocolo MQTT se basa en el modelo publish-subscribe, la recuperación de la información enviada por los nodos requiere de un componente que se suscriba activamente a los topics relevantes.

El último componente a destacar es un sistema configurable con gestión de usuarios que permite el análisis gráfico e histórico de los datos persistidos en una base de datos.

Esta arquitectura modular permite integrar nuevos sensores, modificar flujos de procesamiento de datos o cambiar la base de datos sin alterar el resto del sistema. En las siguientes secciones se detallarán el rol de cada componente.

Sondas y Gateway IoT (Cliente MQTT Publicador)

El sistema cuenta con un conjunto de sondas distribuidas, encargadas de medir variables físicas en campo. Estas sondas no se comunican directamente con el backend, sino que transmiten sus lecturas utilizando un protocolo propietario hacia un nodo central denominado gateway.

El gateway actúa como punto de convergencia entre la red de sondas y la infraestructura de comunicaciones basada en MQTT. Su función principal es recibir los datos enviados por las sondas, interpretarlos según el protocolo de origen, y publicarlos en el broker MQTT correspondiente. Esto implica establecer una conexión persistente con el broker, seleccionar

el topic adecuado para cada tipo de dato o fuente, y asegurar la entrega de los mensajes según la calidad de servicio (QoS) configurada.

El diseño basado en gateways intermedios permite desacoplar completamente la lógica de transmisión de las sondas del sistema de backend, otorgando flexibilidad para integrar tecnologías heterogéneas sin comprometer la estructura general. Además, facilita la segmentación por zonas, la incorporación de protocolos propietarios o especializados, y la implementación de lógicas de validación o agregación local antes de publicar los datos en la red MQTT.

Este enfoque convierte al gateway en un componente clave dentro de la arquitectura del sistema, tanto como puente entre tecnologías como también en su rol activo en la publicación confiable de datos hacia el backend.

EMQX (Broker MQTT)

Existen diversas implementaciones de brokers MQTT en el mercado, tanto de código abierto como comerciales, tales como Mosquitto, HiveMQ, RabbitMQ con soporte MQTT, entre otros. Sin embargo, EMQX (EMQX, 2025) se destaca por ofrecer un equilibrio notable entre escalabilidad, seguridad, visibilidad operativa y facilidad de despliegue, lo cual fue determinante para su elección en este proyecto.

EMQX está diseñado para soportar decenas de miles de conexiones concurrentes incluso en una sola instancia, y su arquitectura permite escalar horizontalmente mediante clústeres distribuidos. Esta capacidad de escalado lo hace especialmente apto para sistemas que pueden crecer en complejidad o en volumen de datos sin necesidad de rediseñar la infraestructura subyacente.

Uno de los aspectos que distingue a EMQX de otras soluciones es su completo panel de administración web. Esta interfaz permite monitorear en tiempo real las conexiones activas, los topics utilizados, el volumen de mensajes transmitidos y otros indicadores clave del sistema, lo que resulta fundamental para el diagnóstico de problemas y la supervisión operativa continua. A esto se suma la capacidad de definir políticas de control de flujo, establecer límites por cliente y configurar reglas personalizadas para la gestión del tráfico.

La seguridad es otro pilar fundamental de EMQX. El broker permite múltiples mecanismos de autenticación, incluyendo credenciales simples, tokens JWT, certificados digitales y

sistemas externos como LDAP. Asimismo, implementa autorización granular mediante listas de control de acceso (ACLs), lo que permite restringir qué clientes pueden publicar o suscribirse a cada topic. Para proteger las comunicaciones, EMQX soporta cifrado TLS/SSL, asegurando la confidencialidad e integridad de los datos transmitidos.

Otro factor que refuerza su elección es la extensibilidad del sistema. EMQX permite integrar plugins personalizados y definir reglas de transformación y enrutamiento de mensajes que se ejecutan directamente en el broker, sin necesidad de intervención del backend. Además, ofrece conectores nativos con bases de datos externas, colas de eventos y sistemas de procesamiento en tiempo real, lo cual facilita su integración con infraestructuras heterogéneas.

Finalmente, EMQX cuenta con una comunidad activa y bien documentada, así como una versión empresarial que extiende sus capacidades para entornos industriales críticos. Esta disponibilidad de soporte técnico y recursos formativos favorece su adopción y mantenimiento a largo plazo.

Node-RED como Cliente MQTT y Plataforma de Procesamiento

El cliente MQTT fue implementado en Node-RED (Node-Red, 2025), actuando como suscriptor de los mensajes publicados por los gateways IoT a través del broker EMQX. Además de su rol como cliente MQTT, Node-RED permite transformar, filtrar, validar o enriquecer los mensajes recibidos antes de enviarlos a otros servicios del backend. Esto incluye la posibilidad de almacenar los datos en bases de datos, invocar APIs externas, generar notificaciones o activar procesos condicionales en función del contenido de los mensajes. En este caso particular, Node-RED está configurado para suscribirse a determinados topics del broker EMQX, procesar los datos recibidos desde los gateways, y canalizarlos hacia una base de datos relacional para su almacenamiento estructurado.

Node-RED representa una herramienta flexible, accesible y poderosa para la gestión de flujos de datos. Su integración nativa con MQTT, su arquitectura orientada a eventos y su soporte para tareas de procesamiento, transformación y persistencia de datos lo convierten en una pieza clave dentro del sistema propuesto. Frente a otras alternativas más complejas o menos accesibles, Node-RED ofrece una curva de aprendizaje suave, una amplia comunidad de usuarios, y un entorno de desarrollo que favorece la iteración rápida y la adaptación a requerimientos cambiantes.

Node-RED es altamente extensible. Su ecosistema incluye miles de nodos disponibles para descarga, que permiten integrar protocolos industriales (como Modbus, OPC-UA), servicios en la nube, herramientas de análisis de datos y plataformas de visualización. También permite la ejecución de funciones personalizadas escritas en JavaScript, lo cual ofrece la flexibilidad necesaria para implementar lógica de negocio compleja cuando las funciones predefinidas no son suficientes.

Grafana como Plataforma de Visualización

La visualización de datos en tiempo real es un componente fundamental en sistemas IoT, ya que permite interpretar el comportamiento del sistema, detectar anomalías y tomar decisiones operativas basadas en información concreta. En este proyecto, la plataforma seleccionada para cumplir ese rol fue Grafana (Grafana, 2025), una solución de código abierto ampliamente adoptada en entornos industriales, científicos y de monitoreo de infraestructura.

Grafana se destaca por su capacidad para construir paneles interactivos a partir de una amplia variedad de fuentes de datos, incluyendo bases de datos SQL, sistemas de series temporales como InfluxDB o Prometheus, y APIs externas. En este caso particular, Grafana se conecta directamente a la base de datos relacional donde se almacenan los datos procesados por Node-RED, permitiendo presentar esa información de manera gráfica y dinámica.

Uno de los motivos principales para elegir Grafana fue su enfoque modular y su interfaz altamente configurable. A través de su sistema de paneles y dashboards, es posible construir visualizaciones personalizadas que se adaptan a las necesidades de cada tipo de usuario, ya sea técnico, operativo o de gestión. Cada panel puede mostrar datos en distintos formatos —como gráficos de líneas, barras, medidores, mapas de calor o tablas— y puede configurarse para actualizarse en tiempo real, permitiendo el seguimiento constante del estado del sistema.

Grafana también permite definir alertas basadas en condiciones específicas sobre los datos, lo que resulta clave en contextos donde es necesario detectar fallos, eventos fuera de rango o comportamientos anómalos. Estas alertas pueden integrarse con servicios externos como correo electrónico, Slack, Telegram o sistemas de notificación propios, proporcionando un canal inmediato de respuesta ante situaciones críticas.

Desde el punto de vista técnico, Grafana ofrece un alto grado de compatibilidad con sistemas heterogéneos, ya que su arquitectura de plugins permite extender sus capacidades para interactuar con nuevas fuentes de datos, nuevos tipos de visualización, y mecanismos de autenticación personalizados. Esta flexibilidad lo convierte en una plataforma apta no solo para monitoreo pasivo, sino también para sistemas complejos donde conviven múltiples tecnologías.

Finalmente, la elección de Grafana también se sustenta en su comunidad activa y en la disponibilidad de documentación oficial, ejemplos y foros de soporte. Esta comunidad no solo facilita la resolución de problemas, sino que también contribuye al desarrollo continuo de nuevas funcionalidades. Además, para escenarios que requieren capacidades avanzadas como autenticación corporativa, control de acceso granular o auditoría de cambios, Grafana ofrece versiones comerciales que amplían sus prestaciones sin alterar su filosofía central.

Aplicación de inteligencia artificial

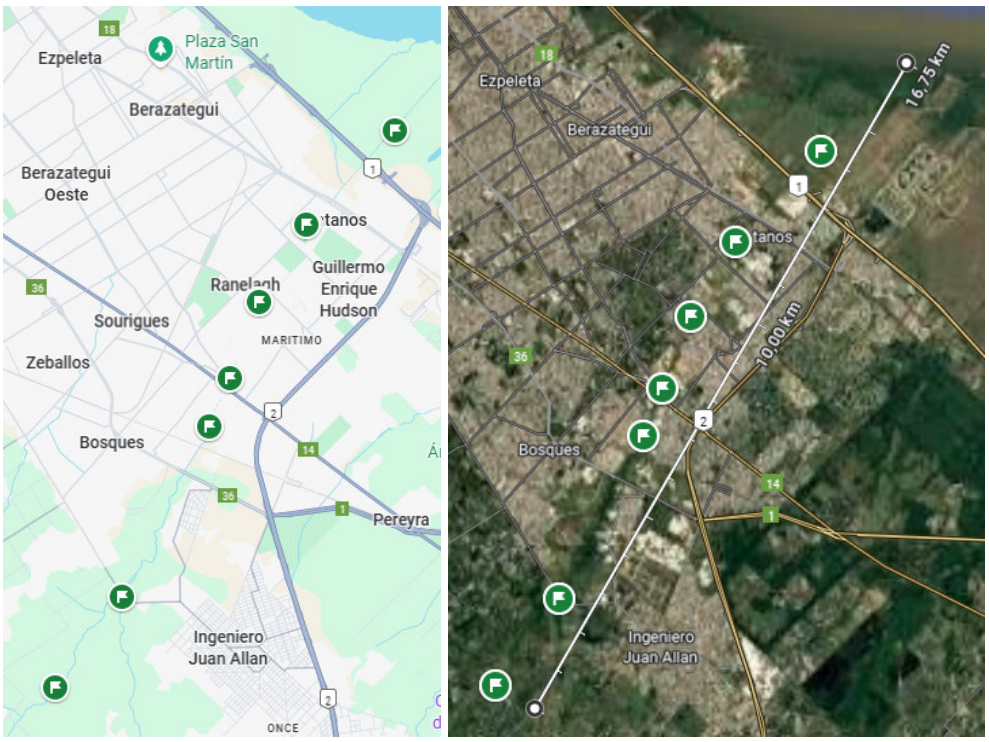
Dataset Utilizado para el Entrenamiento del Modelo de IA

El entrenamiento del modelo de inteligencia artificial diseñado en el marco de este proyecto se realizó a partir de dos conjuntos de datos provistos por investigadores del área de Inteligencia Artificial de la Universidad Nacional Arturo Jauretche (UNAJ).

Se recibieron ambos datasets de investigadores del Instituto de Limnología "Dr. Raúl A. Ringuelet" (ILPLA), dependiente del CONICET y la Universidad Nacional de La Plata. La utilización de datasets armados por personal de un instituto científico como el ILPLA garantiza un alto grado de confiabilidad en los datos, y promueve la sinergia entre diferentes instituciones públicas en el desarrollo de tecnologías orientadas al monitoreo ambiental.

Estructura del Dataset-A

El conjunto de datos está compuesto por 121 registros recolectados entre 2022 y 2023, cada uno correspondiente a una muestra puntual recolectada en el campo bajo condiciones controladas.



Imágen 19. Puntos de recolección manual de muestras.

Cada fila del dataset representa una observación o muestreo individual, e incluye un total de más de 40 variables que permiten caracterizar de forma exhaustiva la calidad fisicoquímica y microbiológica del agua. Las columnas se agrupan en distintas categorías:

Identificación y localización del sitio de muestreo:
SITIO, LATITUD, LONGITUD, FECHA, HORA.

Parámetros físico-químicos clásicos:
pH, TEMPERATURA (ºC), CONDUCTIVIDAD (uS cm^{-1}), TURBIDEZ (NTU), OD (mg/L de oxígeno disuelto) y OD (%).

Sólidos y pigmentos:
SST (mg/L de sólidos suspendidos totales), SDT (mg/L de sólidos disueltos totales), Clorofila-a (µg/L), Feofitina-a (µg/L).

Microbiología:

E. coli (CFU/100mL), Coliformes no-E. coli, Coliformes Totales, Salmonella, Serovar, ATB, E. coli ctxR, E. coli ciproR, BLEE, K. pneumoniae.

Demanda de oxígeno y nutrientes:
DQO (demanda química de oxígeno en mg O_2/L), DBO (demanda bioquímica de oxígeno en mg O_2/L), NITRATO, NITRITO, AMONIO, FOSFATO, NITRÓGENO TOTAL, POTASIO TOTAL.

Iones y metales:
$[Cl^-]$, $[SO_4^{2-}]$, $[Ca^{2+}]$, $[K^+]$, $[Mg^{2+}]$, As, Hg, Pb, Cd, Ni, Cr.

Otros indicadores ambientales:
% INHIBICIÓN, WQI (Water Quality Index).

Estructura del Dataset-B

El Dataset-B, con 552 registros, supera en volumen al Dataset-A, lo que aporta mayor solidez a los análisis realizados. Lo destacable, es que contiene las mismas variables que miden nuestras sondas: turbidez, pH, conductividad y temperatura.

El dataset-B no contiene el campo Escherichia coli, que como veremos en las próximas secciones será de mucha relevancia. Por el contrario, agrega una variable para la concentración de cianobacterias, que no estaba presente en el Dataset-A. De hecho, este conjunto de datos había sido procesado cuando se recibió. Por lo tanto, también contiene un campo que categoriza la concentración de cianobacterias.

Si bien ambos datasets fueron recolectados en el mismo periodo, entre 2022 y 2023, no fueron tomadas de la misma cuenca hídrica. Aunque sí, en zonas geográficas relativamente cercanas, pertenecientes a la zona sur del conurbano bonaerense.

Variables de interés

El conjunto de datos es la fuente fundamental para la construcción de un modelo de inteligencia artificial, para nuestro caso, capaz de clasificar muestras en función de su calidad y posible uso. No se abordó la totalidad de las variables, las principales columnas de interés

para el prototipo funcional fueron aquellas también disponibles a través del sistema IoT (pH, temperatura, conductividad y turbidez), lo cual permite establecer una correlación con las mediciones en tiempo real.

Si bien inicialmente se consideró la utilización del índice WQI (Water Quality Index) como etiqueta para la clasificación supervisada, finalmente se optó por los valores de Escherichia coli (CFU/100 mL) como variable objetivo. Esta decisión fue tomada a partir de recomendaciones de especialistas en áreas vinculadas a la calidad del agua quienes aportaron criterios desde la lógica de negocio y del análisis ecológico del problema, priorizando su relevancia sanitaria y su representatividad como indicador de contaminación microbiológica.

Dada la cantidad de registros, se utilizaron también las columnas DQO mg O2/litro, DBO mg O2/litro, FOSFATO mg L-1, N TOTAL mg L-1y P TOTAL mg L-1. Una vez validadas las categorías se procedió a construir el modelo definitivo cuyas entradas son las variables censadas y sus salidas las categorías en cuestión.

Tratamiento de los Datos

Se generó un nuevo dataset filtrado, incluyendo exclusivamente las columnas seleccionadas como relevantes para los objetivos del modelo de clasificación bajo los criterios antes mencionados. Para asegurar la trazabilidad de las transformaciones aplicadas, cada etapa del preprocesamiento fue documentada mediante la creación de nuevas hojas y planillas de cálculo, conservando versiones intermedias del dataset. Esta práctica permite auditar los cambios realizados y facilitar futuras modificaciones sin comprometer la integridad de los datos originales.

Durante la revisión exploratoria del conjunto de datos, se identificó la presencia de valores ausentes en algunas columnas numéricas continuas. Para abordar este problema, se aplicó un enfoque de imputación basado en el promedio aritmético de cada variable, considerando únicamente los registros válidos para el cálculo. Esta técnica, si bien es sencilla, resulta apropiada cuando la proporción de datos faltantes es baja y los datos no presentan una dispersión significativa, minimizando el riesgo de distorsión en los resultados del modelo.

Se descartaron técnicas más complejas de imputación (como KNN o interpolación por regresión múltiple) debido a la cantidad limitada de registros y a la prioridad por mantener un enfoque simple y reproducible en esta etapa exploratoria del desarrollo. Se dejó documentado que para una implementación futura se podrían considerar estrategias más robustas para el tratamiento de valores faltantes.

Además, con el objetivo de facilitar su posterior uso como variables de entrada en el modelo, se realizó la transformación de datos categóricos nominales en identificadores numéricos. En particular, se asignó un código numérico único a cada sitio de muestreo (representando las distintas ubicaciones geográficas) y otro a cada fecha de recolección, convirtiéndolos en variables discretas tratables por los algoritmos de clasificación. Esta codificación permitió incorporar la dimensión espacio-temporal de las muestras en el proceso de entrenamiento, sin introducir ambigüedades semánticas propias de los formatos de texto o fecha. Aunque, como se verá más adelante, el modelo final solamente empleó como entradas las variables censadas por las sondas.

Una vez tratadas las inconsistencias y seleccionadas las variables relevantes, se procedió a estructurar los datos en formato CSV (Comma-Separated Values). Este formato, de texto plano y estructurado por delimitadores, es ampliamente compatible con bibliotecas como pandas, numpy y scikit-learn, utilizadas en el entorno de desarrollo del proyecto.

El archivo CSV fue luego cargado en un notebook de Google Colab, plataforma basada en Jupyter que permite ejecutar código Python en la nube. Este entorno fue elegido por su facilidad de integración con Google Drive, su soporte para aceleración por hardware (GPU/TPU), por permitir el trabajo remoto y facilitar el seguimiento de los tutores.

Este flujo de trabajo permitió dejar el conjunto de datos listo para el entrenamiento del modelo supervisado, respetando buenas prácticas de reproducibilidad, limpieza de datos y separación entre datos vistos y no vistos durante el aprendizaje.

Análisis exploratorio del dataset-A

Análisis Exploratorio de Correlaciones

Como parte del análisis exploratorio de datos, se construyó una matriz de correlación entre las variables numéricas del conjunto de datos, con el objetivo de identificar patrones de asociación relevantes que orienten tanto la comprensión del fenómeno como la posterior selección de atributos para el modelo de inteligencia artificial.

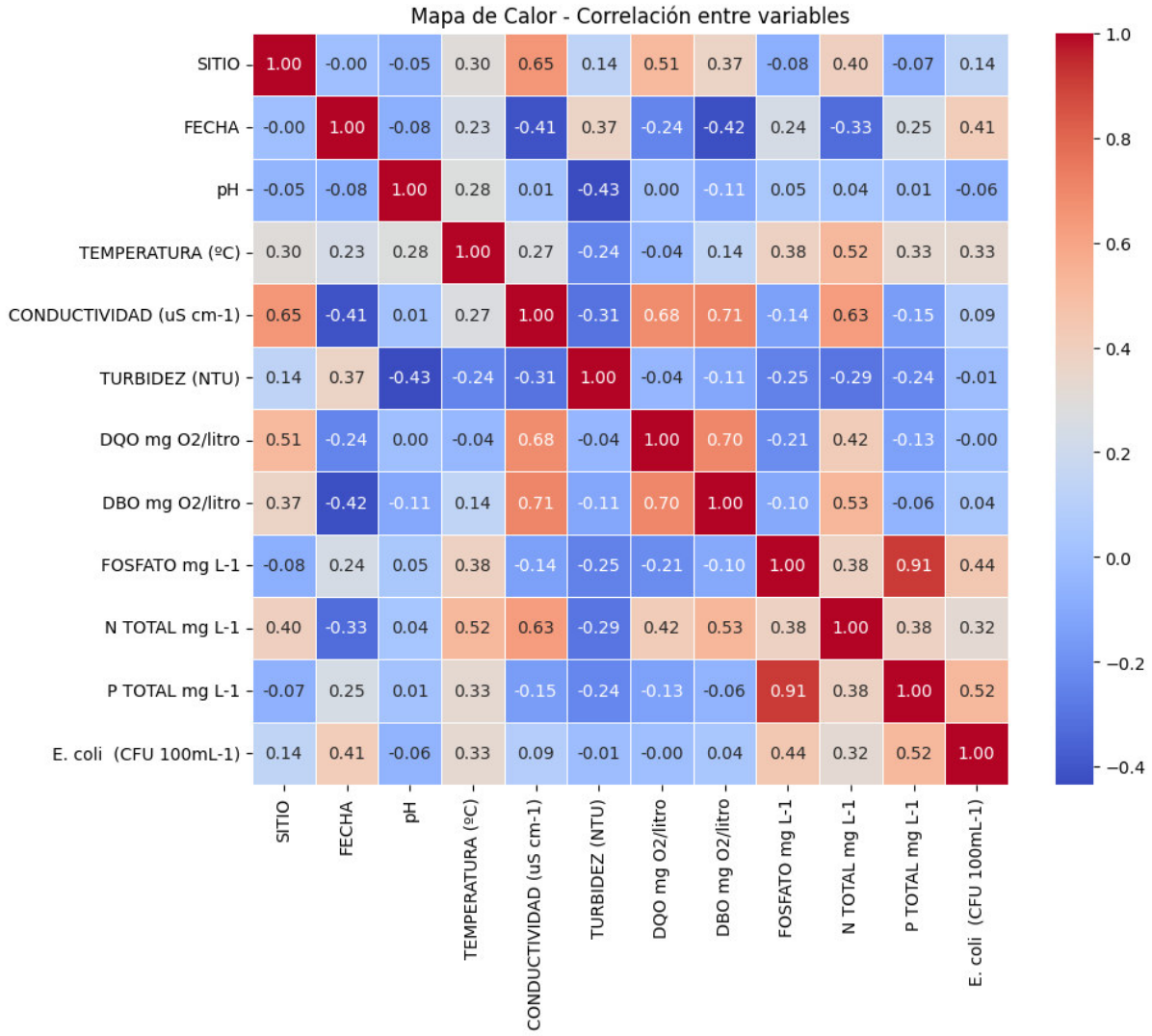
La representación gráfica de dicha matriz mediante un mapa de calor permitió detectar relaciones estadísticamente significativas entre ciertos parámetros fisicoquímicos y microbiológicos de calidad de agua.

Entre los hallazgos más destacados se observan correlaciones altas y positivas entre variables que responden a mecanismos comunes de contaminación. Por ejemplo, se verificó una fuerte relación entre Fosfato y Fósforo Total ($r = 0.91$), coherente con su composición química, y entre DBO y DQO ($r = 0.70$), indicadores clásicos de carga orgánica. También se identificaron asociaciones relevantes entre Conductividad y DBO ($r = 0.71$), así como entre Temperatura y DBO ($r = 0.71$), lo que sugiere una mayor carga orgánica en condiciones de alta temperatura o salinidad.

En el rango de correlaciones moderadas, se destaca la relación entre E. coli y variables como Fósforo Total ($r = 0.52$) y Fosfato ($r = 0.44$), lo que podría reflejar una interacción entre contaminación microbiológica y nutrientes asociados a vertidos cloacales o actividad antrópica. Asimismo, E. coli y Temperatura ($r = 0.33$) mostraron una asociación consistente con lo reportado en literatura sobre proliferación bacteriana en medios cálidos.

Por otro lado, algunas variables como pH y turbidez presentaron bajas correlaciones con el resto de los parámetros, lo que sugiere que su comportamiento no responde directamente a los mismos factores que afectan a los indicadores de contaminación orgánica. El pH, en particular, mostró correlaciones negativas leves con DBO y DQO, lo que podría interpretarse como una ligera tendencia hacia la acidificación en presencia de mayor carga orgánica, aunque sin un vínculo estadísticamente fuerte.

Finalmente, se detectó una correlación negativa entre Conductividad y la variable Fecha (r = -0.41), lo que podría deberse a efectos estacionales —por ejemplo, dilución de sales disueltas durante periodos de lluvia—, reforzando la importancia de considerar el componente temporal en la modelización.



Imágen 20. Mapa de calor del dataset A.

Consideraciones para Modelos de IA

Desde el punto de vista del modelado con técnicas de aprendizaje automático, el análisis de correlaciones cumple un rol importante en la selección de atributos. Cuando dos variables presentan una correlación muy alta, pueden ser consideradas redundantes desde el punto de

vista informativo, ya que aportan contenido similar al modelo. En estos casos, se puede optar por conservar una sola de ellas para reducir la complejidad y evitar sobreajuste (overfitting).

Sin embargo, la presencia de correlación no siempre implica que una variable deba eliminarse. Algunas técnicas de aprendizaje, como los árboles de decisión, son robustas frente a la colinealidad, mientras que otras, como la regresión lineal o los modelos basados en distancias (e.g., KNN), pueden verse afectadas negativamente. Por ello, el análisis de correlación constituye una herramienta de diagnóstico que debe complementarse con evaluaciones de desempeño durante la validación del modelo.

En esta etapa exploratoria, se utilizó la matriz de correlación principalmente para identificar relaciones subyacentes entre variables y guiar la comprensión del fenómeno, sin realizar aún una reducción automática de atributos. No obstante, los patrones identificados ofrecen una base sólida para decisiones futuras relacionadas con la simplificación del modelo, el análisis de importancia de variables y la detección de posibles sesgos estructurales en los datos.

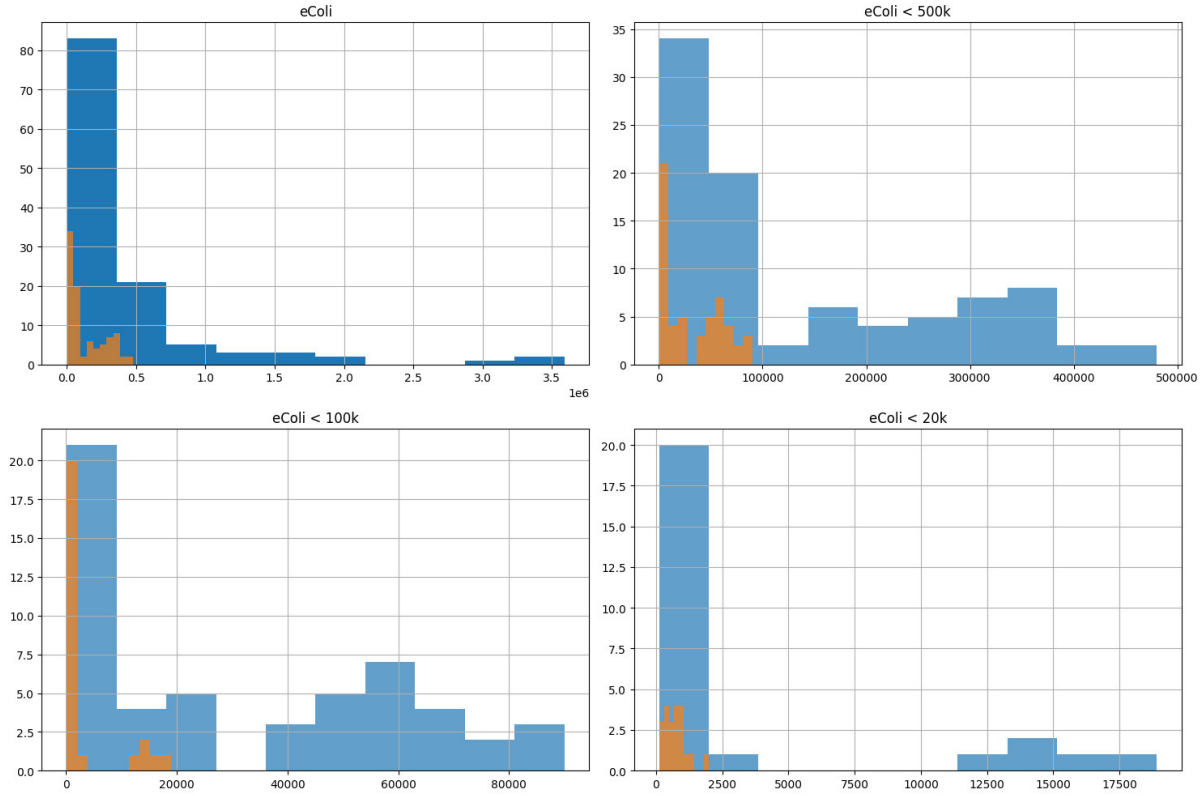
Definición de Categorías para la Clasificación de E. Coli

Con el objetivo de construir un modelo capaz de clasificar muestras en función de su nivel de contaminación microbiológica, se abordó un análisis sistemático del comportamiento de la variable E. coli. Este proceso incluyó la exploración visual mediante histogramas y diagramas de caja (boxplots), así como la evaluación de diferentes esquemas de particionado que dieran lugar a categorías útiles y representativas.

Exploración inicial

La distribución de valores de E. coli mostró una notable asimetría positiva, con una gran concentración de observaciones en valores bajos y una cola extendida hacia niveles muy elevados. Para analizar esta estructura, se aplicó un enfoque progresivo de "zoom" sobre los valores inferiores del histograma. A medida que se restringía el rango observado, se observaba un patrón recurrente de concentración inicial seguido de dispersión creciente, en una dinámica que evocaba cierta autosimilaridad o comportamiento fractal.

Esta característica justificó la búsqueda de particiones que revelaran distribuciones internas más uniformes o normales dentro de subconjuntos del dominio original. Se identificó que los rangos más altos presentan una caída marcada en frecuencia, lo que sugiere una distribución decreciente hacia el infinito. En contraste, los rangos más bajos presentaban comportamientos más simétricos y balanceados.



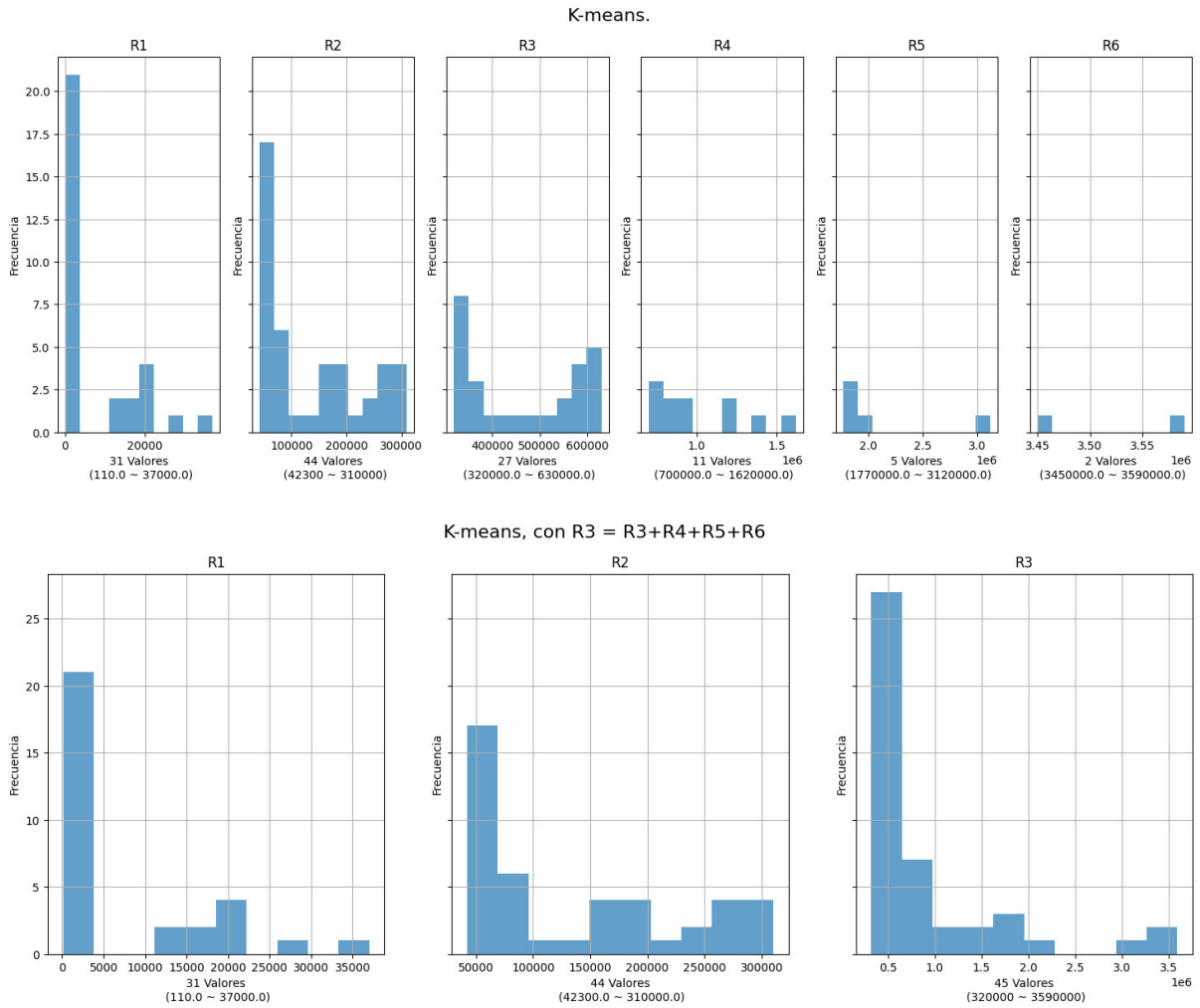
Imágen 21. Histogramas de los valores para E. Coli.

Exploración complementaria mediante K-means

Como estrategia adicional, se aplicó el algoritmo K-means con el objetivo de identificar agrupamientos inherentes en la distribución de los valores. Sin embargo, los grupos resultantes reproducen una estructura ya evidenciada en los análisis de los histogramas previos, caracterizada por una fuerte concentración en los valores bajos y una disminución progresiva en las frecuencias hacia los valores altos.

Esta situación expuso un patrón donde no solo las magnitudes disminuyen, sino también la cantidad de elementos por grupo, lo que podría inducir una clasificación artificial basada en una baja densidad más que en una segmentación estadísticamente significativa. Por esta

razón, se optó por no continuar con este enfoque como base principal para la definición de clases.



Imágen 22. Categorías con K-Means.

Búsqueda de categorías

A partir del análisis inicial se definieron, como punto de partida (posteriormente se le aplicaron diversas transformaciones), dos esquemas de categorización:

- Un esquema intuitivo, basado en potencias de diez, con rangos denominados Muy Bajo, Bajo, Moderado, Alto y Muy Alto, siguiendo criterios sanitarios esperables.
- Un esquema balanceado, construido mediante la redistribución de los límites del esquema anterior, con el objetivo de lograr una cantidad similar de observaciones en cada clase.

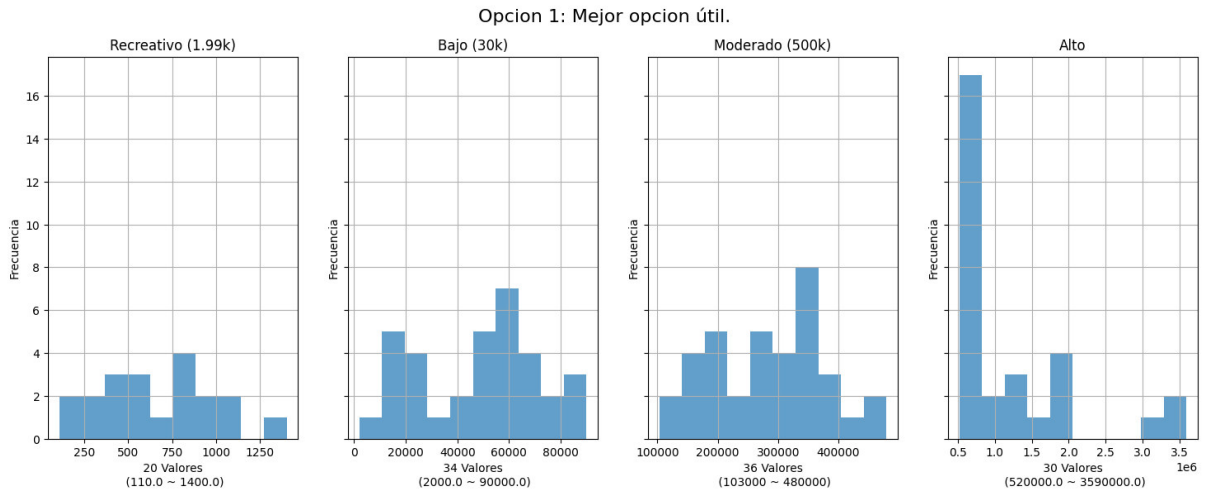
Ambos conjuntos de rangos fueron analizados principalmente con histogramas, para identificar subestructuras internas dentro de cada grupo, como subclusters o valores atípicos. En este proceso se identificaron categorías con distribuciones aproximadamente normales, ya sea por la unión de rangos adyacentes o por la segmentación de agrupamientos internos sugeridos por los propios datos.

A lo largo del análisis, se consideró no solo la solidez estadística de los rangos propuestos, sino también su utilidad desde una perspectiva práctica, considerando el monitoreo sanitario, la toma de decisiones operativas y la interpretación por parte de perfiles no técnicos.

Finalmente, el análisis con boxplots permitió cerrar el proceso de ajuste, al validar la simetría o asimetría de las distribuciones, identificar valores atípicos y evaluar la homogeneidad dentro de cada categoría. Estos elementos sirvieron como criterio adicional para confirmar o revisar las divisiones propuestas.

Propuestas de categorización

Esquema ideal (de cuatro clases): Proporciona una categorización detallada, con un primer rango orientado al uso recreativo (valores muy bajos), y niveles progresivos de contaminación hasta llegar a un rango superior abierto. Esta propuesta presenta cardinalidades razonablemente balanceadas, con la excepción del rango recreativo, cuya menor frecuencia se considera aceptable dado su significado práctico.



Imágen 23. Esquema de cuatro categorías para E. Coli.

Esquema reducido (de dos clases): Alternativa pensada para contextos con escasez de datos o necesidad de simplificación. Los rangos fueron ajustados para lograr una distribución equilibrada que conserve valor informativo, aunque con menor granularidad. Esta clasificación estaba prevista como recurso de respaldo en caso de que el modelo no obtuviera buenos resultados con la clasificación original, priorizando la robustez derivada de conjuntos más numerosos.

Modelado y evaluación de clasificadores

Para predecir las categorías definidas, se aplicaron dos modelos de clasificación supervisada: HistGradientBoostingClassifier y MLPClassifier. En el caso del Dataset-A, ambos modelos fueron entrenados utilizando dos esquemas de clasificación: uno con las cuatro categorías originales y otro con un esquema reducido que agrupa las dos primeras categorías. Para el Dataset-B, se emplearon igualmente dos esquemas, uno reducido de dos clases y otro extendido de cuatro clases, permitiendo así comparar el desempeño de los modelos según el nivel de granularidad en la clasificación.

Para todos los casos el procedimiento general consistió en dividir los datos en conjuntos de entrenamiento y testeo con estratificación por clases, entrenar el modelo y evaluar su desempeño sobre el conjunto de prueba. Se observaron mejoras al aplicar técnicas de normalización o estandarización, particularmente en configuraciones con múltiples clases.

Descripción del MLPClassifier

El MLPClassifier (Multilayer Perceptron Classifier) es un algoritmo de aprendizaje supervisado basado en redes neuronales artificiales. Pertenece a la clase de modelos feedforward, en los cuales la información fluye en una única dirección: desde la capa de entrada, a través de una o más capas ocultas, hasta la capa de salida.

Cada neurona en la red realiza una combinación lineal de sus entradas, seguida de una función de activación no lineal (comúnmente ReLU o función logística), lo que permite al modelo aprender relaciones complejas y no lineales entre los atributos y las clases objetivo. El aprendizaje se realiza mediante retropropagación del error (backpropagation), ajustando

los pesos internos de la red con el objetivo de minimizar una función de pérdida (usualmente la entropía cruzada en clasificación).

Una característica destacable del MLPClassifier es su sensibilidad a la escala de los datos, lo que hace recomendable aplicar transformaciones como la normalización (MinMaxScaler) o la estandarización (StandardScaler) para asegurar una convergencia más eficiente y evitar que algunas características dominen el proceso de aprendizaje.

En el contexto de este trabajo, el MLPClassifier se destaca por su capacidad de modelar patrones complejos en entornos no lineales y su flexibilidad para adaptarse a distintas configuraciones del problema de clasificación. Sin embargo, debido a su naturaleza paramétrica y su dependencia de una correcta inicialización y configuración (número de capas, cantidad de neuronas, función de activación, etc.), se lo aplicó con distintos preprocesamientos y evaluaciones experimentales para garantizar su efectividad en este caso de estudio.

Durante la etapa inicial el modelo fue probado bajo tres configuraciones: entrenamiento directo sin preprocesamiento, entrenamiento con los atributos estandarizados mediante StandardScaler y entrenamiento con los atributos normalizados mediante MinMaxScaler. Se entrenó el modelo y se evaluó su desempeño sobre el conjunto de prueba, observando tanto el porcentaje de aciertos como las probabilidades de predicción para casos individuales.

Descripción del HistGradientBoostingClassifier

Es una variante optimizada del algoritmo de Gradient Boosting que hace uso de histogramas para representar las características. Este enfoque permite discretizar los valores continuos en bins, reduciendo el costo computacional y acelerando el proceso de búsqueda de divisiones óptimas en los árboles de decisión que componen el modelo.

Gradient Boosting es una técnica de aprendizaje ensamblado (ensemble learning) que construye un modelo fuerte a partir de múltiples modelos débiles —en este caso, árboles de decisión de baja profundidad— entrenándolos secuencialmente. Cada nuevo árbol se ajusta para corregir los errores cometidos por la suma de los árboles anteriores, mediante la minimización del gradiente de una función de pérdida (como la log loss en clasificación).

El HistGradientBoostingClassifier, parte de la biblioteca sklearn.ensemble, incorpora mejoras como soporte automático para variables categóricas, manejo eficiente de valores faltantes y regularización incorporada. Estas características lo hacen particularmente adecuado para tareas de clasificación donde se desea equilibrio entre precisión y escalabilidad, sin requerir un preprocesamiento exhaustivo.

En este estudio, se destacó debido a su robustez frente a valores extremos, su capacidad para capturar relaciones no lineales y su excelente rendimiento incluso con esquemas de clases no balanceadas.

En este caso, la evaluación inicial se realizó con la métrica ROC AUC, tanto en la variante binaria como multiclase.

Evaluación con y sin valores extremos

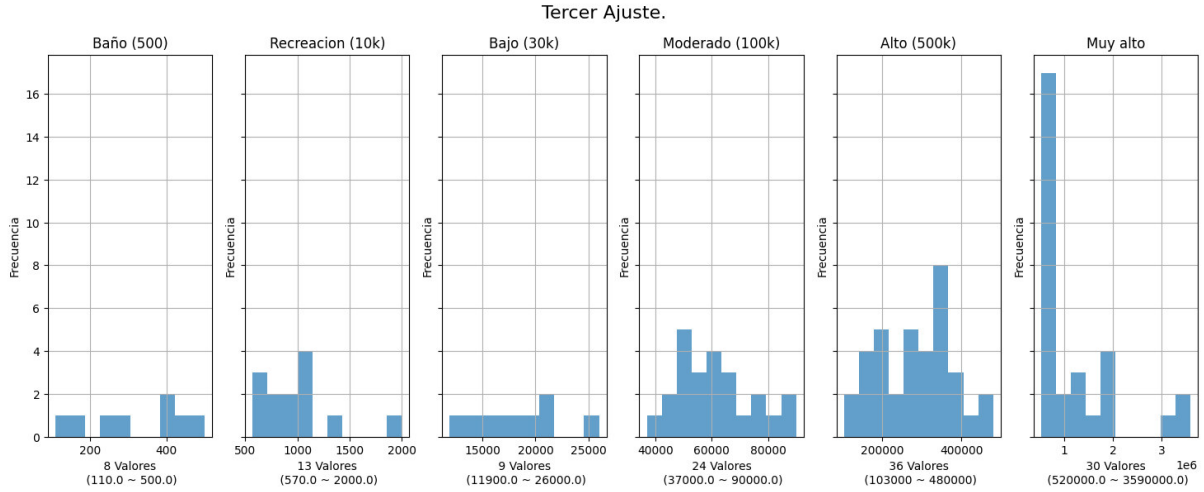
Se realizaron pruebas tanto incluyendo como excluyendo valores extremos detectados en los intervalos de las variables predictoras. Este análisis permitió evaluar la robustez del modelo frente a outliers. En general, se observó una baja diferencia en la precisión al excluir estos valores, aunque en ciertos casos los resultados fueron prácticamente similares, lo que sugiere una tolerancia moderada-alta del modelo a valores atípicos.

Comparación con categorías generadas por K-means

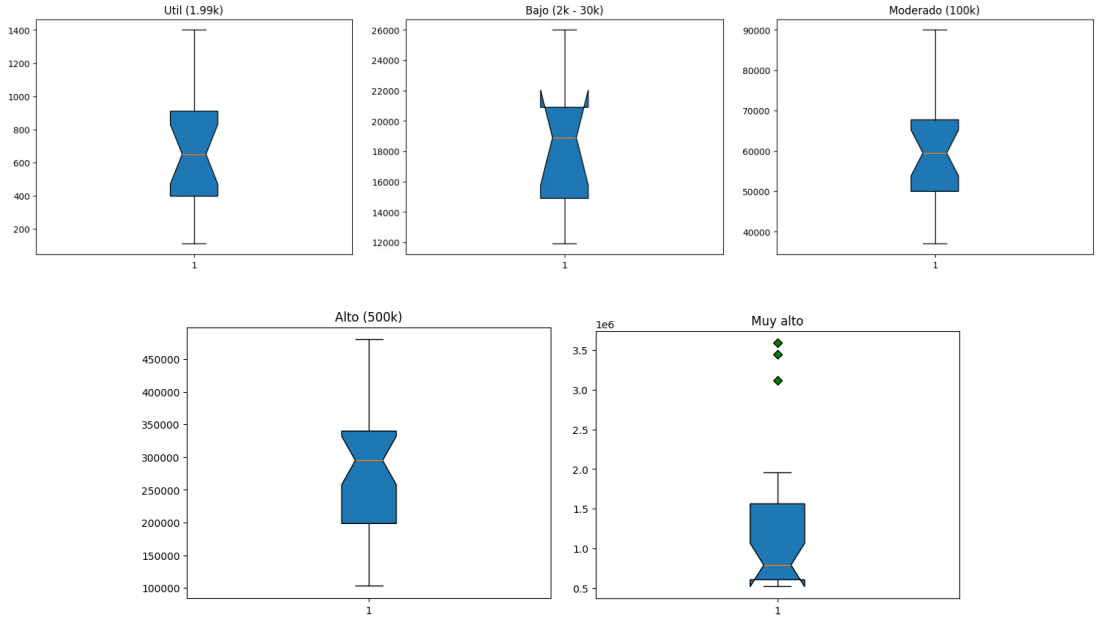
También se probó el desempeño de los modelos sobre las clases obtenidas mediante el agrupamiento K-means. Usando MLPClassifier, tal como se anticipa a partir del análisis exploratorio, los resultados fueron significativamente inferiores respecto a los esquemas basados en histogramas, reflejando que los agrupamientos derivados de K-means no capturan adecuadamente la estructura estadística ni la relevancia práctica de los datos.
Sin embargo, con HistGradientBoostingClassifier se logró buenos resultados, con una diferencia menor respecto a los esquemas diseñados manualmente. Esto indica que el modelo de boosting tiene una mayor capacidad para adaptarse a distribuciones de clases menos estructuradas.

Ajuste y evolución del esquema de categorías

En base a los resultados de los modelos, se propusieron ajustes al esquema original de cuatro clases para el datasetB. Este proceso derivó en un esquema alternativo de seis clases, cuya distribución no es completamente balanceada en términos de cardinalidad. Sin embargo, dicha versión extendida demostró ser capaz de capturar mayor riqueza en la variabilidad de los datos, logrando una segmentación más útil para aplicaciones específicas, a pesar del costo de la desigualdad entre las clases.



Imágen 24. Esquema de seis categorias para E. Coli.



Imágen 25. Boxplot para las primeras cinco categorías.

Optimización del modelo

Matriz de Confusión

Se comenzó construyendo la matriz de confusión, una herramienta fundamental para interpretar el comportamiento del modelo. Esta matriz resume visualmente las predicciones correctas e incorrectas, comparando las etiquetas reales con las predichas por el clasificador.

En el contexto de este trabajo, permitió identificar cuáles clases tienden a solaparse y en qué medida. Por ejemplo, se observaron casos en los que valores reales catalogados como "Moderado" eran predichos como "Alto", revelando fronteras difusas entre clases adyacentes en términos cuantitativos.

Este tipo de error es esperable en escenarios donde las fronteras entre clases se definen por rangos de valores continuos. La matriz permitió así no solo medir la proporción de aciertos, sino también evidenciar la estructura interna de los errores, proporcionando una base sólida para analizar si estos errores eran aleatorios o sistemáticos.

Además, la matriz se utilizó de manera comparativa entre los modelos entrenados sobre distintas categorizaciones del conjunto de datos (cuatro y seis clases), permitiendo analizar cómo varía la capacidad de discriminación de los modelos según el nivel de granularidad de las clases.

Curvas ROC y Precisión-Recuperación

Para complementar la información provista por la matriz de confusión, se analizaron dos tipos de curvas ampliamente utilizadas en clasificación: la curva ROC (Receiver Operating Characteristic) y la curva de Precisión-Recuperación. Aunque estas herramientas son más habituales en contextos binarios, en este trabajo se adaptaron mediante técnicas como la estrategia "uno contra el resto" (OvR) y "uno contra uno" (OvO) para extender su aplicación al caso multiclase.

La curva ROC muestra la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR), lo cual es especialmente útil para visualizar la capacidad de un

modelo de distinguir entre clases en diferentes umbrales de decisión. En problemas con clases balanceadas, el área bajo la curva (AUC) proporciona una medida integral del desempeño. En nuestro caso, aunque algunas clases presentaban diferencias en su frecuencia, la curva ROC fue útil para comparar enfoques de modelado alternativos.

Por otro lado, la curva de Precisión-Recuperación (PR) fue particularmente reveladora en contextos con clases menos representadas. Esta curva ilustra cómo varía la precisión del modelo a medida que cambia la sensibilidad (recall), y permite evaluar con mayor fidelidad qué tan efectivo es un modelo cuando se enfrenta a clases minoritarias. Dado que algunas clases como "Muy alto" o "Recreativo" estaban escasamente representadas, esta visualización fue clave para identificar si los modelos estaban siendo verdaderamente capaces de capturar correctamente estos casos.

Validación Cruzada

Para estimar el rendimiento de los modelos de clasificación en condiciones distintas a las del conjunto de entrenamiento, se aplicaron tres técnicas de validación cruzada: K-Fold, Stratified K-Fold y Leave-One-Out (LOO). Estas técnicas permiten medir la capacidad de generalización del modelo sin requerir un conjunto de prueba separado, reutilizando el conjunto de datos disponible de manera controlada.

Para todas las combinaciones de modelo y técnica de validación, se evaluó el rendimiento utilizando como métrica principal la exactitud promedio (accuracy), complementada con el desvío estándar para cuantificar la variabilidad entre iteraciones. Para el caso del dataset-B, este análisis se realizó tanto para la categorización en 4 clases como para la categorización en 6 clases, permitiendo evaluar no sólo la performance media, sino también la estabilidad del modelo bajo distintas particiones del conjunto de datos.

K-Fold Cross Validation

En el enfoque K-Fold, el conjunto de datos se divide en K subconjuntos de tamaño similar. En este trabajo se utilizó K = 5. En cada iteración, uno de los subconjuntos se utiliza para validación, mientras que los K - 1 restantes se usan para entrenar el modelo. El proceso se repite K veces, rotando el subconjunto asignado a la validación, y los resultados se promedian para obtener una métrica final.

Este método se utiliza ampliamente debido a su eficiencia computacional y a que proporciona una estimación del rendimiento promedio del modelo en distintos subconjuntos del dominio de entrada.

Stratified K-Fold

Una limitación del esquema K-Fold estándar es que no garantiza la misma proporción de clases en cada pliegue, lo que puede ser problemático en casos de desbalance de clases. Para evitar este problema, se empleó también Stratified K-Fold, una variante que asegura que la distribución de clases en cada partición sea similar a la distribución global del conjunto de datos.

Esto resulta particularmente relevante en problemas de clasificación donde algunas clases tienen una baja frecuencia relativa. Al preservar la proporción de clases, esta técnica permite obtener estimaciones más representativas del comportamiento del modelo en presencia de clases poco frecuentes.

Leave-One-Out (LOO)

La técnica Leave-One-Out consiste en utilizar una única instancia como conjunto de validación y el resto de los datos como conjunto de entrenamiento. Este procedimiento se repite tantas veces como instancias haya en el conjunto original. Es una técnica exhaustiva, que permite analizar el comportamiento del modelo ante mínimas variaciones en los datos de entrenamiento.

Se minimizó su uso tanto como se pudo debido a la cantidad de tiempo de procesamiento que requería su empleo. Dado que requiere entrenar el modelo una vez por cada instancia, su costo computacional es elevado. No obstante, fue posible aplicarla en este caso debido al tamaño moderado del conjunto de datos. LOO ofrece una estimación del error con baja varianza, aunque puede verse afectada por el alto sesgo en algunos tipos de modelos.

Optimización de Hiperparámetros

Con el fin de mejorar el rendimiento de los modelos entrenados, se implementó un procedimiento sistemático de búsqueda de hiperparámetros, orientado a identificar las

combinaciones más adecuadas para cada clasificador en base a un conjunto definido de configuraciones. Para ello se utilizó la técnica de búsqueda en malla (Grid Search), que explora de forma exhaustiva todas las combinaciones posibles de valores dentro de una grilla predefinida de hiperparámetros.

El proceso de búsqueda se integró con las estrategias de validación cruzada previamente detalladas (K-Fold, Stratified K-Fold y Leave-One-Out), permitiendo que cada configuración fuese evaluada bajo múltiples particiones del conjunto de datos. Esta integración asegura que la selección de parámetros no se vea afectada por una única división entrenamiento-prueba, y mejora la estimación del comportamiento del modelo ante nuevos datos.

Hiperparámetros considerados

Para el clasificador basado en redes neuronales artificiales (MLPClassifier), los hiperparámetros evaluados incluyeron:

- Estructura de la red, mediante distintas configuraciones de capas ocultas (cantidad y tamaño de las mismas).
- Función de activación, considerando tanto tanh como ReLU.
- Algoritmo de optimización, explorando SGD (descenso por gradiente estocástico) y Adam (adaptativo).
- Tasa de aprendizaje, tanto fija (constant) como adaptable (adaptive).
- Parámetro de regularización L2 (alpha), que controla la penalización sobre los pesos para evitar sobreajuste.

Estas combinaciones permiten evaluar cómo influyen distintos diseños de arquitectura y dinámicas de entrenamiento sobre el rendimiento final del modelo.

En el caso del clasificador HistGradientBoostingClassifier se evaluaron los siguientes parámetros:

- Profundidad máxima de los árboles (max_depth), que determina la complejidad del modelo base.
- Número máximo de iteraciones (max_iter), equivalente al número de árboles en el ensamble.

- Tasa de aprendizaje (learning_rate), que regula el impacto de cada iteración sobre la predicción final.
- Cantidad mínima de muestras por hoja (min_samples_leaf), que controla el tamaño mínimo de subdivisión al construir los árboles, con impacto sobre la generalización.

Estos parámetros estructurales fueron seleccionados por su influencia directa sobre el equilibrio entre capacidad de ajuste y riesgo de sobreajuste, especialmente relevante en modelos de ensamble.

Métricas de Evaluación

Cada combinación de hiperparámetros fue evaluada utilizando cinco métricas distintas, con el objetivo de caracterizar el rendimiento desde múltiples perspectivas:

- Accuracy (exactitud global): proporción total de aciertos.
- Precisión ponderada: mide la proporción de predicciones positivas correctas sobre el total de predicciones positivas, ponderando por el tamaño de cada clase. Por ejemplo, si el modelo predijo 100 veces la presencia de cianobacterias y acertó en 80 casos, la precisión es del 80%. Esta métrica es clave para controlar los falsos positivos.
- Recall ponderado: mide la proporción de casos positivos reales que el modelo detectó correctamente, ponderando por el tamaño de cada clase. Por ejemplo, si había 100 muestras con cianobacterias y el modelo detectó 85, el recall es 85%. Esta métrica es clave para controlar los falsos negativos.
- F1-score ponderado: media armónica entre precisión y recall, balanceando ambos criterios.
- ROC AUC: área bajo la curva ROC, empleada únicamente en contextos binarios, para evaluar la capacidad discriminativa del modelo.

Las métricas se calcularon tanto durante la validación cruzada (para seleccionar los mejores parámetros) como sobre un conjunto de prueba independiente, con el objetivo de verificar que el comportamiento observado se mantuviera fuera del entorno de entrenamiento.

La elección y el análisis de estas métricas no deben considerarse de forma aislada ni meramente técnica: es fundamental contextualizarlas en función de la lógica del problema que se busca abordar. En este caso, el objetivo del modelo es contribuir a la inferencia de

niveles de contaminación en cursos de agua de la provincia de Buenos Aires, una problemática donde las implicancias prácticas de los errores de predicción son significativamente asimétricas.

En situaciones de monitoreo ambiental, ciertos falsos negativos, por ejemplo predecir que no está contaminada cuando sí lo está, puede derivar en consecuencias sanitarias graves. Del mismo modo sucede con ciertos falsos positivos, como considerar que una muestra es apta cuando no lo es. En contraste, hay otros que podrían implicar una sobreestimación del riesgo, pero sin comprometer directamente la salud pública: por ejemplo, si se considera no apta cuando sí lo es. Las métricas de precisión y recall contemplan estos casos críticos, pero ninguna de las dos los contempla a todos.

Asimismo, el F1-score ponderado resulta útil como indicador compuesto, ya que sintetiza el equilibrio entre precisión y recall, especialmente cuando se trabaja con clases desbalanceadas, como suele ocurrir en contextos donde los eventos de interés (por ejemplo, altos niveles de contaminación) son menos frecuentes. En cambio, métricas como el accuracy pueden ser engañosas si se interpretan sin considerar esta desproporción entre clases, dado que una alta exactitud podría estar explicada por una correcta clasificación de las clases mayoritarias, ocultando un bajo desempeño en aquellas que son críticas desde la perspectiva ambiental.

Por su parte, la precisión ponderada puede ser relevante en contextos donde las intervenciones ante una predicción positiva tienen costos altos o consecuencias operativas relevantes, ya que mide la proporción de verdaderos positivos sobre el total de predicciones positivas.

Despliegue del modelo

El despliegue del modelo consistió en trasladar la solución de un entorno de desarrollo a un entorno productivo, garantizando su accesibilidad y funcionamiento estable.
Se optó por un flujo claro y reproducible que incluyó: exportación de los modelos entrenados, empaquetado del servicio como API, pruebas de funcionamiento en entorno local y validación final en un servidor remoto (VPS).

Este enfoque asegura que la transición desde la fase de entrenamiento hasta la operación en campo se realice de manera controlada y escalable.

Flujo de trabajo

El proceso de despliegue comenzó con el entrenamiento del modelo en un entorno controlado (Google Colab), donde se utilizó joblib.dump para exportar el modelo entrenado en formato serializado.

Posteriormente, se preparó un conjunto de archivos necesarios para el despliegue, que incluyen el modelo serializado, el código de la API en FastAPI (FastAPI, s.f.), las dependencias listadas en requirements.txt, y archivos para la orquestación y contenedorización (Dockerfile y docker-compose).

La validación inicial se realizó localmente, ejecutando la API en un entorno controlado y verificando la correcta respuesta a solicitudes mediante Postman. Esta etapa permitió asegurar que el modelo y la API funcionaban correctamente antes del despliegue definitivo.

Finalmente, el sistema fue desplegado en un servidor VPS, donde se repitieron las pruebas con Postman para validar la comunicación remota y la correcta operación del servicio en condiciones reales.

Elección de FastAPI para la PPS

FastAPI se destaca por su rapidez en el desarrollo, tipado explícito que mejora la robustez del código, y su documentación automática que facilita el uso y prueba de las APIs. Su integración nativa con librerías de Machine Learning en Python, como scikit-learn o joblib, permite un flujo eficiente desde el entrenamiento hasta la implementación, sin necesidad de migrar modelos a otros lenguajes o plataformas.

Estas características lo convierten en una opción ideal para prototipos, proyectos académicos y servicios de baja a mediana escala, donde la agilidad y la flexibilidad son prioritarias.

Arquitectura de microservicios como punto de integración

La arquitectura de microservicios facilita la integración de diversas tecnologías y lenguajes dentro de una solución única, permitiendo que componentes específicos se desarrollen con herramientas y frameworks que mejor se adapten a sus requerimientos particulares. En este sentido, FastAPI puede implementarse como el microservicio encargado de la inferencia del modelo de Machine Learning, mientras que la lógica de negocio principal se desarrolla en un entorno más robusto como C# o Java.

Este enfoque aporta ventajas relevantes para sistemas con demandas variadas, ya que cada microservicio puede desplegarse y escalar de forma independiente, favoreciendo la flexibilidad en el mantenimiento y la actualización. Además, posibilita la elección del stack tecnológico más adecuado para cada función, optimizando el rendimiento global y la eficiencia del sistema.

La combinación de FastAPI con tecnologías consolidadas resulta especialmente útil en contextos donde los servicios de inteligencia artificial requieren respuestas rápidas y capacidad de adaptación continua, o cuando se trata de aplicaciones híbridas que mantienen la capa central de negocio en plataformas maduras y utiliza Python para gestionar modelos de Machine Learning, aprovechando su ecosistema especializado.

Por otro lado, en sistemas monolíticos o en aquellos con bajo nivel de complejidad y requisitos estrictos de uniformidad tecnológica, esta arquitectura distribuida y heterogénea puede no ser necesaria ni conveniente, por lo que su adopción debe evaluarse en función del contexto y los objetivos específicos del proyecto.

Resultados relevantes para el proyecto

Resultados de la evaluación del protocolo en escenarios multihop

Se sometió el protocolo a diferentes escenarios diseñados para verificar su capacidad de mantener la comunicación multihop con la participación de un único nodo repetidor (R). El objetivo fué determinar la robustez y estabilidad del protocolo ante condiciones realistas de

operación, donde la conectividad no siempre es estable y los nodos pueden experimentar caídas temporales o retrasos en la comunicación.

En esta sección se presentan los resultados obtenidos en los dos escenarios más representativos. Las pruebas contemplaron tres dimensiones principales: la recuperación tras fallas del gateway o de los nodos intermedios, la consistencia del intercambio de mensajes bajo condiciones de carga variable, y, la pérdida de mensajes en función de la distancia lógica respecto al gateway (G).

Escenario A: Camino feliz

Se evaluó el protocolo bajo condiciones ideales, verificando la comunicación multihop desde una sonda (S) hasta el gateway (G) a través de un repetidor (R). Se registraron cambios de estado, llegada de mensajes y cambios de receptor para analizar la estabilidad de la ruta y la consistencia en la devolución de ACK.

Descripción de la prueba de campo:

1. Se encendió primero el gateway (G).
2. Se incorporó el repetidor (R) dentro del alcance de G, completando el reconocimiento y alcanzando ConnectedState y envío de datos.
3. Se añadió la sonda (S), ubicada fuera del alcance directo de G pero conectada a través de R.
4. Se enviaron los primeros DataRequest desde S y se verificó la correcta propagación y recepción de ACKs a lo largo de la ruta inversa.

Eventos registrados en S

17:54:45) ReconnectingState
17:54:45) ConnectionStatus: Datasource=3, MsgNro=4, Server=2, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=163
17:54:39) Recevier change: 3
17:54:45) ConnectedState
17:54:49) DataRequest: Datasource=3, MsgNro=5, Server=1, NodeListSize=1, NodeList=3, Payload=7HeLoRa!
17:54:55) DataRequest: Datasource=3, MsgNro=6, Server=1, NodeListSize=2, NodeList=32, Payload=71HeLoRa!
17:54:56) ConnectionStatus: Datasource=3, MsgNro=7, Server=2, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=0
17:54:56) Recevier change: 3

Eventos registrados en R

*** ReconnectingState
*** ConnectionStatus: Datasource=1, MsgNro=2, Server=3, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=0
*** Recevier change: 1
*** ConnectedState
*** ConnectionStatus: Datasource=1, MsgNro=3, Server=3, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=0
*** Recevier change: 1
*** ConnectionStatus: Datasource=1, MsgNro=4, Server=3, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=0
*** Recevier change: 1
*** RecognitionRequest: Datasource=2, MsgNro=1
*** ConnectionStatus: Datasource=1, MsgNro=5, Server=3, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=0
*** Recevier change: 1
*** DataRequest: Datasource=2, MsgNro=2, Server=3, NodeListSize=1, NodeList=2, Payload=71HeLoRa!
*** ConnectionStatus: Datasource=1, MsgNro=6, Server=3, NodeListSize=1, NodeList=2, Distance=0, OtherRSSI=0
*** Recevier change: 1
*** ConnectionStatus: Datasource=1, MsgNro=7, Server=3, NodeListSize=0, NodeList=, Distance=0, OtherRSSI=0
*** Recevier change: 1

Eventos registrados en G

17:53:51) GatewayState
17:54:18) RecognitionRequest: Datasource=3, MsgNro=1
17:54:28) DataRequest: Datasource=3, MsgNro=2, Server=1, NodeListSize=1, NodeList=3, Payload=5HeLoRa!
17:54:38) DataRequest: Datasource=3, MsgNro=3, Server=1, NodeListSize=1, NodeList=3, Payload=6HeLoRa!
17:54:49) DataRequest: Datasource=3, MsgNro=5, Server=1, NodeListSize=1, NodeList=3, Payload=7HeLoRa!
17:54:55) DataRequest: Datasource=3, MsgNro=6, Server=1, NodeListSize=2, NodeList=32, Payload=71HeLoRa!
17:54:59) DataRequest: Datasource=3, MsgNro=8, Server=1, NodeListSize=1, NodeList=3, Payload=8HeLoRa!

Interpretación de resultados:

- La sonda S se conectó correctamente a través del repetidor R, completando el reconocimiento inicial.
- Todos los DataRequest enviados por S fueron retransmitidos por R y recibidos por G, demostrando estabilidad en la ruta multihop y la consistencia en la entrega de mensajes.
- Todos los DataRequest originados por R llegaron al gateway sin interferir con las retransmisiones.
- No se detectaron pérdidas ni duplicados en los mensajes registrados, confirmando que bajo condiciones ideales el protocolo cumple su función de manera robusta.

Dispositivo	Timestamp	Evento Detalle
Gateway	17:53:51	GatewayState
Repetidor	17:54:18	ReconnectingState
Repetidor	17:54:18	Envía Reconocimiento.
Gateway	17:54:18	Recibe Reconocimiento de R.
Gateway	17:54:18	Envía Ack a R.
Repetidor	17:54:18	Recibe Ack de G.
Repetidor	17:54:18	Setea como Receptor a G.
Repetidor	17:54:18	ConnectedState
R + G	17:54:28	R y G intercambian Datos por Ack. Se actualiza RSSI.
R + G	17:54:38	R y G intercambian Datos por Ack. Se actualiza RSSI.
Sonda	17:54:39	ReconnectingState
Sonda	~ 17:54:40	PausedState
Sonda	17:54:45	Envía Reconocimiento.
Repetidor	17:54:45	Recibe Reconocimiento de S.
Repetidor	17:54:45	Envía Ack a S.
Sonda	17:54:45	Recibe Ack de R.
Sonda	17:54:45	Setea como Receptor a R.
Sonda	17:54:45	ConnectedState
R + G	17:54:49	R y G intercambian Datos por Ack. Se actualiza RSSI.
S	17:54:49	Recibe un paquete de R hacia G y lo ignora.
Sonda	17:54:55	Envía Datos a R.
Repetidor	17:54:55	Recibe Datos de G.
Repetidor	17:54:55	Retransmite los datos de R hacia G.
Gateway	17:54:55	Recibe Datos de G (retransmitidos) y los ignora.
Gateway	17:54:55	Recibe Datos de G (retransmitidos).
Gateway	17:54:55	Envía Ack a R.

Repetidor	17:54:55	Recibe Ack de G y actualiza RSSI.
Repetidor	~17:54:56	Retransmite Ack de G hacia R.
Sonda	17:54:56	Recibe Ack a R y actualiza RSSI.
R + G	17:54:59	R y G intercambian Datos por Ack. Se actualiza RSSI.

Escenario B — Recuperación y consistencia bajo carga

Se evaluó el protocolo bajo condiciones de falla temporal, verificando la capacidad de la red para restablecer la comunicación multihop y mantener la consistencia de los mensajes desde una sonda (S) hasta el gateway (G) a través de un repetidor (R). Se registraron cambios de estado, llegada de mensajes, cambios de receptor y pérdida de paquetes para analizar la robustez del protocolo frente a reinicios y desconexiones temporales.

En esta segunda sección se omiten detalles como registros de ejecución o tablas intermedias, dado que la metodología aplicada es análoga a la expuesta previamente. En la sección anterior, el énfasis radica en el valor del mecanismo en sí, el cual reforzaba la comprensión de los contenidos presentados en el informe. Sin embargo, repetirlo en esta segunda sección solo agregaría complejidad a la presentación de resultados en lugar de aportar valor.

Reinicio rápido de gateway (G):

Con G encendido y S conectada a través de R, se reinicio G.

Observaciones:

- Se observó cómo R y S detectaron el reinicio del gateway, invalidaron sus receptores y reiniciaron el reconocimiento.
- Se constató que la red pudo volver al estado de conexión total.

Falla temporal de repetidor (R):

Con G encendido y S conectada a través de R, se apagó R y se dejó transcurrir suficiente tiempo para que S detectara la ausencia y entrará en estado conectando.

Observaciones:

- Se observó la cantidad de mensajes perdidos hasta que S detectara la desconexión, resultado acordes a la configuración de timeouts y frecuencia de envío seleccionada en cada caso.
- Al volver a encender R, se observó cómo la sonda y el gateway restablecieron la comunicación multihop, reanudando el envío de DataRequest.

Dashboard

En esta sección se presentan capturas de pantalla de las interfaces del dashboard utilizadas para el monitoreo de los parámetros sensados.

En todos los casos, para cada parámetro se incluyó un gráfico de línea, que permite observar la evolución histórica de los valores, así como un indicador que facilita la visualización rápida del último valor registrado.

Se configuraron un total de siete dashboards: tres de ellos muestran los parámetros individuales de cada sonda, con cuatro variables por dashboard, y los cuatro restantes permiten visualizar los valores de cada parámetro particular provenientes del conjunto completo de sondas, mediante gráficos por sonda.

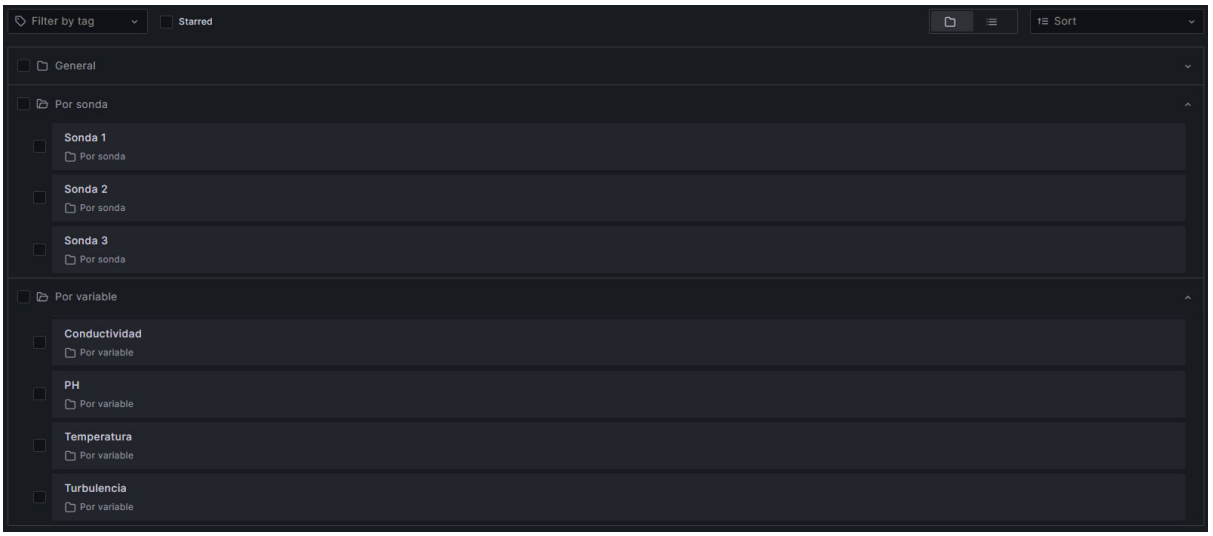


Image 26. Menú que permite seleccionar entre los diferentes dashboards.

Imagen 27. Dashboard para visualizar los datos censados por una de las sondas.

Aplicación de inteligencia artificial

Todos los resultados obtenidos a lo largo del proceso aportan información valiosa que puede enriquecer futuros trabajos de carácter académico o de investigación. Sin embargo, dado que el enfoque principal de este proyecto es eminentemente práctico, se decidió centrar la presentación en los hallazgos más relevantes para el diseño implementado. Como excepción, se incluyen aquí los resultados relacionados con la predicción de E. coli a partir de parámetros no censados por las sondas y la opción de seis categorías de E.Coli, dado que constituyen un hallazgo concreto y significativo, con valor científico y potencial para futuras investigaciones.

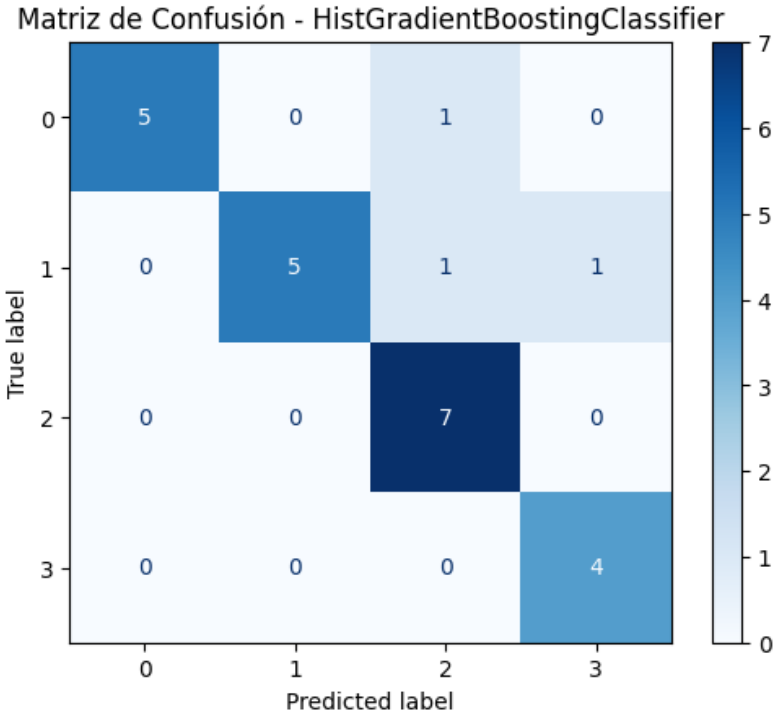
Para la predicción de E. coli y cianobacterias, se seleccionó el modelo HistGradientBoostingClassifier como la herramienta más efectiva, tanto para ambos conjuntos de variables de entrada como para las distintas configuraciones analizadas.

Para el análisis fueron contemplados múltiples aspectos, destacando lo detallado anteriormente respecto a precisión, recall y f1 score, como factor de mayor ponderación para la toma de decisiones.

La matriz de confusión se utilizó como recurso gráfico para la visualización y análisis de los resultados, con foco en los casos críticos, o sea los que están por debajo de la diagonal. Esta lectura condice con el recién mencionado factor de mayor ponderación.

Finalmente, es relevante destacar que todas las demás métricas evaluadas respaldaron de manera consistente la elección de este modelo, reafirmando su idoneidad para el propósito del proyecto.

Predicción de E. Coli con cuatro categorías



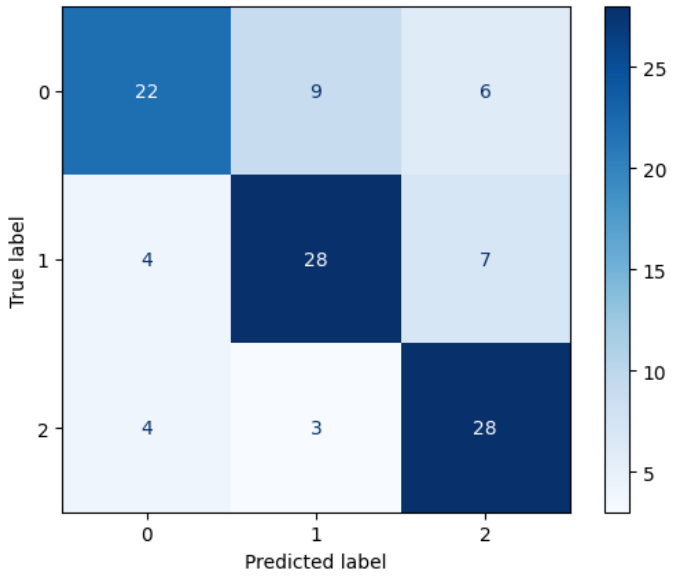
Imágen 28. Matriz de confusión para el esquema de cuatro categorías para E. Coli.

Se destaca que no haya falsos positivos por debajo de la diagonal de la matriz de confusión, son todos ceros. Esto indica que el modelo nunca clasificó erróneamente una muestra contaminada como de menor gravedad.

```
ROC AUC OvR - Macro: 0.9664
ROC AUC OvR - Ponderado: 0.9608
ROC AUC OvO - Macro: 0.9663
ROC AUC OvO - Ponderado: 0.9645

Cardinalidades:
Alto                  30
Bajo (30k)            33
Moderado (500k)       36
Recreativo (1.99k)    21
Name: count, dtype: int64

Exactitud (Accuracy): 0.8750
Precisión Macro: 0.8944
Precisión Ponderada: 0.9019
Recall Macro: 0.8869
Recall Ponderado: 0.8750
F1 Score Macro: 0.8766
F1 Score Ponderado: 0.8737

Métricas por clase:
Clase 0: Precisión=1.0000, Recall=0.8333, F1 Score=0.9091
Clase 1: Precisión=1.0000, Recall=0.7143, F1 Score=0.8333
Clase 2: Precisión=0.7778, Recall=1.0000, F1 Score=0.8750
Clase 3: Precisión=0.8000, Recall=1.0000, F1 Score=0.8889
```

Imágen 29. Métricas para el esquema de cuatro categorías para E. Coli.

Predicción de cianobacterias



Imágen 30. Matriz de confusión para la categorización de cianobacterias..

En este modelo se observa una mayor presencia de valores fuera de la diagonal en comparación con el modelo para E. Coli, lo que indica que las predicciones son menos precisas que el anterior. No obstante, la mayoría de las predicciones se concentran en la diagonal, y los errores tienden a situarse por encima de ella, evitando subestimar la gravedad de la contaminación. Además, todas las métricas evaluadas presentan valores superiores a 0,7, lo que confirma un desempeño sólido del modelo.

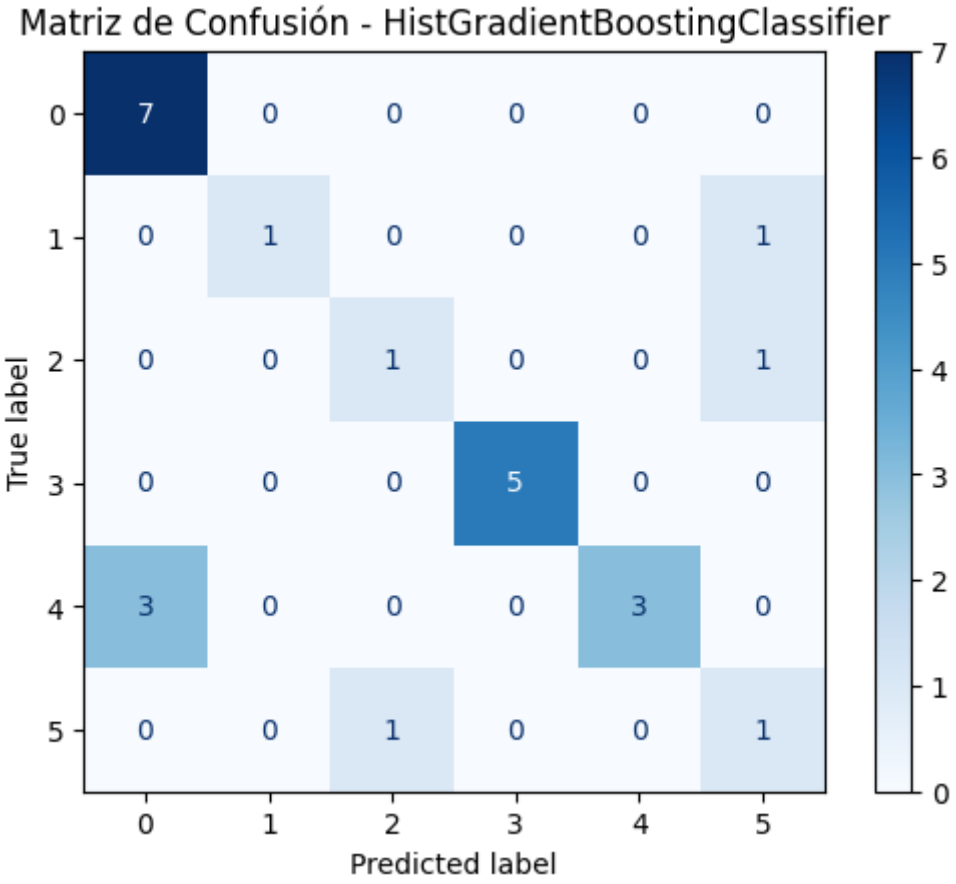
```
ROC AUC OvR - Macro: 0.8224
ROC AUC OvR - Ponderado: 0.8213
ROC AUC OvO - Macro: 0.8234
ROC AUC OvO - Ponderado: 0.8226

Cardinalidades:
1    186
2    192
3    174
Name: count, dtype: int64

Exactitud (Accuracy): 0.7027
Precisión Macro: 0.7054
Precisión Ponderada: 0.7057
Recall Macro: 0.7042
Recall Ponderado: 0.7027
F1 Score Macro: 0.7008
F1 Score Ponderado: 0.7003

Métricas por clase:
Clase 0: Precisión=0.7333, Recall=0.5946, F1 Score=0.6567
Clase 1: Precisión=0.7000, Recall=0.7179, F1 Score=0.7089
Clase 2: Precisión=0.6829, Recall=0.8000, F1 Score=0.7368
```

Imágen 31. Métricas para la categorización de cianobacterias.

Predicción de E.Coli con 6 categorías

Esta categorización representa el esquema ideal y, en general, obtuvo buenos resultados. No obstante, la métrica más relevante para nuestro caso, el F1 Score, alcanzó un valor de 0,676 (siendo que la meta era obtener al menos 0,7), acompañado por el recall en 0.66. Al analizar la matriz de confusión, se observan pocos valores fuera de la diagonal; sin embargo, esto probablemente se deba a la escasa cantidad de registros disponibles para distribuir entre las

distintas clases, lo que limita la representatividad de algunas celdas. Estos dos factores fueron los principales a la hora de optar por la alternativa de cuatro categorías. Se decidió colocarla en esta sección ya que es una categorización a considerar si se obtiene un dataset más poblado.



Imágen 32. Métricas para el esquema de seis categorías para E. Coli.

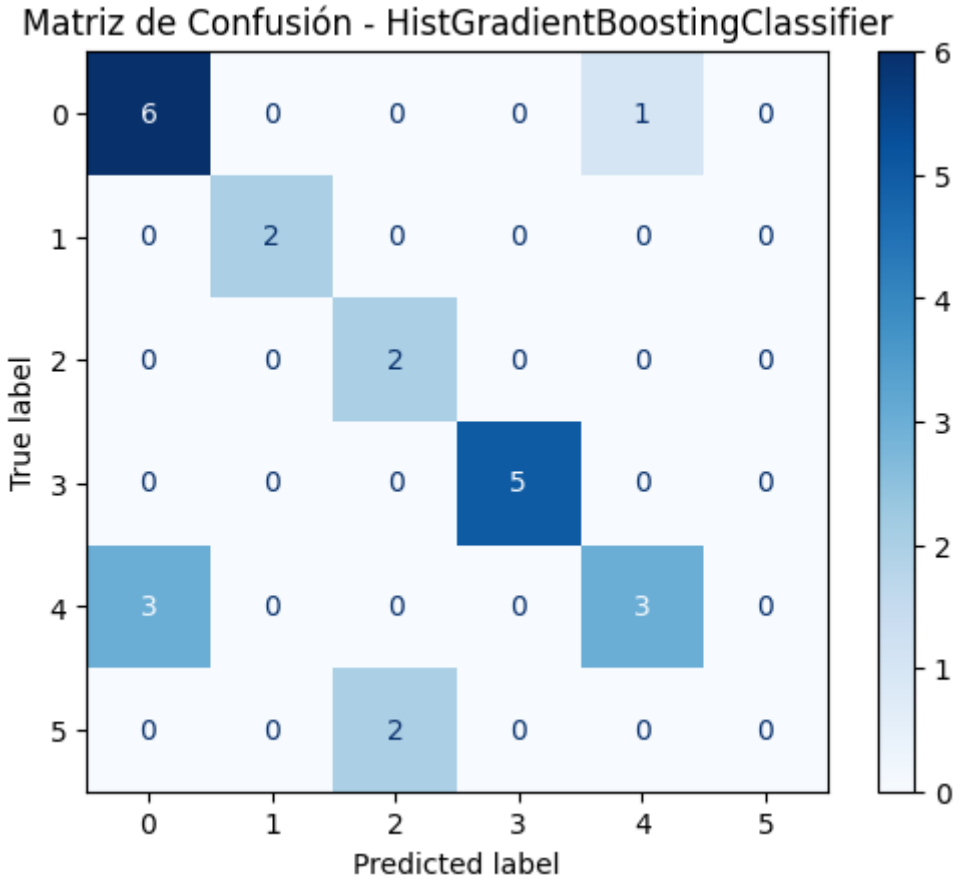
```
ROC AUC score (One-vs-Rest) is 0.9570848541436776
ROC AUC score (One-vs-One) is 0.9570848541436776
```

```
Exactitud: 0.75
Recuperación (Recall): 0.6666666666666666
Precisión: 0.7555555555555555
F1 Score: 0.6761437908496731
```

Imágen 33. Métricas para el esquema de seis categorías para E. Coli.

Predicción E.Coli a partir de parámetros no medibles con las sondas

Propuesta de seis categorías

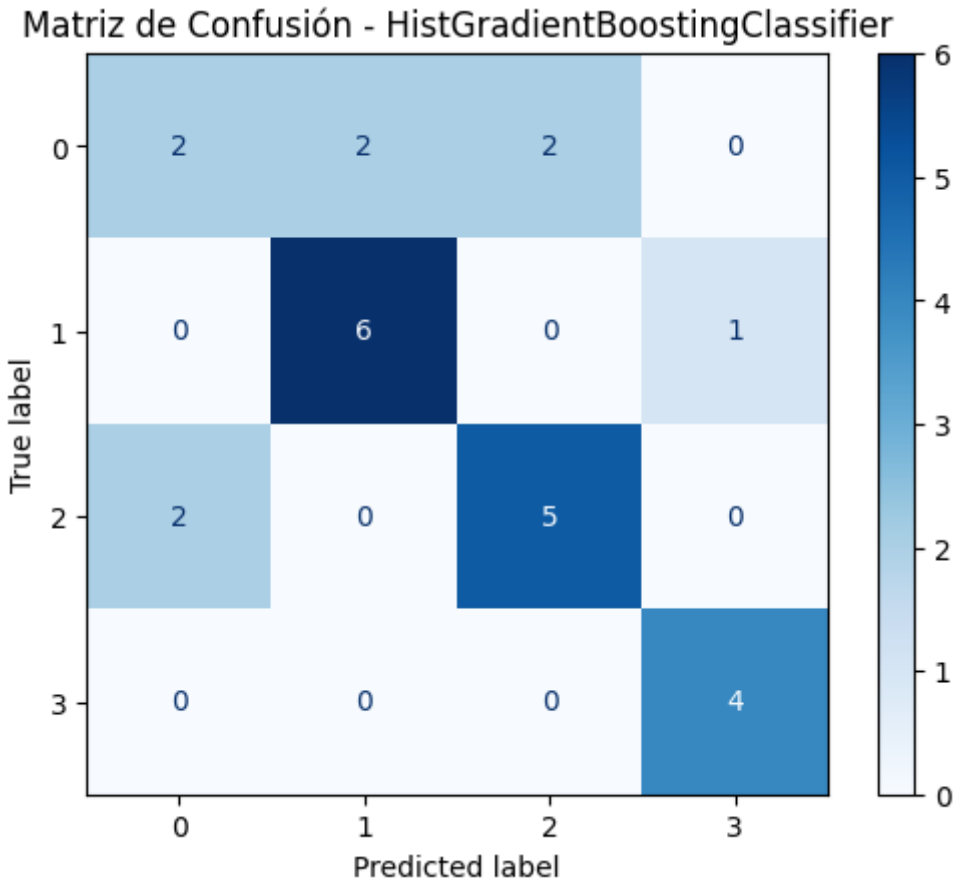


Imágen 34. Matriz de confusión, 6 categorías, entradas no censadas por las sondas.

```
ROC AUC score (One-vs-Rest) is 0.9412483848758358
ROC AUC score (One-vs-One) is 0.9412483848758358
```

```
Exactitud: 0.75
Recuperación (Recall): 0.7261904761904763
Precisión: 0.6527777777777778
F1 Score: 0.6694444444444444
```

Imágen 35. Métricas, 6 categorías, entradas no censadas por las sondas.



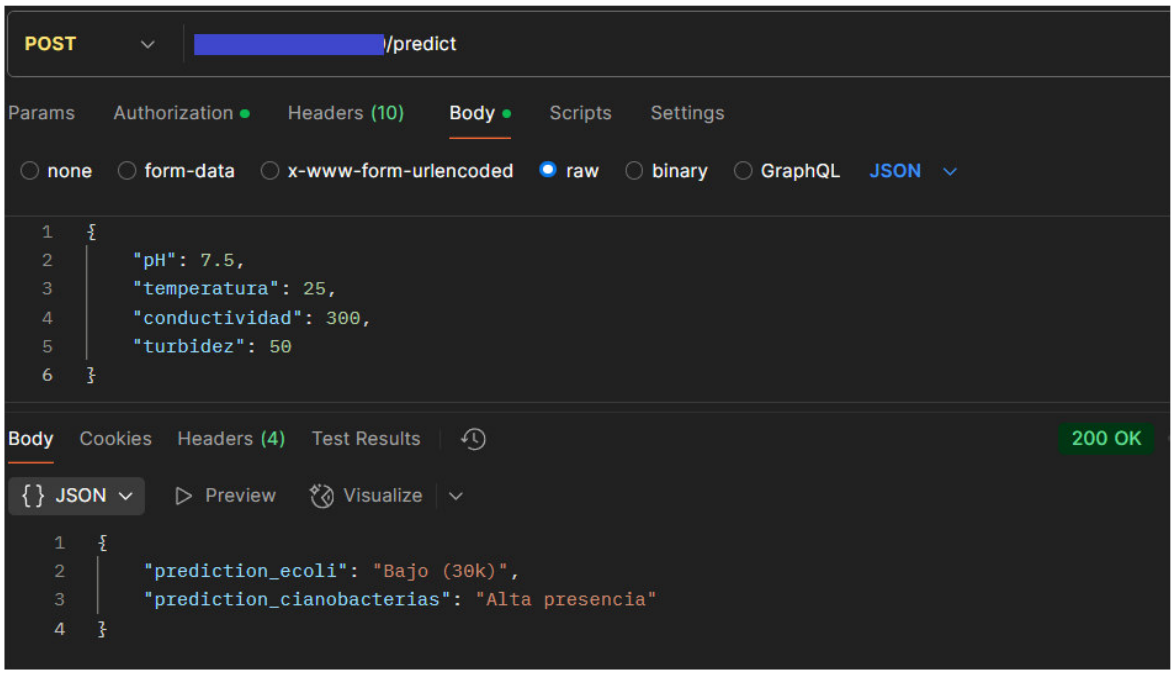
Imágen 36. Matriz de confusión, 4 categorías, entradas no censadas por las sondas.

```
ROC AUC score (One-vs-Rest) is 0.902855586679116
ROC AUC score (One-vs-One) is 0.902855586679116
```

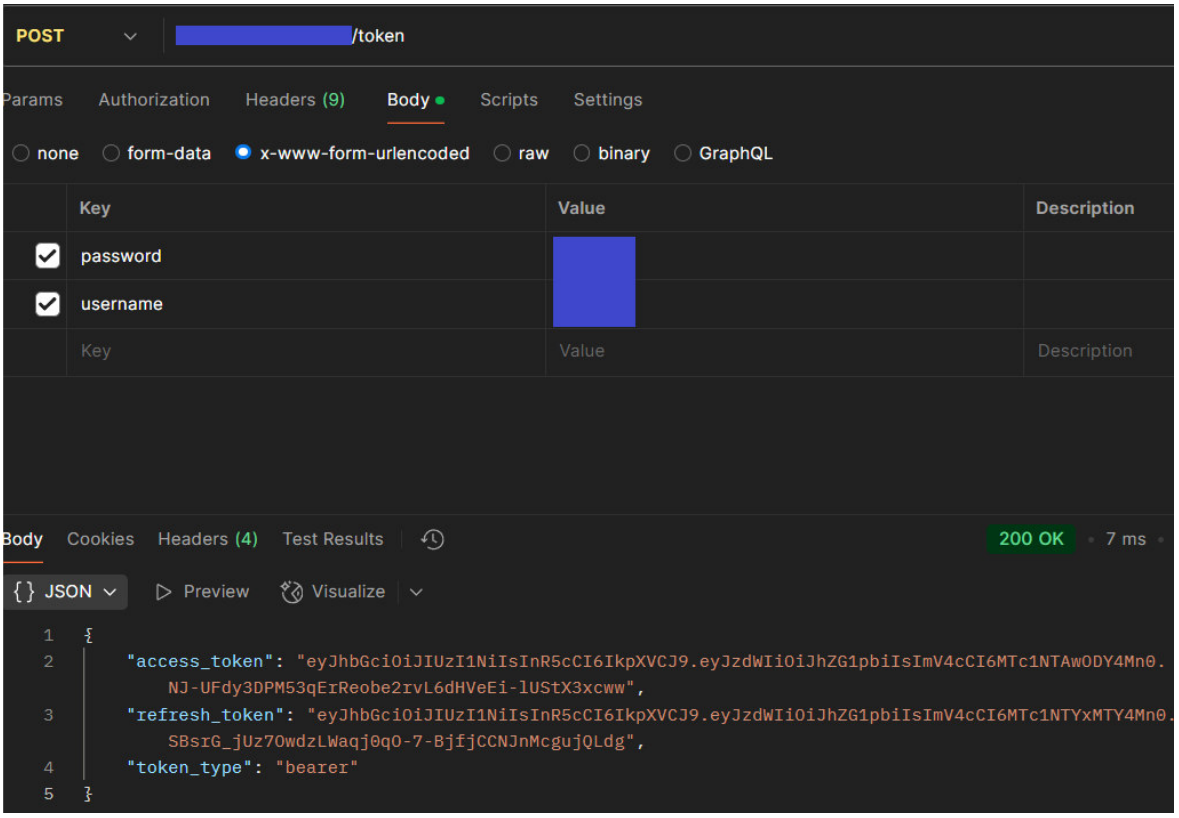
```
Exactitud: 0.7083333333333334
Recuperación (Recall): 0.7261904761904762
Precisión: 0.6910714285714286
F1 Score: 0.7007936507936509
```

Imágen 37. Métricas, 6 categorías, entradas no censadas por las sondas.

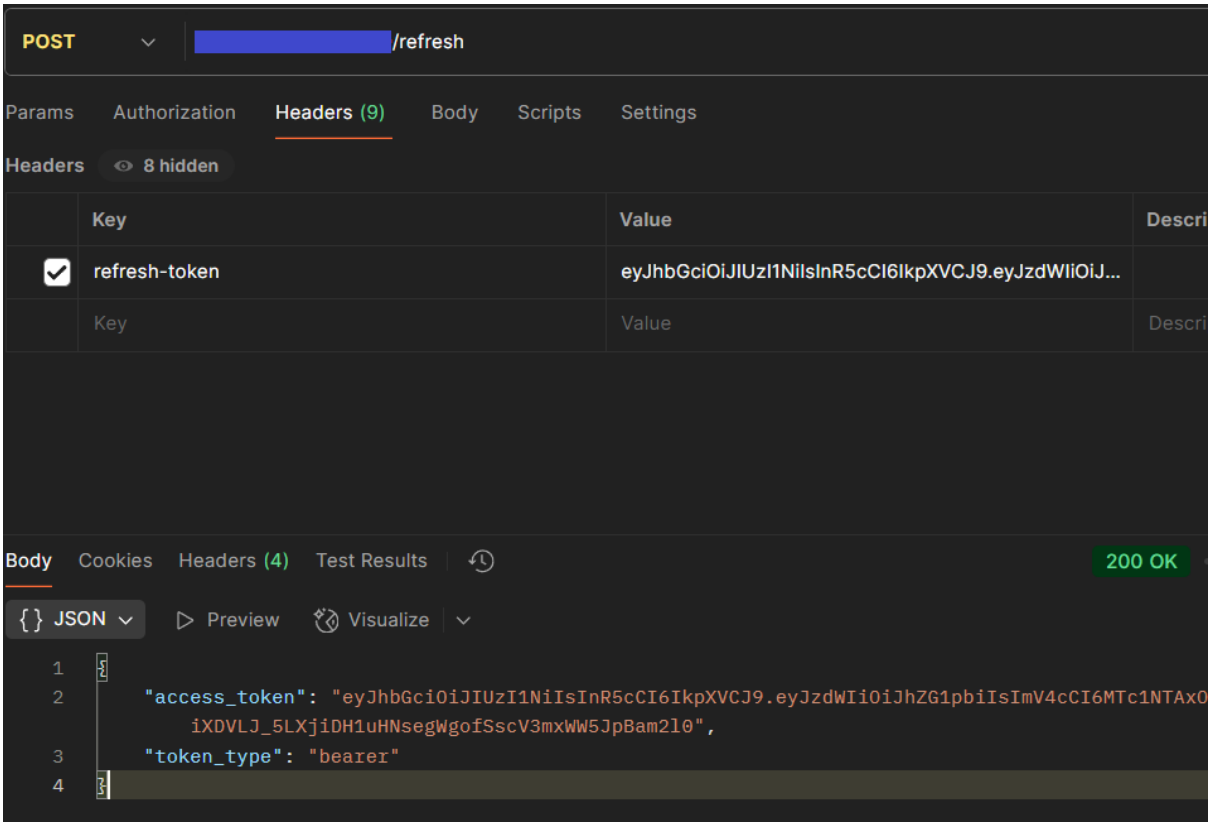
Despliegue de la API RESTful



Imágen 38. Envío de solicitud por Postman al endpoint para predecir usos del agua.



Imágen 39. Envío de solicitud por Postman al endpoint para autenticación..



Imágen 40. Envío de solicitud por Postman al endpoint para renovar el token JWT.

Conclusiones

El trabajo realizado puede organizarse en tres aportes principales. Primero, se contribuyó al equipo de investigación del área de IoT de la UNAJ, mediante un caso de estudio en el que todas las decisiones de diseño se centraron en aspectos específicos de IoT, incluyendo desde la provisión de librerías para sistemas embebidos hasta técnicas de procesamiento de datos provenientes de dispositivos conectados. Segundo, se aportó al área de inteligencia artificial, donde se mejoraron los resultados previamente obtenidos sobre el dataset de cianobacterias y, por primera vez, se construyeron modelos de clasificación para el dataset de E. coli. Cabe destacar que estos desarrollos tecnológicos, concebidos desde la carrera de informática, proveen herramientas concretas para investigadores de otras áreas no tecnológicas de la UNAJ.

En paralelo, se desarrolló un protocolo de comunicación propietario que requirió una investigación independiente para fundamentar su diseño, por no existir líneas de investigación previas en la universidad. Se definió una solución de capas 2 y 3 que, aunque

no acoplada a LoRa, contempla aspectos relacionados con entornos inalámbricos de recursos limitados. Su implementación en C++, inspirada en la arquitectura hexagonal y basada en un patrón de estados, permitió separar con claridad las responsabilidades.

La elección de construir un protocolo propietario respondió a la necesidad de balancear robustez técnica, bajo costo y flexibilidad. Alternativas como Wi-SUN ofrecen prestaciones industriales y escalabilidad probada, pero imponen costos elevados, dependencia de hardware especializado y menor control sobre el stack. En cambio, la propuesta desarrollada constituye una opción más accesible y ajustada a entornos de investigación y prototipado, contribuyendo con la comunidad a la exploración de soluciones IoT de bajo coste y alta adaptabilidad.

En cuanto a la arquitectura IoT, la implementación se apoyó en servicios ampliamente utilizados en la industria, lo que permitió acelerar el desarrollo y maximizar resultados con un esfuerzo mínimo. La incorporación de un broker MQTT como núcleo del intercambio de mensajes aseguró un sistema de comunicación ligero, eficiente y altamente compatible con dispositivos heterogéneos, características esenciales en aplicaciones de monitoreo ambiental.

La incorporación de inteligencia artificial amplió el alcance del sistema hacia un análisis predictivo y preventivo. Se construyeron dos modelos de clasificación: uno para predecir niveles de cianobacterias, alcanzando métricas superiores a 0,7 con tres categorías, y otro para la detección de E. coli, con un desempeño cercano a 0,9 en cuatro categorías. Ambos modelos fueron desplegados como servicios accesibles mediante una API RESTful, protegida con autenticación JWT y operativa en un servidor VPS administrado por el equipo de investigación del área IoT de la UNAJ, lo que garantiza su integración con sistemas externos y su escalabilidad futura.

En conclusión, los resultados alcanzados evidencian que la propuesta no solo responde a la necesidad de mejorar la recolección y análisis de parámetros críticos en cuerpos de agua, sino que también sienta las bases para soluciones IoT escalables, de bajo coste y con potencial de aplicación en distintos escenarios, incluyendo la gestión ambiental.

Referencias

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. Computer Networks, 54(15), 2787–2805.

Augustin, A., Yi, J., Clausen, T., & Townsley, W. (2016). A study of LoRa: Long range & low power networks for the Internet of Things. Sensors, 16(9), 1466.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347–2376.

Cilfone, A., Davoli, L., Belli, L., & Ferrari, G. (2019). Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies. Future Internet, 11(4), 99.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J. P., & Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550. https://doi.org/10.17487/RFC6550

Bluetooth SIG. (2023). Mesh Protocol Specification v1.1. https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

Shelby, Z., Bormann, C., & Frank, B. (2008). 6LoWPAN: The Wireless Embedded Internet. Wiley.

Thubert, P. (2012). Routing Requirements for Low-Power and Lossy Networks (RPL). RFC 6550. https://doi.org/10.17487/RFC6550

Wi-SUN Alliance. (2021). Wi-SUN Field Area Network (FAN) Profile Specification. https://www.wi-sun.org/

Zigbee Alliance. (2012). Zigbee Specification. https://zigbeealliance.org/solution/zigbee/

BACnet International. (s.f.). BACnet: Building Automation and Control Network. https://bacnetinternational.org/

EMQX. (s.f.). Connecting MQTT-SN devices using EMQX. https://www.emqx.com/en/blog/connecting-mqtt-sn-devices-using-emqx

Shelby, Z., Bormann, C., & Frank, B. (2014). Constrained Application Protocol (CoAP). RFC 7252. https://doi.org/10.17487/RFC7252

APHA (2017). Standard Methods for the Examination of Water and Wastewater. American Public Health Association.

Bonetto, A. A. (1994). Ecología de las aguas continentales. EUDEBA.

Chapman, D. (1996). Water Quality Assessments – A Guide to Use of Biota, Sediments and Water in Environmental Monitoring. UNESCO/WHO/UNEP.

Liu, Y., Wang, J., Yu, H., & Yang, Y. (2018). "IoT and Big Data-based water quality monitoring system". IEEE Access.

OMS (2011). Guidelines for Drinking-water Quality, Fourth Edition. World Health Organization.

Tundisi, J. G., & Tundisi, T. M. (2008). Limnología. Oficina de Textos.

UNESCO (2020). Water security and the sustainable development goals.

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine). Recuperado el 20 de agosto de 2025, de

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Alistair Cockburn (2005). “Hexagonal architecture.”

OASIS. (2019). MQTT Version 3.1.1. OASIS Standard. Recuperado el 20 de agosto de 2025, de https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

LoRaMesher. (s.f.). GitHub repository. Recuperado el 20 de agosto de 2025, de https://github.com/LoRaMesher/LoRaMesher

Meshtastic. (s.f.). GitHub repository. Recuperado el 20 de agosto de 2025, de https://github.com/meshtastic/meshtastic-device

AirSpayce. (s.f.). RadioHead Packet Radio library. Recuperado el 20 de agosto de 2025, de http://www.airspayce.com/mikem/arduino/RadioHead/

LoRaNet. (s.f.). Documentation and stack overview. Recuperado el 20 de agosto de 2025, de https://www.loranet.org/

MySensors. (s.f.). MySensors documentation. Recuperado el 20 de agosto de 2025, de https://www.mysensors.org/

Link Labs. (s.f.). Symphony Link protocol overview. Recuperado el 20 de agosto de 2025, de https://www.link-labs.com/symphony

Wirepas. (s.f.). Wirepas Mesh Overview. Recuperado el 20 de agosto de 2025, de https://www.wirepas.com/

Espressif. (s.f.). ESP-MESH Documentation. Recuperado el 20 de agosto de 2025, de https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/esp-wifi-mesh.html

Bluetooth SIG. (s.f.). Mesh Protocol Specification v1.1. Recuperado el 20 de agosto de 2025, de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

OpenThread. (s.f.). Open-source implementation of Thread networking protocol. Recuperado el 20 de agosto de 2025, de https://openthread.io/

BACnet International. (s.f.). BACnet: Building Automation and Control Network. Recuperado el 20 de agosto de 2025, de https://bacnetinternational.org/

MQTT-SN . (s.f.). Connecting MQTT-SN devices using EMQX. Recuperado el 20 de agosto de 2025, de https://www.emqx.com/en/blog/connecting-mqtt-sn-devices-using-emqx

EMQX. (s.f.). EMQX. Recuperado el 20 de agosto de 2025, de https://www.emqx.com

IEEE. (s.f.). IEEE 802.15.4-2020: Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs). Recuperado el 20 de agosto de 2025, de https://standards.ieee.org/ieee/802.15.4/5050/

IEEE. (s.f.). IEEE 802.11ah-2016: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Amendment 2: Sub 1 GHz License Exempt Operation. Recuperado el 20 de agosto de 2025, de https://standards.ieee.org/ieee/802.11ah/4960/

CoAP. (s.f.). Constrained Application Protocol (CoAP) information portal. Recuperado el 20 de agosto de 2025, de https://coap.space/

Contiki-NG. (s.f.). OS for connected, memory-constrained systems. Recuperado el 20 de agosto de 2025, de https://www.contiki-ng.org/

RIOT OS. (s.f.). RIOT Operating System for IoT. Recuperado el 20 de agosto de 2025, de https://www.riot-os.org/

FastAPI, S. (s.f.). FastAPI. Recuperado el 20 de agosto de 2025, de https://fastapi.tiangolo.com/

Node-RED. (s.f.). Flow-based programming for the Internet of Things. Recuperado el 20 de agosto de 2025, de https://nodered.org/

Grafana Labs. (s.f.). Grafana: The open observability platform. Recuperado el 20 de agosto de 2025, de https://grafana.com/