Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén
Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

MONITORIZACIÓN DE INSTALACIONES INDUSTRIALES

Alumno

Rubén Guijarro Ortega

Tutor

Pedro José Sánchez Sánchez
(Departamento de Informática)

Junio, 2023

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Don Pedro José Sánchez Sánchez, tutor del Trabajo Fin de Grado titulado: '**Monitorización de instalaciones industriales**', que presenta Don Rubén Guijarro Ortega, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

En Jaén, Junio de 2023

El alumno: El tutor:

Rubén Guijarro Ortega Pedro José Sánchez Sánchez

(Página intencionalmente en blanco)

AGRADECIMIENTOS

Agradezco a mis padres, por el incondicional apoyo recibido en toda mi educación y mis proyectos, dándome ánimos en todo momento.

A mis abuelos, que estarían encantados de poder verme finalizar los estudios universitarios.

A mis compañeros y amigos Diego y Fernando, que me han ayudado desde el principio del camino.

A mi tutor, Pedro José Sánchez Sánchez, por su paciencia y atención, animándome desde el primer momento en la realización del proyecto pese a mis dificultades.

A ti Álvaro, en especial, por estar siempre apoyándome en todo lo que hago.

Gracias a todos.

FICHA DEL TRABAJO FIN DE GRADO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Ingeniería del Software
Mención (solo TFG)	Informática Empresarial
Idioma	Español
Tipo	Específico
TFG en equipo	No
Autor/a	Rubén Guijarro Ortega
Fecha de asignación	10/11/2021
Descripción corta	El presente TFG pretende abordar un problema de digitalización y monitorización de los sistemas ya existentes en los entornos industriales. Se plantea el desarrollo de un prototipo de sistema de monitorización. Este sistema estará formado por varios elementos independientes, dotándole de mayor capacidad de adaptación a la instalación existente. Los módulos incluyen capacidades de obtención de datos, tratamiento, envío, almacenamiento y visualización de los mismos.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
IEEE 9ª edición	Estilo de referencias y citas de IEEE

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
*Estas normas se han utilizado **como base o referencia** para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como **proyecto** la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.*	

Contenido

Índice de ilustraciones

Índice de tablas

1 INTRODUCCIÓN

El avance de la tecnología en el sector industrial ha llevado este entorno a lo que se conoce como Industria 4.0, donde la conectividad, la obtención de información, análisis y toma de decisiones adquieren una importancia crítica.

Se conoce como Industria 4.0 o Cuarta Revolución Industrial al avance tecnológico experimentado por los entornos industriales, donde las tecnologías de la información adquieren una mayor importancia. Maximizar la conectividad punto a punto, la disponibilidad de los datos, su obtención en tiempo real y tratamiento de los mismos, desarrollo de inteligencia de negocio, optimización de procesos operativos para reducir costes, generar valor añadido, son algunos de los objetivos de esta nueva era industrial. La trascendencia de esta evolución va más allá del sector secundario, con previsiones de gran impacto en el sector primario y terciario en el futuro inmediato.

En el presente trabajo se abordará una solución aplicable a un entorno con ciertas tecnologías obsoletas, poniendo en valor las actualmente disponibles, su potencial de digitalizar y monitorizar una instalación industrial.

Mediante un análisis del entorno específico, se detallan los problemas a resolver, así como la valoración de diferentes soluciones. Posteriormente, se realiza la elección de una solución, previa valoración de sus fortalezas y debilidades. Una vez elegida esta solución, se lleva a cabo la planificación y desarrollo de un proyecto donde estudiar la viabilidad de la solución elegida. Por último, se expone un resumen económico, las conclusiones sobre el proyecto planteado y su viabilidad como solución en el mundo real.

1.1 Motivación

La adaptación de los medios operativos en el entorno industrial hacia las tecnologías de la información puede suponer una fuerte barrera de entrada a la Industria 4.0. Las características específicas de los elementos de la instalación, los recursos humanos y su capacitación, la logística, la infraestructura de cada empresa individual o las relaciones comerciales con clientes y otras empresas son componentes de la cadena de valor afectados por la transición a este nuevo paradigma.

Este proyecto nace como evolución natural de las necesidades del entorno laboral en el que me encuentro durante su realización, que se sitúa en el sector de las energías renovables y las instalaciones energéticas. Se necesita realizar procesos de puesta en marcha de los sistemas, labores de vigilancia activa y respuesta ante incidentes, en ocasiones, en zonas de difícil observación directa o en condiciones adversas como altas temperaturas y trabajo en altura. Para ello, se dota a la instalación de equipos de control local que no son capaces de ampliar sus capacidades paralelamente a la evolución de las necesidades del cliente ni de la empresa. Además, se precisa de la generación de informes, histórico de datos o estudio comercial del funcionamiento de estos productos; objetivos que, en ocasiones, resultan imposibles de alcanzar.

Las tecnologías disponibles para la obtención y tratamiento de la información ofrecen amplias oportunidades para la digitalización y mejora continua en este sector.

La finalidad del presente trabajo es el estudio de una situación de partida donde se afrontan problemas comunes en las instalaciones industriales, desarrollando una solución con capacidad de disminuir las barreras de entrada a la Industria 4.0.

1.2 Objetivos del trabajo

La elaboración de este proyecto busca el acercamiento del desarrollo de software a un desafío del mundo laboral, con el fin de desgranarlo en secciones cuya dificultad sea asumible. Para ello, se plantea una serie de objetivos específicos:

- Identificar y documentar los problemas específicos del caso objeto de estudio
- Recopilar información de las tecnologías aplicables en el proceso de mejora y desglose por niveles.
- Analizar las mejores opciones mediante criterios definidos y medibles. Comparación de los resultados, elección de candidatos y justificación.
- Búsqueda de otras soluciones similares. Comparativa de ventajas e inconvenientes.
- Elaboración de una solución donde se aborden las necesidades planteadas.

1.3 Estructura de la memoria

La presente memoria se ha dividido en capítulos siguiendo la siguiente estructura:

- En el capítulo 1, se realiza una introducción al proyecto desarrollado, identificando los principales objetivos del proyecto. Se detalla además la organización temporal de las tareas a realizar para llevarlo a cabo.
- En el capítulo 2, se expone el problema y las carencias identificadas. Se realiza una labor de investigación y búsqueda de soluciones para las deficiencias conocidas.
- En el capítulo 3, Se realiza un desglose según la metodología incremental y se establecen unas metas entregables.
- En el capítulo 4, se expone los diferentes elementos de diseño seleccionados para cada incremento según los requisitos especificados. Se explica en detalle las características propias de la implementación.
- En el capítulo 5 se incluyen las conclusiones y líneas futuras de actuación.

Finalmente, se incluye la bibliografía, los anexos y el glosario de términos.

1.4 Planificación temporal

Con el fin de organizar el tiempo disponible para realizar el proyecto, se ha dividido el tiempo disponible en diferentes tareas generales. Estas tareas tienen una estimación de tiempo orientativa según el esfuerzo que pueda requerir cada una.

La elaboración del trabajo se divide en el tiempo según la Tabla 1 Planificación temporal TFG.

Tarea	Tiempo estimado
Análisis de soluciones	2 semanas
Búsqueda de tecnologías	1 semanas
Toma de requisitos y creación de diagramas	3 semanas
Desarrollo del proyecto	4 semanas
Documentación	6 semanas
Tiempo total	**16 semanas (4 meses)**

Tabla 1 Planificación temporal TFG

2 ANTECEDENTES Y ESTADO DEL ARTE

Para seguir un criterio único, se trabajará bajo la aplicación de la norma ISA 95, utilizada por la "International Society of Automation", ISA. La norma ISA 95 es un estándar cuya función es facilitar la integración de sistemas de información industrial a todos los niveles, desde el inicio o fabricación hasta el nivel empresarial[1].

En el presente trabajo se prestará especial dedicación a los siguientes niveles:

- Nivel 1: Entrada/Salida, Dispositivos y Sensores
- Nivel 2: Interfaz Hombre Máquina (HMI), Sistemas de Supervisión y Obtención de Datos (SCADA)

Los dispositivos **IoT** tienen un gran protagonismo en la obtención de datos sobre el Nivel 1, nivel más cercano al hardware. Pese a sus múltiples puntos fuertes como la versatilidad, la conectividad, su coste reducido o las bajas necesidades energéticas, presentan una serie de limitaciones y retos inherentes a su naturaleza.

Es necesario adaptar los desarrollos a sus limitaciones de potencia computacional, espacio de memoria escaso y espacio de almacenamiento reducido, en muchas ocasiones, no ampliable. Otras funciones básicas presentes en ordenadores completos como son las tecnologías de red IPv6 o descubrimiento de dispositivos en red no están disponibles de forma usual. La incapacidad de mantener programas pesados en segundo plano dificulta la implementación de medidas de seguridad más allá de aquellas de carácter pasivo. Aplicaciones pesadas como antivirus, sistemas avanzados de detección de vulnerabilidades o control de puntos finales de acceso no son aplicables en términos generales.

El control de energía y eficiencia energética es un punto clave en estos dispositivos, ya que no poseen sistemas de control de energía adaptativos. La actividad y consumo energético viene determinada por las necesidades hardware específicas en cada implementación, así como las llamadas a procesos del sistema que activen diferentes modos de funcionamiento.[2]

2.1 Descripción de la situación de partida

Se presenta como elemento objetivo una instalación industrial provista de tecnologías que pueden no enmarcarse en la norma ISA 95, quedando en una situación tecnológica deficiente y con un valor de negocio mermado. Esta situación presenta unas características donde la adopción de tecnologías modernas permite devolver, e incluso incrementar, el valor de negocio original de la instalación.

2.1.1 Descripción del entorno actual

En la Tabla 2 Resumen de niveles ISA se encuentra el resumen de elementos de una instalación industrial en el Nivel 1 y Nivel 2 del esquema ISA 95.

Nivel ISA	Grupo		Entradas	Salidas
Nivel 1	Autómata Programable	PLC	Sensores y detectores	Interruptores, válvulas, motores, leds
	Terminal Remoto	RTU	Señales de los PLC	Telemetría, otros PLC
	Dispositivo Electrónico Inteligente	IED	Sensores, equipos eléctricos	Funciones de protección y control eléctrico
	Sistemas de Control Distribuido	DCS	Programación centralizada, sensores y detectores	Interruptores, válvulas, motores, leds
Nivel 2	Unidad de Control Central	MTU	Datos de los RTU	Señales de control, interfaz, alarmas
	Sistemas de Control y Adquisición de Datos	SCADA	Datos de las MTU y los RTU	Interfaz, señales de control, informes

Tabla 2 Resumen de niveles ISA

2.1.2 Deficiencias y carencias identificadas

A partir de la Tabla 2 Resumen de niveles ISA, se realiza un análisis del estado del sistema y sus puntos de mejora. Las carencias se organizan en la Tabla 3 Relación de debilidades, impacto y soluciones, que servirá de punto de partida para la elaboración de la solución final.

Se considera como debilidades del sistema aquellos comportamientos no deseables que se pretende evitar repetir y aquellos que se dará solución con el nuevo sistema.

El impacto considerado mide la gravedad de cada debilidad. Los niveles utilizados indican:

- **Alto**: La debilidad debe quedar cubierta en el primer entregable para que la solución se considere válida.
- **Medio**: La debilidad debe quedar cubierta. Puede cubrirse en los sucesivos entregables parciales.
- **Bajo**: Se plantea cubrir la debilidad, aunque no es necesaria para considerar finalizado el proceso de desarrollo.

Debilidades del sistema	Impacto considerado	Solución Tipo	Tabla detalle
DS1: Obtención de medidas físicas	Alto	Obtención de medidas digitales, computables por sistemas de la información	Tabla 20 Obtención de medidas físicas
DS2: Transmisión de datos con protocolos propietarios	Alto	Transmisión mediante tecnologías de red e internet	Tabla 21 Transmisión de datos
DS3: Almacenamiento de datos en sistemas locales	Medio	Solución de almacenamiento basado en la nube	Tabla 22 Almacenamiento local de datos
DS4: Comunicación con el sistema mediante interacción física	Medio	Comunicación basada en tecnologías de la información	Tabla 23 Interacción física con el sistema
DS5: Presentación de datos en interfaces no accesibles	Bajo	Presentación mediante interfaces usables y accesibles	Tabla 24 Visualización de datos
DS6: Cohesión de sistemas e interdependencia de sus elementos	Medio	Evadir soluciones integrales o masivas	Tabla 25 Sistema monolítico
DS7: Accesibilidad de la información	Bajo	Uso de tecnologías reconocidas, estructuración de la información	Tabla 26 Accesibilidad de la información

Tabla 3 Relación de debilidades, impacto y soluciones

3 PLANIFICACIÓN DEL DESARROLLO

Para analizar el sistema a desarrollar, se ha seguido una estructura documental basada en la obtención de requisitos.

Se han elaborado tres tipos de requisitos, de forma que los aspectos más relevantes del sistema queden analizados.

A partir de estos requisitos, se han elaborado las tareas a realizar, así como la temporización de las mismas durante el periodo de realización del proyecto.

3.1 Requisitos generales

Podemos establecer los requisitos generales como el conjunto de características tanto necesarias como deseables del sistema. Cada requisito general puede estar relacionado con varias partes del sistema de forma transversal.

Resumen Requisitos Generales del Sistema
Tabla 27 Tratamiento de la información
Tabla 28 Información accesible
Tabla 29 Compatibilidad externa
Tabla 30 Sistema modular
Tabla 31 Registro de nuevos dispositivos
Tabla 32 Portabilidad del sistema
Tabla 33 Integración en niveles superiores
Tabla 34 Estabilidad del desarrollo
Tabla 35 Sistema inteligente
Tabla 36 Seguridad del sistema

Tabla 4 Requisitos generales del sistema

3.2 Especificación de requisitos del sistema

3.2.1 Requisitos funcionales

Los requisitos funcionales establecen las necesidades de negocio del sistema. El desarrollo de cada requisito aporta valor de forma notable al conjunto.

Resumen requisitos funcionales del sistema
Tabla 37 Gestión remota de dispositivos
Tabla 38 Administración de datos
Tabla 39 Gestión de datos en la nube
Tabla 40 Visualización de datos
Tabla 41 Generación de informes

Tabla 5 Requisitos funcionales del sistema

3.2.2 Requisitos no funcionales

Los requisitos no funcionales del sistema establecen atributos como calidad, rendimiento o seguridad. Cuando estos requisitos se cumplen, aumentan el valor de negocio de los requisitos funcionales relacionados.

Resumen requisitos no funcionales del sistema
Tabla 42 Registro de actividad
Tabla 43 Independencia de componentes
Tabla 44 Copias de seguridad
Tabla 45 Control de acceso
Tabla 46 Implantación del sistema

Tabla 6 Requisitos no funcionales del sistema

3.2.3 Restricciones técnicas

Se considera una restricción técnica cualquier limitación externa a valorar o restricción interna de obligado cumplimiento para el correcto funcionamiento del sistema.

Resumen restricciones técnicas del sistema
Tabla 47 Tecnología de contenedores
Tabla 48 Integración continua
Tabla 49 Minimización de tiempos

Tabla 7 Restricciones técnicas del sistema

3.2.4 Requisitos de integración

Los requisitos de integración establecen capacidades de comunicación con otros sistemas.

Resumen requisitos de integración del sistema
Tabla 50 Interfaces de comunicación software
Tabla 51 Interfaces de comunicación de la instalación

Tabla 8 Requisitos de integración del sistema

3.3 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

3.4 Descripción de la solución propuesta

La monitorización propuesta se basa en la implementación de un sistema modular, basado en la nube, y dispositivos de bajo coste que estarán ubicados en sistema industrial objetivo.

Los dispositivos de obtención de datos son microcontroladores programables. Disponen de capacidades de conectividad inalámbrica y por cable, así como opciones de expansión mediante módulos compatibles.

El software implementado en los microcontroladores deberá obtener mediciones de la instalación monitorizada. Mediante el software implementado en cada uno de ellos, estos datos serán formateados para ser enviados a un gestor de mensajes.

El gestor de mensajes es el encargado de recibir mensajes y canalizarlos, pudiendo almacenarlos temporalmente, conocer su estado y suministrarlos a los sistemas que los requieran.

Los sistemas que realizan la labor de extracción de datos del gestor de mensajes serán los agentes. Los agentes podrán tratar los datos obtenidos, formatearlos e introducirlos en las bases de datos. Para respaldar el sistema, se implementará un método de sincronización de datos de una instancia local a una instancia de base de datos en la nube.

La base de datos permite almacenamiento de los mismos, suministrados por los agentes, y los expone a sistemas externos de visualización y consulta mediante accesos seguros.

Los sistemas externos permitirán al usuario la consulta de los datos, extracción de métricas e inferencia de nuevas mediciones.

3.5 Material y métodos

Para la elaboración del proyecto, se ha hecho uso de los siguientes equipos:

- Un ordenador Macbook con sistema operativo macOS 12.
- Una Raspberry Pi 3 con sistema operativo Raspbian.
- Una placa ESP32[3] con cable micro-usb
- Un ordenador con sistema operativo Windows 10

3.6 Tecnologías utilizadas

Para el desarrollo de este proyecto, se han empleado las siguientes tecnologías, agrupadas según el área funcional a la que pertenecen:

Lenguajes de programación:

- *Toit*. Lenguaje utilizado por la plataforma *Toit*[4] y *Jaguar*[5]. Es el lenguaje base de la plataforma y es de código abierto.
- Flux[6]. Lenguaje funcional de *scripts* utilizado por la base de datos *InfluxDB[7]*, compatible con múltiples sistemas como *Grafana* mediante *plugins*.
- *Python*. Lenguaje utilizado en el desarrollo del módulo de *Odoo[8]*

Librerías:

- *Toit* mqtt[9]. Librería utilizada para la implementación del cliente *MQTT[10]* en el microcontrolador *ESP32*.
- *Toit* gpio[11]. Librería utilizada para el control de los pines ubicados en el microcontrolador *ESP32*.
- *Infludb-client[12].* Librería empleada para obtener datos de *InfluxDB* desde *Odoo.*

Software:

- *Docker for Desktop[13]*. Software utilizado para el despliegue de contenedores y visualización de registro de eventos.

- *Microsoft Word[14]*. Elaboración de la memoria.
- *Visual Paradigm[15]*. Elaboración de diagramas del proyecto.
- *Visual Studio Code[16]*. *IDE* utilizado para el desarrollo de todo el proyecto.

Servicios web:

- *Scopus*[17]. Plataforma web de consulta a diferentes recursos científicos reconocidos.
- *ClickUp[18]*. Elaboración de diagramas del proyecto.

3.7 Metodología de desarrollo de software

Podemos diferenciar las metodologías de desarrollo de software en dos categorías conocidas como **metodologías tradicionales** y **metodologías ágiles[19]**.

Las metodologías tradicionales contemplan 5 fases en su ciclo de vida:

- Requisitos: Definición de funcionalidades que debería cumplir el software.
- Diseño: Definición de la estructura del proyecto.
- Desarrollo: Elaboración del *software* para alcanzar los requisitos.
- Pruebas: Evaluación de errores del *software* y cumplimiento de requisitos.
- Mantenimiento: Prevención y corrección de errores posteriores a la entrega del *software.*

Todas las fases anteriores conllevan la elaboración de la documentación asociada.

Las diferencias entre metodologías tradicionales radican en la organización de dichas fases y la granularidad de las mismas. En el siguiente apartado se detallan las características principales, puntos fuertes y puntos débiles de algunas de las metodologías tradicionales más utilizadas:

- **Cascada:** Las fases tienen lugar de forma continuada. Como ventaja establece una metodología de fácil seguimiento. Su principal desventaja es la rigidez ante los cambios, ya que los requisitos se definen desde el inicio y cada fase se inicia una vez se ha concluido la anterior.
- **Incremental:** Las fases se desarrollan en cascada, pero el entregable es considerablemente más reducido. Cada incremento suele integrar una

nueva funcionalidad completa al producto. Su principal ventaja es la capacidad de generar valor al cliente en poco tiempo, reducción de costes y flexibilidad ante cambios. Sus desventajas son similares a las metodologías ágiles, como la visión parcial del sistema o la obsolescencia de la documentación y las funcionalidades.

- **Prototipado:** Es un modelo similar al incremental. Su principal ventaja radica en la flexibilidad y comunicación con el cliente, especialmente útil en proyectos cuyas funcionalidades deseadas no se definen con claridad. Entre sus desventajas se encuentran la necesidad de alta interacción con el cliente o costes adicionales en demostraciones del producto.
- **Espiral:** Es un modelo similar al Prototipado, con especial dedicación a la gestión de riesgos. Su principal desventaja es el aumento de toma de decisiones, y su punto fuerte el control de calidad, costes y recursos.
- **Proceso unificado:** Esta metodología es un marco de trabajo de carácter iterativo o incremental, con especial enfoque en la arquitectura e identificación de riesgos, vertebrado por los casos de uso elaborados con el cliente. Su principal ventaja es la adaptabilidad a proyectos, equipos u organizaciones completas y la comunicación continua con el cliente. Sus principales desventajas vienen dadas por su adaptabilidad y generalidad, que dificultan el refinamiento en los procesos de desarrollo.

Por otro lado, las metodologías ágiles comparten una serie de características que pueden ser sintetizadas en los siguientes principios:

- **Incremental:** Se desarrollan elementos software de poca extensión en un corto periodo de tiempo.
- **Cooperación / comunicación:** Se mantiene retroalimentación constante con el cliente respecto a la evolución del proyecto.
- **Sencillez:** La organización de la metodología debe ser fácil de entender y aplicar.
- **Adaptación:** La metodología debe ser capaz de integrar cambios factibles de forma rápida.

A continuación, se resumen algunas de las metodologías ágiles más utilizadas, sus ventajas y desventajas:

- **Desarrollo de Software Adaptativo:** Se basa en la concurrencia de tareas de gestión y desarrollo del equipo. Tiene un componente característico de auto-aprendizaje y mejora continua como evaluación de nuevos incrementos. Su principal ventaja es la alta adaptabilidad, y su desventaja es la dificultad para identificar tareas y organizar el trabajo.
- **Lean:** Es una adaptación de la corriente de pensamiento elaborada por *Taiichi Ohno.* Sus principales características son la eliminación de lo superfluo, la creación de conocimiento, minimizar la toma de decisiones, realización de entregas rápidas, creación de equipo, búsqueda de la calidad y optimización global. Estos principios constituyen sus principales ventajas; como contraparte, sus desventajas derivan de una alta dependencia de empleados comprometidos o la necesidad de empleados capacitados en puestos críticos.
- **Programación extrema:** Se desarrollan necesidades software inmediatas mediante las fases de planificación, diseño, codificación y pruebas. Requiere de comunicación continua con el cliente y las necesidades desarrolladas posteriormente deben ser integradas con las funcionalidades existentes en las fases anteriores. Los roles del equipo de trabajo se definen de forma estricta para facilitar el reparto de tareas, aunque pueden ser compartidos. Presenta ciertas ventajas como la programación en pareja, la comunicación entre todos los componentes y la velocidad de entrega. Como principal desventaja se destaca la inversión extra de tiempo en integración de nuevas características o cambios no previstos.
- **Scrum:** Es un marco de trabajo que busca la entrega adaptativa del mayor valor para el cliente permitiendo descomponer proyectos complejos en tareas asumibles. Sus principales ventajas son la comunicación con el cliente y cumplimiento de expectativas, la gestión del riesgo, la adaptación a los cambios o la motivación del equipo. Como desventajas podemos mencionar la necesidad de conocimiento de la metodología ola estimación errónea de esfuerzos por cualificación insuficiente del equipo.

- **Kanban / Scrumban:** En ocasiones, debido a la escasa definición de procesos y roles, Kanban es mayormente considerado una herramienta visual aplicable a otras metodologías como Scrum y utilizada en conjunto. Scrumban adopta Kanban como herramienta visual de gestión del trabajo, tomando los roles y reuniones definidos en Scrum. Como Scrumban, su principal ventaja es la adición de una herramienta visual que permite la autogestión por parte del equipo de las tareas que se han definido. Como desventaja, la falta de experiencia del equipo en definir las tareas puede ocasionar desbalances en la carga de trabajo a lo largo del tiempo.

Para el desarrollo del proyecto será de aplicación una **metodología incremental**. El empleo de esta metodología se justifica debido a la arquitectura modular del sistema, donde sus componentes pueden ser desplegados por separado, y posteriormente, interconectados.

La metodología incremental se caracteriza por presentar sus requisitos generales al inicio, construyendo los incrementos a partir de los mismos al elaborar la valoración de las tareas a realizar y la consecución de la planificación temporal.

Esta metodología permite adaptar en cierta medida los requisitos de incrementos futuros que puedan verse afectados por incrementos previos, aunque debe limitarse a los requisitos generales para evitar demoras excesivas en el desarrollo del proyecto.

Para enmarcar el desarrollo del proyecto en esta metodología, el periodo de elaboración se organizará en incrementos, y cada incremento en fases, según se detalla en la Tabla 9 Fases del incremento.

Fase	Descripción
Objetivos del incremento	Se establecen los objetivos a completar en la fase, así como los requisitos relacionados.
Análisis y Diseño	Conceptualización de una solución capaz de cumplir los objetivos.
Implementación	Elaboración de una solución a partir del diseño previo.
Pruebas	Comprobación de adecuación de la solución a los objetivos del incremento.

Tabla 9 Fases del incremento

3.8 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

La distribución de tareas se presenta en la Ilustración 1 Diagrama WBS reducido, en un primer nivel. El diagrama **WBS** completo puede encontrarse en el Ilustración 27 Diagrama WBS completo.

La estimación de esfuerzo en horas viene dada en el apartado, donde se detallan las horas estimadas para las tareas del diagrama WBS.

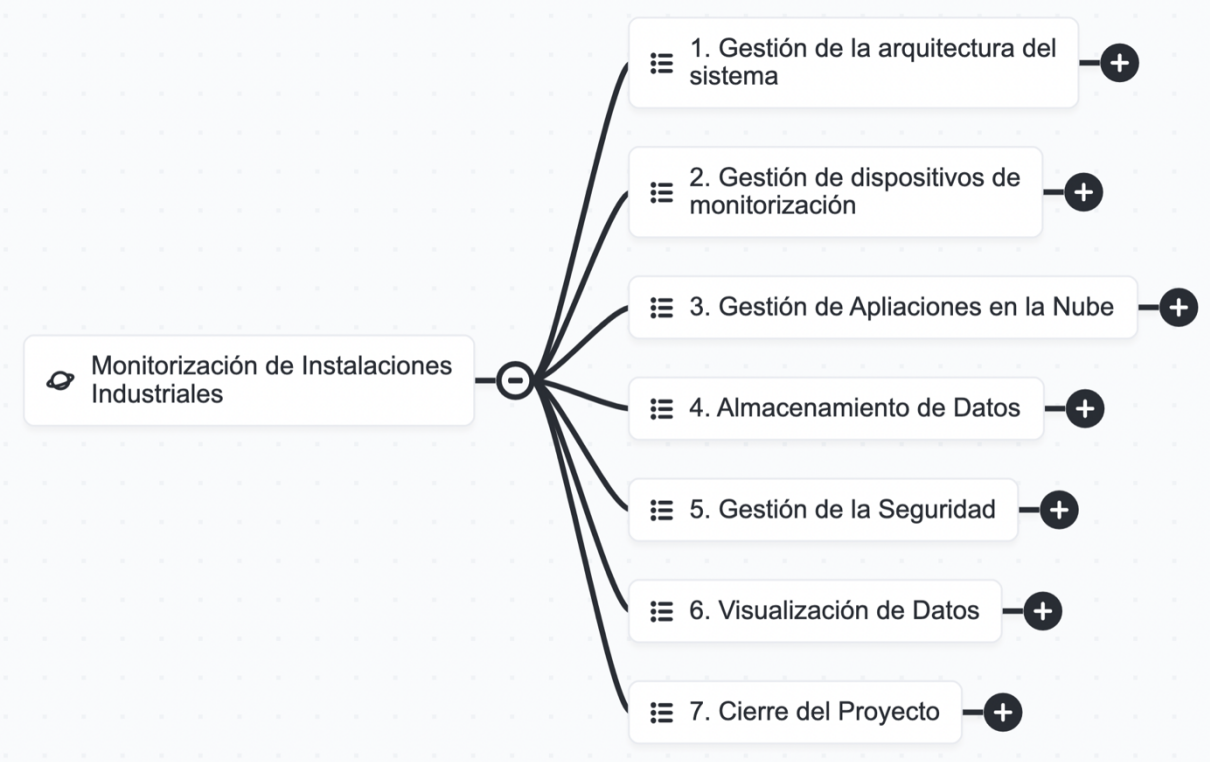

Ilustración 1 Diagrama WBS reducido

3.9 Planificación temporal

Se ha estimado el tiempo para llevar a cabo las tareas del proyecto, quedando resumidas en la siguiente tabla:

Tareas	Estimación en Horas
1. Gestión de la arquitectura del sistema	
1.1 Análisis del proyecto	40
1.2 Análisis de objetivos y alcance	10
1.3 Diseño y planificación del proyecto	26
2. Gestión de dispositivos de monitorización	
2.1 Despliegue de aplicaciones en dispositivos	12
2.2 Plataforma Toit	20
2.3 Registro e integración de dispositivos	4
2.4 Pruebas de usuario	3
3. Gestión de aplicaciones en la nube	
3.1 Gestión de contenedores	28
3.2 Despliegue de aplicaciones: Mosquitto	16
3.3 Pruebas de usuario	6
4. Almacenamiento de datos	
4.1 Configuración InfluxDB	12
4.2 Almacenamiento desde Telegraf	12
4.3 Consultas	3
4.4 Pruebas de usuario	
5. Visualización de datos	
5.1 Diseño del dashboard	6
5.2 Presentación de datos y actualización	4
5.3 Pruebas de usuario	4
6. Integración Odoo	
6.1 Configuración del entorno	10
6.2 Desarrollo del módulo	32
6.3 Pruebas	6
7. Cierre del proyecto	
7.1 Despliegue de la versión 1.0	4
7.2 Pruebas	4

Tabla 10 Estimación de esfuerzo en tiempo

Se ha dividido la carga de trabajo en 4 iteraciones, cuya distribución se ha escogido en función de los objetivos establecidos. El resto del tiempo hasta completar las 300 horas se ha dedicado a la elaboración de la documentación.

Incremento	Grupo de tareas	Horas
Periodo previo	1	76
Incremento 1	2	39
Incremento 2	3.1 y 3.2	44
Incremento 3	3.3 - 4 - 5	47
Incremento 4	6 - 7	56
Horas totales		262

Tabla 11 Organización de tareas en incrementos

En la Ilustración 2 Diagrama de Gantt del proyecto se presenta la distribución temporal de las tareas expuestas en la Tabla 10 Estimación de esfuerzo en tiempo.

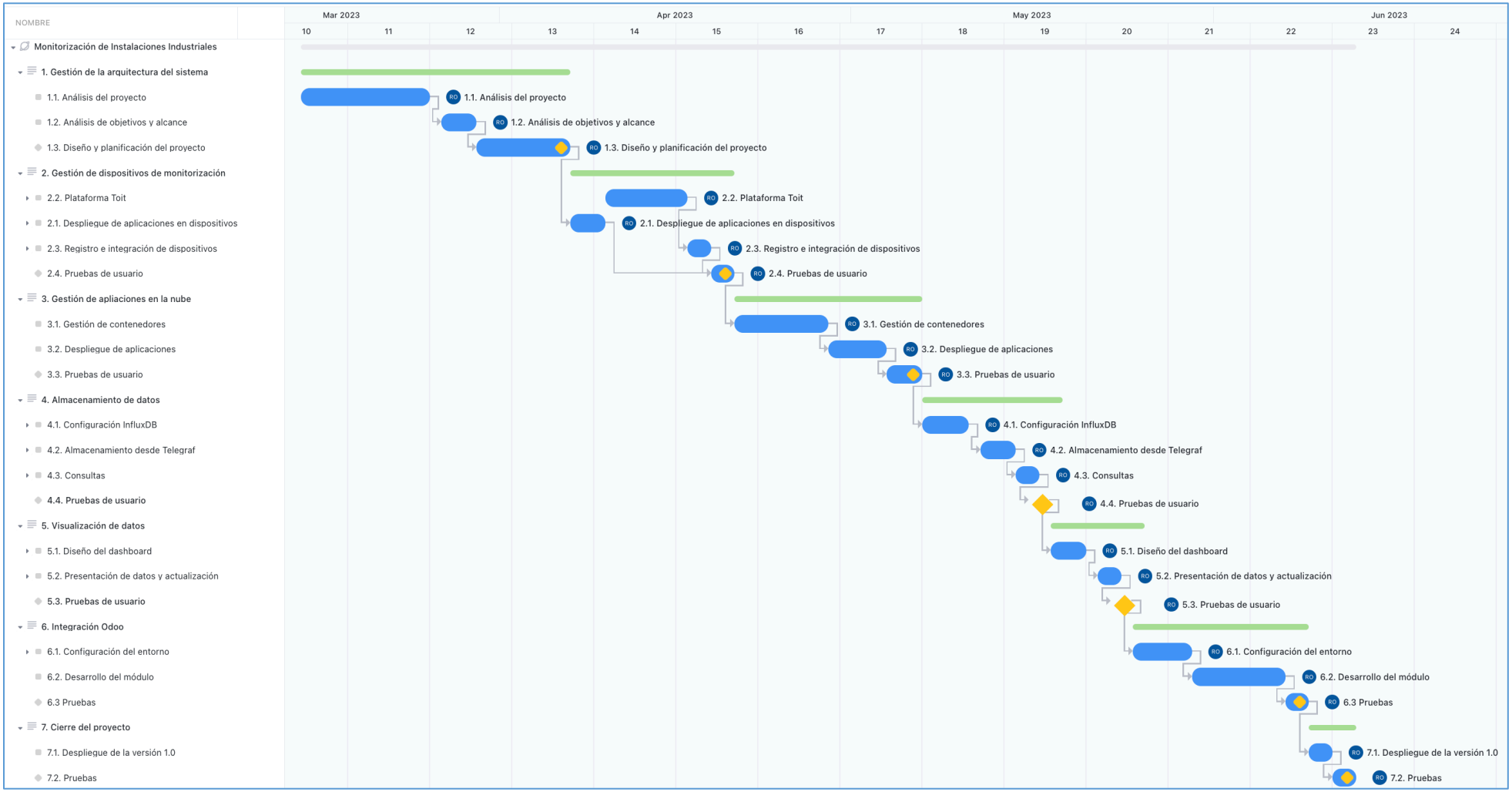

Ilustración 2 Diagrama de Gantt del proyecto

3.9.1 Mecanismos de control de calidad

El aseguramiento de la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

4 DESARROLLO DE INCREMENTOS

4.1 Visión general del sistema

El sistema se organiza según el modelo arquitectónico de la Ilustración 3 Arquitectura general del sistema.

Es un sistema donde la información se transmite desde los microcontroladores, situados en la instalación monitorizada, hasta la visualización adecuada de estos datos y exposición en el **ERP**.

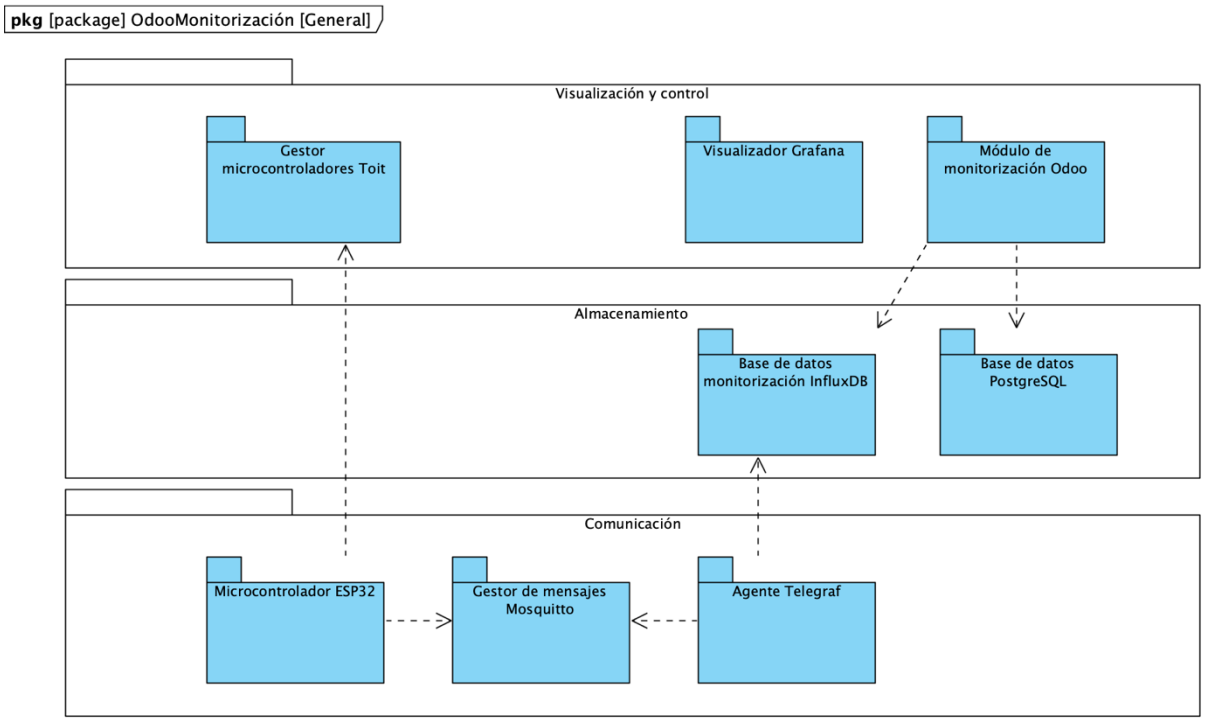

Ilustración 3 Arquitectura general del sistema

4.2 Periodo previo

En el primer periodo de elaboración se ha realizado la especificación de la arquitectura a seguir. Se han establecido las tecnologías imprescindibles a utilizar en el proyecto, así como el tiempo necesario de aprendizaje y estudio de alternativas ante los problemas encontrados en el planteamiento de soluciones.

Por último, se han definido los requisitos y elaborado los distintos diagramas que definen el sistema, desde lo general a lo concreto.

4.3 Primer incremento

En este primer incremento se llevará a cabo la gestión de dispositivos de monitorización. Se trabajará en el despliegue en el microcontrolador desde la web de administración de *Toit*.

4.3.1 Análisis y Diseño

El entorno de trabajo de Toit nos permite registrar dispositivos en el sistema, monitorizar su estado, desplegar aplicaciones desde la nube al microcontrolador y actualizar el ***firmware***, las aplicaciones desarrolladas y configuraciones de conectividad.

Haciendo uso de las funcionalidades disponibles, se prepara un dispositivo *ESP32* con el *firmware* de Toit. Una vez registrado, se desarrolla una pequeña aplicación que permite obtener medidas de los sensores integrados en la placa. Estos datos, en pasos posteriores, serán enviados a otros sistemas.

El microcontrolador debe obtener las mediciones de los sensores, tratando los datos y enviándolos al gestor de mensajes.

El esquema de funcionamiento básico se detalla en la Ilustración 4.

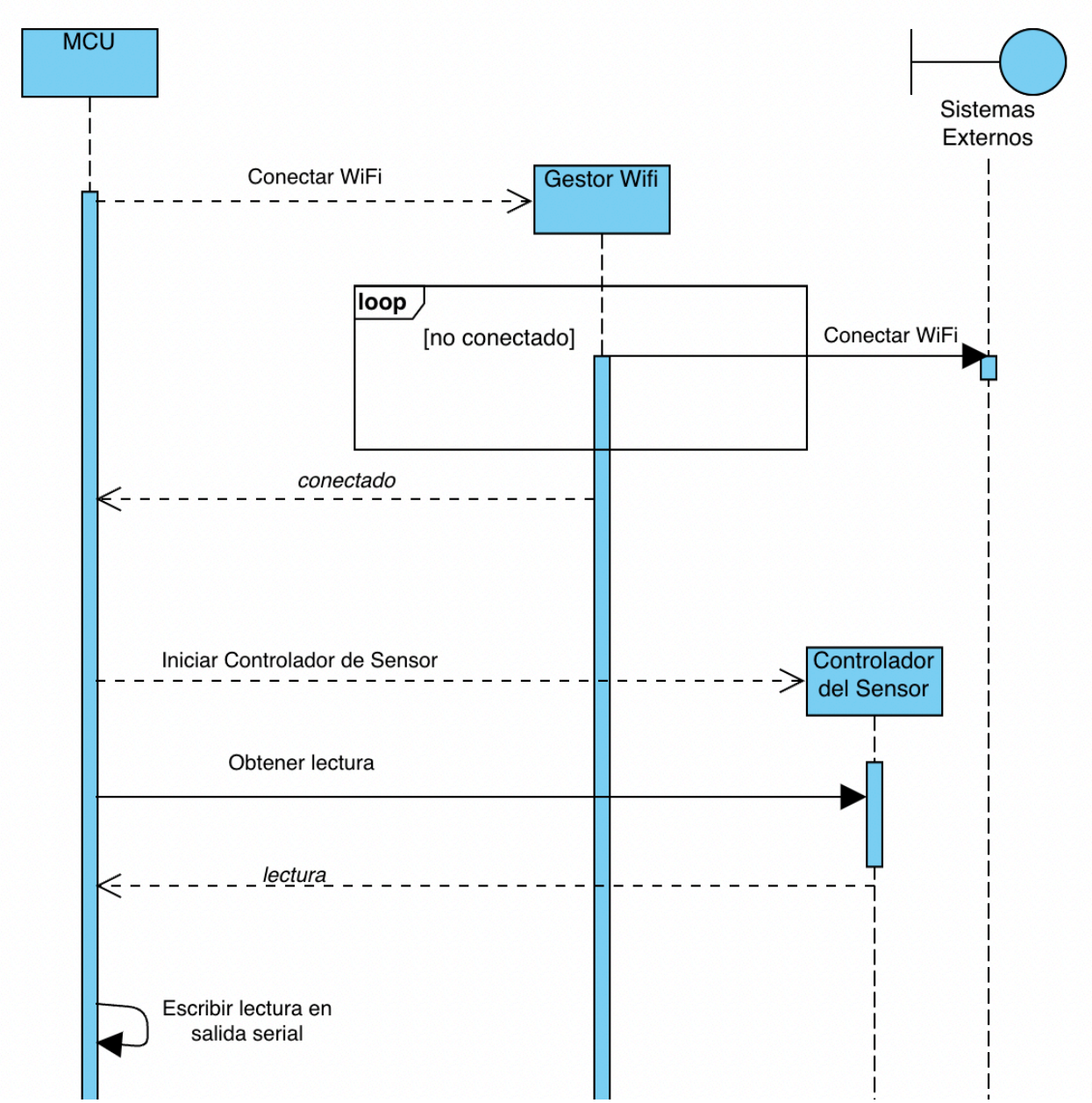

Ilustración 4 Diagrama de secuencia sobre el envío de una medición

La unidad microcontroladora (MCU) siempre se encarga de iniciar el ciclo, por lo que es importante mantener el elemento Gestor Wifi en todas las implementaciones para no perder la comunicación con la placa y poder realizar comprobaciones y actualizaciones.

4.3.2 Implementación

Como requisito previo, es necesario registrarse en el sitio web del proyecto y acceder al panel principal de usuario. Permite registro alternativo con cuenta de *Google[20]* o *GitHub[21]*.

Para instalar el *firmware* en el microcontrolador *ESP32* es necesario descargar desde el sitio web de *Toit* la interfaz de consola y configurarla en el sistema. También será necesario instalar los controladores de puerto serie de *Silicon Labs [22].*

El proceso de instalación hace uso del navegador de forma conjunta con la herramienta descargada. Se debe conectar la placa por ***USB*** en el ordenador donde se ha instalado la herramienta para realizar un primer aprovisionamiento del *firmware* y aplicar una configuración de red válida que permita acceder a las funcionalidades *online* posteriormente.

El proyecto *Toit* tiene disponible su versión de código abierto conocida como *Jaguar*. Tanto la versión estándar privada como la versión de código abierto permiten iniciar nuevas aplicaciones instaladas en el microcontrolador sin la necesidad de reiniciar el dispositivo manualmente, dotándolo de gran flexibilidad y velocidad.

Tras completar la configuración básica de funcionamiento, se desarrolla una aplicación de prueba que obtiene unas mediciones mediante los sensores integrados en la placa. Esta aplicación es la que se ha definido anteriormente en la Ilustración 4 Diagrama de secuencia sobre el envío de una medición.

Por defecto, la salida estándar al utilizar el *firmware* de *Toit*, se presenta en su plataforma web.

La plataforma web se estructura como un panel de administración desde el cual se tiene acceso a todas las funcionalidades en el menú lateral y la ventana activa en la zona principal como podemos ver en la Ilustración 5 Sección de configuración inicial del microcontrolador.

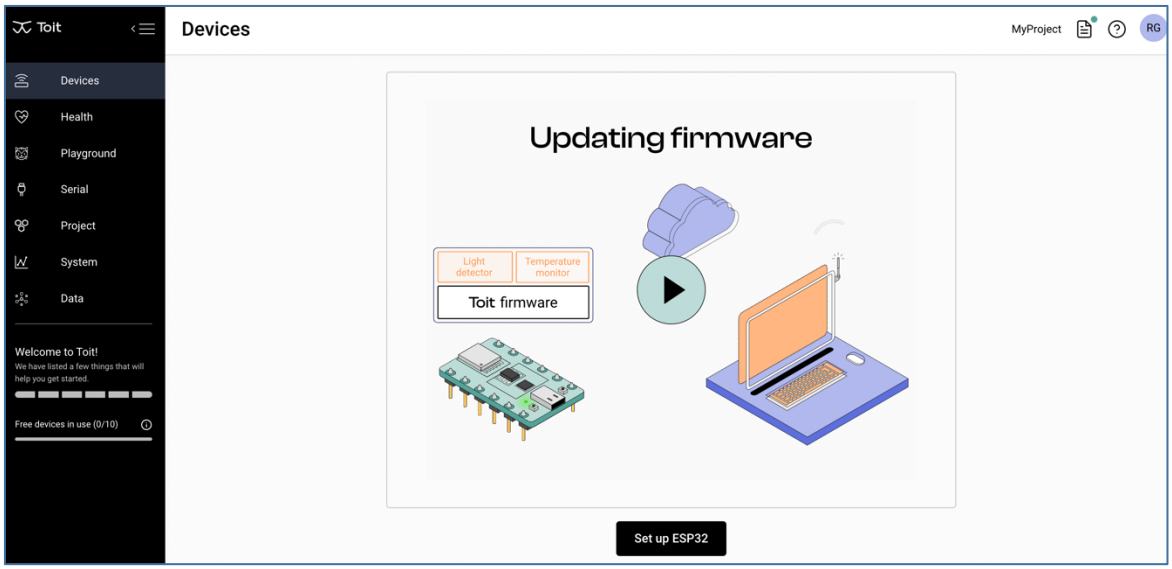

Ilustración 5 Sección de configuración inicial del microcontrolador

Podemos ver los mensajes registrados por la plataforma en la opción "Dispositivos", seleccionando alguno de nuestros dispositivos activos y entrando en la sección "*Logs*".

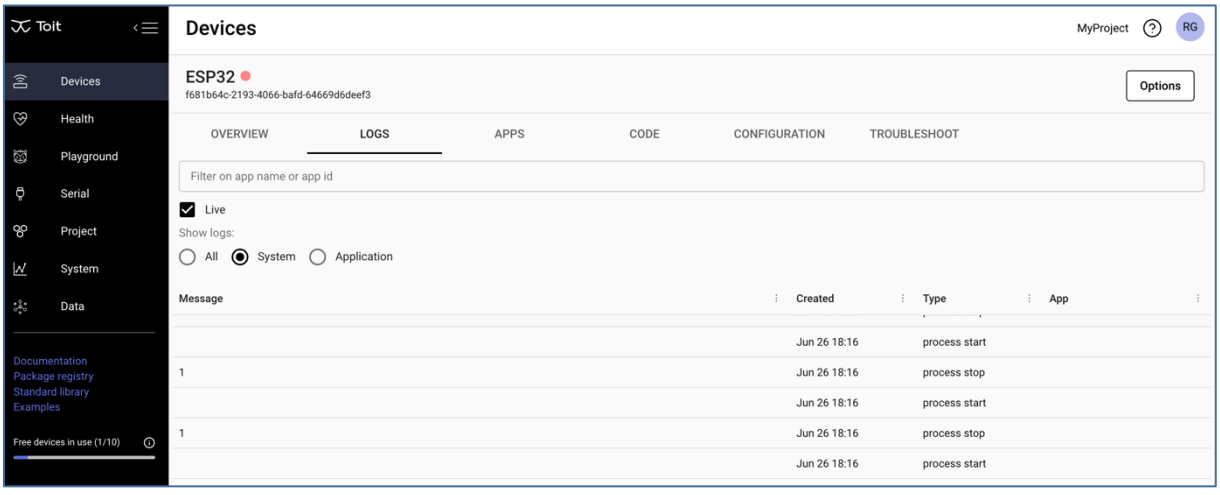

Ilustración 6 Registro de eventos de un microcontrolador

Otras secciones como "*Playground*" permiten el desarrollo de código con resaltado en color de forma *online*. También es posible obtener datos de salida mediante puerto serie *USB* en tiempo real.

La plataforma posee una implementación propia de la librería *PubSub*[23] de *Google*. Los mensajes publicados por los dispositivos registrados en el proyecto son aceptados por la nube privada, estando disponibles mediante la librería propia *PubSub* de *Toit*[24] o mediante ***token*** de acceso.

La administración de los dispositivos se puede realizar por completo en el sitio web. Toda la configuración aplicada es actualizada vía ***OTA***, no siendo necesario volver a conectar el dispositivo al ordenador. Desde el menú de la Ilustración 7 es posible seleccionar el *firmware* que utilizará el microcontrolador, establecer las conexiones *wifi* o el nivel de detalle de los mensajes enviados al registro de eventos.

También es posible seleccionar el tiempo máximo que el dispositivo puede permanecer sin conexión, un parámetro interesante en instalaciones cuyas necesidades de supervisión son críticas o aquellas donde el dispositivo se ha implantado con una batería o suministro limitado de energía.

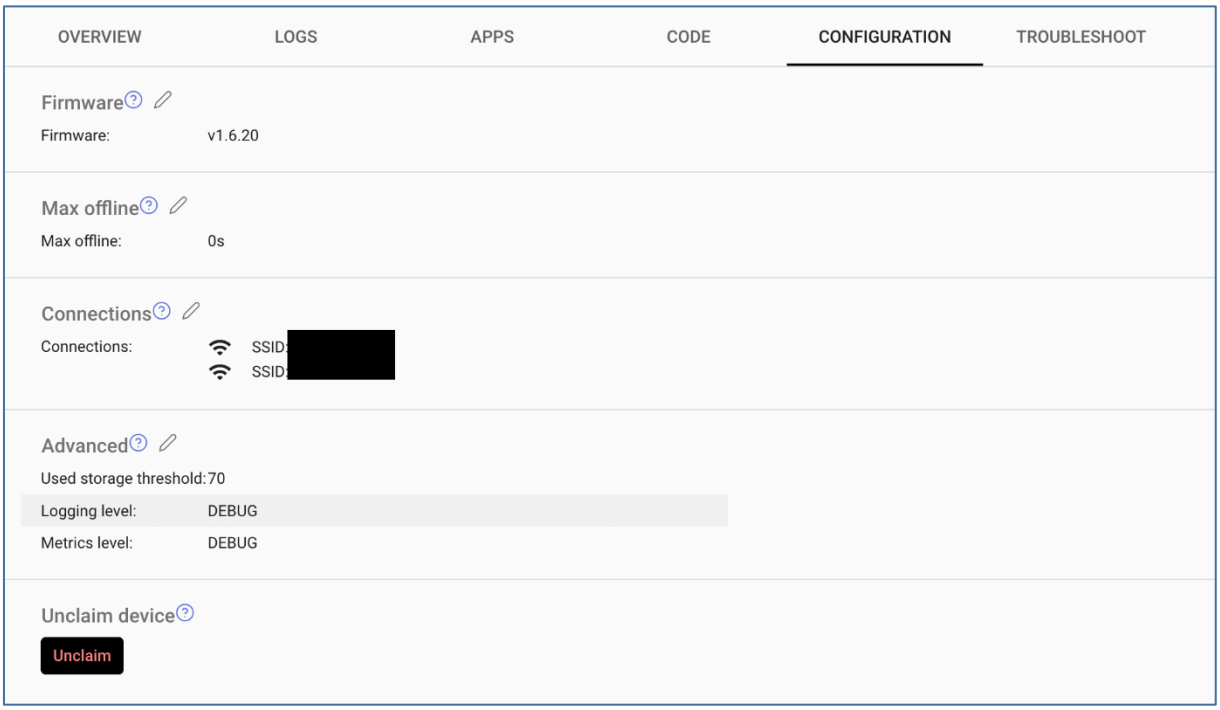

Ilustración 7 Panel detalle de la configuración del microcontrolador

4.3.3 Pruebas

Para la batería de pruebas se ha registrado varios programas en el microcontrolador, procedentes de las librerías ejemplo de la plataforma.

En concreto, se han desplegado las aplicaciones descritas en la Tabla 12 Aplicaciones de prueba desplegadas.

Librería	Ejemplo	Descripción	Resultado	
GPIO	Blinking led	Apagar y encender un led en la placa	Error	El led de la placa no es posible gestionarlo.
PubSub	Publish	Envía un mensaje al gestor de mensajes de la plataforma	Correcto	El resultado se muestra en la ventana de suscripciones
BLE	Scanning	Búsqueda de dispositivos bluetooth	Correcto	Se muestran dispositivos detectados
Programa implementado		Envío de lectura de pines capacitivos	Correcto	El sistema detecta la interacción externa

Tabla 12 Aplicaciones de prueba desplegadas

4.4 Segundo incremento

En el segundo incremento se llevan a cabo las configuraciones de las distintas herramientas y aplicaciones que basaran sus funcionalidades en la nube.

4.4.1 Análisis y Diseño

Las soluciones basadas en la nube gozan de bondades únicas, así como de dificultades añadidas. Para este proyecto, la solución será de carácter remoto, sin realizar un despliegue en una gran plataforma comercial como, puede ser *Google Cloud[25]*, *AWS[26]* o *Azure[27]*.

Esta solución se aloja en un ordenador conectado en la misma red que los microcontroladores encargados de realizar las operaciones de obtención de datos, simulando una conexión remota. Las comunicaciones se estructuran según el diagrama de la Ilustración 8.

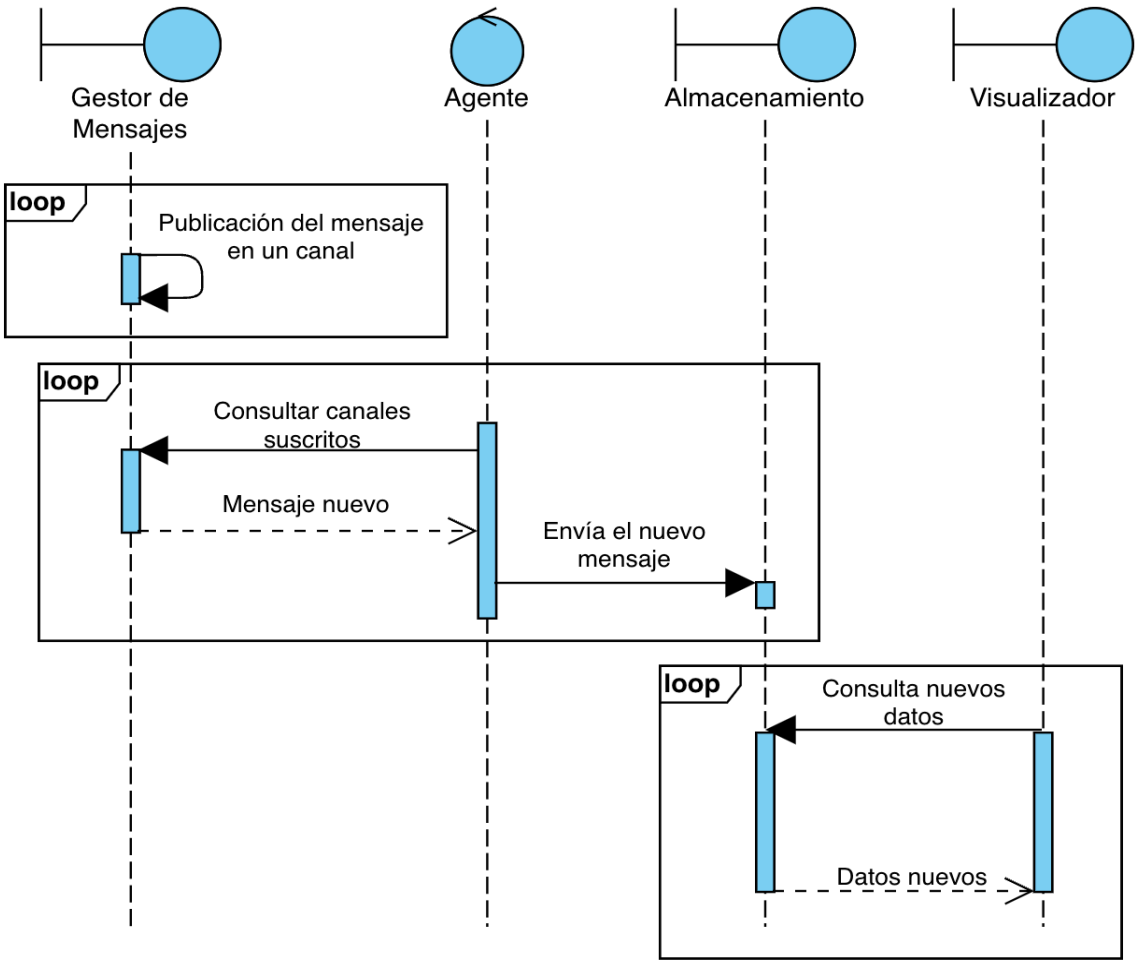

Ilustración 8 Diagrama general de tratamiento remoto de datos

Todo el sistema está diseñado para ser implementado mediante contenedores. Esto garantiza el correcto despliegue en un mayor número de plataformas, la escalabilidad de las aplicaciones unitarias que conforman el sistema y la mejora futura de cada una de ellas.

Un contenedor es una unidad software que alberga todo lo necesario para que las aplicaciones desarrolladas no alteren su correcto funcionamiento ante posibles cambios de entorno que tienen lugar tras avances en fases de desarrollo, despliegue en servidores o despliegue de servicios y aplicaciones en la nube, entre otros. La información puede estar conformada por archivos de configuración, datos básicos de entrada o de inicio, así como código fuente desarrollado o aplicaciones completas con sus dependencias.

Para esta solución se ha utilizado como tecnología de contenedores *Docker*. Algunas de las principales utilidades son:

1. Aislamiento de aplicaciones: Permite crear contenedores que aíslan las aplicaciones del sistema operativo y de otras aplicaciones en ejecución, lo que permite que múltiples aplicaciones se ejecuten en la misma máquina sin interferir entre sí.

2. Portabilidad: Las aplicaciones contenidas son altamente portables, lo que significa que se pueden adaptar fácilmente a diferentes entornos como pueden ser de desarrollo, pruebas o producción sin cambios en su comportamiento.

3. Eficiencia: Los contenedores son en cuanto a recursos como memoria o almacenamiento, lo que implica que se pueden ejecutar muchas instancias de una aplicación en una sola máquina.

4. Escalabilidad: Se pueden crear y ejecutar múltiples instancias de contenedores de una aplicación para satisfacer las condiciones de disponibilidad, respuesta o rendimiento del sistema de forma dinámica y automatizada.

Las unidades que conforman el sistema remoto son:

- *Broker* o gestor de mensajes
- Agente
- Almacenamiento de datos
- Visualizador
- ERP

En este incremento se acometerá la configuración del gestor de mensajes, por lo que los elementos Agente, Almacenamiento de datos, Visualizador y ERP serán elaborados en profundidad en los siguientes incrementos.

La principal función del *broker* o gestor de mensajes es la de recibir los datos de sistemas publicadores, mantener estos mensajes y distribuirlos cuando son reclamados por agentes suscriptores. En la solución elegida, se utilizará el gestor de mensajes conocido como *Mosquitto[28]*.

La recepción de un mensaje se produce en una cola de mensajes, identificada mediante un *topic* o tema con el nombre identificativo del dispositivo que realiza el envío. Se utiliza como *topic* el asunto del mensaje enviado al gestor de mensajes. Cuando se recibe un mensaje nuevo, el gestor realiza una búsqueda de coincidencias con el asunto del mensaje en los temas que tiene definidos, guardando el mensaje en la cola correspondiente.

El tiempo que un mensaje permanece en la cola es dependiente de la configuración, pudiendo establecerse diferentes criterios y políticas de Calidad de Servicio o *QoS*, por sus siglas en inglés Quality of Service[29]. Los niveles se especifican en la Tabla 13 Niveles de Calidad de Servicio.

Niveles de Calidad de Servicio	
Nivel 0	Una vez se ha enviado el mensaje, se elimina de la cola.
Nivel 1	Se garantiza el envío del mensaje al menos una vez. La respuesta ante el envío no es responsabilidad del gestor de mensajes, lo que puede ocasionar envíos múltiples si no se confirma la recepción al gestor. Requiere más recursos que el nivel anterior.
Nivel 2	Se garantiza la recepción del mensaje una vez y solo una. El uso de este requiere de un nivel bastante elevado de recursos en comparación a los niveles previos.

Tabla 13 Niveles de Calidad de Servicio

Las políticas de calidad de servicio son aplicables tanto a agentes publicadores como agentes suscriptores del gestor de mensajes.

Para el gestor elegido, la política de calidad del servicio utilizada por defecto es el nivel 0. Este nivel no envía confirmación al recibir el mensaje. Las conexiones para el envío se realizan mediante protocolo TCP para mantener el orden de los mensajes.

Para esta solución, se ha otorgado la funcionalidad de solicitar los mensajes a los agentes suscriptores.

4.4.2 Implementación

El sistema remoto hace uso de repositorios de imágenes de contenedores. El desarrollo de esta sección consistirá en la implementación del archivo de configuración y especificación de estas aplicaciones como conjunto, los archivos de configuración individuales de cada aplicación, y las acciones básicas de configuración para la puesta en marcha.

Para la puesta en marcha de los contenedores definidos, se hará uso de la herramienta *docker-compose,* incluida en las librerías de *Docker[30].* Su archivo de configuración utiliza el formato ***YAML***, que en esta implementación llamaremos *compose.yaml*. Este nombre es reconocido por el editor utilizado, *Visual Studio Code,* de forma automática al utilizar la extensión de *Docker*. El archivo se estructura según la Ilustración 9 Estructura básica del archivo utilizado por docker-compose.

```yaml
services:
  nombre_del_servicio:
    image: imagen_a_utilizar
    container_name: nombre_del_contenedor
    volumes:
      - ruta_de_carpeta_local:ruta_de_carpeta_dentro_del_contenedor
    ports:
      - puerto_aplicación_docker:puerto_interno_del_contenedor
    networks:
      - nombre_de_red
networks:
  nombre_de_red:
volumes:
  nombre_del_servicio_cuyo_volumen_se_instancia
```

Ilustración 9 Estructura básica del archivo utilizado por docker-compose

En este archivo se definen los contenedores en una sola red ya que, en su despliegue, se utilizará como un contenedor único.

Para el despliegue de las aplicaciones definidas en este archivo, es necesario operar en una interfaz de consola, ejecutar el comando de despliegue, situándose en el directorio del archivo o bien, especificando su ruta en el sistema. En la Ilustración 10 se observa que el comando se ejecuta en el directorio del archivo *compose.yaml* y se aplica el indicador *-d,* eludiendo las salidas por consola realizadas por la aplicación *Docker*.

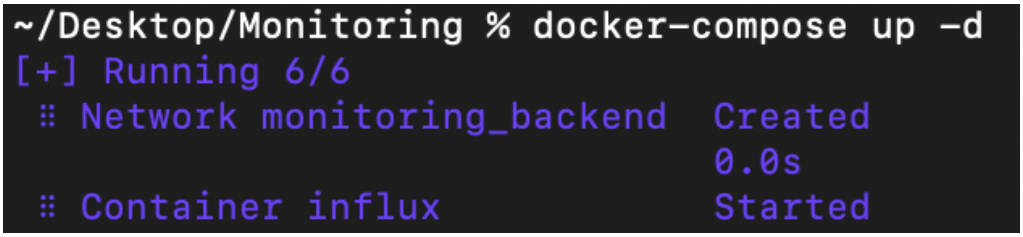

Ilustración 10 Inicialización del contenedor conjunto

En el archivo de configuración del gestor de mensajes se establecen los parámetros recogidos en la Tabla 14 Parámetros de configuración mínimos del gestor de mensajes.

Parámetro	Valor	Descripción
persistence	True / False	Permite el almacenamiento de los mensajes enviados por un cliente hasta alcanzar un máximo.
persistence_location	Ruta de archivo	Ruta de archivo donde se creará la base de datos.
log_type	error, warning, notice, information	Permite establecer el nivel de detalle de los registros de eventos.
log_dest	Ruta de archivo	Ruta de archivo donde se creará el registro de eventos.
listener	Número entre el 1 y el 65536.	Puerto a través del que se realizan las peticiones. Suele utilizarse el 1883 para conexiones normales y el 8883 para conexiones seguras.
allow_anonymous	True / False	Establece la directiva de acceso por usuario registrado o anónimo.

Tabla 14 Parámetros de configuración mínimos del gestor de mensajes

Con estos parámetros mínimos de configuración ya es posible la creación de una instancia del gestor de mensajes.

mosquitto > config > mosquitto.conf

```
1   persistence true
2   persistence_location /mosquitto/data/
3   log_type all
4   log_dest file /mosquitto/log/mosquitto.log
5   listener 1883
6   allow_anonymous true
```

Ilustración 11 Configuración básica del gestor de mensajes

Una vez creado el archivo de configuración específico de la aplicación, se detalla sus parámetros de configuración en el archivo *compose.yaml*, como se muestra en la Ilustración 12.

```
1   services:
2     mosquitto:
3       image: eclipse-mosquitto
4       container_name: mosquitto
5       volumes:
6         - ./mosquitto:/mosquitto
7         - ./mosquitto/data:/mosquitto/data
8       ports:
9         - 1883:1883
10      networks:
11        - backend
```

Ilustración 12 Fragmento del archivo compose.yaml

Se procede al primer inicio de la aplicación mediante consola como se ha detallado en la Ilustración 10. En este primer despliegue no se ha procedido a la creación de las credenciales de acceso al gestor de mensajes. Este aspecto se contempla en la Ilustración 13 Creación de usuarios en el gestor de mensajes.

4.4.3 Pruebas

Para comprobar el correcto funcionamiento del gestor de mensajes se han realizado una serie de tareas, resumidas en la tabla siguiente:

Pruebas del segundo incremento		Resultado
Tarea 1	Establecer conexión con el bróker mediante interfaz de consola de comandos.	Correcto
Tarea 2	Iniciar sesión con un usuario registrado en el bróker, publicar un mensaje.	Correcto
Tarea 3	Iniciar sesión en una terminal diferente, obtener el mensaje anterior	Correcto
Tarea 4	Comprobar la recepción y envío de mensajes en tiempo real mediante ambas terminales.	Correcto
Tarea 5	Conectar el dispositivo de monitorización. Comprobar mediante suscripción con la terminal cliente que los mensajes llegan correctamente al topic definido por el dispositivo.	Correcto

Tabla 15 Pruebas del segundo incremento

4.5 Tercer incremento

En este incremento se realiza la configuración de la base de datos, la configuración del agente y la configuración del cuadro de mandos para la presentación de datos.

4.5.1 Análisis y Diseño

Continuando con la Ilustración 8 Diagrama general de tratamiento remoto de datos es necesario determinar los siguientes elementos:

- **Almacenamiento de datos**: La base de datos tiene como principal función el almacenamiento estructurado y la consulta de datos. Los datos obtenidos en el proceso de monitorización de sistemas se caracterizan por ser series temporales. Para esta tipología de datos, se ha elegido *InfluxDB*, una base de datos especializada en almacenamiento y consulta de esta tipología.
- **Agente**: La función del agente es servir datos entre aplicaciones de forma activa. El agente elegido para esta solución es Telegraf. El agente realiza las peticiones de datos al gestor de mensajes, actuando como suscriptor. Las suscripciones al gestor de mensajes pueden ser a un único *topic,* a todos los *topics,* o a un subconjunto filtrado de los mismos. En la presente solución, realizaremos una suscripción a todos los *topic.* Esta suscripción de carácter general permite la inclusión de nuevos microcontroladores y toda la pila de monitorización desarrollada sin tener que realizar cambios en la configuración del agente.
- **Visualizador de datos**: La función del visualizador de datos es unificar y presentar todos los orígenes de datos en un solo espacio.

  Para la solución escogida, se ha utilizado *Grafana*. La aplicación es configurada con los orígenes de datos a utilizar, y se establecen las condiciones de consulta de estas ubicaciones de datos.

Una vez se han establecido los orígenes de datos, se elaboran los paneles de control. Los paneles se diseñan buscando la coherencia con los datos que van a ser presentados.

4.5.2 Implementación

InfluxDB permite la gestión mediante Organizaciones, espacios de trabajo compartidos a nivel de usuario donde se asignan diferentes recursos como *buckets*, *dashboards* o usuarios pertenecientes a la organización.

Los datos almacenados en *InfluxDB* se almacenan en *buckets*, cuya característica principal es el tiempo de retención. Estas localizaciones de datos temporales permiten configurar la retención de los datos almacenados y su borrado automatizado.

En *InfluxDB* es posible trabajar con varios métodos de comunicación dedicados a la escritura de datos:

- Carga desde archivo. El formato de archivo debe ser ***CSV,*** soportando la inclusión de cabeceras de tabla, propiedades de tabla, propiedades de fila o la inclusión de tablas múltiples en un solo archivo.
- Escritura directa mediante clientes. *InfluxDB* dispone de varios clientes en diferentes lenguajes diseñados para comunicarse con esta base de datos. Los clientes utilizan el protocolo usual de comunicación de *InfluxDB,* conocido como *Line Protocol.*

Para interconectar las aplicaciones que vamos a desplegar es necesario crear credenciales de acceso y registrarlas en las configuraciones de cada una.

En primer lugar, se realizará la creación de un usuario de acceso en el gestor de mensajes. Será necesario ejecutar mediante interfaz de consola el comando de la Ilustración 13 Creación de usuarios en el gestor de mensajes, donde se solicitará una contraseña. Esta contraseña será guardada en el archivo *password.txt* indicado.

```
mosquitto_passwd -c /mosquitto/config/password.txt < nombre_de_usuario >
```

Ilustración 13 Creación de usuarios en el gestor de mensajes

Por último, se detiene el servicio en la aplicación de *Docker*, y se modifica el archivo de configuración del gestor de mensajes. Se deben agregar las líneas que se detallan en la Ilustración 14 Activación de la autenticación del gestor de mensajes.

```
## Authentication ##
allow_anonymous false
password_file /mosquitto/config/password.txt
```

Ilustración 14 Activación de la autenticación del gestor de mensajes

A continuación, se prepara la base de datos. La configuración de *InfluxDB* nos permitirá establecer los datos de acceso y crear credenciales para aplicaciones externas como agentes u otros sistemas. La configuración inicial se detalla en el Apéndice C: Configuración Inicial de InfluxDB.

Una vez configurada la base de datos y el acceso desde el gestor de mensajes, veremos las opciones de configuración y visualización de elementos en el cuadro de mandos.

Para el desarrollo del cuadro de mandos se ha utilizado la herramienta *Grafana.* Esta herramienta de software libre es una aplicación web cuya función principal es la obtención de datos de diversas fuentes para presentar la información al usuario mediante gráficas o paneles.

Sus principales utilidades incluyen:

1. Visualización de datos: *Grafana* es una herramienta muy útil para presentación de datos provenientes de una amplia variedad de fuentes, incluyendo bases de datos, sistemas de monitorización de infraestructuras, servicios en la nube, entre otros. Los datos se pueden representar en gráficos, paneles, tablas y mapas.
2. Análisis de datos: Permite la creación de estadísticas en tiempo real y alertas, que permiten a los usuarios obtener información valiosa sobre el rendimiento y la salud de sus sistemas y aplicaciones.

3. Personalización: Otorga a los usuarios capacidad de adaptar la herramienta a sus necesidades específicas. Los usuarios pueden crear paneles personalizados, integrar plugins o aplicaciones de terceros y personalizar el aspecto y la funcionalidad de sus gráficos.
4. Colaboración: Los equipos pueden realizar tareas cooperativas a través de funciones de compartición de paneles y el acceso a diferentes roles y permisos.

Ventajas de *Grafana*:

1. Facilidad de uso, gracias a su interfaz de usuario intuitiva que facilita la creación de gráficos y paneles.
2. Integrable, mediante una gran cantidad de adaptadores disponibles.

Algunos inconvenientes asociados a *Grafana*:

1. Complejidad técnica: A pesar de su facilidad de uso, la configuración de consultas o elaboración de gráficos avanzada presenta dificultades para usuarios menos experimentados o aquellos que no estén familiarizados con el lenguaje de consultas de base de datos.
2. Escalabilidad: A medida que se agregan más fuentes de datos y se crean más paneles y gráficos, la herramienta puede volverse más lenta y requerir más recursos.
3. Dificultad de depuración: Cuando surge un problema puede ser difícil determinar si el problema se encuentra en la propia aplicación, en el sistema operativo o en la fuente de datos elegida.

La obtención de datos se puede realizar mediante acceso directo a las bases de datos soportadas de forma nativa o mediante extensiones de la aplicación, realizadas en gran número por los usuarios. La obtención de datos es programable de forma periódica, por lo que la información representada es dinámica.

Para la instalación de *Grafana* en nuestro sistema, procederemos accediendo a la ruta por defecto, donde se nos solicitará un registro.

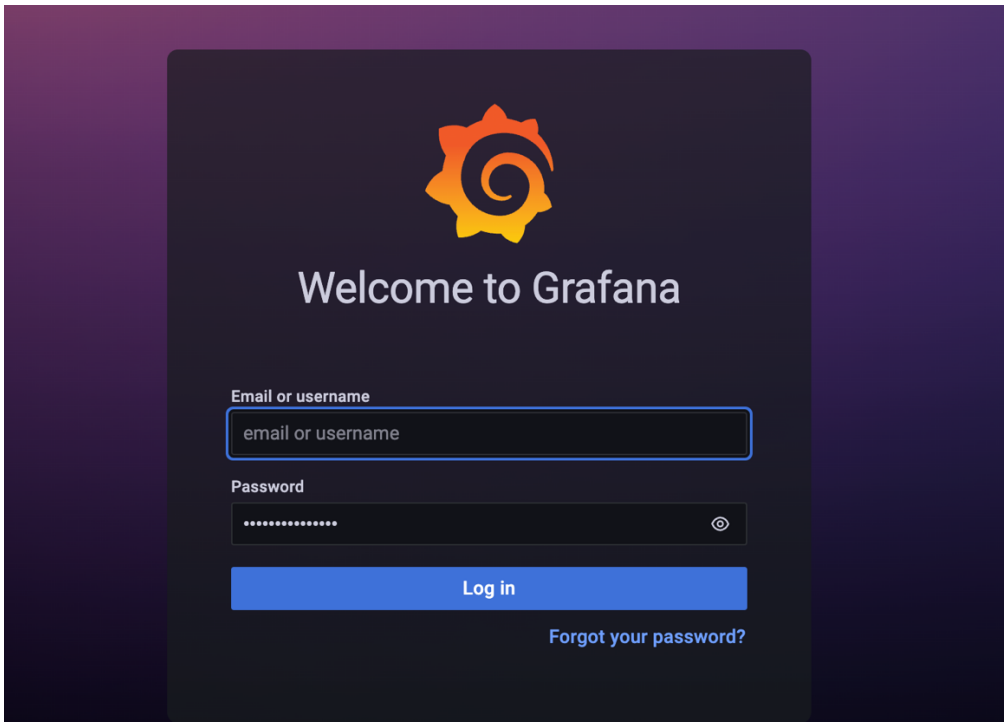

Ilustración 15 Login de Grafana

Este primer registro se realiza por defecto mediante el usuario especificado en el archivo de configuración, tal y como se especifica en la Ilustración 16 Usuario administrador por defecto de Grafana.

```
260  ################################### Security ############################
261  [security]
262  # disable creation of admin user on first start of grafana
263  disable_initial_admin_creation = false
264
265  # default admin user, created on startup
266  admin_user = rgo00020
267
268  # default admin password, can be changed before first start of grafana, or in profile settings
269  admin_password = ujaenpwd
270
```

Ilustración 16 Usuario administrador por defecto de Grafana.

Una vez realizado el inicio de sesión se muestra el cuadro de mandos principal.

Este cuadro de mandos se compone de paneles personalizables, que son los utilizados para mostrar los datos. A continuación, se creará un panel donde se podrán obtener y mostrar datos que pueden proceder de múltiples orígenes. En este caso, el origen de datos a configurar previamente es *InfluxDB.* La configuración se detallará en el Apéndice E: Configuración de orígenes de datos en Grafana.

El nuevo panel, mostrado en la Ilustración 17 Mediciones en tiempo real obtiene los datos generados por los microcontroladores, plasmándolos en tiempo real de forma gráfica.

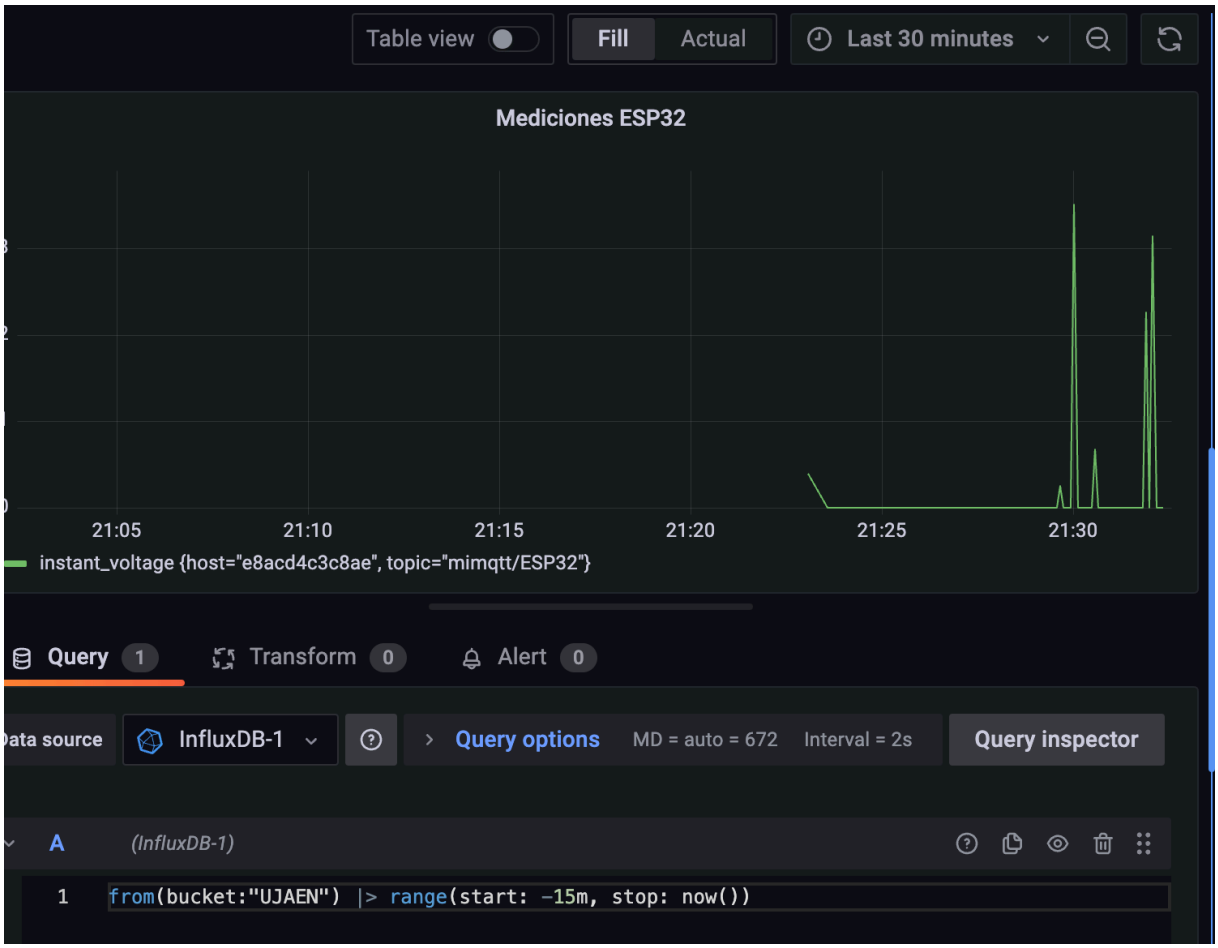

Ilustración 17 Mediciones en tiempo real

4.5.3 Pruebas

En primer lugar, se han realizado las pruebas sobre obtención desde los microcontroladores, alojados temporalmente en el gestor de mensajes.

A continuación, se han realizado las pruebas de consulta y visualización de los datos almacenados en *InfluxDB*.

Pruebas del tercer incremento		Resultado
Tarea 1	Listado de todos los *topics* del gestor de mensajes.	Correcto
Tarea 2	Con el dispositivo de monitorización activo, se realizan lectura y almacenamiento en *InfluxDB* mediante *Telegraf*. Comprobación de la existencia de estos datos.	Correcto
Tarea 3	Consulta externa de *InfluxDB* desde *Grafana.*	Correcto
Tarea 4	Mostrar los datos obtenidos en la Tarea 3 en un panel. Actualización en tiempo real	Correcto

Tabla 16 Pruebas del tercer incremento

4.6 Cuarto incremento

En este último incremento se realiza la integración en el sistema *Odoo ERP*, desde la creación de la aplicación dedicada hasta la generación de informes personalizados para el usuario.

Odoo es de código abierto, aportando un valor único y diferenciador de sus competidores del sector como *SAP Hana[31]* o *SAGE[32].* Esta característica ha hecho posible el crecimiento de una comunidad de desarrollo muy activa.

La versión de pago o *Enterprise* ofrece más de 80 aplicaciones con soporte directo y funcionalidades exclusivas, como la aplicación móvil, únicamente disponible en esta versión. El contrato funciona por suscripción, según los usuarios inscritos en el sistema y las aplicaciones *Enterprise* instaladas. El modo de suscripción fue actualizado con la salida de *Odoo* 16.0 en 2022, liberando todo el conjunto de aplicaciones, quedando incluido en pago por licencia de usuario.

Está disponible para todas las plataformas, incluso en formato contenedor para facilitar el despliegue en la nube. Utiliza *Python* como lenguaje principal, *PostgreSQL[33]* como motor de base de datos y *XML* para la creación de interfaces web e introducción de datos estáticos en el lado del servidor. La arquitectura sigue un patrón *MVC* (Modelo-Vista-Controlador), ampliamente conocido en el desarrollo software.

En su última versión se anunció el ***framework*** *Odoo Web Library* para la creación de componentes de interfaz mediante *JavaScript,* siguiendo las líneas de otros *frameworks* como *React[34]* o *Angular[35]*.

Para facilitar la comprensión de este sistema y su amplitud, se detallarán los aspectos más importantes a tener en cuenta en las diferentes fases del incremento.

4.6.1 Arquitectura

La arquitectura recomendada se resume en la Tabla 17 Arquitectura recomendada para una aplicación Odoo.

Carpeta	Elemento	Descripción
models	modelo.py	Archivo donde se definen los modelos y sus métodos. Los modelos aquí definidos suelen tener persistencia en base de datos.
security	ir.model.access.csv	Archivo de declaración de permisos del sistema. Contempla los permisos de lectura, creación, escritura y borrado del modelo.
views	vista.xml	Archivo de definición de elementos visuales, menús o acciones del usuario que ejecutan métodos de clase.
wizard	modelo.py	Archivo de definición de modelos y métodos. Son modelos transitorios, sin persistencia
	view.xml	Se definen acciones y vistas interactivas con el usuario que facilitan el uso de la aplicación mediante formularios.
data	dato.xml	Archivos para carga de datos. Los archivos habituales contienen datos de modelos o datos de configuración.
i18n	es.po	Archivo para la traducción de términos.
(sin carpeta)	__manifest__.py	Descripción de la aplicación. Este archivo es obligatorio, ya que se definen las dependencias con otras aplicaciones, el orden de creación de objetos en el sistema (permisos, vistas, datos, elementos de menu) y archivos estáticos.

Tabla 17 Arquitectura recomendada para una aplicación Odoo

Odoo utiliza mapeo objeto-relacional, conocido como *ORM* por sus siglas en inglés *Object-Relational Mapping.* Este mecanismo permite abstraer las interacciones con la base de datos, facilita la modificación de los modelos y mejora la seguridad.

Todos los modelos heredan de la clase *model,* la cual tiene unos atributos por defecto como son identificador, usuario creador del registro, fecha de creación, última fecha de modificación y usuario que realizó la última modificación.

4.6.2 Interfaz

El motor de plantillas de *Odoo* se encarga de generar las interfaces y otros elementos como acciones y registros por defecto a partir de las vistas definidas en *XML*, transformándolas en documentos *HTML.*

Las vistas son modificables gracias a la herencia de plantillas y la funcionalidad *XPATH,* (en inglés *XML Path Language),* que permite la alteración de la estructura del documento en toda su extensión.

Por defecto, *Odoo* contempla varias vistas plantilla que facilitan su implementación en la construcción de nuevas interfaces.

La vista “Formulario” se utiliza para la visualización en detalle del registro elegido, para crear un registro nuevo o para editar un registro ya existente. Permite la visualización de aquellos registros múltiples asociados al registro principal.

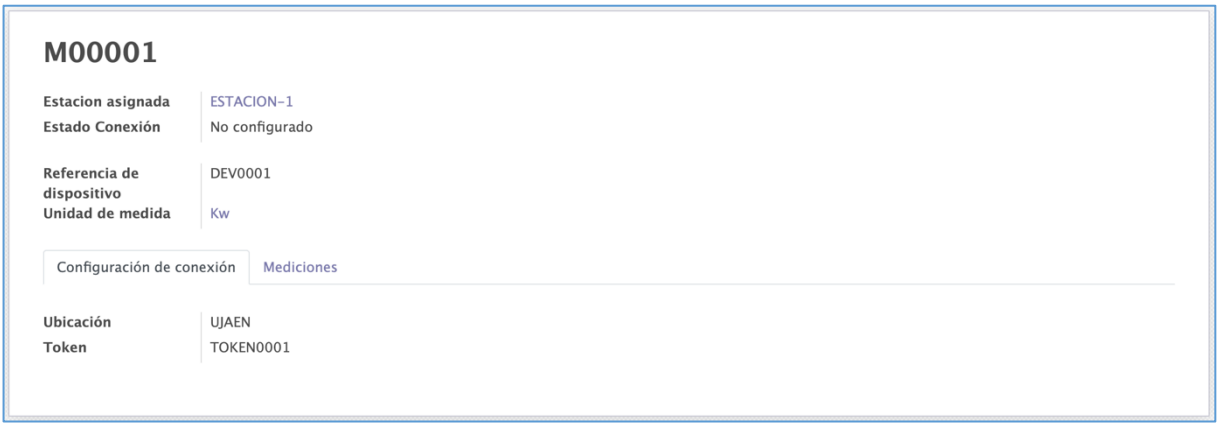

Ilustración 18 Vista Formulario sobre un medidor

La vista “Lista” permite ver un conjunto de registros del modelo elegido organizados en filas. En este tipo de vista se presentan algunos campos en columnas, y permite la agrupación de datos dinámica, operaciones como suma o promedio, filtrado de datos, creación rápida de registros nuevos y edición múltiple de registros.

Ilustración 19 Vista Lista sobre medidores

La vista “Actividad” permite la visualización de datos en una tabla de doble entrada. Esta vista solo está presente en los modelos que tengan un *log* asociado y permitan planificar actividades en el mismo.

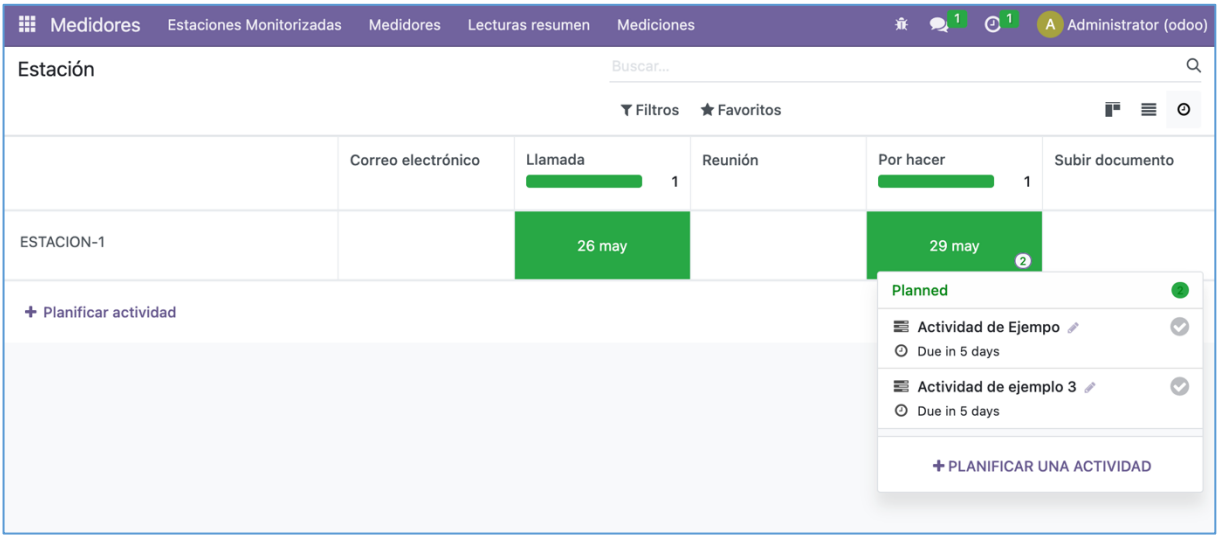

Ilustración 20 Vista Actividad sobre estaciones

La vista “Kanban” permite la visualización de múltiples registros en formato tarjeta agrupados en columnas. Estas columnas se denominan Etapas, y las tarjetas permiten mostrar un pequeño fragmento de información del registro que representan.

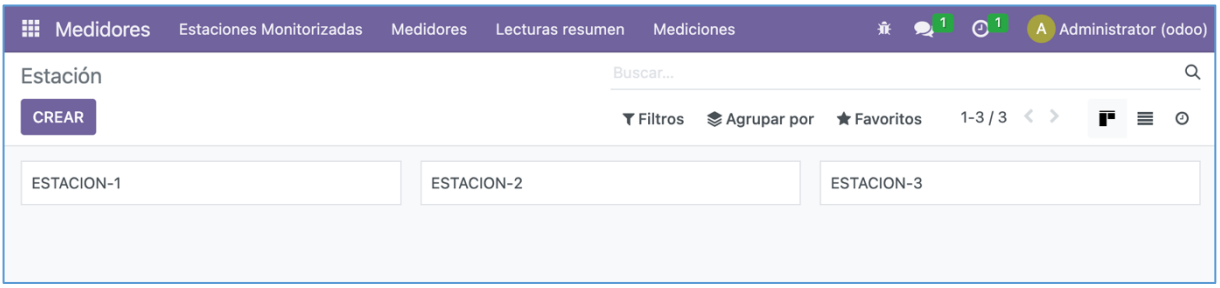

Ilustración 21 Vista Kanban de estaciones

La vista “Pivote” se utiliza para visualizar registros de forma interactiva. Es de gran utilidad para el tratamiento y estudio de datos, permitiendo agrupaciones en varios niveles, filtrado de datos e incluso exportación manteniendo el formato.

	Total					
	noviembre 2022	diciembre 2022			enero 2023	
		Albaranes completados	Albaranes pendientes	Indefinido		
	Cuenta	Cuenta	Cuenta	Cuenta	Cuenta	Cuenta
Total	**2.447**	**1.230**	**126**	**104**	**1.480**	**5.387**
Preparado	60		26	1	105	**192**
Cancelado	190	45	9		35	**279**
En espera	8				12	**20**
Hecho	2.135	1.185	64	103	1.183	**4.670**
Esperando otra operación	54		27		145	**226**

Ilustración 22 Vista pivote sobre albaranes

4.6.3 Seguridad

En el ámbito de la seguridad, *Odoo* gestiona el acceso a modelos mediante una lista *ACL (*en inglés *Access Control List).* Esta lista de acceso define los siguientes campos:

Campos ACL	
id	Identificador *XML* del permiso. Debe ser único en el sistema.
name	Nombre descriptivo del permiso. En ocasiones se define según el grupo.
model_id	Identificador *XML* del modelo sobre el que se aplica el permiso.
group_id	Identificador *XML* del grupo de seguridad que se aplica el permiso.
perm_read	Permiso de lectura sobre el modelo.
perm_write	Permiso de escritura sobre el modelo.
perm_create	Permiso de creación sobre el modelo.
perm_unlink	Permiso de borrado/desenlace. El borrado de un registro es dependiente de las relaciones con otros modelos y cómo se restringe esta acción en la base de datos.

Tabla 18 Campos ACL

Es posible gestionar y modificar estas listas dentro de la propia interfaz del sistema, en el apartado “Permisos de acceso” en los ajustes del modelo.

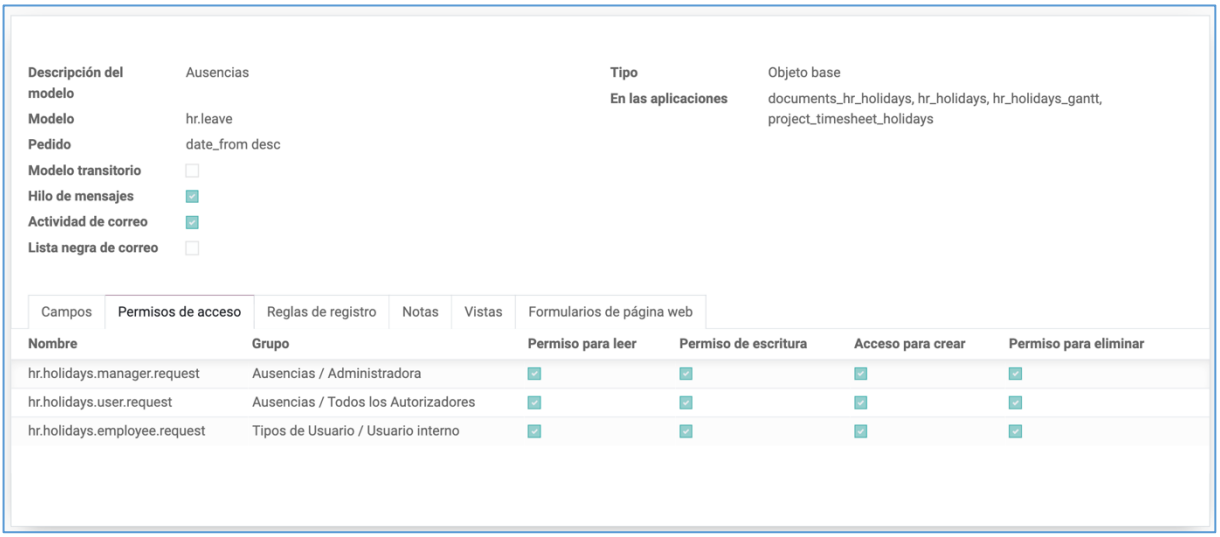

Ilustración 23 Lista de acceso en la interfaz de usuario

4.6.4 Análisis y Diseño

La aplicación desarrollada para este proyecto hace uso de gran parte de la arquitectura que ofrece *Odoo*: creación de nuevos modelos de datos, interfaces, conexiones remotas o funciones del servidor, ya que no existen aplicaciones oficiales que puedan cubrir total o parcialmente las necesidades de negocio.

Se necesita crear un módulo capaz de gestionar la información generada por los elementos de monitorización.

- Debe poder obtener todos los datos de cada medidor por separado de forma autónoma
- Debe poder agrupar los datos para facilitar su comprensión y análisis

El proceso se divide en dos fases diferenciadas:

- **Fase de obtención de datos**: El servidor ejecuta una acción planificada, según el periodo de ejecución configurado en la misma, para obtener nuevas mediciones.

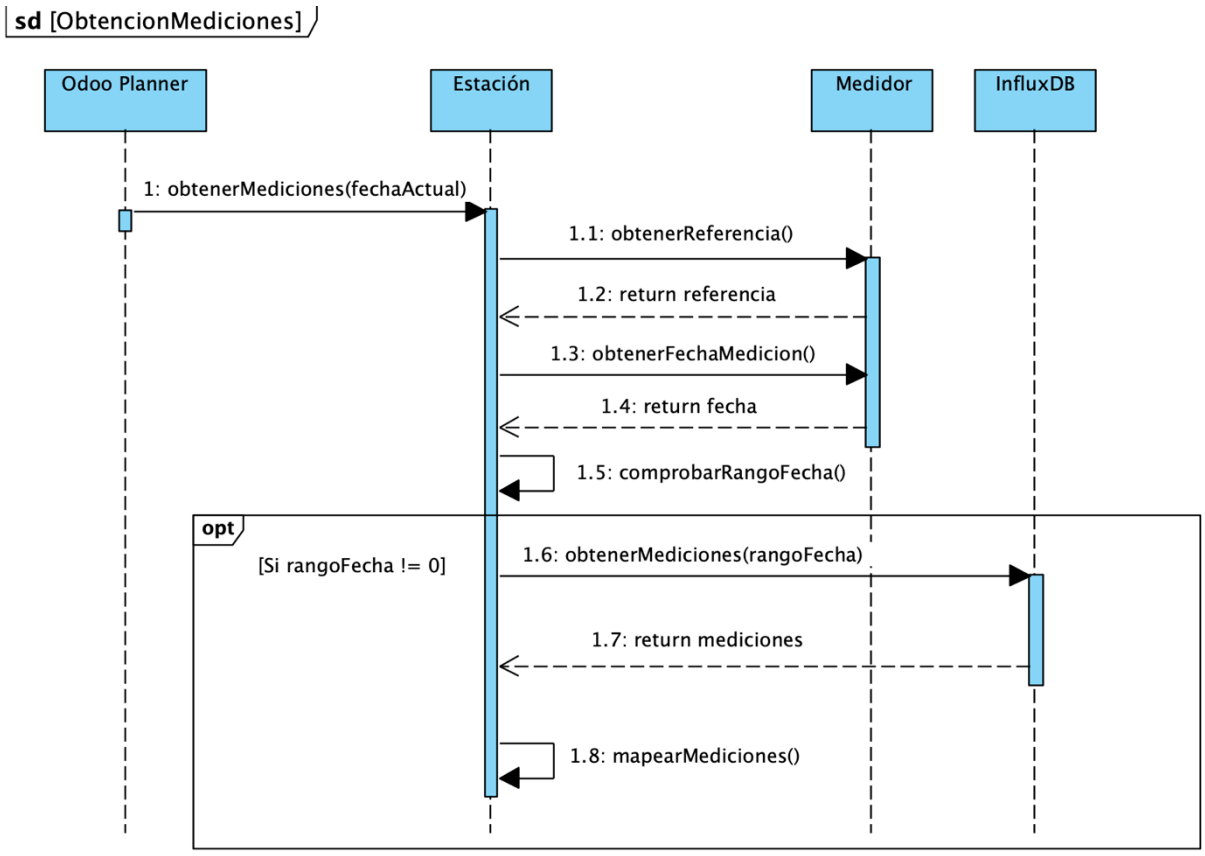

Ilustración 24 Obtención de mediciones en Odoo desde InfluxDB

Se obtiene la última fecha de consulta en cada medidor, y obtiene la última fecha de registro de mediciones para elaborar el rango hasta la fecha actual de ejecución. La dirección de consulta en base de datos se obtiene del identificador del medidor registrado en la estación. Estos datos son los que utilizan para realizar la consulta a la base de datos remota. Una vez se obtiene la respuesta, se formatean las mediciones y se registran para su posterior procesado. Este proceso queda plasmado en la Ilustración 24 Obtención de mediciones en Odoo desde InfluxDB.

- **Fase de resumen de datos**: El servidor ejecuta una acción planificada, según el periodo de ejecución configurado en la misma, que resume las mediciones no procesadas en anteriores ejecuciones. Consulta las mediciones no procesadas hasta la fecha de ejecución de la acción, generado una lectura con el promedio de las mediciones.

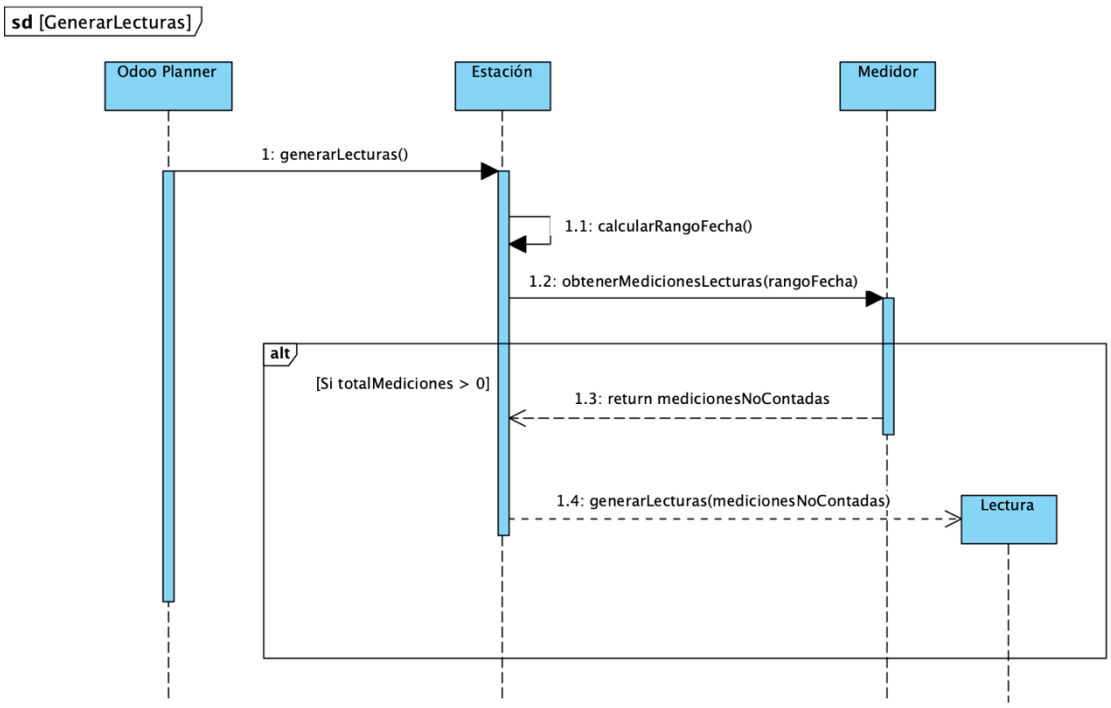

Ilustración 25 Generación de lecturas resumen

A partir de estas dos funciones principales, se detallan las principales entidades y sus relaciones en la Ilustración 26 Diagrama de clases del módulo de monitorización.

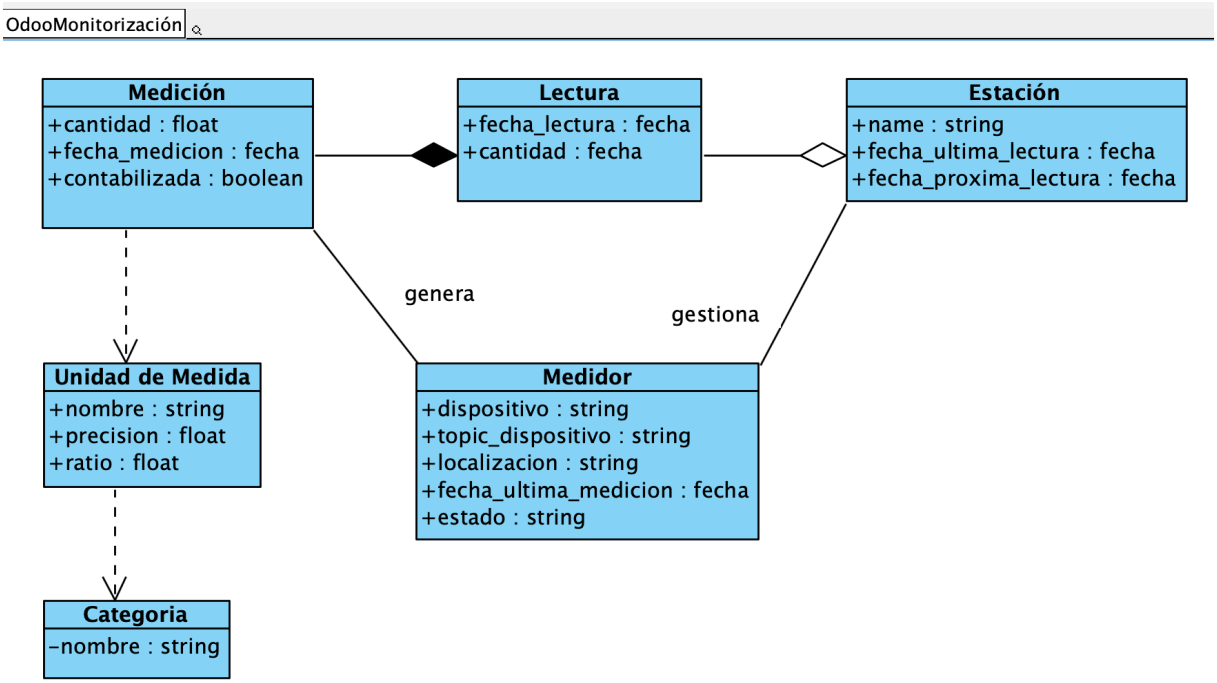

Ilustración 26 Diagrama de clases del módulo de monitorización

4.6.5 Implementación

En primer lugar, se han desarrollado los modelos de datos, atendiendo a la Ilustración 26 Diagrama de clases del módulo de monitorización.

Para poder registrar correctamente los modelos en la base de datos es necesario establecer sus restricciones de seguridad en el archivo de permisos.

Una vez han quedado registrados, se han creado los archivos de datos de ejemplo, que permitirán la realización de pruebas al final de este incremento.

A continuación, se han desarrollado las interfaces de visualización: vistas de lista y vistas de formulario, las cuales permitirán visualizar y manipular ciertos datos. La visibilidad de elementos y capacidad de edición se ha controlado para algunos datos como la secuencia de medidores.

Una vez creados todos los elementos base, se han implementado las dos funcionalidades centrales del módulo: la obtención de datos y la capacidad de tratamiento de los mismos. Estas funciones quedan registradas como acciones planificadas que el servidor ejecutará de forma periódica.

Por último, se ha llevado a cabo la traducción del módulo a partir del archivo generado por el propio sistema mediante la herramienta de importación y exportación nativa.

4.6.6 Pruebas

Partiendo de los datos generados en las pruebas realizadas en los anteriores incrementos, se realizan las siguientes pruebas:

Pruebas del cuarto incremento		Resultado
Tarea 1	Instalación de la librería *influxdb-client* a partir de la especificación de la imagen de *Odoo* en su archivo *Dockerfile*. Mediante consola de comandos, se accede al contenedor de *Odoo*, se ejecuta una *Python Shell* interactiva. Se realiza la importación de *influxdb-client.*	Correcto
Tarea 2	Conexión mediante *influxdb-client* a la base de datos *InfluxDB*. Utilizando el cliente de la librería, obtener un listado de *topics* o alguna información como *buckets* u organización del usuario utilizado.	Correcto
Tarea 3	Lectura de mediciones mediante *influxdb-client* desde *Odoo*, síntesis de datos.	Correcto
Tarea 4	Creación de un registro en el modelo destinado a tal efecto.	Correcto
Tarea 5	Ejecución planificada de la obtención de datos en un periodo de tiempo predeterminado.	Correcto
Tarea 6	Realización de varias ejecuciones de resumen de datos sobre lecturas.	Correcto

Tabla 19 Pruebas del cuarto incremento

5 RESULTADOS Y DISCUSIÓN

5.1 Conclusiones y líneas futuras de actuación

Este proyecto plantea una solución factible de bajo costo, con capacidad de ser integrada en sistemas de diversa índole de forma satisfactoria.

La gran cantidad de tecnologías utilizadas en este prototipo puede dificultar la escalabilidad de la solución para un uso real, ya que requeriría de mayor conocimiento experto en cada uno de los aspectos implicados.

Como líneas futuras de actuación se pueden plantear los siguientes aspectos:

- Estandarización en el desarrollo de aplicaciones para los microcontroladores.
- Funcionalidades en *Odoo* que cubran necesidades específicas de la empresa que implanta el sistema.
- Desarrollo de una plataforma de gestión privada para evitar la dependencia con empresas externas.
- Creación de una plataforma altamente escalable mediante tecnologías de computación en la nube.
- Integración con sistemas de alerta y gestión de incidencias.
- Desarrollo de aplicaciones con capacidades de comunicación inteligente entre microcontroladores.

BIBLIOGRAFÍA

[1] J. F. Bolivar, *Infraestructuras críticas y sistemas industriales: Auditorías de seguridad y fortificación*.
[2] Neto, Walter L., «Wireless Protocols in Device Communication in the Industrial Internet of Things: Systematic Review», vol. 12957 LNCS, sep. 2021, doi: 10.1007/978-3-030-87013-3_28.
[3] «ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems». https://www.espressif.com/en/products/socs/esp32 (accedido 6 de marzo de 2022).
[4] «Toit - IoT software platform for the ESP32». https://toit.io/ (accedido 21 de marzo de 2022).
[5] «Jaguar: Live reloading for your ESP32». Toit language. Accedido: 25 de marzo de 2022. [En línea]. Disponible en: https://github.com/toitlang/jaguar
[6] «Flux - Influx Query Language». https://docs.influxdata.com/flux/v0.x/ (accedido 10 de abril de 2023).
[7] «InfluxDB», *InfluxData*. https://www.influxdata.com/home/ (accedido 23 de abril de 2023).
[8] «Open Source ERP and CRM | Odoo», *Odoo S.A.* https://www.odoo.com/es_ES (accedido 7 de mayo de 2023).
[9] «Toit MQTT». https://docs.toit.io/tutorials/mqtt (accedido 15 de abril de 2023).
[10] «MQTT - The Standard for IoT Messaging». https://mqtt.org/ (accedido 11 de marzo de 2023).
[11] «Toit GPIO». https://docs.toit.io/peripherals/gpio (accedido 15 de abril de 2023).
[12] «influxdb-client-python». InfluxData. Accedido: 2 de junio de 2023. [En línea]. Disponible en: https://github.com/influxdata/influxdb-client-python
[13] «Docker for Desktop». https://www.docker.com/products/docker-desktop/ (accedido 2 de septiembre de 2022).
[14] «Microsoft Word». https://www.microsoft.com/es-ww/microsoft-365/word (accedido 20 de marzo de 2023).
[15] «VIsual Paradigm». https://www.visual-paradigm.com/ (accedido 4 de abril de 2023).
[16] «Visual Studio Code». https://code.visualstudio.com/ (accedido 14 de septiembre de 2022).
[17] «Scopus preview - Scopus - Welcome to Scopus». https://www.scopus.com/ (accedido 1 de marzo de 2022).
[18] «ClickUp», *clickup.com*. https://app.clickup.com/20652524/home (accedido 2 de mayo de 2023).
[19] A. López Gil, «Estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software», info:eu-repo/semantics/bachelorThesis, 2018. Accedido: 15 de marzo de 2023. [En línea]. Disponible en: http://uvadoc.uva.es/handle/10324/32875
[20] «Google». https://www.google.es/ (accedido 26 de junio de 2023).
[21] «GitHub», *GitHub*. https://github.com/ (accedido 26 de junio de 2023).
[22] «CP210x USB to UART Bridge VCP Drivers - Silicon Labs». https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers (accedido 30 de marzo de 2022).

[23] «Pub/Sub para la integración de aplicaciones y datos | Cloud Pub/Sub», *Google Cloud*. https://cloud.google.com/pubsub?hl=es-419 (accedido 31 de marzo de 2022).
[24] «Standard libraries - Toit». https://libs.toit.io/pubsub/library-summary (accedido 3 de abril de 2022).
[25] «Google Cloud», *Google Cloud*. https://cloud.google.com/?hl=es (accedido 26 de junio de 2023).
[26] «AWS | Cloud Computing - Servicios de informática en la nube». https://aws.amazon.com/es/ (accedido 26 de junio de 2023).
[27] «Microsoft Azure». https://azure.microsoft.com/es-es (accedido 26 de junio de 2023).
[28] «Eclipse Mosquitto», *Eclipse Mosquitto*, 8 de enero de 2018. https://mosquitto.org/ (accedido 26 de junio de 2023).
[29] «Quality of Service | Cedalo Platform». https://docs.cedalo.com/mosquitto/quality-of-service (accedido 21 de febrero de 2023).
[30] «Docker Compose», *Docker Documentation*, 23 de junio de 2023. https://docs.docker.com/compose/ (accedido 26 de junio de 2023).
[31] «SAP Hana», *SAP*. https://www.sap.com/spain/products/technology-platform/hana.html (accedido 26 de junio de 2023).
[32] «Software de gestión empresarial | Sage». https://www.sage.com/es-es/ (accedido 26 de junio de 2023).
[33] «PostgreSQL», *PostgreSQL*, 26 de junio de 2023. https://www.postgresql.org/ (accedido 26 de junio de 2023).
[34] «React». https://es.react.dev/ (accedido 26 de junio de 2023).
[35] «Angular». https://angular.io/ (accedido 26 de junio de 2023).
[36] «Developer — Odoo 16.0 documentation». https://www.odoo.com/documentation/16.0/developer.html (accedido 4 de mayo de 2023).

APÉNDICES

Apéndice A: Diagrama WBS

El diagrama WBS completo muestra todas las tareas que componen la realización del proyecto.

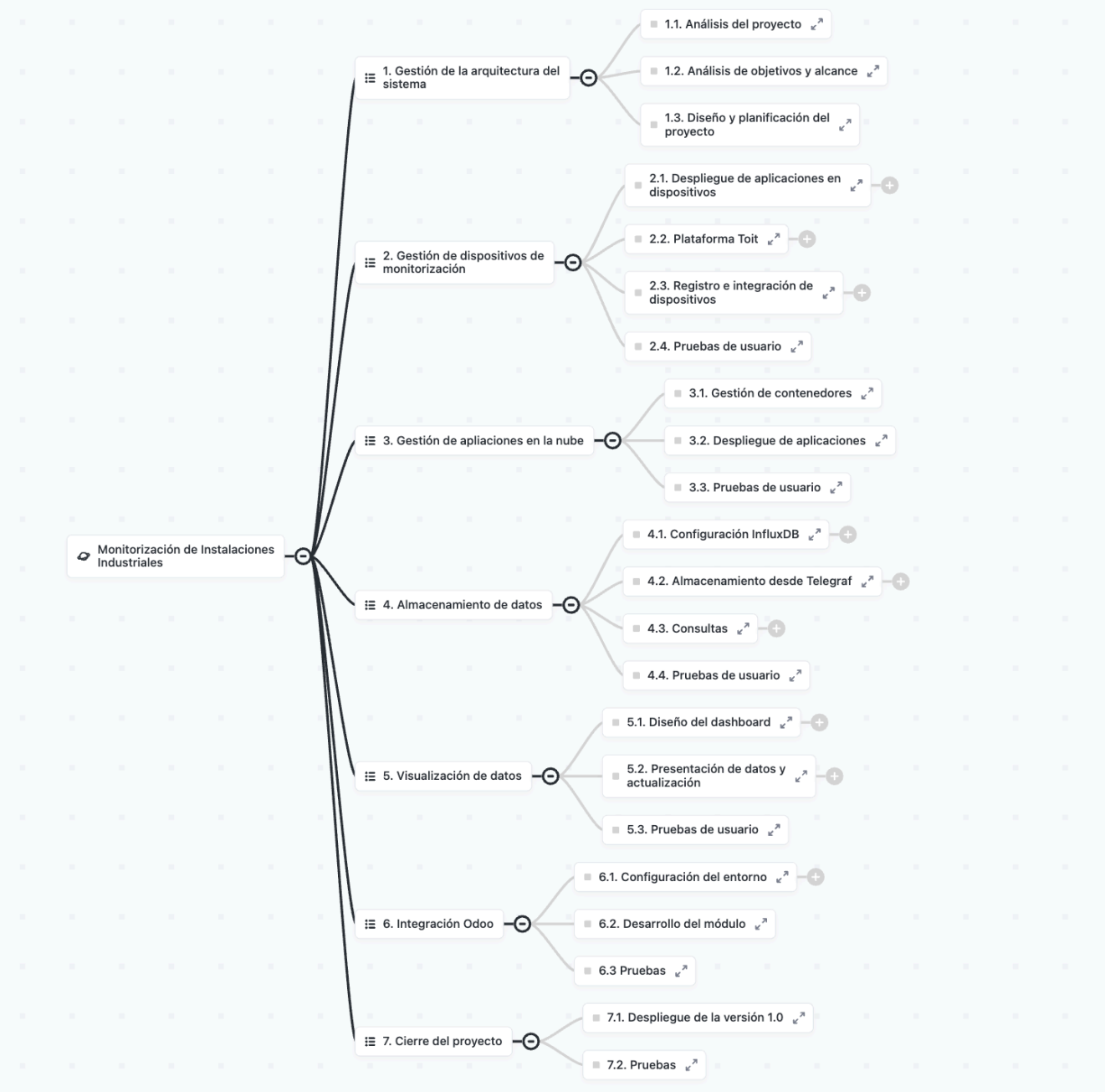

Ilustración 27 Diagrama WBS completo

Apéndice B: Debilidades y Requisitos del sistema

B.1 Debilidades y carencias

DS1	Obtención de medidas físicas
Versión	1.0
Impacto	Crítico
Descripción	El sistema obtiene mediciones físicas, mediante sensores. Estas medidas son procesadas como señales físicas por otros elementos del sistema.
Comentarios	Las medidas físicas no son almacenables y dificultan la adaptabilidad.

Tabla 20 Obtención de medidas físicas

DS2	Transmisión de datos
Versión	1.0
Impacto	Crítico
Descripción	Los elementos medidores del sistema transmiten los datos mediante protocolos propietarios
Comentarios	Genera un alto grado de dependencia de la tecnología utilizada. Estrecha el abanico de tecnologías y soluciones a utilizar para resolver problemas.

Tabla 21 Transmisión de datos

DS3	Almacenamiento local de datos
Versión	1.0
Impacto	Alto
Descripción	El sistema almacena el histórico de datos de forma local exclusivamente.
Comentarios	Puede provocar pérdida total de los datos, ya que no cumple con las recomendaciones generales de copias de seguridad.

Tabla 22 Almacenamiento local de datos

DS4	Interacción física con el sistema
Versión	1.0
Impacto	Alto
Descripción	El sistema trabaja mediante actuadores físicos, por lo que requiere interacción de un actor externo.
Comentarios	Los actuadores no están supervisados por elementos inteligentes que limiten o que faciliten la actuación externa.

Tabla 23 Interacción física con el sistema

DS5	Visualización de datos
Versión	1.0
Impacto	Medio
Descripción	El sistema presenta los datos en tiempo mediante interfaces en elementos empotrados en la propia instalación, o bien en aplicaciones de visualización con interfaces estáticas o poco intuitivas.
Comentarios	El software es privado, no se puede personalizar.

Tabla 24 Visualización de datos

DS6	Sistema monolítico
Versión	1.0
Impacto	Alto
Descripción	El sistema presenta una estructura monolítica, por lo que todas las opciones de configuración quedan limitadas a las ofrecidas por el sistema implantado.
Comentarios	El software presenta carencias que no pueden ser resueltas con otras aplicaciones de manera rentable.

Tabla 25 Sistema monolítico

DS7	Accesibilidad de la información
Versión	1.0
Impacto	Medio
Descripción	El sistema no presenta interfaces de comunicación estándar reconocidas en el mercado.
Comentarios	La comunicación existente utiliza protocolos privados. Se requiere de elementos hardware adicionales, con el consecuente incremento de coste y, en ocasiones, una disminución de las prestaciones.

Tabla 26 Accesibilidad de la información

B.2: Requisitos generales del sistema

RG1	Tratamiento de la información
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema debe realizar las tareas de procesado, almacenamiento y exposición de los datos obtenidos mediante monitorización en una ubicación remota
Requisitos hijos	- RG2: Información accesible - RG7: Integración en niveles superiores

Tabla 27 Tratamiento de la información

RG2	Información accesible
Versión	1.0
Dependencias	- RG1: Tratamiento de la información
Descripción	El sistema debe presentar la información de forma centralizada, para facilitar el acceso a agentes externos de procesado de datos, visualización o actuación.
Requisitos hijos	- RG7: Integración en niveles superiores

Tabla 28 Información accesible

RG3	Compatibilidad externa
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema debe ser compatible con diversas tecnologías de conectividad, para maximizar la flexibilidad y adaptabilidad.
Requisitos hijos	- RG5: Registro de nuevos dispositivos

Tabla 29 Compatibilidad externa

RG4	Sistema modular
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema debe ser modular, permitiendo la escalabilidad y adaptándose a los cambios en las necesidades del mismo.
Requisitos hijos	- RG6: Portabilidad del sistema

Tabla 30 Sistema modular

RG5	Registro de nuevos dispositivos
Versión	1.0
Dependencias	- RG3: Compatibilidad externa
Descripción	El sistema debe facilitar a los operarios la integración y puesta en marcha de nuevos elementos hardware-software.
Requisitos hijos	Ninguno.

Tabla 31 Registro de nuevos dispositivos

RG6	Portabilidad del sistema
Versión	1.0
Dependencias	- RG4: Sistema modular
Descripción	El sistema debe poder implantarse en múltiples plataformas sin necesidad de aplicar adaptaciones de su estructura.
Requisitos hijos	Ninguno.

Tabla 32 Portabilidad del sistema

RG7	Integración en niveles superiores
Versión	1.0
Dependencias	- RG2: Información accesible
Descripción	El sistema debe ofrecer mecanismos de integración de sus funcionalidades en sistemas de mayor nivel como CMS o ERP.
Requisitos hijos	- RG9: Sistema inteligente

Tabla 33 Integración en niveles superiores

RG8	Estabilidad del desarrollo
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema debería utilizar librerías y aplicaciones en versiones estables de su desarrollo, evitando siempre que sea posible aquellas que estén en desuso u obsoletas.
Requisitos hijos	- RG10: Seguridad del sistema

Tabla 34 Estabilidad del desarrollo

RG9	Sistema inteligente
Versión	1.0
Dependencias	- RG7: Integración en niveles superiores
Descripción	El sistema debería advertir de anomalías en las mediciones, anticiparse a posibles fallos, aplicar acciones preventivas y extraer datos relevantes para decisiones futuras.
Requisitos hijos	Ninguno.

Tabla 35 Sistema inteligente

RG10	Seguridad del sistema
Versión	1.0
Dependencias	- RG8: Estabilidad del desarrollo
Descripción	El sistema debería disponer de elementos y medidas de seguridad en toda su extensión.

Tabla 36 Seguridad del sistema

B.3: Requisitos funcionales del sistema

RF1	Gestión remota de los dispositivos
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema permitirá agregar nuevos dispositivos a monitorizar mediante una provisión software mínima. Permitirá modificar el *firmware* para agregar nuevas funcionalidades, apagar o encender los dispositivos programáticamente y llevar un control de los dispositivos en activo.
Importancia	Alta

Tabla 37 Gestión remota de dispositivos

RF2	Administración de datos almacenados
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema dispondrá de herramientas de administración para las distintas tipologías de datos generados y las ubicaciones de almacenamiento de los mismos.
Datos específicos	• Datos directos como mediciones o indicadores de estado • Datos de obtención indirecta, como informes periódicos o patrones de comportamiento generados por el sistema • Datos de configuración e implantación, como direcciones de red, nombres de dominio o procesos de automatización e integración continua • Secretos, como claves de administración, parejas de claves o claves API • Seguridad, como listados de usuarios o permisos de acceso
Importancia	Media

Tabla 38 Administración de datos

RF3	Gestión de datos en la nube
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema procesará los datos en la nube. El sistema tendrá una copia de datos existentes en la nube.
Importancia	Alta
Comentarios	Esta copia de los datos forma parte de una correcta política de copias de seguridad, deslocalizando una de las copias a realizar.

Tabla 39 Gestión de datos en la nube

RF4	Visualización de datos
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema presentará los datos de forma clara y entendible en un *dashboard* adaptable.
Importancia	Media
Comentarios	Las interfaces de presentación de datos deberán ser coherentes en toda la extensión del sistema. La información debe ser presentada acorde al nivel de abstracción, siendo más técnica en niveles inferiores y más resumida en los niveles más altos.

Tabla 40 Visualización de datos

RF5	Generación de informes
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema generará informes entendibles por el usuario sobre los datos generados por la monitorización de la instalación. Estos informes se generarán como documento imprimible.
Importancia	Media
Comentarios	Los informes serán generados en los niveles más altos del sistema, de forma atractiva a un usuario final con perfil no técnico.

Tabla 41 Generación de informes

B.4: Requisitos no funcionales del sistema

RNF1	Registro de actividad
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema mantendrá un registro de actividad durante la ejecución de sus componentes en formato *log de eventos*.
Comentarios	No siendo prioritario su almacenamiento, esta funcionalidad estará disponible en caso de necesidad.

Tabla 42 Registro de actividad

RNF2	Independencia de componentes
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema se compondrá de módulos interconectados, con una cierta autonomía, para mitigar los fallos de comunicación. El sistema administrará estos componentes de forma automatizada.

Tabla 43 Independencia de componentes

RNF3	Copias de seguridad
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema dispondrá de mecanismos de recuperación ante desastres mediante copias de seguridad de datos.
Comentarios	La capacidad de respuesta del sistema estará condicionada por la infraestructura elegida en entornos de producción.

Tabla 44 Copias de seguridad

RNF4	Control de acceso
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema establecerá elementos de seguridad en todos sus componentes.
Comentarios	El acceso a un componente concreto no debe posibilitar el acceso total al sistema.

Tabla 45 Control de acceso

RNF5	Implantación del sistema
Versión	1.0
Dependencias	Ninguna.
Descripción	El entorno en producción del sistema deberá ser en la nube. El despliegue de componentes en entornos locales deberá estar justificado por la necesidad concreta de la implantación.
Comentarios	La deslocalización del sistema mejora su estabilidad, por lo que se recomienda el uso de nubes públicas o híbridas.

Tabla 46 Implantación del sistema

B.5: Restricciones técnicas

RT1	Tecnología de contenedores
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema se basará en contenedores, desplegando sus componentes en plataformas destinadas a ello.
Comentarios	El uso de herramientas de orquestación de contenedores permite minimizar el tiempo de inactividad.

Tabla 47 Tecnología de contenedores

RT2	Integración continua
Versión	1.0
Dependencias	Ninguna.
Descripción	Se utilizarán procesos de *CI/CD* en el ciclo de desarrollo de los componentes del sistema, reduciendo el riesgo e impacto de fallos en las actualizaciones.

Tabla 48 Integración continua

RT3	Minimización de tiempos
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema minimizará los tiempos entre obtención y presentación de datos en las interfaces de usuario.

Tabla 49 Minimización de tiempos

B.6: Requisitos de integración

RI1	Interfaces de comunicación software
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema seguirá las recomendaciones para exposición de *API* dedicadas al envío de datos a otros sistemas de mayor nivel (*CMS, ERP*).
Comentarios	La integración vertical es clave para el usuario final.

Tabla 50 Interfaces de comunicación software

RI2	Interfaces de comunicación de la instalación
Versión	1.0
Dependencias	Ninguna.
Descripción	El sistema dispondrá de adaptadores de datos para la integración de componentes de monitorización *IT/OT* disponibles en la instalación y compatibles con los protocolos de comunicación del nuevo sistema.

Tabla 51 Interfaces de comunicación de la instalación

Apéndice C: Configuración Inicial de InfluxDB

Este anexo detalla el proceso de configuración inicial de la base de datos. Para poder llevar a cabo los pasos que se detallan, deben cumplirse una serie de requisitos previos:

- Tener *Docker* instalado en el sistema con permisos suficientes de ejecución.
- Tener configurado el archivo *docker-compose.yaml* correctamente para disponer de acceso a la subred.
- Tener libres los puertos de la máquina host que vayan a ser usados por la aplicación.
- Tener el contenedor previamente inicializado y estado activo

Para la configuración inicial de *InfluxDB* seguiremos los siguientes pasos:

Creación del usuario inicial. Deberemos acceder al portal web de administración de la base de datos. La primera vez que se accede se nos mostrará la Ilustración 28 Página de primer registro de InfluxDB.

En este formulario podemos crear una Organización y el primer almacén de datos o *Bucket*.

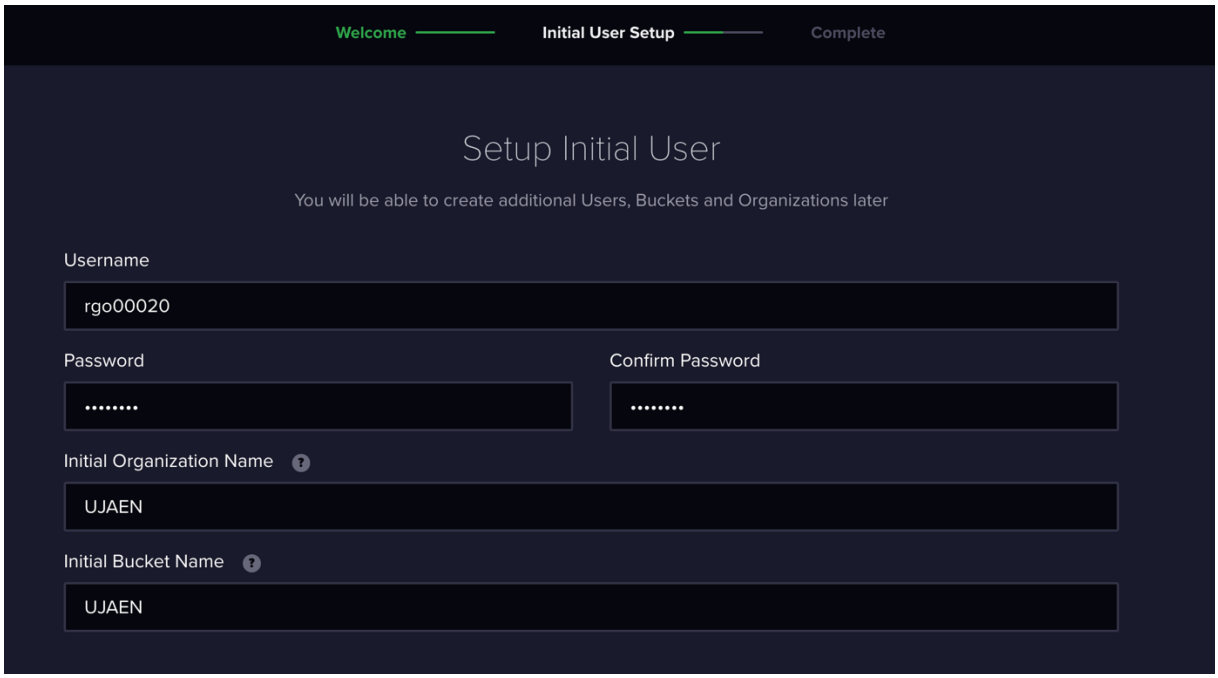

Ilustración 28 Página de primer registro de InfluxDB

Una vez creado el primer usuario, *InfluxDB* nos muestra un tutorial para facilitar la configuración. En la primera opción mostrada en Ilustración 29 Tutorial guiado de InfluxDB, se nos muestran los distintos orígenes de datos aceptados.

Seleccionaremos *TELEGRAF*, y crearemos una configuración.

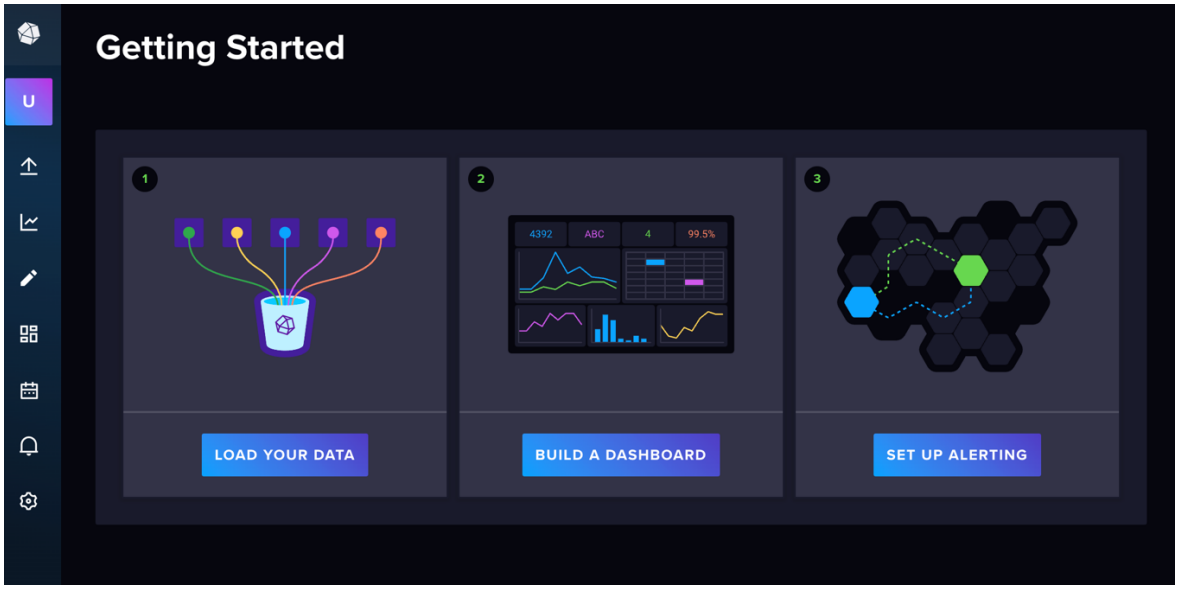

Ilustración 29 Tutorial guiado de InfluxDB

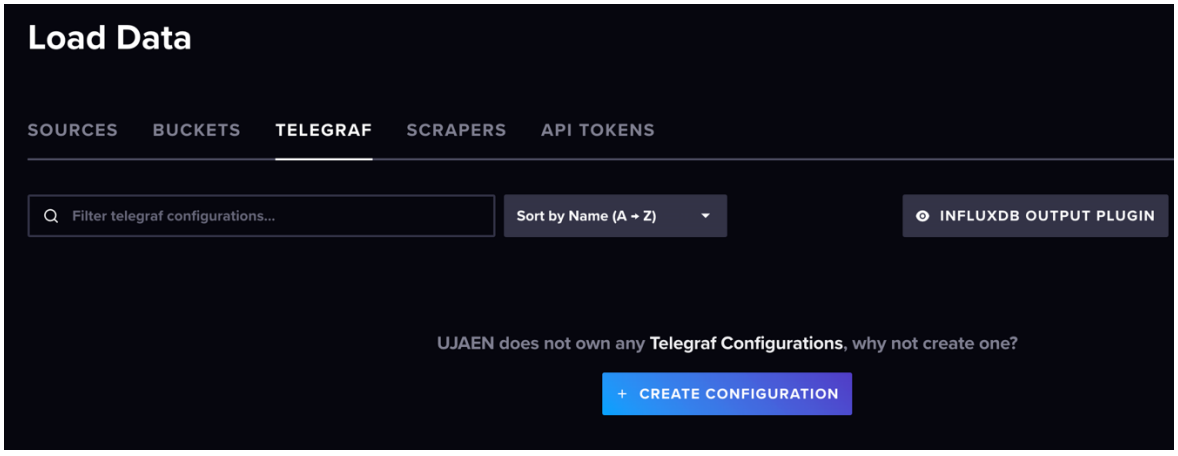

Ilustración 30 Opciones de configuración de orígenes de datos

En el asistente de configuración, elegiremos el *bucket* donde van a ser almacenados los datos obtenidos. Ayudándonos del buscador, seleccionaremos *MQTT Consumer*.

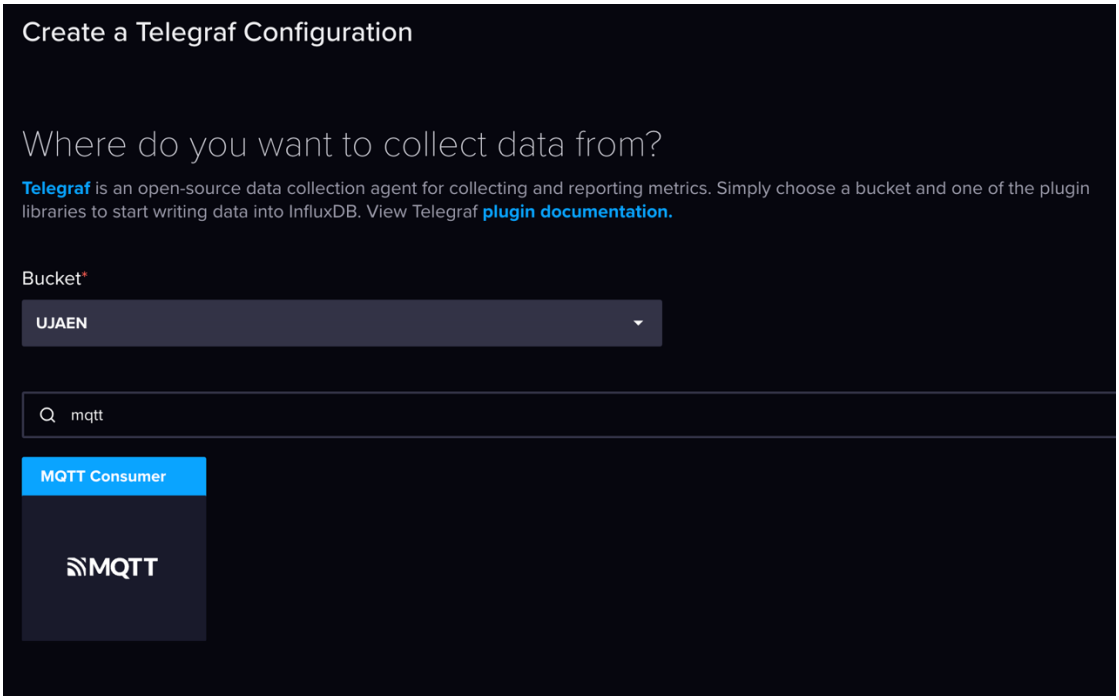

Ilustración 31 Asistente de configuración de InfluxDB – Telegraf

Una vez elegido, *InfluxDB* nos mostrará un cuadro de diálogo donde se nos mostrará un *token* de acceso al mismo como se observa en la Ilustración 32 Generación del token de acceso a InfluxDB. Este *token* es necesario para que *Telegraf* pueda conectarse a *InfluxDB*.

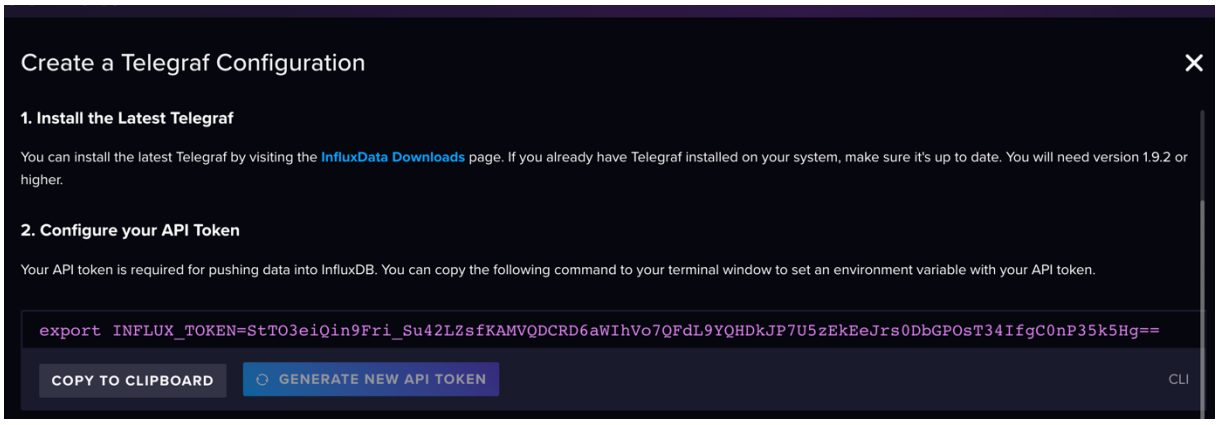

Ilustración 32 Generación del token de acceso a InfluxDB

Podemos administrar los *tokens* generados en el menú *API TOKENS.*

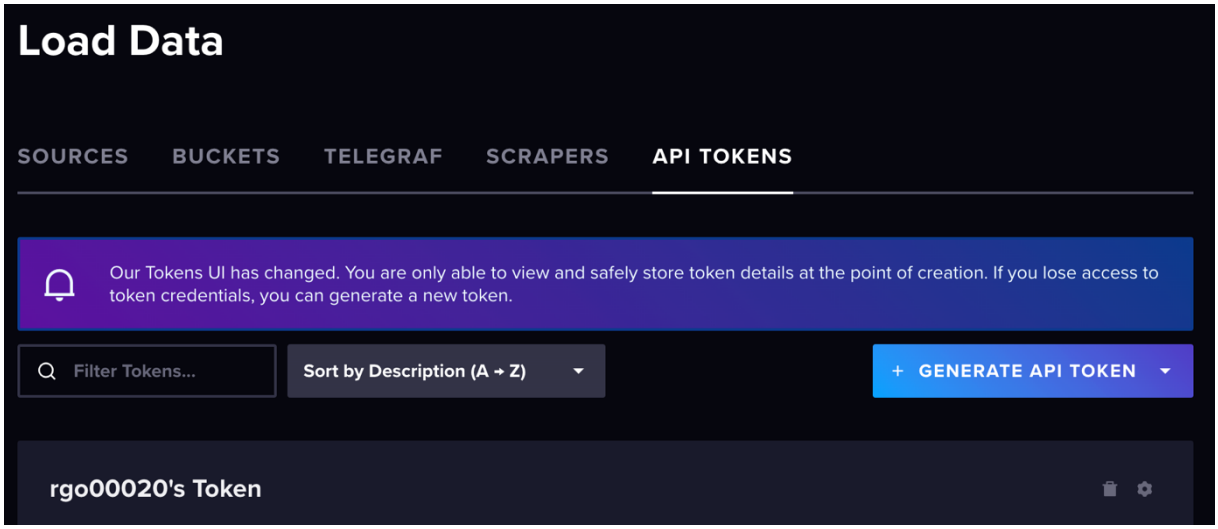

Ilustración 33 Listado de tokens generados

Cada elemento de este listado puede ser administrado individualmente, de forma que se pueden activar o desactivar los permisos de lectura y escritura en los diferentes elementos a los que se otorga acceso.

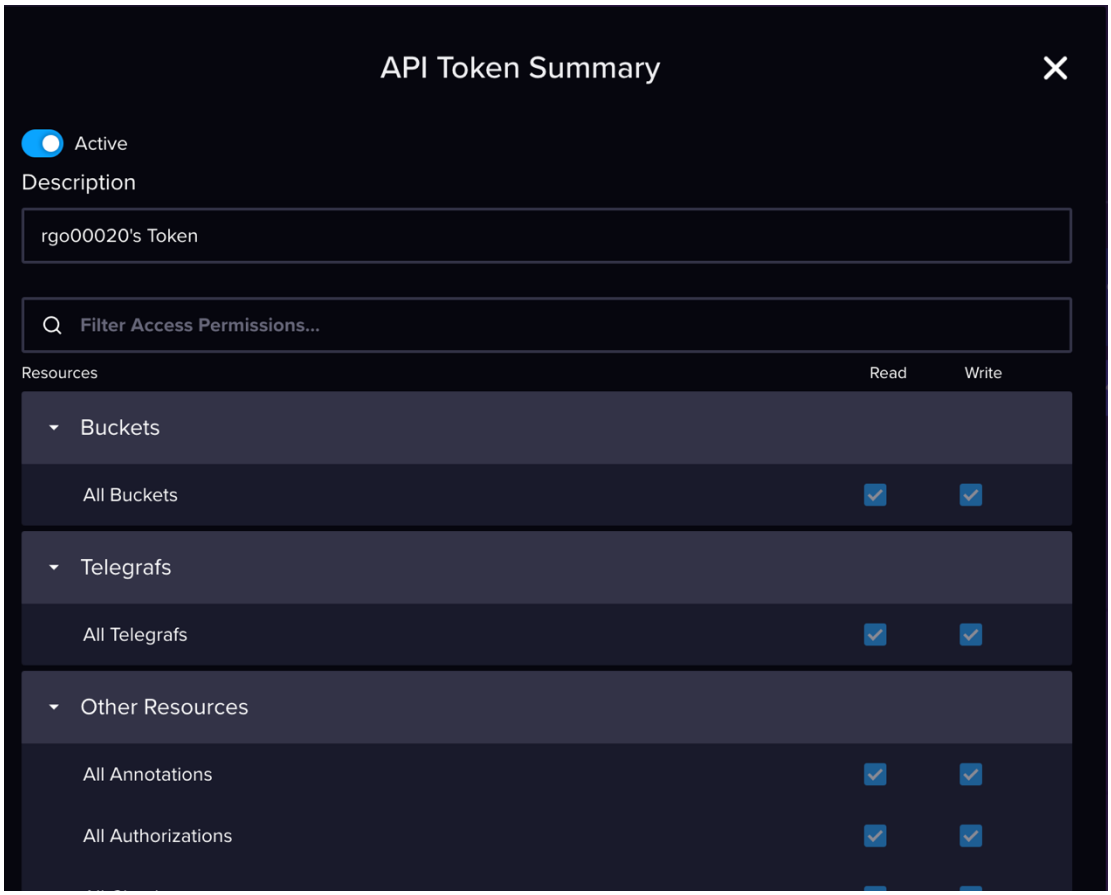

Ilustración 34 Control de permisos de lectura y escritura de los tokens

Por razones de seguridad, en las últimas versiones de *InfluxDB* no es posible consultar el valor de un *token*. El *token* muestra su valor en el momento de la generación. En caso pérdida, será necesario el borrado y la generación de un nuevo *token*.

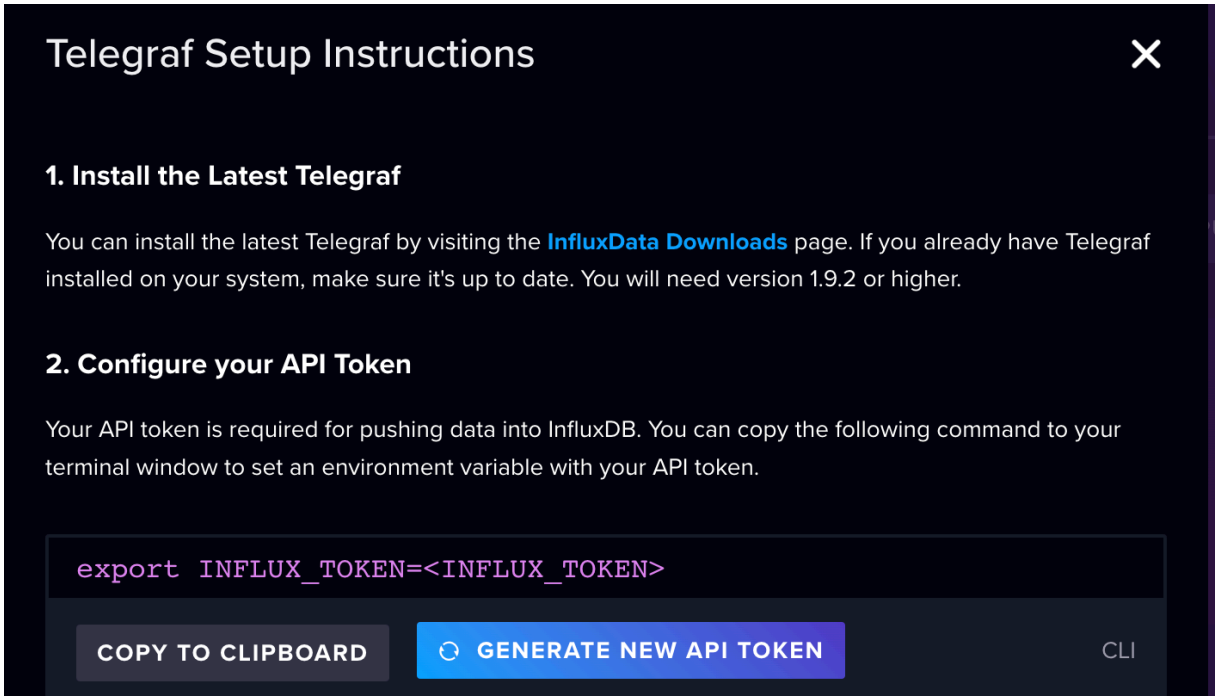

Ilustración 35 Generación de un nuevo token de acceso

Apéndice E: Configuración de orígenes de datos en Grafana

Este anexo detalla el proceso de configuración de orígenes de datos. Para poder llevar a cabo los pasos que se detallan, deben cumplirse una serie de requisitos previos:

- Tener el contenedor de la base de datos previamente inicializado y en estado activo

Para configurar *InfluxDB* como origen de datos seguiremos estos pasos:

Deberemos tener iniciada la sesión en el portal. Accederemos al apartado "Configuración" y a continuación "Orígenes de datos".

Aquí crearemos un nuevo origen de datos, buscando *InfluxDB* en el listado de aplicaciones.

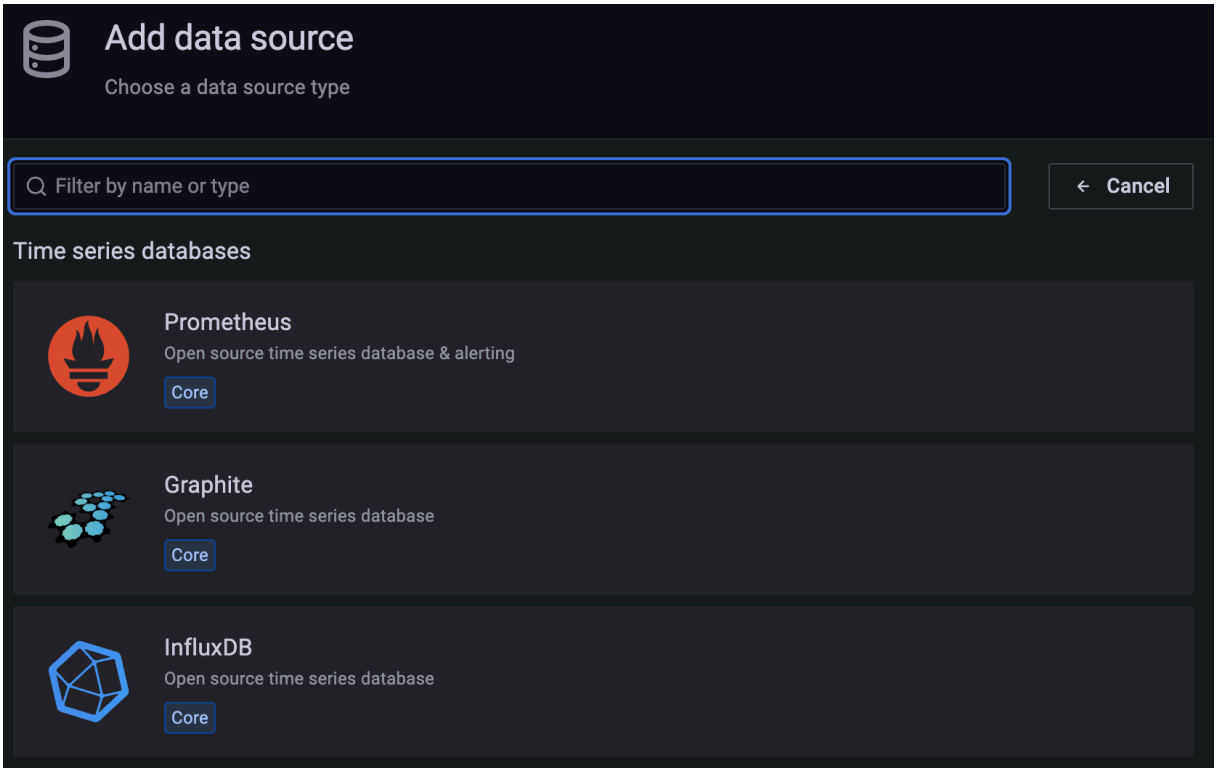

Ilustración 36 Listado de orígenes de datos

Una vez seleccionado el origen de datos, la Ilustración 37 Configuración de conexión a InfluxDB muestra las opciones concretas del mismo.

En esta configuración seleccionaremos *Flux* como lenguaje de consulta, ya que aporta seguridad al estar basado en clave previamente compartida.

Se ha de utilizar una clave generada en *InfluxDB* en el apartado mostrado en la Ilustración 38 Configuración de autenticación en InfluxDB, pudiendo ser la misma utilizada en *Telegraf,* o una exclusiva para el servicio, siguiendo las prácticas recomendadas.

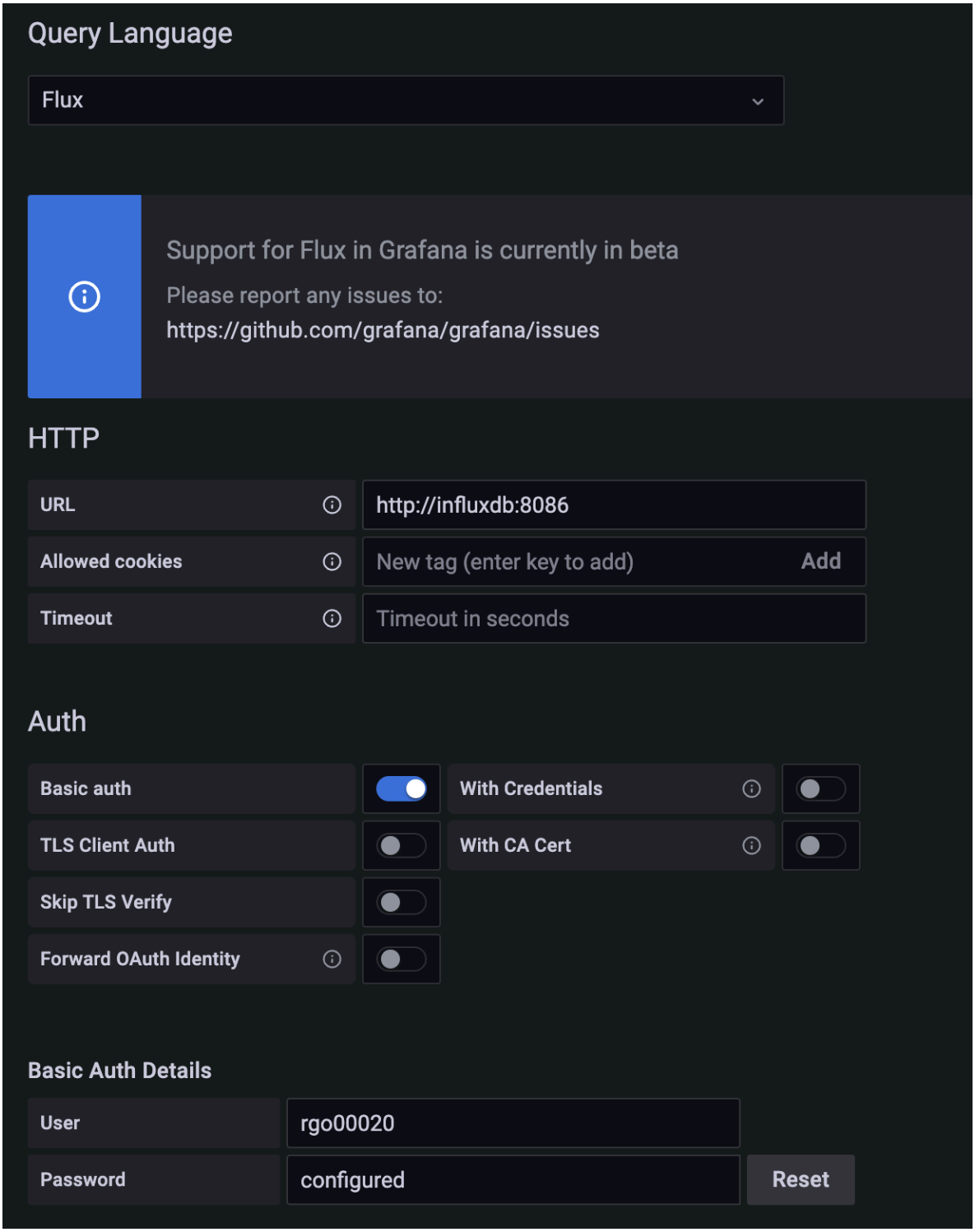

Ilustración 37 Configuración de conexión a InfluxDB

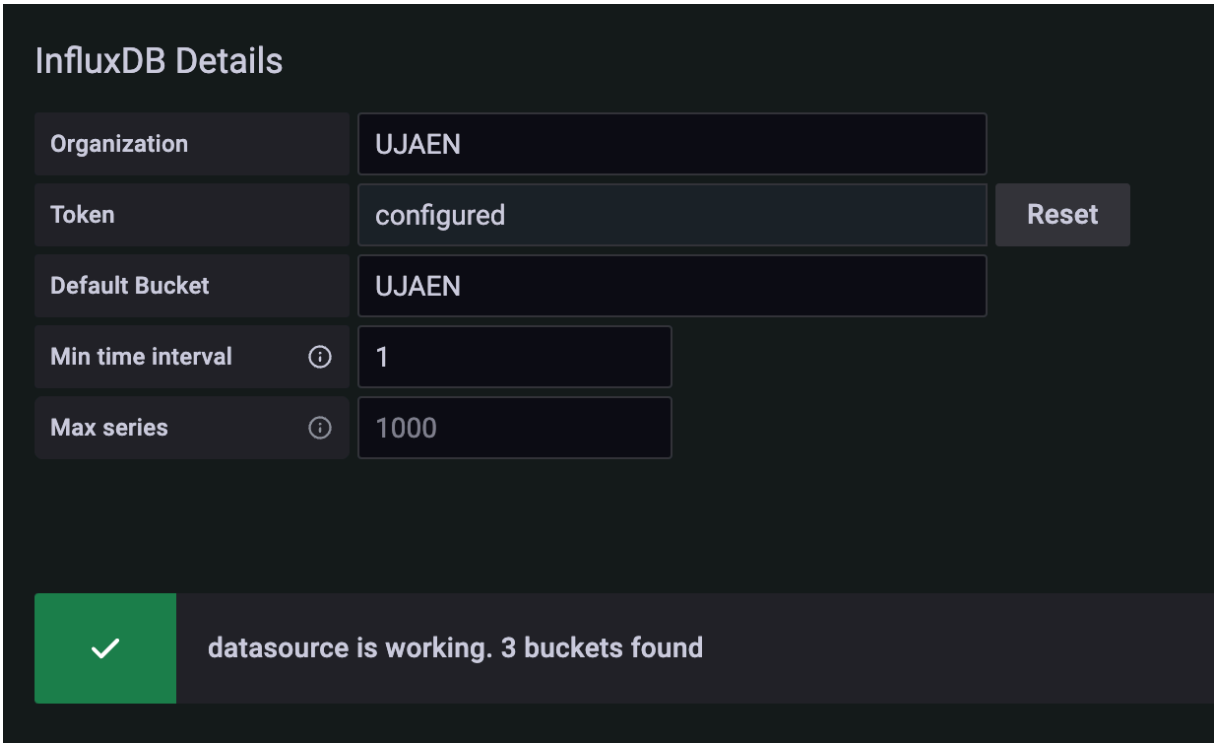

Ilustración 38 Configuración de autenticación en InfluxDB

DEFINICIONES Y ABREVIATURAS

IoT: En inglés, *Internet of Things,* interconexión digital de personas, animales y cosas (electrodomésticos, coches, etc.) con internet.

WBS: En inglés, *Work Breakdown Structure,* diagrama estructurado de organización y desglose de tareas del proyecto.

ERP: En inglés, *Enterprise Resource Planner,* sistema diseñado para la total administración de los recursos empresariales.

FIRMWARE: *Software* con interacción directa con el *hardware.* Suele permanecer inalterable durante el funcionamiento del resto del *software.*

USB: En inglés, *Universal Serial Bus,* puerto serie universal utilizado para comunicaciones con la placa microcontroladora y transferencia de datos.

TOKEN: Del inglés, cadena de caracteres no repetible utilizada como identificador de identidades. Puede tener asociadas otras características en la aplicación que la genera.

OTA: Del inglés, *Over The Air,* utilizado para las comunicaciones inalámbricas u *online* entre sistemas.

YAML: En inglés, *Yet Another Markup Language,* formato de serialización de datos estructurados legible por humanos, inspirado en el lenguaje de marcado legible *XML.*

CSV: En inglés, *Comma Separated Values,* valores separados por comas. El separador por defecto puede variar.

FRAMEWORK: Del inglés, marco de trabajo utilizado en el ámbito software para agilizar los procesos de desarrollo.