E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Desarrollo de una plataforma de gestión de sensores IoT y cerraduras para contenedores de recogida RSU basada en entorno FIWARE

Grado en Ingeniería Informática

Trabajo Fin de Grado

Leire Mendaza Izco

Jose Javier Astrain Escola

Pamplona, 11/06/2020

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

En esta memoria, se va a explicar el sistema de gestión de dispositivos IoT para contenedores de recogida de residuos sólidos urbanos, basado en el entorno FIWARE, realizado en la empresa i3i Ingeniería Avanzada.

Se desea obtener información a tiempo real de diferentes características de los contenedores, recogidos por dispositivos IoT instalados en dichos contenedores, además de poder gestionar ciertos atributos.

La herramienta FIWARE empleada, es una plataforma impulsada por la Unión Europea, para el desarrollo y despliegue global de aplicaciones de Internet del Futuro. Para gestionar los datos de los diferentes objetos IoT, se empleará el componente Orion Context Broker FIWARE.

El objetivo principal del proyecto es proveer de una plataforma, para gestionar y visualizar datos que informan del estado real de los contenedores.

Para la realización de esta plataforma, se han construido dos entornos, entorno con componentes FIWARE y entorno WEB, mediante la aplicación de automatización de despliegue de aplicaciones Docker.

Abstract

In this document, I am going to explain the management system for IoT devices for urban solid waste collection containers, based on the FIWARE environment, carried out at the company i3i ingeniería avanzada.

The aim is to obtain real-time information on different characteristics of the containers, collected by IoT devices installed in said containers, in addition to being able to manage certain attributes.

The FIWARE tool used is a platform promoted by the European Union for the development and global deployment of Internet applications of the Future. To manage the data of the different IoT objects, the Orion Context Broker component of FIWARE will be used.

The main objective of the project is to provide a platform to manage and visualize data that informs the real state of the containers.

For the realization of this platform, two environments have been built, a FIWARE components environment and a WEB environment, through the docker application.

Contenido del documento

Índice de imágenes

Índice de tablas

1. Introducción

Antecedentes

Para finalizar mis estudios de grado de ingeniería informática, he realizado prácticas curriculares desde el 2 de febrero hasta el 31 de mayo de 2021 en la empresa i3i Ingeniería Avanzada. I3i es una empresa de carácter dinámico y joven que se dedica a diferentes áreas de negocio. Entre ellas se encuentra el área dedicada al mundo IoT y en el que se ha centrado mi proyecto.

La empresa, me propuso realizar un proyecto que consiste en desarrollar una plataforma de gestión de dispositivos IoT para contenedores de recogida de residuos sólidos urbanos (RSU) basada en entorno FIWARE. Además, se ha hecho uso de Docker como sistema de virtualización, para desarrollar y levantar los entornos de trabajo.

IoT o Internet de las cosas, es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet. Este proyecto, se centra en 3 dispositivos definidos y creados por la empresa, denominados, sensores, cerraduras y nodrizas.

FIWARE es una plataforma impulsada por la Unión Europea, para el desarrollo y despliegue global de aplicaciones de Internet del Futuro. La aplicación de Orion Context Broker, uno de los componentes principales de FIWARE, ha sido necesario para el desarrollo de este proyecto. [1]

Docker es una aplicación para automatizar el despliegue de aplicaciones dentro de contenedores software y la virtualización de aplicaciones en múltiples sistemas operativos. [2]

Debido a mi desconocimiento de estas tecnologías, el reto de este proyecto ha sido la labor de investigación que he tenido que realizar para desarrollarlo, de lo cual he aprendido mucho y me ha enriquecido profesionalmente.

Descripción del problema

Los trabajadores que recogen los residuos de los contenedores tienen problemas a la hora de calcular cuándo y cada cuánto deben realizar la recogida de los residuos. Esto ocurre porque no existe forma de calcular tiempos regulares en los que se deben recoger los restos.

Si la recogida no se realiza en el tiempo adecuado, surgen dos problemas.

1. Si la recogida se realiza asiduamente, sin tener en cuenta la ocupación del contenedor, los vehículos de recogida generan una fuente importante de contaminación del aire, a causa de las emisiones de monóxido de carbono, hidrocarburos y óxidos de nitrógeno. Además, las horas trabajadas de los empleados, se desperdician.
2. Sin embargo, si se tarda más de lo necesario, la acumulación de los residuos, pueden llegar a generar problemas complicados de gestionar como son: malos olores, suciedad en la calle por la imposibilidad de tirar residuos dentro del contenedor, aparición de ratas, etc. Esto puede derivar en una situación peligrosa e incómoda para los ciudadanos.

Objetivo

La finalidad de este proyecto es ayudar a minimizar las rutas de recogida para reducir la huella de carbono y fomentar el reciclaje de la ciudadanía, así como, maximizar las horas de trabajo de los gestores. Además, se pretende facilitar a los gestores de residuos de contenedores de recogida RSU, el cumplimiento de diferentes objetivos de desarrollo sostenible.

Los objetivos de desarrollo sostenible son una iniciativa impulsada por Naciones Unidas para la continuidad a la agenda de desarrollo tras los Objetivos de Desarrollo del Milenio. Son 17 objetivos, y uno de los objetivos que se pretende cumplir con este proyecto es el número 11, el cual consiste en construir ciudades y comunidades sostenibles.[3]

Concretamente, el indicador 11.6.1 de este objetivo define el siguiente propósito: "Proporción de desechos sólidos urbanos recogidos periódicamente y con una descarga final adecuada respecto del total de desechos sólidos urbanos generados." [4]

Para este cumplimiento, se pretende automatizar el control de los contenedores, mediante el desarrollo de una aplicación de gestión. Para ello, la empresa i3i ingeniería avanzada, ha desarrollado unos dispositivos con capacidad de recoger información del estado de los contenedores en los que serán instalados.

El proyecto explicado en este documento consiste en desarrollar una plataforma con capacidad de recoger los datos de estos dispositivos y comunicarlos a los usuarios.

Estado del arte

En el mercado ya existen otras soluciones para el problema expuesto.

El crecimiento de las ciudades y el estilo de vida actual implica que se generen cada vez muchos más residuos y el deseo de combatir esto ha provocado que diferentes empresas hayan invertido en el desarrollo de este tipo de soluciones en diferentes países del mundo.

Después de realizar una búsqueda de sistemas que solucionen el problema de la gestión de contenedores de residuos, se han encontrado las siguientes soluciones.

1. Empresa Sensoneo [5]

   La empresa Sensoneo situada en Eslovaquia, es un proveedor de soluciones de gestión de residuos inteligentes de nivel empresarial que permite a las ciudades y empresas gestionar sus residuos de forma rentable, ser más responsables con el medio ambiente y mejorar el bienestar de las personas.

   Esta empresa, provee de una solución completa y bastante extendida por Europa central.

   La solución consiste en monitorear los contenedores con sensores y optimizar la capacidad y las rutas de recogida de residuos. Además, ofrecen una aplicación móvil inteligente, que informa a los ciudadanos que la utilizan, donde se encuentra el contenedor vacío más cercano y adecuado para el tipo de residuos que quieren tirar para ayudar a reducir contenedores desbordados.

Por otro lado, para monitorizar los datos recogidos por los sensores a tiempo real, utilizan Smart Analytics y Smart Routes Planning, herramientas dedicadas a analizar datos y optimizar rutas respectivamente.

2. Empresa Contenur [6]

La empresa Contenur con central en Madrid, ofrece soluciones Smart para combatir el problema. Concretamente, se ofrecen tres servicios:

- Control de acceso: contenedores equipados con cerraduras electrónicas, con apertura mediante elementos inteligentes de identificación y control (tarjetas personalizadas o smartphone).
- Recycla: sistema integral que permite obtener información inmediata de la separación de residuos realizada por cada ciudadano y de su comportamiento a la hora de depositar los residuos, haciendo posible actuar allí donde es necesario para conseguir un reciclaje más eficaz, ahorrando costes y generando beneficios.
- Sensorización: Contenedores que permiten implementar soluciones IoT, identificación mediante RFID, sensores de llenado… Todo ello permite optimizar la planificación de los procesos de recogida y la gestión eficiente de los contenedores.

Sin embargo, la empresa I3I ingeniería avanzada, ha decidido plantear y desarrollar su propia solución, tanto a nivel electrónico, desarrollando sus propios dispositivos (sensores y cerraduras) como a nivel software, para gestionar la comunicación con los dispositivos y gestionar la información recogida.

Descripción de la propuesta

Para solucionar el problema presentado, se pretende desarrollar una aplicación de gestión, de manera que facilite a los usuarios que la utilicen, la visualización y gestión de datos.

La propuesta desarrollada consiste en recoger y guardar en bases de datos, mediante una plataforma con componentes FIWARE, los datos enviados por los mecanismos instalados en los contenedores de residuos.

Estos mecanismos, son dispositivos IoT a los cuales se les ha dotado de conexión a Internet y cierta inteligencia software, sobre los que se pueden medir parámetros físicos, como puede ser, temperatura del contenedor, distancia a la que se encuentran los residuos …

Además, se mostrarán los datos recogidos sobre una aplicación para facilitar la visualización y su posterior gestión. En este proyecto se ha decidido desplegar un entorno web.

Para la correcta ejecución de esta propuesta se establecen los siguientes bloques a ejecutar:

- Plantear e implementar un sistema de virtualización que sirva de entorno de desarrollo de la plataforma (Docker o similar).

- Elegir y parametrizar las herramientas del entorno FIWARE y otros, para cubrir las necesidades del cliente.
- Desarrollar las partes principales de la plataforma (interfaz, bases de datos, context brokers, etc…).
- Testear rendimientos de la plataforma.
- Plantear la adecuación del sistema para entorno nube tipo Azure / AWS.

2. Análisis

Análisis de requisitos

Partiendo de las indicaciones recogidas, después de realizar varias reuniones con el gerente y el responsable informático de la empresa i3i ingeniería avanzada, se ha decidido crear una aplicación web en la cual los usuarios, después de identificarse mediante un correo electrónico y una contraseña, podrán realizar las siguientes acciones, planteadas para facilitar el cumplimiento de los objetivos a desempeñar:

1. Acceder a una herramienta de visualización de series temporales, en este caso Grafana, para analizar el estado de los contenedores.
2. Ver el listado de dispositivos, de los cuales se han recogido datos
    a. Ver atributos estáticos de dichos dispositivos.
    b. Modificar los atributos de los dispositivos.

Para la correcta descripción de las acciones mencionadas, se ha optado por realizar diagramas UML. Concretamente los diagramas de casos de uso, los cuales facilitan la descripción de las acciones y actividades:

Caso de uso: Uso de la plataforma

En el primer caso de uso, se definen las diferentes acciones que se pueden realizar dependiendo del nivel de privilegios que tiene los actores definidos en el sistema.

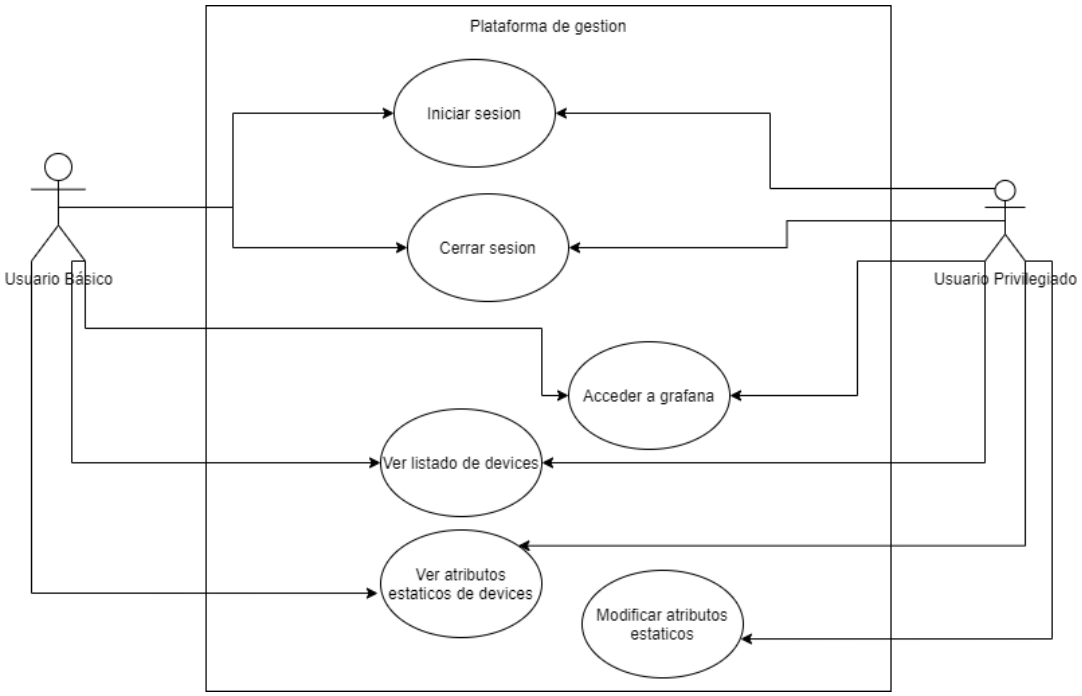


Ilustración 1 - Caso de uso "uso de la plataforma"

Los actores definidos son los siguientes:

Nombre del actor	Propiedades
Usuario Básico	Usuario que tiene acceso a las acciones Iniciar y Cerrar Sesión, Acceder a Grafana y ver listado de dispositivos, así como los atributos.
Usuario Privilegiado	Usuario que tiene acceso a las mismas acciones que el usuario básico y además puede modificar los valores de los atributos estáticos de los dispositivos.

Tabla 1 -Actores de la aplicación

Caso de uso: Inicio de sesión

El inicio de sesión consta de la identificación del usuario mediante el correo electrónico y la contraseña. Estos datos se validan contra una base de datos y la correcta identificación, dará acceso a la plataforma.

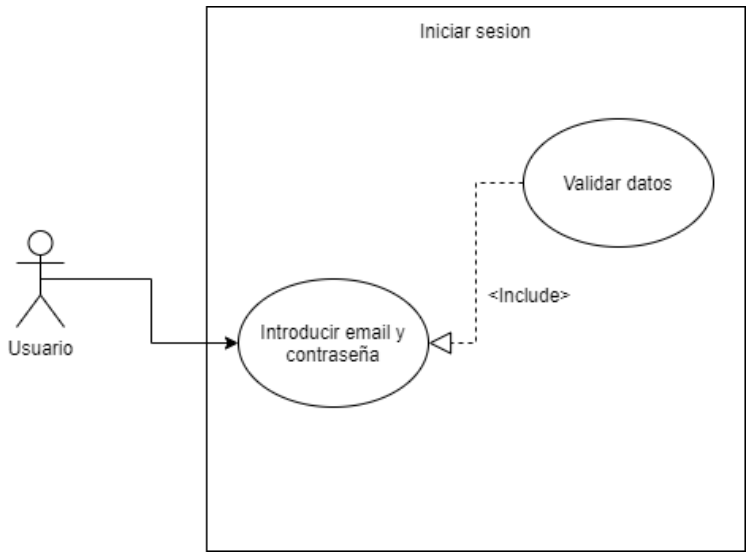


Ilustración 2 - Caso de uso "iniciar sesión"

Cualquier usuario tiene acceso a la acción iniciar sesión, por esto no se ha hecho ninguna división de actores.

Caso de uso: Mostrar lista de dispositivos

Para mostrar la lista de dispositivos, el usuario debe seleccionar el tipo de dispositivo del cual quiere obtener el listado. Una vez seleccionado, se muestra una colección de dispositivos, con su nombre identificativo.

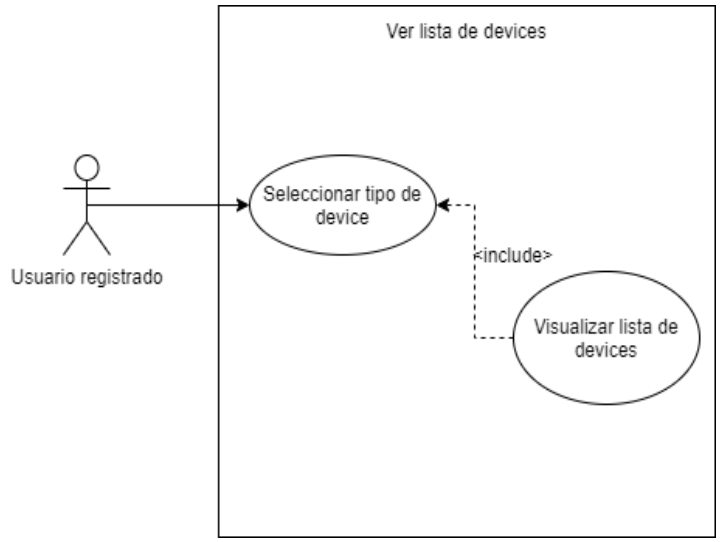


Ilustración 3 - Caso de uso "ver lista de dispositivos"

Cualquier usuario, que previamente haya realizado el inicio de sesión correctamente, tiene acceso a la visualización del listado de dispositivos.

Caso de uso: Mostrar atributos estáticos

Para visualizar los atributos estáticos de un dispositivo, se debe seleccionar el dispositivo del cual se quiere obtener información detallada. Una vez pulsado, se visualizará una colección de atributos estáticos asociados a este objeto. Por cada atributo se mostrará su nombre y su valor.

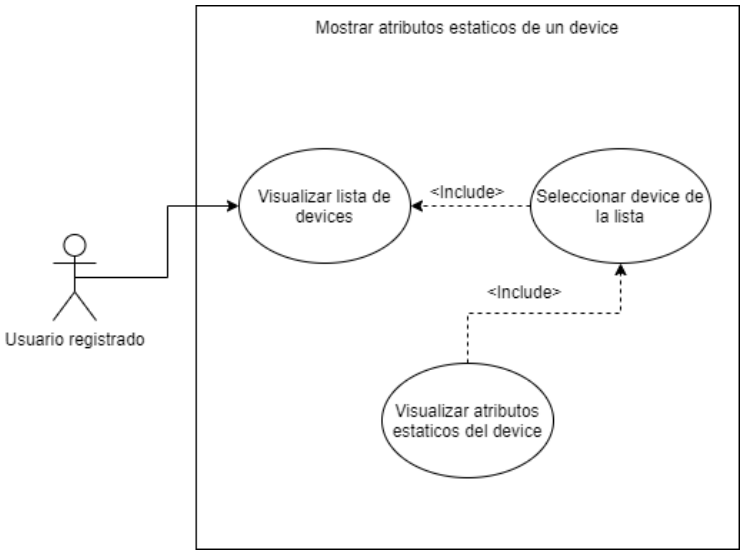


Ilustración 4 - Caso de uso "mostrar atributos estáticos del dispositivo"

Cualquier usuario que previamente haya realizado el inicio de sesión correctamente, tiene acceso a la visualización de los atributos estáticos de los dispositivos.

Caso de uso: Modificar atributos estáticos

Para la modificación de los atributos, una vez se visualizan los atributos estáticos de un dispositivo, existe la posibilidad de introducir un nuevo valor y pulsar un botón submit para actualizar dicho valor en la base de datos.

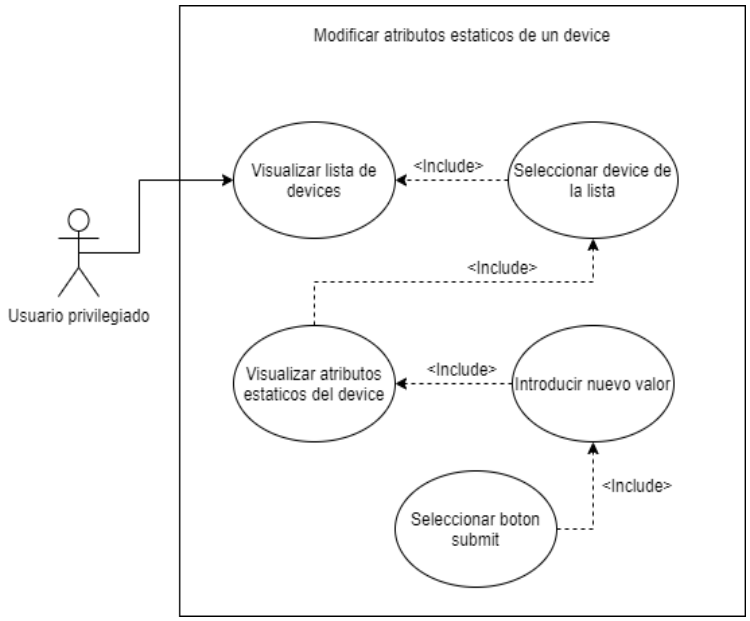


Ilustración 5 - Caso de uso "modificar atributos estáticos del dispositivo"

Únicamente el usuario con privilegios de modificación y que previamente haya iniciado sesión correctamente puede modificar los atributos estáticos de los dispositivos.

Caso de uso: Visualizar Grafana

Para visualizar Grafana, el usuario debe seleccionar dicha opción en el menú. Una vez seleccionada, se mostrará el visualizador de series temporales.

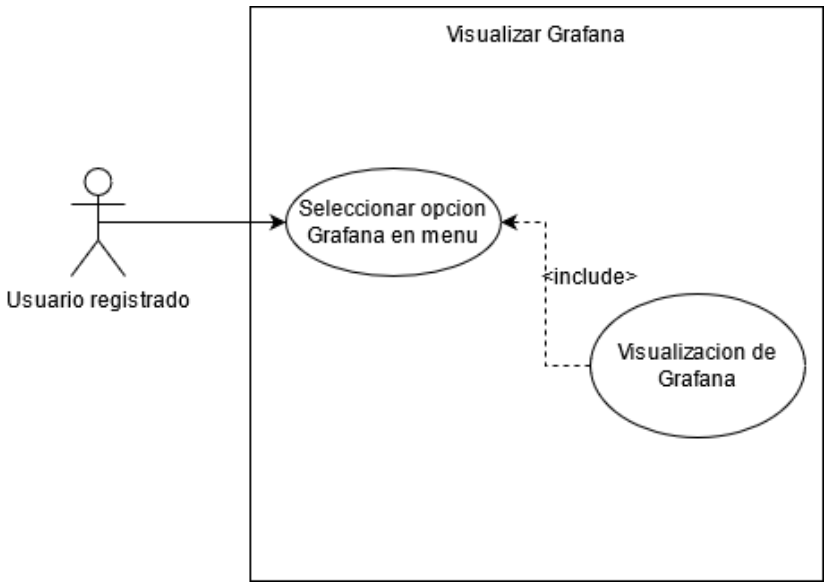


Ilustración 6 - Caso de uso "visualizar Grafana"

Cualquier usuario que previamente haya realizado el inicio de sesión correctamente, tiene acceso a la visualización de Grafana.

Caso de uso: Cerrar sesión

Para cerrar sesión en la aplicación, el usuario debe seleccionar el botón cerrar sesión que se visualiza con el texto cerrar sesión.

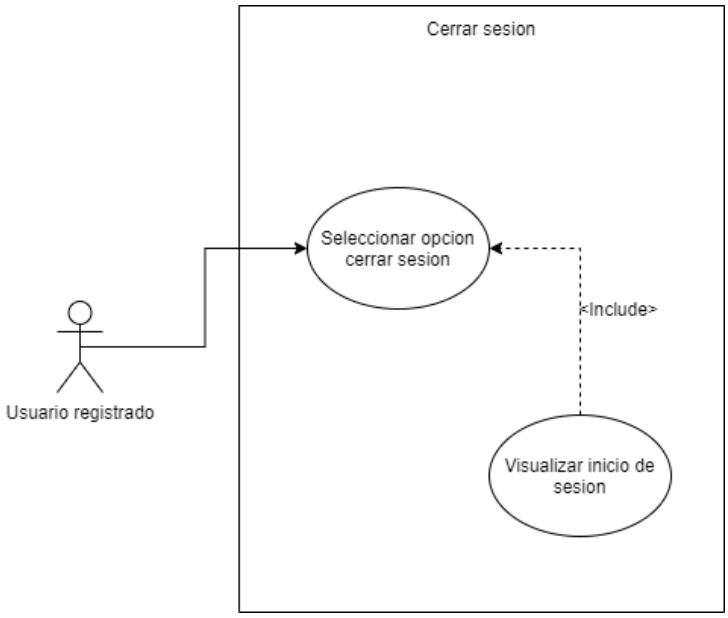


Ilustración 7 - Caso de uso "cerrar sesión"

Cualquier usuario que previamente haya realizado el inicio de sesión correctamente, tiene acceso a la opción cerrar sesión.

Análisis del contexto

Para el correcto entendimiento de las acciones ya mencionadas, se ha decidido realizar diagramas de secuencia de las acciones que se consideran más complejas. Las acciones no mencionadas, son triviales.

Los diagramas de secuencia se utilizan para modelar la interacción entre objetos de un sistema. Es por ello, que se ha optado por el uso de este tipo de diagrama UML para la explicación de las acciones a profundizar.

En los diagramas que se exponen a continuación, se muestra el funcionamiento de diferentes patrones de diseño aplicados, los cuales se explican más adelante en el apartado “Diseño - Arquitectura Software”. También se muestran componentes como iot-agent y mongodb en los cuales se profundiza en el apartado “Diseño - Arquitectura del sistema”.

Diagrama de secuencia: Mostrar atributos estáticos

En el siguiente diagrama se muestra el procedimiento que se ejecuta cuando el usuario desea visualizar los atributos estáticos de un dispositivo:

El usuario realiza la petición de mostrar atributos estáticos, la cual recoge el controlador. A continuación, este decide que debe hacer una llamada a la función que procede. Cuando el

controlador recibe la respuesta, éste llama a una nueva función. Una vez se prepara la respuesta que el usuario verá, se muestran los atributos estáticos solicitados por el usuario.

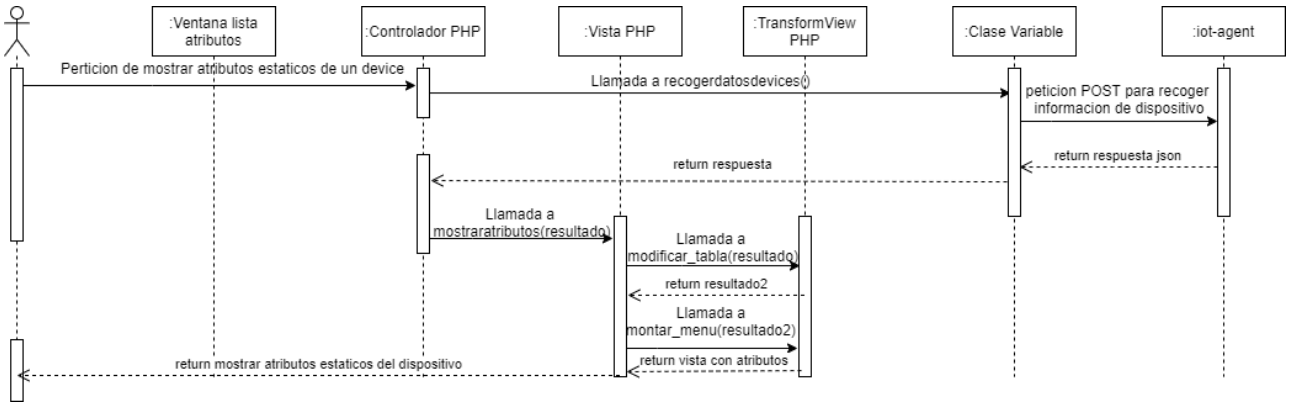


Ilustración 8 - Diagrama de secuencia "mostrar atributos estáticos"

Diagrama de secuencia: Modificar atributos estáticos

En el siguiente diagrama, se explica el proceso que se ejecuta cuando el usuario modifica el valor de un atributo estático:

El usuario manda una petición de modificar valor de un atributo, junto con el atributo y el valor a aplicar. El controlador lo recoge y decide llamar a la función que procede. Esta función, conecta con la base de datos y realiza una consulta con el fin de modificar el valor del atributo. La base de datos comunica una respuesta, y cuando llega al Controlador, si no se ha modificado ningún atributo, se responde con un mensaje de error y si por el contrario se ha modificado algún atributo, se comunica que la modificación ha sido realizada con éxito.

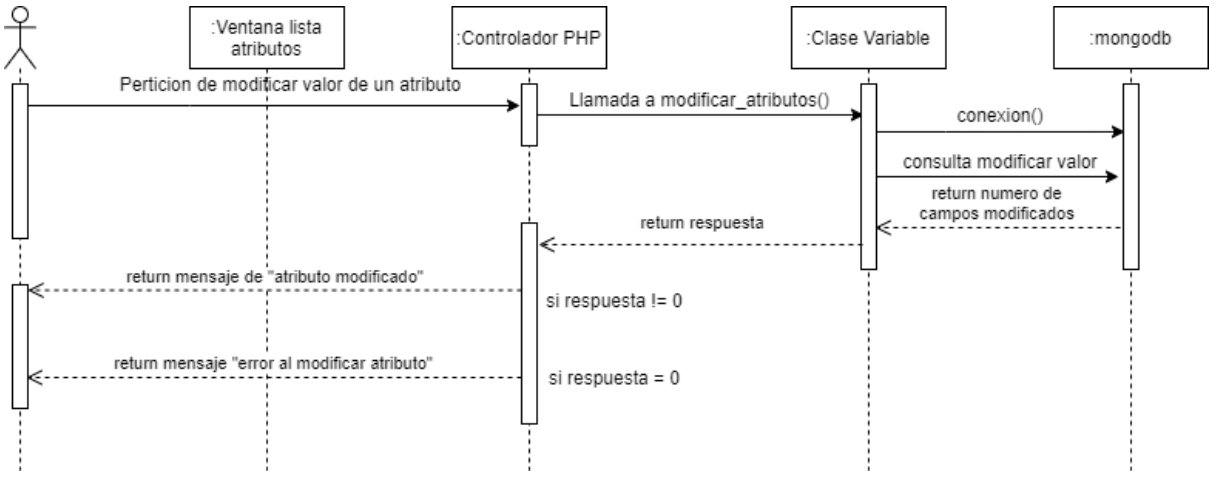


Ilustración 9 - Diagrama de secuencia "modificar atributos estáticos"

Análisis de riesgos

Con el fin de minimizar el impacto de ciertos problemas que podían surgir en el transcurso del proyecto, se realizó un análisis preliminar de riesgos (APR). La realización del análisis se ha realizado con el propósito de mitigar la probabilidad de que suceda, así como, en el caso de no poder evitar el suceso, moderar su impacto.

Este análisis consiste en detectar los riesgos que pueden presentarse en el proyecto, cuál es el impacto que puede tener y las acciones a tomar frente al riesgo, si este llegase a suceder.

Además, se ha realizado una valoración numérica para identificar cual es la gravedad de que suceda el riesgo, entre 1 y 5, siendo 1 un riesgo que tiene un impacto leve y 5 un riesgo que impacta gravemente en el proyecto.

Por otro lado, se ha evaluado la probabilidad que tiene el riesgo de que suceda. Se ha realizado mediante una evaluación numérica de 1 a 3, siendo 1 nada probable y 3 muy probable.

Los riesgos que considero que pueden surgir al inicio del proyecto son los siguientes:

RIESGO	IMPACTO	VALORACIÓN	PROBABILI-DAD	ACCIONES
Falta de documentación para las tareas a realizar.	Imposibilidad de realizar las tareas.	5	2	Planificar tiempo de proyecto para la búsqueda de documentación.
Falta de conocimiento con las herramientas a utilizar, así como, Docker, FIWARE, IoT, AWS, Bootstrap, …	Imposibilidad de realizar las tareas con rapidez.	4	3	Búsqueda de información y realización de cursos formativos. Apoyarme en tutor y compañeros de empresa.
Falta de material /recursos para realizar proyecto.	Imposibilidad de realizar tareas hasta obtener material.	4	1	Solicitar y contemplar con la empresa la adquisición del material.

Tabla 2 - Riesgos inicio

Los riesgos que valoro que pueden impactar durante la ejecución del proyecto podrían ser:

RIESGO	IMPACTO	VALORACIÓN	PROBABILI-DAD	ACCIONES
Elaboración de un diseño web, sin la	Falta de profesionalidad en el diseño de la web.	1	3	Hacer uso de plantillas y herramientas como Bootstrap.

colaboración de un diseñador.				
Cambio de requisitos durante la ejecución del proyecto.	Complicación de finalizar el proyecto a tiempo.	5	1	Revisar con los responsables los requisitos al inicio del proyecto, asentarlos antes de su ejecución.
Fallos en la integración de los componentes desarrollado.	Incremento del tiempo para finalizar proyecto.	3	2	Dedicar tiempo extra a la búsqueda de soluciones.

Tabla 3 - Riesgos durante

Los riesgos para tener en cuenta cuando se acerca la finalización del proyecto son:

RIESGO	IMPACTO	VALORACIÓN	PROBABILI-DAD	ACCIONES
No realizar y completar los requisitos del proyecto en el tiempo estimado.	Proyecto no finalizado.	5	1	Dedicar horas extra y recalcular tiempo y tareas.

Tabla 4 - Riesgos finalización

3. Diseño

Arquitectura del sistema

El sistema desarrollado se compone de tres arquitecturas principales: Arquitectura de los objetos IoT instalados en los contenedores, arquitectura de la plataforma y arquitectura de la aplicación de gestión desarrollada en entorno web.

Los objetos IoT, son aquellos objetos cotidianos que, mediante una conexión a internet, pueden en este caso, recoger datos y enviarlos.

Los datos recogidos por estos objetos son enviados a la plataforma, la cual almacena esta información en bases de datos temporales. Por último, la aplicación de gestión desarrollada es la encargada de comunicar a los usuarios con los datos a través de un entorno web.

El diagrama que describe el sistema mencionado es el siguiente:

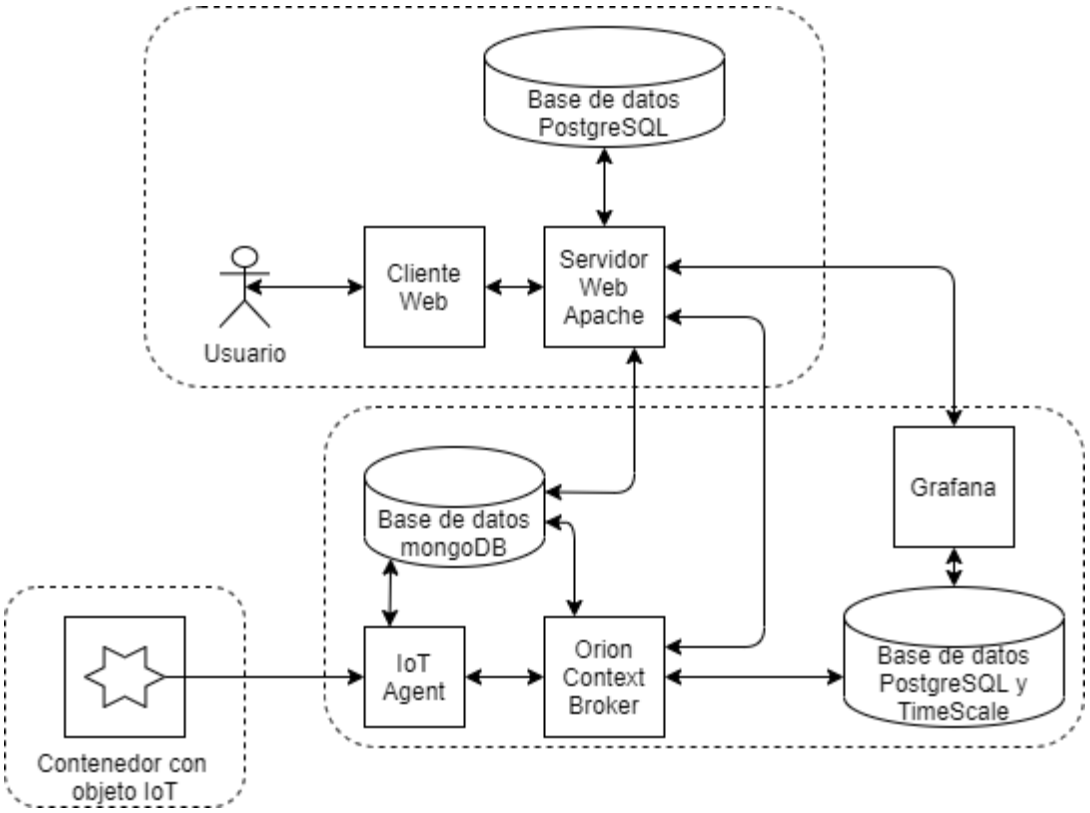


Ilustración 10 - Arquitectura del sistema

A continuación, se profundiza en la estructura de cada una de las arquitecturas aplicadas. Y se explican los componentes que las forman.

Arquitectura de objetos IoT

Un dispositivo IoT consiste en un objeto al que se le ha dotado de conexión a Internet y cierta inteligencia software, sobre el que se pueden medir parámetros físicos o actuar remotamente y que por tanto permite generar un ecosistema de servicios alrededor del mismo. [7]

Con el propósito de recoger datos a tiempo real de los contenedores de gestión RSU, se pretende instalar dispositivos IoT, sensores, cerraduras y nodrizas, los cuales serán los encargados de enviar los datos requeridos. En el mercado existen una infinidad de dispositivos definidos con esa finalidad. Sin embargo, en este proyecto se han implementado dispositivos propios.

La razón por la que la empresa i3i ha llevado a cabo esta decisión es, porque lo que generalmente se ofrece en el mercado, es la compra de los servicios de dichos dispositivos y no los dispositivos en sí. Esto no cumple con lo que la empresa quiere presentar. Lo que ellos quieren ofrecer es libertad del producto, de manera que el comprador tenga independencia en el futuro. Esta independencia implica que, si el cliente quiere desarrollar futuras funcionalidades, no tiene por qué realizarlas en la empresa i3i.

La estructura de los dispositivos que se desean instalar en los contenedores es similar a la que se define en la siguiente imagen:

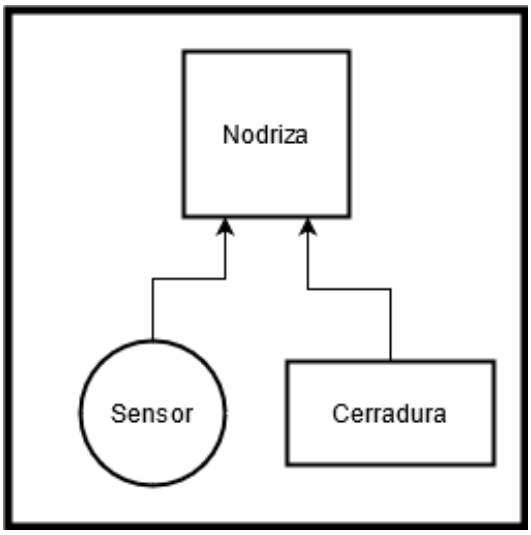


Ilustración 11 - Arquitectura de los dispositivos IoT

El objeto denominado nodriza, tiene como objetivo recoger los datos generados por el sensor y la cerradura y enviarlos a la plataforma que se ha creado.

Los objetos denominados como sensor y cerradura por su parte tienen como objetivo leer y generar la información requerida y mandarla a la nodriza.

Por cada uno de los dispositivos nombrados, se desea almacenar diferentes datos, los cuales se mencionan más adelante.

Arquitectura de la plataforma

Con la finalidad de automatizar la recogida de los datos enviados por los objetos IoT se ha construido una arquitectura, basada en el componente de FIWARE Orion Context Broker.

FIWARE intenta proveer de una arquitectura totalmente abierta, pública y libre, así como de un conjunto de especificaciones que permita a los desarrolladores, proveedores de servicios, empresas y otras organizaciones desarrollar productos que satisfagan sus necesidades, sin dejar de ser abierta e innovadora.

La arquitectura de la plataforma se basa en las arquitecturas estudiadas en la documentación oficial de FIWARE. Una vez adaptada a los propósitos y herramientas que se desean utilizar en el proyecto, la estructura principal de la arquitectura implementada es la que se describe en la siguiente figura: [8]

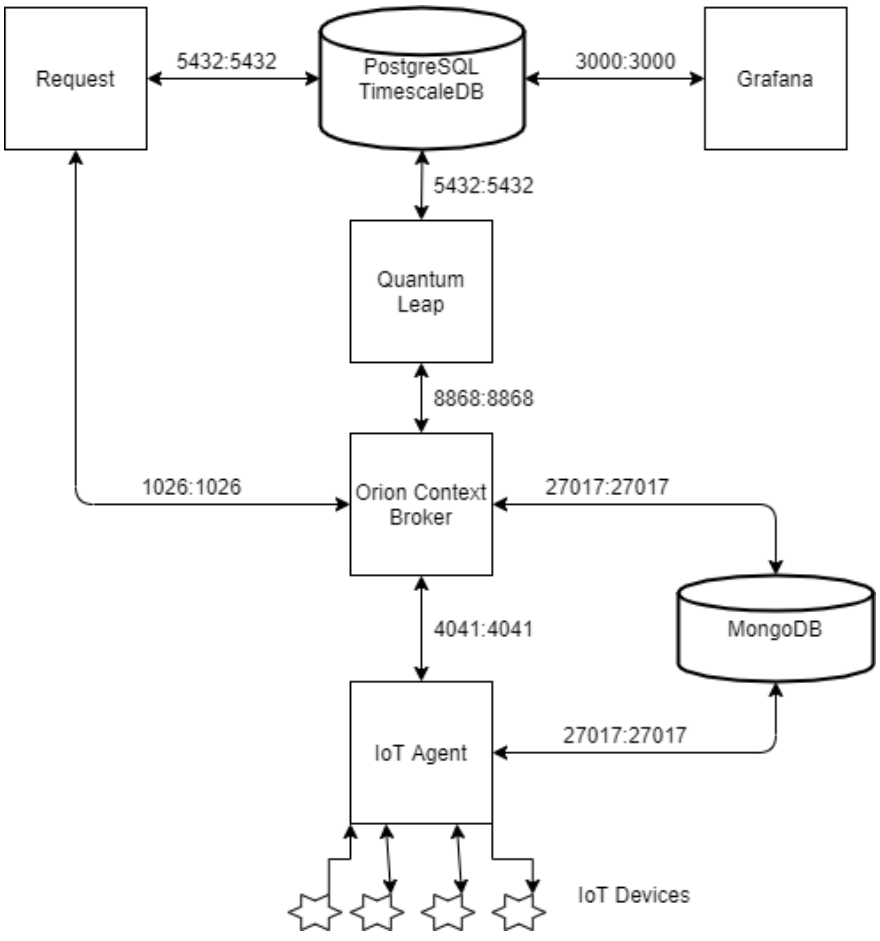


Ilustración 12 - Arquitectura de la plataforma

Los dispositivos IoT, envían los datos requeridos a IoT-agent. IoT agent, es un componente que permite a los objetos enviar sus datos para que sean controlados por un Context Broker. [9]

En este caso se ha hecho uso de Orion Context Broker el cual se encarga de administrar, consultar y actualizar la información. Estos dos componentes, se conectan mediante el puerto 4041. A su vez, también se conectan con la BBDD mongoDB mediante el puerto 27017. [10][11]

MongoDB es una base de datos no SQL que se utiliza para almacenar datos del contexto.

La estructura interna que conecta Orion Context Broker y MongoDB es la que se define en la siguiente imagen:

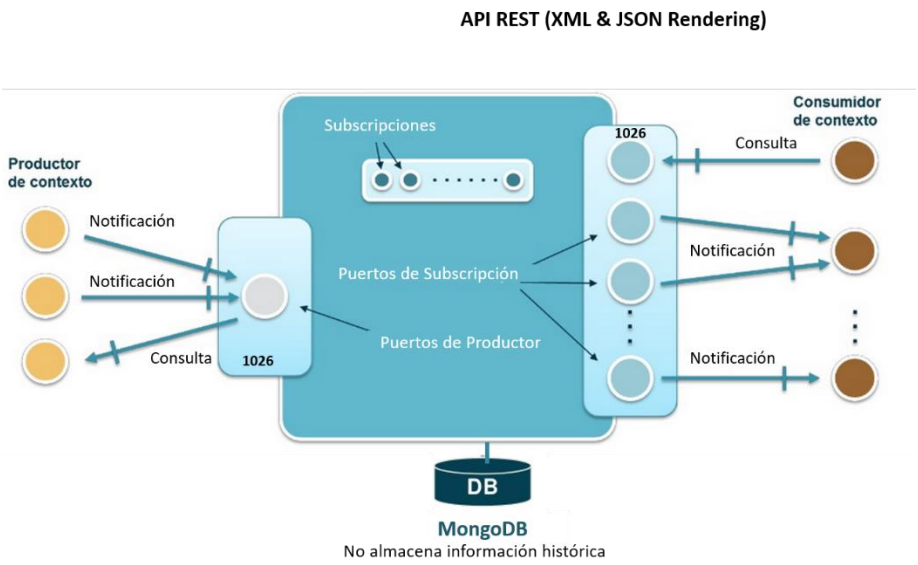


Ilustración 13 - Orion Context Broker

A continuación, Orion Context Broker, requiere almacenar los datos enviados por los dispositivos en otra base de datos.

La base de datos seleccionada para este proyecto ha sido TimescaleDB. TimescaleDB es una base de datos diseñada para almacenar y procesar datos en forma de series temporales. TimescaleDB es una extensión de PostgreSQL, de manera que como se explicará más adelante, se han tenido que añadir las dos bases de datos a la estructura. [12][13]

El almacenamiento en la base de datos se hace a través de QuantumLeap. La función de este componente es controlar los datos que se van a almacenar. Por ejemplo, vigila que el tipo de los datos a almacenar sea el correcto. Además, es el encargado de estructurar dichos datos para recogerlos en las bases de datos temporales. [14]

Orion y Quantumleap, se conectan mediante el puerto 8868 y la conexión a la base de datos se realiza mediante el puerto 5432.

Por último, en este proyecto, se ha visto necesario la implementación de una herramienta de visualización de datos, con el fin de que el usuario pueda estudiar los datos de una manera más clara. Para ello, se ha incluido la herramienta Grafana. La conexión entre la bbdd y el software de visualización se realiza mediante el puerto 3000. [15]

Los puertos que se han definido en la arquitectura no son aleatorios, al igual que la estructura, se han aplicado los puertos que se explican en la documentación oficial.

Arquitectura de la aplicación web

La aplicación web desarrollada para la visualización y gestión de los datos enviados a tiempo real por los objetos IoT tiene la siguiente estructura:

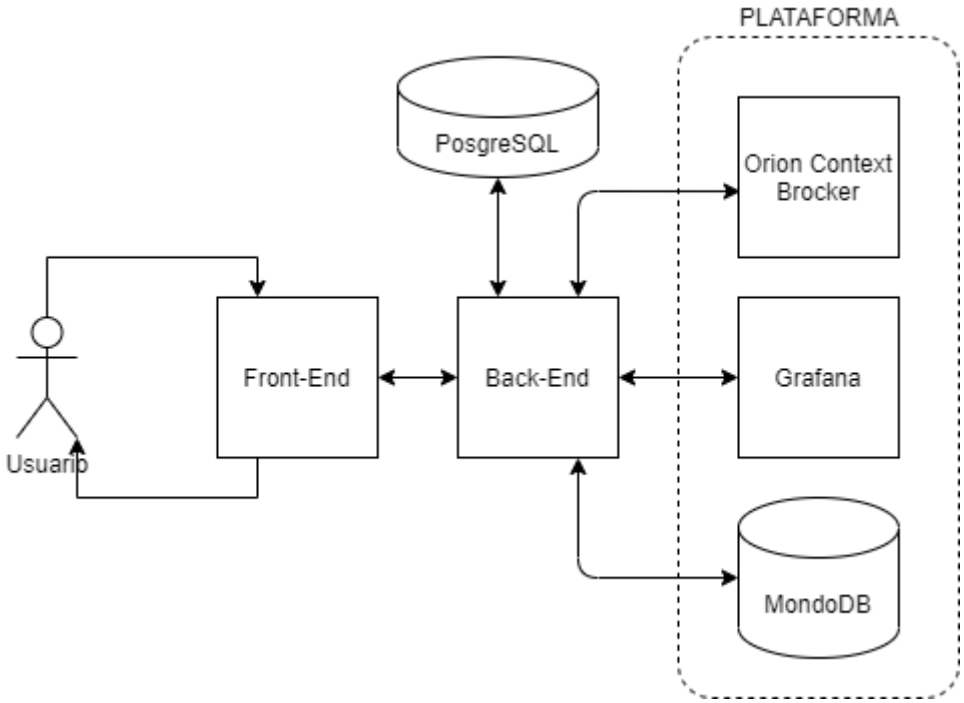


Ilustración 14 - Arquitectura de la aplicación

Con el propósito de facilitar la visualización y gestión de los objetos, se ha desarrollado un entorno web que comunica directamente con la plataforma mencionada en el punto anterior. De esta manera, los usuarios registrados, pueden visualizar los datos mediante el acceso a la herramienta Grafana y en el caso de ser usuarios privilegiados, podrán modificar los valores estáticos.

Los usuarios registrados, están almacenados en la base de datos PostgreSQL.

Arquitectura software

Con el fin de facilitar y estructurar el desarrollo del entorno web se han empleado diferentes patrones de diseño. La elección de estos patrones de diseño ha sido personal, ya que la empresa no puso limitaciones o pautas a la hora de programar.

Por un lado, se ha utilizado el patrón de diseño Modelo-Vista-Controlador (MVC). La razón principal del uso de este patrón ha sido su comodidad y sencillez. Esto se debe a que, en las asignaturas realizadas en la Universidad Pública de Navarra, se ha hecho uso de este patrón para estructurar entornos web. Además, se considera que es un patrón que estructura claramente el código desarrollado y en este caso, ha facilitado la búsqueda de errores.

A continuación, y de forma automática, se aplicó el patrón Page Controller, para centrar la recogida de todas las peticiones realizadas por el usuario. Esto ha ayudado a tener el código más estructurado.

Por último, conforme se iba desarrollando el entorno, se vio la necesidad de aplicar un patrón de vistas, en concreto el patrón Transform View. Al ser un entorno que no es estático, se realizan muchas modificaciones en la imagen de éste. Para facilitar y centrar estas modificaciones, se optó por utilizar el patrón mencionado.

1. MVC

Este patrón consiste en realizar 3 divisiones lógicas en el código (modelo, vista y controlador). El modelo, es el encargado de realizar la carga, almacenamiento y el tratamiento de los datos (que pueden estar en una BBDD). La vista es la encargada de imprimir los datos que devuelve el modelo. El controlador es la clase encargada de organizar y realizar las llamadas necesarias tanto al modelo como a la vista y de interactuar con el usuario.

Por otro lado, el modelo se ha querido estructurar por clases, diferenciando las clases “usuario” y “variable”.

2. Page Controller

Se ha utilizado este patrón de diseño, para recoger todas las peticiones http (get y post) que se realizan, así como para gestionar dichas peticiones y proceder a ejecutar los métodos apropiados tanto de la vista como del modelo. Este elemento forma parte del controlador y tiene el nombre de “index.php”.

3. Transform View

Este patrón, es el responsable de montar la estructura de los archivos html necesarios y mandarlos a la vista, la cual está encargada de mostrar los html finales.

La estructura general y cómo se comunican los diferentes componentes se define en la siguiente imagen:

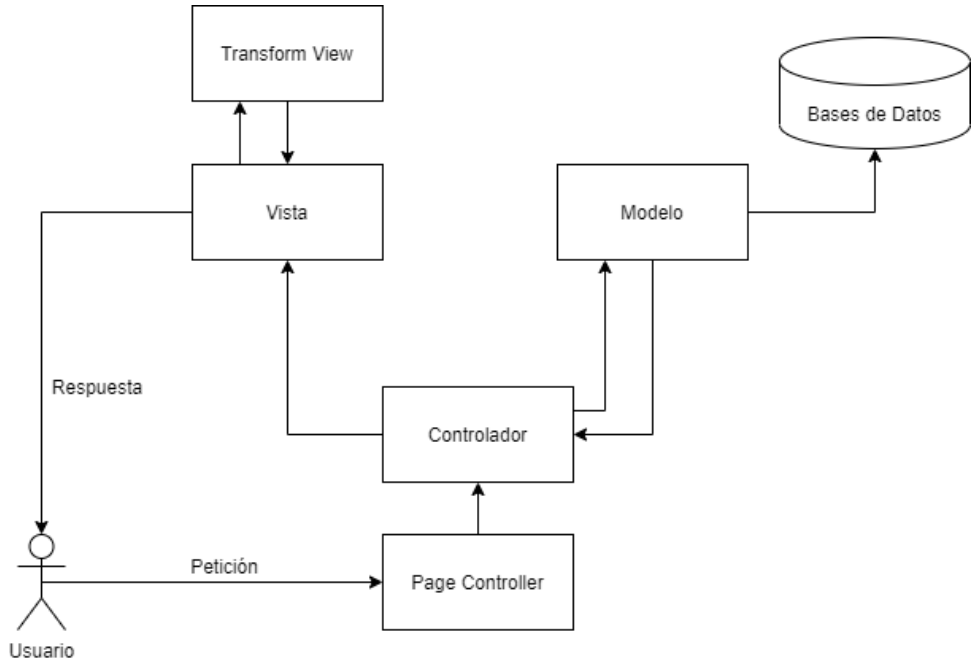


Ilustración 15 - Patrones de diseño

El usuario realiza una petición, la cual es capturada por el PageControler. El controlador, decide cuál es la acción que hay que tomar respecto a la petición recibida.

Dependiendo de la acción a realizar, el controlador se comunica con el modelo y con la vista para devolver una respuesta adecuada al usuario.

Modelo y estructura de datos

Para facilitar la portabilidad de datos en diferentes ámbitos como IoT, se han ido creando diferentes modelos de datos. Para modelar el modelo definido para el almacenamiento de los datos enviados por los dispositivos IoT, se ha utilizado el modelo de datos NGSI-LD. [16]

NGSI-LD es un modelo de información y API para editar, consultar y subscribirse a información de contexto.

Un ejemplo que define este modelo es el siguiente:

```
{
    "devices": [
        {
            "device_id":   "cerradura001",
            "entity_name": "urn:ngsi-ld:cerradura:001",
            "entity_type": "Thing",
            "timezone":    "Europe/Madrid",
            "attributes": [
                { "object_id": "numero_tarjeta", "name": "numero_tarjeta", "type": "Integer" },
                { "object_id": "falso_wakeup", "name": "falso_wakeup", "type": "Integer"},
                { "object_id": "bateria", "name": "bateria", "type": "Number"},
                { "object_id": "estado_carpeta", "name": "estado_carpeta", "type": "Integer" },
                { "object_id": "timestamp_cerradura", "name": "timestamp_cerradura", "type": "DateTime"}
            ],
            "static_attributes": [
                { "name":"ubicacion", "type": "Text", "value": "urnieta"}
            ]
        }
    ]
}
```

Ilustración 16 - Modelo de datos NGSI-LD

Mediante este modelo de datos, se han definido los dispositivos utilizados, con los atributos tanto estáticos como dinámicos que se desean almacenar.

A continuación, se presenta la información que se requiere almacenar de cada dispositivo:

Variable	Tipo	Valores	Descripción
SENSOR			
Variables dinámicas			
ir	Integer	0 – 1	Detección de fuego. Si hay fuego, cogerá valor 1. Si no hay fuego, valor 0.
distancia	Number	Número real	Distancia entre el sensor y los residuos.
temperatura	Number	Número real	Temperatura del contenedor.
voltaje pila	Number	Número real	Voltaje de la pila.
switch magnético	Integer	0 – 1	Si el switch está activo o no. Si lo está, cogerá valor 1, si no, valor 0.
tilt	Integer	0 – 1	Detección de vuelque. Es un pequeño objeto que se mueve dependiendo del ángulo. Si vuelca, cogerá valor 1, si no, valor 0.
tiempo despierto	Integer	Número entero	Cantidad de segundos que ha estado despierto.
missfire	Integer	Número entero	La cantidad de veces que ha fallado en el envío de datos.
timestamp sensor	DateTime	Fecha y hora	Fecha y hora a la que se han enviado estos datos.

Variables estáticas			
tipo de contenedor	Integer	0, 1, 2, 3, 4	Define el tipo de contendor: 0-sin definir 1-organico 2-resto 3-plasticos 4-papeles
número de contenedor	Integer	Un número entero	El número identificativo del contenedor.
CERRADURA			
Variables dinámicas			
número de tarjeta	Integer	Número entero	Número de la tarjeta que abre la cerradura.
falso wakeup	Integer	0 - 1	Si se despierta por contacto de la tarjeta o no. Si se despierta cuando no hay contacto con tarjeta, es un falso despertar, por lo tanto, cogerá valor 1.
batería	Number	Número real.	El porcentaje de batería que tiene la cerradura.
estado de tapeta	Integer	0 - 1	Si la tapa está abierta o cerrada. Si está abierta, cogerá valor 1, si está cerrada, valor 0.
timestamp cerradura	DateTime	Fecha y hora	Fecha y hora a la que se han enviado estos datos.
Variables estáticas			
ubicación	Text	Texto que describe la localización en la que debe estar la cerradura	Ubicación en la que debe estar la cerradura.
NODRIZA			
Variables dinámicas			
batería	Number	Valor real	Batería de la nodriza.
rssi	Text	2 – 3	Indica el tipo de señal. Señal 2G, cogerá valor 2, señal 3G cogerá valor 3.

tipo señal	Text	2 – 3	Indica el tipo de señal. Señal 2G, cogerá valor 2, señal 3G cogerá valor 3.
fallos conexión	Text	2 – 3	Tipo de señal que ha provocado fallo. 2G/3G.
posición gps	Text	Texto que describe la localización de la cerradura	Ubicación/Posición en la que se encuentra la Nodriza. Es un valor aproximado.
timestamp nodriza	DateTime	Fecha y hora	Fecha y hora a la que se han enviado estos datos.
Variables estáticas			
ubicación	Text	Texto que describe la localización de la cerradura	Ubicación en la que debe estar la nodriza.

Tabla 5 - Atributos de los dispositivos

Por otro lado, durante el desarrollo de la plataforma web, se vio necesario el control de los usuarios con acceso a la plataforma, por lo tanto, se decidió optar por la creación de una tabla llamada usuario, y otra tabla llamada privilegio_usuario.

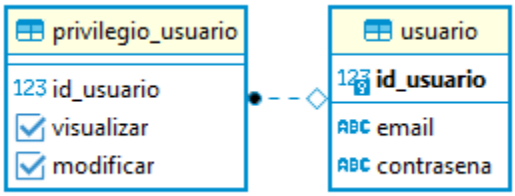


Ilustración 17 - Tablas de la aplicación

El diagrama Entidad – Relación del modelo descrito es el siguiente:

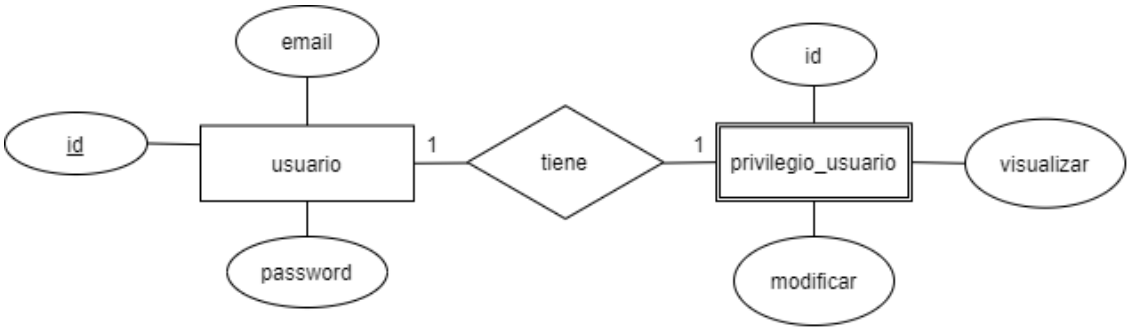


Ilustración 18 - Diagrama Entidad-Relación

En la tabla usuario, se almacenan los usuarios que tienen acceso a la plataforma.

Campo	Tipo	Descripción
id_usuario	Integer	Clave primaria. Campo que define el id del usuario. Es auto incremental.
email	Text	Campo que define el correo electrónico del usuario. No pude ser nulo y es único.
password	Text	Campo que almacena la contraseña del usuario. Se almacena encriptada y no puede ser nula.

Tabla 6 - Campos de usuario

En la tabla privilegio_usuario, se almacenan los privilegios de los usuarios.

Campo	Tipo	Descripción
id_usuario	Integer	Es clave extranjera, relacionada con el campo id_usuario de la tabla usuario.
visualizar	Boolean	Campo que indica si el usuario con id id_usuario tiene permiso para visualizar datos. Si tiene permisos, coge valor true.
modificar	Boolean	Campo que indica si el usuario con id id_usuario tiene permiso para modificar datos. Si tiene permisos, coge valor true.

Tabla 7 - Campos de privilegio_usuario

4. Desarrollo

Metodologías de desarrollo

La aplicación y cumplimiento de una metodología de desarrollo, es complicado. Una metodología se compone de múltiples reglas y procedimientos a llevar a cabo y no todos son adecuados para acomodarlos en el desarrollo.

Por lo tanto, en este proyecto no ha habido aplicación formal y completa de ninguna metodología de desarrollo. Sin embargo, se han aplicado diversas prácticas de diferentes metodologías.

Por un lado, se ha aplicado el desarrollo en cascada. Este ha consistido en estructurar el proyecto en bloques y seguir un orden lógico en el desarrollo de dichos bloques. Los bloques en los que se ha estructurado el proyecto son los siguientes:

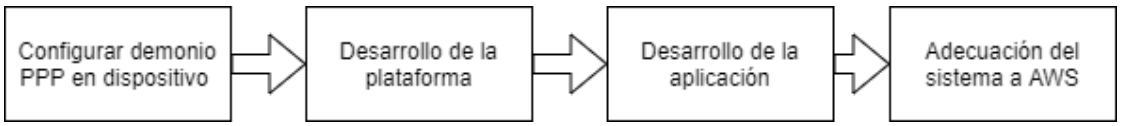


Ilustración 19 - Desarrollo en cascada

Sin embargo, por cada uno de los bloques se han aplicado diversas prácticas para un desarrollo más incremental.

De esta manera, por cada bloque se han planteado tareas cortas, se han analizado, se han desarrollado y se han testado. Tras su finalización, se ha realizado reuniones con la empresa, para su revisión y validación.

Específicamente, se han realizado reuniones una vez por semana, en la que se ha llevado a cabo una revisión del trabajo desarrollado hasta el momento. En estas reuniones se han concretado las siguientes tareas a realizar y correcciones o mejoras a aplicar en las tareas ya desarrolladas.

Herramientas de desarrollo empleadas

Sistema de control de versiones

La empresa, no hace uso de ningún sistema de control de versiones. La herramienta utilizada para compartir la totalidad del proyecto (código, documentos, versiones, entregables, …) ha sido OneDrive. De esta manera, el conjunto de modificaciones está a disposición de todos los participantes del proyecto.

Como propuesta, recomendaría el uso de un sistema de control de versiones como git. Sin embargo, al ser un proyecto que he realizado junto a trabajadores de la empresa los cuales no se sienten cómodos con esta práctica, finalmente no se ha aplicado.

Recursos utilizados

Recursos Hardware	Ordenador portátil	Se ha utilizado ordenador portátil para desarrollar todas las partes del proyecto. Modelo: Acer Procesador: Intel(R) Core (TM) i5-4200U CPU Memoria RAM: 8,00 GB Tarjeta gráfica: Almacenamiento: 475 GB Sistema Operativo: Windows 10 Pro
	Servidor	Se ha utilizado un servidor Intel, para almacenar el proyecto desarrollado. Además, se ha utilizado para montar una máquina virtual mediante la plataforma de virtualización Proxmox.
	Raspberry Pi y Quectel ec25-e LTE	Raspberry Pi es una serie de ordenadores de placa reducida, ordenadores de placa única u ordenadores de placa simple de bajo coste desarrollado por la Raspberry Pi Foundation. [17] Se ha utilizado, para realizar pruebas en Amazon Web Services. y para la configuración del demonio PPP y VPN en un el usb Quectel ec25-e

Tabla 8 - Recursos hardware

Recursos Software	Máquina virtual Proxmox [18]	Se ha creado una máquina virtual Ubuntu 18.04 LTS, mediante Proxmox, para levantar los contenedores Docker desarrollados: Imagen: iso/ubuntu-18.04.1 Memoria: 4GB Tamaño de disco: 20GB
	Visual Studio Code [19]	El entorno seleccionado para desarrollar ha sido Visual Studio Code, ya que soporta gran variedad de lenguajes. Además, tiene librerías que permiten realizar otras acciones, las cuales han sido adecuadas para este proyecto. Principalmente se ha hecho uso de las siguientes librerías: **REST Client:** Esta librería, ofrece la posibilidad de ejecutar peticiones desde la misma herramienta. El procedimiento consiste en, crear un fichero con extensión http. A continuación, se crea una petición, y automáticamente encima, aparece un pequeño botón para ejecutarla. Se pueden añadir múltiples peticiones separándolas por ###  *Ilustración 20 - Rest Client*

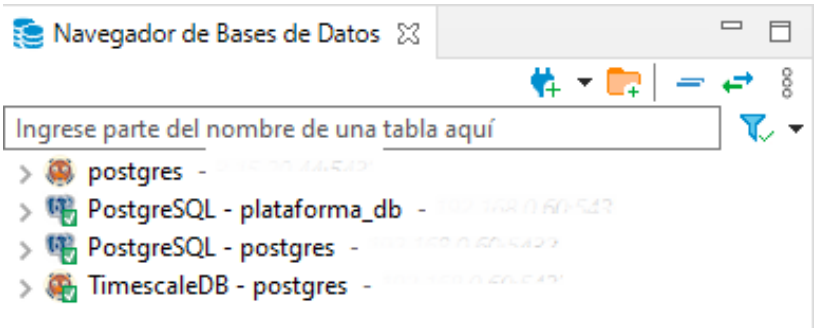
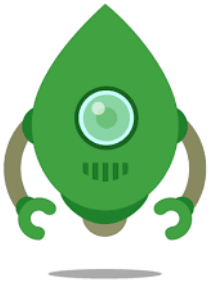
		Docker: Esta librería, permite programar archivos Dokerfile y docker-compose.yml, entre otros, de manera sencilla y clara.  *Ilustración 21 - Docker*
	DBeaver [20]	El entorno seleccionado para visualizar las Bases de Datos ha sido DBeaver. Ofrece una interfaz agradable para la visualización de una gran variedad de sistemas de gestión de bases de datos. En este proyecto, se ha utilizado para la visualización de PostgreSQL y TimeScale, los sistemas de gestión de bases de datos utilizados.  *Ilustración 22 - DBeaver*
	Robo 3T 1.3.1 [21]	Para la visualización del sistema de bases de datos NoSQL, MongoDB, se ha optado por la GUI gratuita Robo 3T 1.3.1. Se ha utilizado esta herramienta por su sencillez y claridad a la hora de visualizar datos.  *Ilustración 23 - Robo 3T*
	Navegador web Chrome [22]	 *Ilustración 24 - Chrome* Para realizar todas las búsquedas relacionadas con el proyecto, se ha utilizado el navegador web Chrome.
	Putty [23]	Para realizar las conexiones SSH con el servidor y la Raspberry Pi, se ha utilizado la herramienta Putty.  *Ilustración 25 - Putty*

Tabla 9 -Recursos software

Tecnologías utilizadas

1. DOCKER y Docker-Compose

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Tanto para el despliegue de la plataforma como para el entorno de desarrollo web, el uso de Docker y Docker-Compose ha sido primordial. Cada uno de estos entornos, se ha levantado mediante archivos docker-compose.yml.

2. Desarrollo WEB

Para el desarrollo del entorno web, se ha hecho uso de diferentes herramientas y lenguajes. La selección de éstas ha sido motivada por la comodidad de uso debido al conocimiento previo de las mismas.

a. Front-End

Para desarrollar el diseño de la aplicación web, se han utilizado los siguientes lenguajes y herramientas de estilo.

El lenguaje de etiquetas de hipertexto HTML, como lenguaje base para el desarrollo de las páginas.

El lenguaje de diseño gráfico CSS, junto con plantillas y herramienta BootStrap, para definir el diseño y la presentación de las páginas.

El lenguaje JavaScript para lanzar eventos, como por ejemplo alertas.

b. Back-End

Para desarrollar la lógica de la aplicación, se ha utilizado el lenguaje de programación de código abierto y adecuado para el desarrollo web, PHP.

c. Servidor Web

Como servidor web HTTP, se ha optado por utilizar Apache.

Un servidor web es un software que forma parte del servidor y tiene como misión principal devolver información (páginas) cuando recibe peticiones por parte de los usuarios. En otras palabras, es el software que permite que los usuarios que quieren ver una página web en su navegador puedan hacerlo. [24]

d. Seguridad

Para aplicar seguridad al entorno web, se ha aplicado nginx como proxy inverso junto con LetsEncrypt, para aplicar el certificado SSL.

Nginx es un servidor web/proxy inverso ligero de alto rendimiento. [25]

Ilustración 26 - Nginx

Let's Encrypt es una autoridad de certificación que se puso en marcha el 12 de abril de 2016 y que proporciona certificados X.509 gratuitos para el cifrado de Seguridad de nivel de transporte (TLS) a través de un proceso automatizado diseñado para eliminar el complejo proceso actual de creación manual, la validación, firma, instalación y renovación de los certificados de sitios web seguros.[26]

Ilustración 27 - Let's Encrypt

Certbot es cliente ACME. Éste puede automatizar la emisión e instalación de certificados con cero tiempos de inactividad. [27]

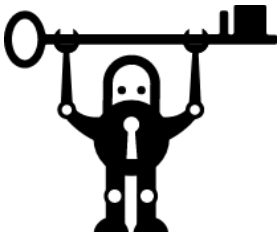


Ilustración 28 - Certbot

3. AWS

Para adecuar el sistema al entorno nube se ha utilizado Amazon Web Services. Para tomar esta decisión, se ha realizado una comparación previa entre el entorno nube Azure y AWS. Esta comparativa, se ha explicado en el anexo. [28]

Ilustración 29 - AWS

Descripción del desarrollo

Para el desarrollo del proyecto, se han definido objetivos individuales. Estos objetivos se han utilizado como guía para dividir el proyecto en fases independientes, de manera que se han ido realizando individualmente e integrando conforme se iban terminando.

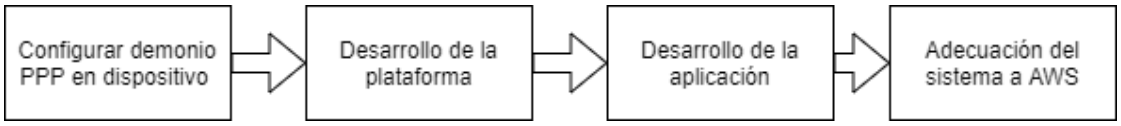


Ilustración 30 - Desarrollo por bloques

Configuración del demonio PPP en el dispositivo Quectel ec25-e

Para realizar la comunicación y el envío de datos de los dispositivos IoT, se planteó el uso de un usb Quectel ec25-e con tarjeta SIM de Movistar y conectado a un modem LTE.

LTE responde a las siglas Long Term Evolution (evolución a largo plazo) y hace referencia a la tecnología de banda ancha inalámbrica que sirve para la transmisión de datos con la finalidad de dar acceso a Internet a los dispositivos móviles. [29]

Para ello, era necesaria la configuración del demonio pppd y la VPN de la empresa en el dispositivo usb.

Pppd es el nombre que se le da a varios programas diferentes y de autores varios, pero que tienen en común el hecho de funcionar como agente intermediario o demonio del protocolo de comunicación PPP, en distintas versiones del sistema operativo UNIX.

El demonio pppd se encarga sobre todo de la traducción de los paquetes de información provistos por el stack de protocolos IP en paquetes aceptables por el estándar PPP, pero también suele dar soporte al control de la sesión PPP.[30]

El protocolo punto a punto o PPP es un protocolo de nivel de enlace de datos, utilizado para establecer una conexión directa entre dos nodos de una red.[31]

Por otro lado, una conexión VPN (Virtual Private Network) es una tecnología de red que sirve para conectar una o más computadoras a una red privada utilizando como medio una red pública como internet, es decir, la Red Privada Virtual permite que estos dispositivos estén conectados entre sí a través de internet de una forma segura, la cual garantiza la integridad y la confidencialidad de la información que se encuentra en dichos dispositivos. [32]

Para facilitar esta configuración, se conectó a una Raspberry Pi, como la que se muestra en la imagen.

Ilustración 30 - usb Quectel conectado a Raspberry Pi

Tras conectarse al sistema operativo Linux de la Raspberry Pi, mediante el cliente SSH Putty se han realizado los siguientes pasos.

1. Crear los scripts necesarios

Una vez conectado el usb a la Raspberry, en el sistema de ésta hay cuatro puertos USB activos, los cuales se pueden visualizar en la dirección /dev/ del sistema. En este caso, el puerto activo y donde se encuentra el flujo activo, es el puerto /dev/ttyUSB3, el cual utilizaremos para configurar el usb.

A continuación, se realiza la configuración del demonio, mediante la creación de los scripts. Para ello es necesario la instalación del paquete ppp en el sistema.

Los scripts que deben crearse son tres: quectel-ppp, quectel-chat-connect y quectel-chat-disconnect. Estos scripts han sido extraídos de la documentación oficial de Quectel ec25-e [33], y su contenido es el siguiente:

Archivo quectel-ppp:

```
/dev/ttyUSB3 115200
user MOVISTAR
password MOVISTAR
# The chat script, customize your APN in this file
connect 'chat -s -v -f /etc/ppp/peers/quectel-chat-connect -T movistar.es'
# The close script
disconnect 'chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect'
# Hide password in debug messages
hide-password
# The phone is not required to authenticate
noauth
# Debug info from pppd
debug
# If you want to use the HSDPA link as your gateway
defaultroute
# pppd must not propose any IP address to the peer
noipdefault
# No ppp compression
novj
novjccomp
noccp
ipcp-accept-local
ipcp-accept-remote
local
# For sanity, keep a lock on the serial line
#lock
#modem
#dump
#nodetach
# Hardware flow control
nocrtscts
remotename 3gppp
ipparam 3gppp
ipcp-max-failure 30
# Ask the peer for up to 2 DNS server addresses
usepeerdns
```

Ilustración 31 - Archivo quectel-ppp

Archivo quectel-chat-connect:

```
ABORT "BUSY"
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
TIMEOUT 30
"" AT
OK ATE0
#OK AT+CPIN=4491
OK ATI;+CSUB;+CSQ;+CPIN?;+COPS?;+CGREG?;&D2
# Insert the APN provided by your network operator, default apn is movistar.es
OK AT+CGDCONT=1,"IP","\T",,0,0
OK ATD*99#
CONNECT
```

Ilustración 32 - Archivo quectel-chat-connect

Archivo quectel-chat-disconnect:

```
ABORT "ERROR"
ABORT "NO DIALTONE"
SAY "\nSending break to the modem\n"
""  +++
""  +++
""  +++
SAY "\nGoodbay\n"
```

Ilustración 33 - Archivo quectel-chat-disconnect

Para la creación de estos scripts, es importante conocer la siguiente información:

- En qué tty está el dispositivo activo: /dev/ttyUSB3
- Cuál es el APN (Nombre del Punto de Acceso) de la tarjeta: movistar.es
- Usuario y contraseña del APN: MOVISTAR MOVISTAR
- PIN de la tarjeta. En este caso, se decidió desactivar el PIN

2. Lanzar los scripts

Una vez los scripts están preparados, se lanzan mediante el comando "sudo pppd call". Para verificar que ha sido lanzado correctamente, se debe crear la interfaz ppp0. Se puede encontrar escribiendo los comandos ifconfig o route –n en la terminal:

```
ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet [redacted]  netmask 255.255.255.255  destination [redacted]
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 15  bytes 584 (584.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 58 (58.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 34 - ppp0

3. Automatizar el proceso con systemctl

Aunque el procedimiento para crear el demonio es el que ya se ha explicado, en este caso, es necesario automatizar este proceso, de manera que cada vez que la máquina se inicie o haya algún error, se lancen los scripts que recién hemos creado.

Systemd es un administrador del sistema que facilita la administración de servicios. Permite el inicio de demonios bajo demanda, realiza un seguimiento de los procesos con el uso de los grupos de control de Linux. [34]

Para ello creamos un archivo en la localización "/etc/systemd/system/pppd-call.service" con el siguiente contenido:

```
[Unit]
Description=i3i - pppd call service

[Service]
Type=simple
Restart=always
RestartSec=10s
User=pi
ExecStart=/bin/bash /home/pi/ejecuta_pppd.sh

[Install]
WantedBy=multi-user.target
```

Ilustración 35 - Archivo pppd-call.service

Por otro lado, para la configuración de la VPN se realizaron los siguientes pasos:

1. Crear script

Para configurar la VPN en el dispositivo, se crea un script con el nombre que se desee en la dirección /etc/ppp/peers/nombre_script.

En este caso, la VPN descrita pertenece a la empresa i3i y el contenido del script lo proporcionó la empresa en cuestión. El contenido de este es el siguiente:

```
pi@raspberrypi:~ $ sudo cat /etc/ppp/peers/i3i-vpn
pty "pptp [redacted]  --nolaunchpppd --debug"
name [redacted]
password [redacted]
remotename PPTP
require-mppe-128
require-mschap-v2
refuse-eap
refuse-pap
refuse-chap
refuse-mschap
noauth
debug
persist
maxfail 0
defaultroute
replacedefaultroute
usepeerdns
```

Ilustración 36 - Archivo de configuración de la vpn

2. Lanzar script

Una vez el script está preparado, se lanzan mediante el comando "pon nombre_script".

Para verificar que ha sido lanzado correctamente, se debe crear la interfaz ppp1. Se puede encontrar escribiendo los comandos ifconfig o route –n en la terminal:

```
ppp1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1396
        inet [redacted]  netmask 255.255.255.255  destination [redacted]
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 7  bytes 94 (94.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26  bytes 878 (878.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 37 - ppp1

3. Automatizar proceso

Por último, igual que el lanzamiento del demonio ppp, también se desea automatizar el proceso de creación de la VPN. Para ello se introduce el comando anterior, en el archivo /etc/rc.local.

El archivo /etc/rc.local nos permite ejecutar un comando o script cada vez que se inicia un sistema tipo Unix/Linux.

Todo comando o script que ejecutemos por medio de rc.local será ejecutado tras el arranque, una vez se hayan ejecutado los comando o scripts del runlevel o nivel de ejecución con el que arranca el sistema. [35]

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

#Establish PP2P VPN Connection
#printf "Establishing VPN connection through ppp0\n"
pon i3i-vpn persist updetach

exit 0
```

Ilustración 38 - Archivo rc.local

Finalmente, esta idea fue descartada por la empresa ya que han encontrado soluciones más adecuadas para abordar la conexión de los dispositivos.

Desarrollo de la plataforma

Para automatizar el proceso de recogida de datos de los contenedores, los dispositivos IoT se conectan a la plataforma descrita en el apartado (Diseño - Arquitectura del sistema – Arquitectura de la plataforma).

La construcción de este entorno se ha realizado mediante un archivo docker-compose.yml.

Un archivo Docker-compose.yml, creado para definir todos los servicios necesarios y ponerlos en marcha con único comando, se compone de la siguiente estructura:

- Versión: Primero se indica la versión del esquema. Lo más correcto es utilizar la última versión.
- Services: En la sección services, se indican todos los servicios que componen el entorno que se desea poner en marcha.
- Volumes: Se deben definir la asignación de volúmenes. La función de los volúmenes es lograr la persistencia y almacenamiento de datos.
- Networks: Se indican las redes a las que se unirán los contenedores, referenciando las entradas debajo de las redes de nivel superior.

En el archivo, se ha indicado la versión, en este caso versión: '3', y los siguientes servicios:

```
mongo:
  image: mongo:3.2
  command: --nojournal
  hostname: mongo
  ports:
    - "27017:27017"
  volumes:
    - "mongodata:/data/db"
```

Ilustración 39 -Docker mongo

La imagen utilizada es mongo con versión 3.2. Se ha optado por la versión 3.2 ya que es la recomendación que se hace en la documentación de FIWARE-ORION. Concretamente, se mencionan como versiones compatibles las 2.6/3.0/3.2.

```
# Orion Context Brocker
orion:
  image: fiware/orion
  links:
    - mongo
  hostname: orion
  container_name: fiware-orion
  depends_on:
    - mongo
  ports:
    - "1026:1026"
  command: -dbhost mongo -writeConcern 0 -logLevel ERROR -disableMetrics
```

Ilustración 40 - Docker orion

La imagen escogida para Orion ha sido fiware/orion.
Se han definido las dependencias con mongo.
Utiliza el puerto 1026, porque así se define en la documentación FIWARE-ORION

```
# iot-agent
iot-agent:
  image: telefonicaiot/iotagent-ul
  hostname: iot-agent
  depends_on:
    - mongo
    - orion
  ports:
    - "4041:4041"
    - "4061:4061"
    - "7899:7896"
  environment:
    - "IOTA_CB_HOST=orion"
    - "IOTA_CB_PORT=1026"
    - "IOTA_NORTH_PORT=4061"
    - "IOTA_REGISTRY_TYPE=mongodb"
    - "IOTA_MONGO_HOST=mongo"
    - "IOTA_MONGO_PORT=27017"
    - "IOTA_MONGO_DB=iotagent-ul"
    - "IOTA_HTTP_PORT=7896"
    - "IOTA_PROVIDER_URL=http://iot-agent:4061"
    - "IOTA_CB_NGSI_VERSION=v2"
    - "IOTA_AUTOCAST=true"
```

Ilustración 41 - Docker iot-agent

La imagen seleccionada para iot-agent es telefonicaiot/iotagent-ul.
Se han definido las dependencias con mongo y orion.

Los puesto que utiliza don 4041, 4061 y 7896.

Se usan los puertos mencionados, porque así se definen en la documentación FIWARE-ORION.

Sin embargo, para que no haya colisión con otros proyectos de la empresa, se ha modificado el puerto interno 7896 por el puerto 7899.

```
# quantumleap timescale postgres
quantumleap:
  image: smartsdk/quantumleap
  hostname: quantumleap
  container_name: fiware-quantumleap
  ports:
    - "8668:8668"
  depends_on:
    - mongo
    - orion
    - postgres
    - timescaledb
  environment:
    - QL_DEFAULT_DB=timescale
    - POSTGRES_HOST=postgres
    - POSTGRES_PORT=5432
    - POSTGRES_DB_NAME=postgres
    - POSTGRES_DB_USER=[redacted]
    - POSTGRES_DB_PASS=[redacted]
```

Ilustración 42- Docker quantumleap

La imagen utilizada para quantumleap ha sido la de smartsdk/quantumleap.

Los puertos que utiliza son 8668:8668.

Depende de mongo, orion, postgresql y timescale.

Se definen parámetros referentes a la base de datos como, la base de datos por defecto timescale, el puerto de la base de datos 5432 etc.

```
# Postgres Database
postgres:
  image: postgres:10.5
  hostname: postgres
  container_name: db-postgres
  restart: always
  ports:
    # Admin UI
    - "4200:4200"
    # Transport protocol
    - "4300:4300"
    - "5432:5432"
  environment:
    POSTGRES_HOST: postgres
    POSTGRES_DB_NAME: postgres
    POSTGRES_DB_USER: [redacted]
    POSTGRES_DB_PASS: [redacted]

  volumes:
    - "/home/leire_tfg/tfg/postgre:/data"
```

Ilustración 43 - Docker PostgreSQL

La imagen seleccionada para postgreSQL ha sido postgres con la versión 10.5.

Los puertos que utiliza son 4200, 4300 y 5432.

Además, se han definido los nombres y credenciales de la base de datos que se utiliza.

```
#Timeserie tables
timescaledb:
  image: timescale/timescaledb:1.7.0-pg10
  hostname: timescaledb
  container_name: db-timescale
  depends_on:
    - postgres
  environment:
    POSTGRES_USER: [redacted]
    POSTGRES_PASSWORD: [redacted]

  volumes:
    - "/home/leire_tfg/tfg/timescale:/data"
```

Ilustración 44 - Docker Timescale

La imagen utilizada para tiemscaledb ha sido timescaledb con versión 1.7.0 haciendo referencia a la versión 10 de postgresql

Timescale depende de postgres y se han definido las credenciales, iguales que la de postgresql.

Como se ha comentado previamente, aunque el almacenamiento se desea realizar en la base de datos temporal Timescaledb, ha sido necesaria la definición del servicio PostgreSQL.

La documentación oficial comunica que para utilizar Timescale, es necesaria la instalación de PostgreSQL, específicamente, QuantumLeap requiere una versión 10 o superior de PostgreSQL, con la extensión de Timescale. [36]

```
grafana:
  build:
    context: ./grafana
  container_name: grafana
  depeds_on:
    - timescaledb
  ports:
    - 3000:3000
  volumes:
    -"/home/leire_tfg/tfg/grafana:/data/grafana"
```

Ilustración 45 - Docker Grafana

La imagen que se utiliza para Grafana, viene definida en el archivo Dockerfile.

Depende de Timescaledb y utiliza el puerto 3000.

Además, se ha decidido eliminar el control de acceso a la plataforma, para facilitar a los usuarios su uso.

```
FROM grafana/grafana:7.1.5-ubuntu

# Disable Login form or not
ENV GF_AUTH_DISABLE_LOGIN_FORM "true"
# Allow anonymous authentication or not
ENV GF_AUTH_ANONYMOUS_ENABLED "true"
# Role of anonymous user
ENV GF_AUTH_ANONYMOUS_ORG_ROLE "Admin"
```

Ilustración 46- Dockerfile Grafana

```
networks:
    default:

volumes:
  mongodata:
  postgres:
  timescale:
  grafana:
```

Ilustración 47 - Docker network y volumes

Se utiliza la configuración por defecto de las networks.

En la sección volumen, se han indicado todos los volúmenes que se utilizan en los servicios.

Una vez el entorno está montado y operativo mediante el comando “docker-compose up -d”, se procede a automatizar la carga de datos mediante la plataforma definida.

Para ello se crean los dispositivos que hacen referencia a los que se van a instalar en los contenedores. Los pasos para crearlos son los siguientes:

1. Creación de un servicio:

Los servicios, sirven para agrupar un conjunto de dispositivos. Su creación se realiza mediante una petición POST a iot-agent:

```
###
Send Request
POST http://{{host}}:4061/iot/services HTTP/1.1
content-type: application/json
fiware-service: [redacted]_demo
fiware-servicepath: /

{
    "services": [
        {
            "apikey": "6abmuu6q235pqt7um7r4",
            "cbrocker": "http://orion:1026",
            "entity_type": "Thing",
            "resource": "/iot/d"
        }
    ]
}
```

Ilustración 48 - Definición de servicio

Para definir este servicio, se indican los siguientes parámetros:

Apykey: es un string aleatorio que servirá para identificar el servicio.

Cbroker: indicamos cual es el context broker que utilizaremos. En este caso, orion.

Entity type: definimos que tipo de objetos se incluirán. Existen tipos definidos como, por ejemplo, lamp. En este caso, como no existen nombres que definan los objetos, utilizaremos Thing.

2. Definición del objeto:

A continuación, se definen los objetos, mediante el modelo de datos previamente explicado NGSI-LD.

En esta petición, se realiza un POST al componente iot-agent, para la definición de un dispositivo, en este caso un sensor.

Para definir el dispositivo, se indican los siguientes parámetros:

Device id: el identificador del dispositivo.
Entity name: nombre del dispositivo.
Entity type: Tipo del dispositivo.
Attributes: atributos dinámicos del dispositivo.
Static attributes: atributos estáticos del dispositivo.

```
###
Send Request
POST http://{{host}}:4061/iot/devices HTTP/1.1
content-type: application/json
fiware-service: [redacted]
fiware-servicepath: /

{
    "devices": [
        {
            "device_id":   "sensor003",
            "entity_name": "urn:ngsi-ld:sensor:003",
            "entity_type": "Thing",
            "timezone":    "Europe/Madrid",
            "attributes": [
                { "object_id": "ir", "name": "ir", "type": "Integer" },
                { "object_id": "distancia", "name": "distancia", "type": "Number" },
                { "object_id": "temperatura", "name": "temperatura", "type": "Number" },
                { "object_id": "voltaje_pila", "name": "voltaje_pila", "type": "Number" },
                { "object_id": "switch_magnetico", "name": "switch_magnetico", "type": "Integer" },
                { "object_id": "tilt", "name": "tilt", "type": "Integer" },
                { "object_id": "tiempo_despierto", "name": "tiempo_despierto", "type": "Integer" },
                { "object_id": "missfire", "name": "missfire", "type": "Integer" },
                { "object_id": "timestamp_sensor", "name": "timestamp_sensor", "type": "DateTime" }
            ],
            "static_attributes": [
                { "name": "tipo_contenedor", "type": "Integer", "value": "0"},
                { "name": "numero_contenedor", "type": "Integer", "value": "3"}
            ]
        }
    ]
}
```

Ilustración 49 - Definición de objeto

3. Provisionamiento del dispositivo:

El provisionamiento de un dispositivo consiste en almacenar los datos enviados por un dispositivo. Como todavía no hay una instalación real de los dispositivos, simulamos la entrada de datos, introduciéndolos manualmente mediante una petición.

Para realizar el provisionamiento, previamente hay que definir una suscripción. El mecanismo de suscripción de Orion Context Broker funciona de tal manera que cada vez determina cuando ha ocurrido un cambio en la información y notifica dicho cambio.

```
Send Request
POST http://{{host}}:1026/v2/subscriptions HTTP/1.1
content-type: application/json
fiware-service: [redacted]
fiware-servicepath: /

{
    "description": "Suscripcion",
    "subject": {
        "entities": [{ "idPattern": ".*" }]

    },
    "notification": {
        "http": { "url": "http://quantumleap:8668/v2/notify" },
        "metadata": ["dateCreated", "dateModified"]

    }
}
```

Ilustración 50 - Definición de subscripción

La suscripción, se crea mediante una petición post a orion.

La definición de esta suscripción, indica que cada vez que haya un cambio en cualquier entidad, se envíe una notificación a quantumleap.

Para introducir datos en la base de datos, se realiza una petición post a iot-agent, en la que se indican el valor de todos los datos dinámicos del dispositivo del cual vamos a insertar datos.

Ilustración 51 - Provisionamiento de objeto

4. Comprobar respuestas correctas:

Para comprobar que este proceso es correcto, por cada petición mencionada, se debe recibir una respuesta cuya cabecera debe indicar que ha sido correcta. Normalmente, mediante el código numérico 200, como se visualiza en la siguiente imagen:

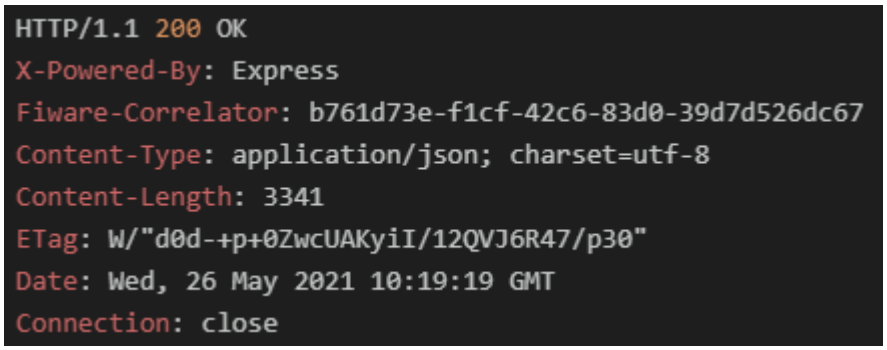


Ilustración 52 - Mensaje 200

Una vez terminado este procedimiento, se crean las bases de datos mencionadas en el apartado "3. Diseño – Modelo de datos" del documento. Estas se estructuran de la siguiente manera:

Por un lado, **MongoDB** almacena la estructura y atributos de los dispositivos creados:

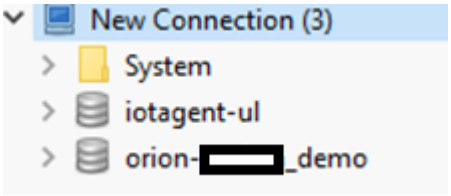


Ilustración 53 - Bases de datos de mongodb

Al poner en marcha la plataforma, se generan automáticamente dos bases de datos, iotagent-ul y orion-xxxx_demo. En este proyecto, solo se ha hecho uso de la base de datos iotagent-ul.
Esta base de datos ha sido fundamental para realizar las modificaciones de los atributos estáticos.

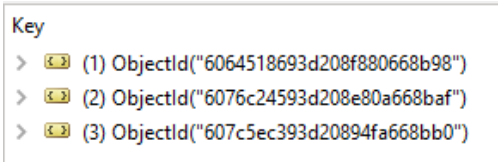


Ilustración 54 - Objetos almacenados en mongodb

Por cada objeto creado, se genera una estructura en la base de datos mencionada.

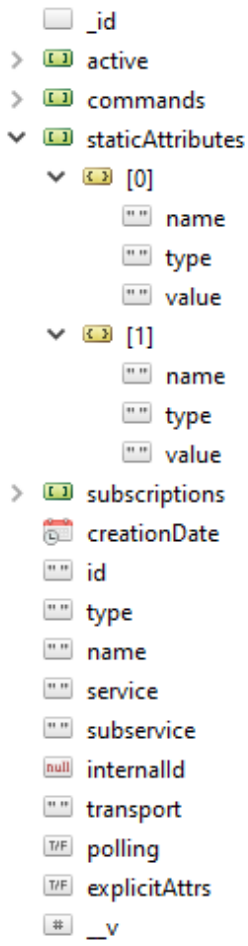


Ilustración 55 - Estructura de objeto almacenada en mongodb

La estructura del objeto que se almacena es la que se visualiza en la imagen.

En esta estructura se destacan los atributos almacenados y el id del objeto.

La división que almacena mongoDB entre los atributos estáticos y dinámicos es de gran utilidad para el proyecto.

Concretamente, para ofrecer al usuario la posibilidad de modificar los atributos estáticos mediante la aplicación, se van a extraer y modificar de esta estructura.

Por otro lado, en las bases de datos **PostgreSQL** y **TimeScale**, se almacenan todos los datos que envían los objetos creados.

Esta estructura, también es generada automáticamente por Orion Context Broker. Concretamente, la primera vez que los objetos envían datos, se genera una estructura similar a la que se visualiza en la siguiente imagen.

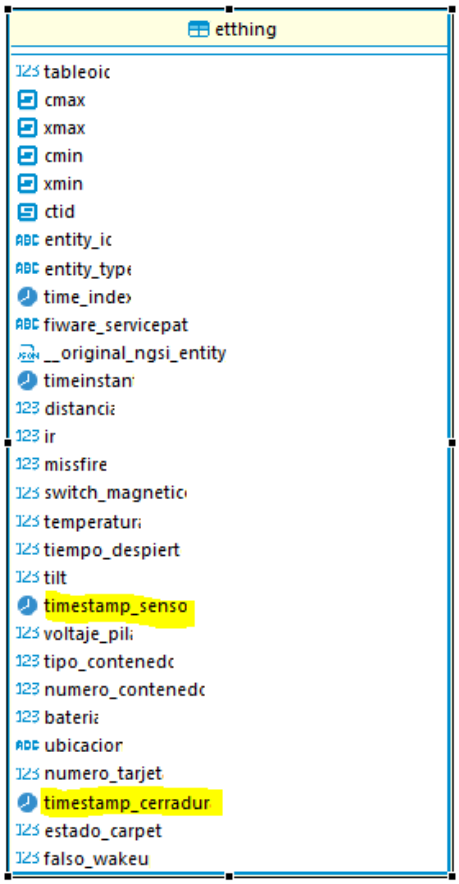


Ilustración 56 - Estructura de objeto almacenada en timescale

En la tabla generada, destaca el nombre "etthing", el cual se genera automáticamente al indicarle que el tipo de objeto es Thing.

Por otro lado, es llamativo que, en la misma tabla, se almacenan los datos de distintos objetos.

Esto es así, porque se agrupan en el mismo servicio, pero como los nombres de los atributos son diferentes por cada tipo de dispositivo, no produce ningún problema.

Por último, también es posible acceder a la herramienta Grafana mediante el puerto 3000.

Es muy sencillo configurar esta herramienta, para visualizar datos recogidos en la base de datos temporal Timescale.

Primero se crea una conexión a la base de datos, Timescale-PostgreSQL:

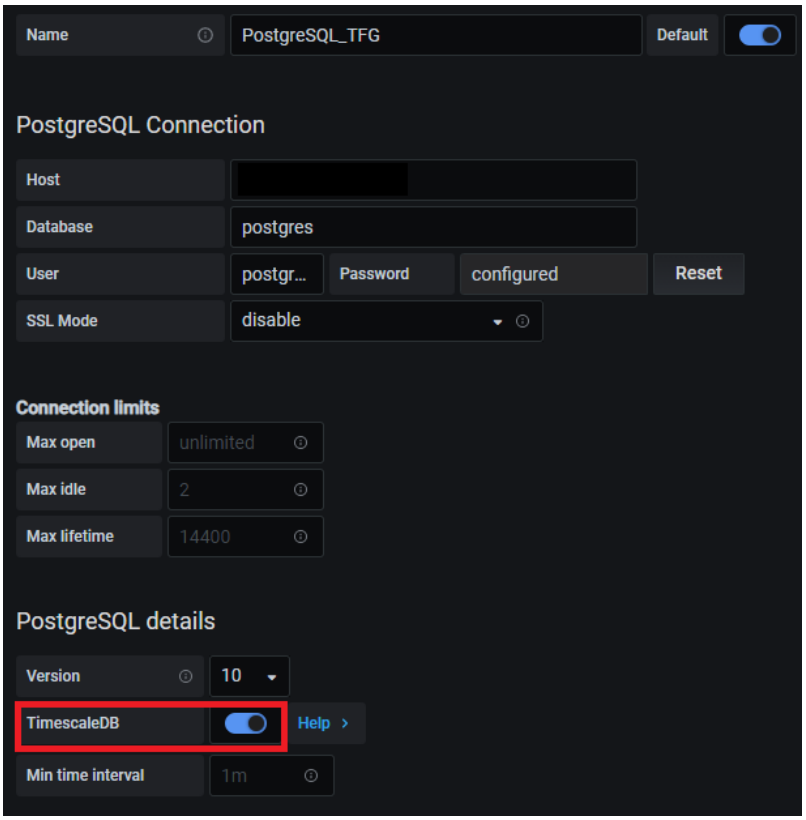


Ilustración 57 - Conexión a base de datos en Grafana

Una vez definida la conexión, se pueden realizar diferentes consultas para mostrar gráficas. Como las que se muestran a continuación:

Mediante esta consulta se quiere mostrar el atributo distancia del sensor 002.

```
SELECT
  time_index AS "time",
  distancia
FROM mt[redacted]_demo.etthing
WHERE entity_id='urn:ngsi-ld:sensor:002'
```

Ilustración 58 - Consulta en Grafana

Además, existe la opción de definir una alerta, de manera que, puede facilitar el control de los datos. Por ejemplo, si el último valor es mayor que 1, notifica visualmente que ha saltado la alerta.

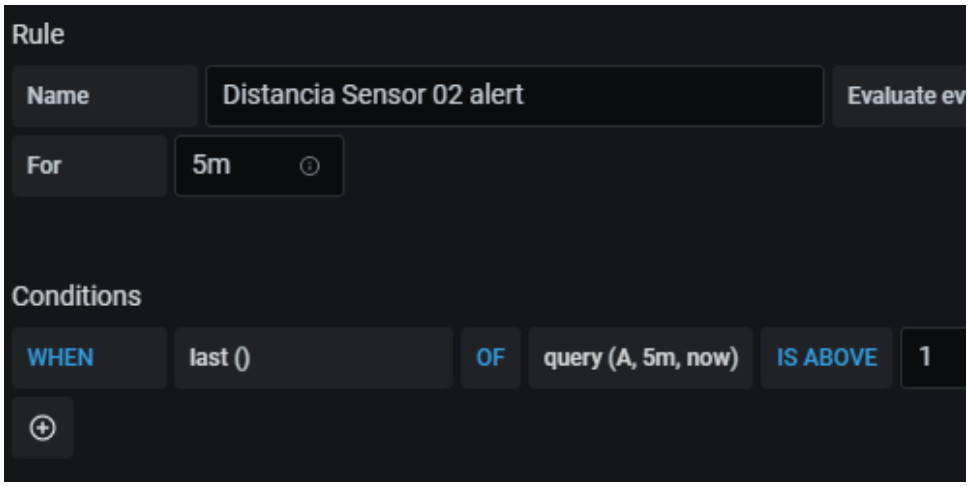


Ilustración 59 - Alerta en Grafana

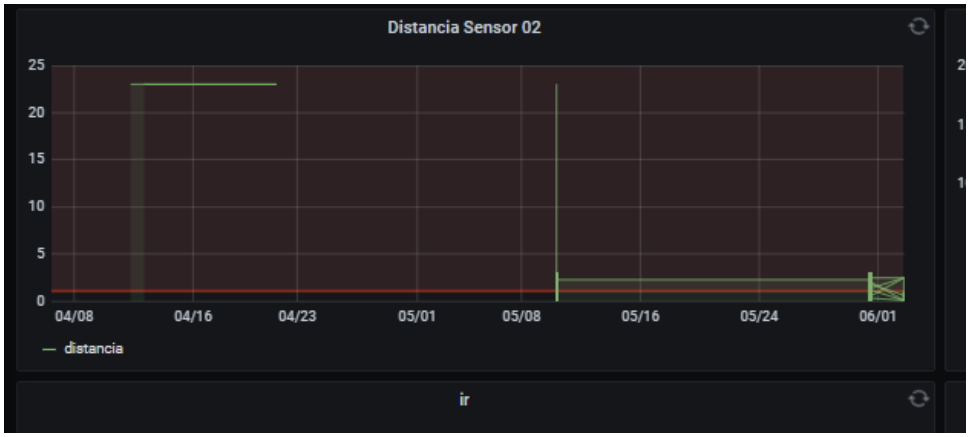


Ilustración 60 - Grafica Grafana

Desarrollo de la aplicación

Una vez terminada la estructura de la plataforma, es indispensable la creación de una aplicación web con el propósito de permitir el acceso a la modificación y visualización de los atributos estáticos, así como facilitar el acceso a Grafana.

A continuación, se explica el procedimiento seguido para el desarrollo de la aplicación dividido en dos secciones. En la primera se expone la creación del entorno web y la programación back-end y front-end llevada a cabo. En la segunda, se expone los pasos seguidos para securizar el entono mediante el certificado SSL.

a. Entorno web

Para montar el entorno de desarrollo web, se han utilizado archivos docker y docker-compose.yml. El contenido de dichos archivos es el siguiente:

```yaml
web:
  #image: php:7.4-apache
  build:
    context: .
    dockerfile: ./Dockerfile
  user: '1000:1000'
  ports:
    - "80:80"
  environment:
    APACHE_RUN_USER: '#1000'
    APACHE_RUN_GROUP: '#1000'
  volumes:
    - ./html:/var/www/html
```

Ilustración 61 - Docker apache

Para construir el servicio web que define el servidor web HTTP Apache, se ha utilizado un el fichero Dockerfile.

Los puertos definidos son 14080:80. La selección del puerto 14080, es aleatoria.

En el archivo Dockerfile, definido en la siguiente imagen, se indica la versión del servidor apache a levantar y la instalación de recursos adicionales necesarios para utilizarlos en el desarrollo php.

Por un lado, para conexiones orientadas a objetos con postgreSQL, se ha instalado "pdo pdo_pgsql"

Además, se requieren conexiones con mongodb, para ello se ha instalado una extensión de mongo.

```
FROM php:8.0.3-apache
RUN apt-get update && apt-get install -y libpq-dev && docker-php-ext-install pdo pdo_pgsql && docker-php-ext-install pgsql pdo pdo_pgsql
RUN pecl channel-update pecl.php.net \
    && pecl install mongodb \
    && cp /usr/local/etc/php/php.ini-production /usr/local/etc/php/php.ini \
    && echo "extension=mongodb.so" >> /usr/local/etc/php/php.ini \
    && docker-php-ext-enable mongodb
```

Ilustración 62 - Dockerfile apache

```yaml
db:
  image: postgres:9.6.1
  ports:
    - "5433:5432"
  environment:
    POSTGRES_PASSWORD: dbpass
    POSTGRES_DB: plataforma_db
```

Ilustración 63 - Docker PostgreSQL

Para almacenar información de los usuarios, se necesita crear una nueva base de datos con nombre "plataforma_db" en el sistema de gestión de bases de datos postgreSQL, versión 9.

Para evitar que los puestos colisionen con los definidos en la plataforma, se define el puerto externo como 5433.

```yaml
volumes:
  html: {}
```

Ilustración 64 - Docker volúmenes

Se indican los volúmenes utilizados en los servicios definidos.

Después de preparar y levantar el entorno de desarrollo, es sencillo comenzar a programar y visualizar los resultados.

Como se ha comentado en el apartado "3. Diseño – Arquitectura del software", la parte backend de la aplicación web desarrollada se ha estructurado mediante el patrón de diseño Modelo-Vista-Controlador.

Para ello, se ha dividido la programación en 3 partes generales.

- El **Controlador**, está compuesto por las clases coger_parametro.php e index.php. Estas clases recogen las peticiones que se realizan en la aplicación, y ejecutan diferentes funciones definidas en las partes Modelo y Vista.

- El **Modelo** por su parte, está compuesto por las clases Usuario.php, Variable.php, Connection.php y Connection2.php

  Usuario.php recoge aquellas funciones que afectan a la base de datos de los usuarios.

  Como aplicación de seguridad, las contraseñas almacenadas en la base de datos PostgreSQL, se han guardado encriptadas.

  Para ello, se ha creado una extensión pgcrypto, de manera que, al realizar las inserciones para crear los usuarios, se ha hecho uso de la función "crypt ('password', gen_salt('bf'))". [37]

  Pgcrypto es un módulo que provee de funciones criptográficas a PostgreSQL. En la función crypt, la cual realiza la encriptación, además de indicar la contraseña, se debe indicar gen_salt, el cual indica que algoritmo se utiliza. En este caso, el algoritmo seleccionado ha sido 'bf'.

  La especificación descrita en la documentación de PostgreSQL del algoritmo es la siguiente:

Algorithm	Max password length	Adaptive?	Salt bits	Description
bf	72	yes	128	Blowfish-based, variant 2a

Ilustración 65 - Algoritmo bf

  Para realizar la comprobación de la contraseña en el código php, se ha realizado la siguiente comparativa:

```
if (hash_equals($datos->contrasena, crypt($password, $datos->contrasena))){
```

Ilustración 66 - Comparación de contraseñas

  Si la contraseña introducida encriptada es igual que la contraseña encriptada almacenada en la base de datos, devuelve TRUE.

  Variable.php recoge aquellas funciones que afectan a los dispositivos y atributos que se muestran en la aplicación.

  En esta clase, se deben destacar las funciones recogerdatos() y modificar_atributos().

La función recogerdatos(), realiza una petición al objeto iot agent. Para facilitar la creación de esta petición en lenguaje php, el editor Visual Studio Code, ofrece una transformación automática de las peticiones mostradas anteriormente, al lenguaje que se desee.

Después de realizar esta transformación, el resultado es el siguiente:

```php
public static function recogerdatos(){
    $device = parametro::coger_parametro("device");
    $curl = curl_init();
    curl_setopt_array($curl, [
        CURLOPT_PORT => "4061",
        CURLOPT_URL => "http://[redacted]:4061/iot/devices",
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_ENCODING => "",
        CURLOPT_MAXREDIRS => 10,
        CURLOPT_TIMEOUT => 30,
        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
        CURLOPT_CUSTOMREQUEST => "GET",
        CURLOPT_HTTPHEADER => [
            "content-type: application/json",
            "fiware-service: [redacted]_demo",
            "fiware-servicepath: /",
            "user-agent: vscode-restclient"
        ],
    ]);
    $response = curl_exec($curl);
    $err = curl_error($curl);
    curl_close($curl);
    if ($err) {
        echo "CURL Error #:" . $err;
    }
    $obj=json_decode($response);
    return $obj;
}
```

Ilustración 67 - Función recogerdatos()

La función modificar_atributos() realiza la modificación del valor de un atributo estático.

En esta función destaca el uso del driver y la función utilizada para realizar esta operación. El driver utilizado y definido en el Dockerfile es MongoDB\Driver. Para la modificación se ha hecho uso de la funcion BulkWrite cuyo uso es el siguiente:

```php
$bulk = new MongoDB\Driver\BulkWrite;
foreach ($rows as $row) {
    if ($row->id==$deviceid){
        $bulk->update(
            ['$and'=>[['staticAttributes.name'=>$variable], ['id'=>$deviceid]]],
            ['$set'=>['staticAttributes.$.value'=>$valor]]
        );
        $result = $conn->executeBulkWrite('iotagent-ul.devices',$bulk);
        $resultado = ($result->getModifiedCount());
    }
}
return $resultado;
```

Ilustración 68 - Update mongo en php

Las clases Connection.php y Connection2.php contienen las conexiones a las bases de datos MongoDB que contiene la información de los dispositivos y PostgreSQL que almacena la información de los usuarios respectivamente.

La conexión a PostgreSQL se realiza orientada a objetos mediante la librería instalada pdo_postgres.

La conexión a mongoDB se realiza mediante el driver y la función "MongoDB\Driver\Manager("mongodb://ip")".

- La **Vista** por otro lado recoge las clases Vista.php y TranformView.php, así como el contenido de las plantillas utilizadas para la presentación del entorno.

Para realizar la parte frontend de la aplicación, se ha hecho uso de plantillas, con el fin de conseguir un diseño más profesional y agradable para el usuario. Las plantillas que se han utilizado son las que ofrece AdminLTE de manera gratuita y libre de uso. [38]

La plantilla utilizada "AdminLTE Bootstrap Admin Dashboard" tiene el diseño completo de un entorno y se ha adaptado para la aplicación de gestión desarrollada.

Concretamente, han sido de gran utilidad la aplicación de las plantillas del menú principal, las tablas y el iframe.

Ilustración 69 - Página principal de la aplicación

b. Aplicación del certificado SSL

Se ha decidido aplicar un estándar de seguridad al sitio web desarrollado mediante la aplicación de un certificado ssl, para que la transferencia de datos sea segura. Mediante el procedimiento que se describe más adelante, se consigue implementar el protocolo https, el cual protege la identidad y la confidencialidad de los datos de los usuarios.

Ilustración 70 - Diagrama de proxy inverso

Como capa de seguridad, delante de la imagen apache levantada mediante docker, se ha aplicado nginx como proxy inverso. Un proxy inverso se ubica frente a un servidor web y recibe todas las solicitudes antes de que lleguen al servidor origen.

El procedimiento ha sido el siguiente:

1. Apuntar DNS a dirección IP.

En este caso, mediante DynDns se ha asociado un DNS con nombre “gestion-rsu.dyndns.org” a la dirección ip de la máquina en la que corre el entorno web. DynDns es una compañía dedicada a las soluciones de DNS en direcciones IP. [39]

2. Crear certificados Certbot

Para crear los certificados ssl, se utiliza Certbot, junto con LetsEncrypt. Para descargarlos, se utiliza el siguiente comando: “sudo certbot –nginx –d dns-de-sitio-web". Si todo ha ido correctamente, en la dirección /etc/letsencrypt/live/dns-de-sitio-web se deben haber creado los certificados necesarios. Concretamente se utilizarán los certificados “fullchain.pem” y “privkey.pem”.

3. Configuración de nginx

A continuación, después de instalar nginx, para configurarlo, es importante conocer dos carpetas situadas en las direcciones /etc/nginx/sites-available y /etc/nginx/sites-enabled.

La primera carpeta /sites-available/ es un repositorio para todos los sitios web que van a atravesar el proxy inverso nginx.

La segunda carpeta /sites-enabled/ contiene todos los links simbólicos de estos sitios web. Un enlace o link simbólico indica el acceso a un directorio o fichero que se encuentra en un lugar distinto dentro de la estructura de directorios.

Para definir la configuración de nginx, se crea un nuevo archivo en la carpeta /sites-available/ cuyo contenido es el siguiente:

```
server {
        root /var/www/html;
        index index.html index.html;
        server_name gestion-rsu.dyndns.org;

        location / {
                proxy_pass http://localhost:[redacted];
                proxy_set_header Host $host;
                #access_log /var/log/nginx/plataforma_access.log main;
                error_log /var/log/nginx/plataforma_error.log;
                #try_files $uri $uri/ =404;
        }

        #listen [::]:443 ssl ipv6only=on; #managed by Certbot
        listen [redacted] ssl; #managed by Certbot
        ssl_certificate /etc/letsencrypt/live/gestion-rsu.dyndns.org/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/gestion-rsu.dyndns.org/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server{
        if ($host = gestion-rsu.dyndns.org) {
                return 301 https://$host$request_uri;
        }

        listen [redacted];
        listen [::]:[redacted];
        server_name gestion-rsu.dyndns.org;
        location / {
                proxy_pass http://localhost:[redacted];
                proxy_set_header Host $host;
                #access_log /var/log/nginx/plataforma_access.log main;
                error_log /var/log/nginx/plataforma8990_error.log;
                #try_files $uri $uri/ =404;
        }
        #access_log  /var/log/nginx/access.log  main;
        #return 404;
}
```

Ilustración 71 - Archivo de configuración nginx

Es importante definir dos servidores. El primero es el que contiene los certificados ssl de lets encrypt. El segundo es el que redirige todo el tráfico http a la versión encriptada de este. Se apunta a localhost:puerto, donde se encuentra corriendo el entorno web.

Para que el archivo funcione, se abren los puertos necesarios en el router de la empresa.

A continuación, se crea el link simbólico en la carpeta /sites-enabled/, con el siguiente comando: “sudo ln -s ../sites-available/nombre-de-archivo-configuracion dns-de-sitio-web”

4. Reiniciar nginx y probar

Una vez reiniciado el nginx, y teniendo el entorno web operativo, buscando el DNS mencionado se despliega el entorno web desarrollado, y además se visualiza el candado característico que indica que el sitio es seguro.

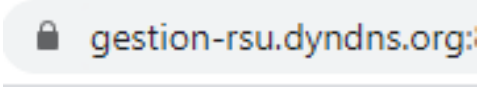


Ilustración 72 - url segura

Adecuar sistema a AWS

Para adecuar el sistema desarrollado a un entorno nube, se ha optado por AWS. Amazon Web Services, posee una gran variedad de servicios. Entre estos servicios, se han buscado aquellos que pueden aportar en el proyecto.

De todos los servicios se ha hecho uso de dos. Maquinas EC2 e IoT Core.

a. Servicio EC2 [40]

Por un lado, ya que el proyecto en su mayoría esta levantado por Docker, se ha buscado como aplicar esto mediante AWS. La solución que ofrece es la virtualización con ec2.

Amazon Elastic Compute Cloud es una parte central de la plataforma de cómputo en la nube de la empresa Amazon.com denominada Amazon Web Services. EC2 permite a los usuarios alquilar computadores virtuales en los cuales pueden ejecutar sus propias aplicaciones.

i. Para crear una de estas máquinas, se realiza el siguiente procedimiento:

Una vez se selecciona la opción de generar una nueva instancia, se selecciona la imagen que se desea crear. Estas imágenes se diferencian por el sistema operativo y demás propiedades. En este caso, se optó por Ubuntu.

A continuación, se selecciona el tipo de instancia. AWS ofrece una versión gratuita.

Por último, está la posibilidad de modificar y ajustar ciertas propiedades como el almacenamiento.

Una vez creada la instancia, se deben descargar las claves para poder acceder a dicha instancia.

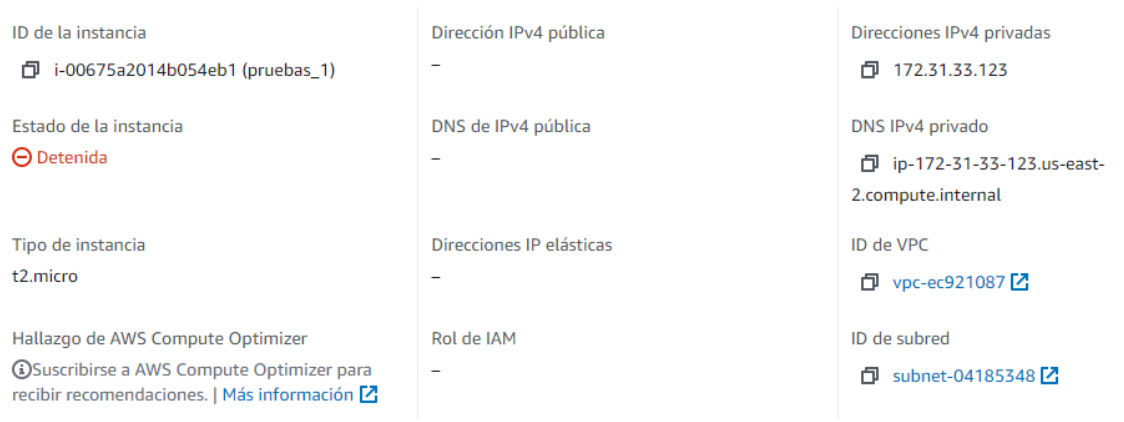


Ilustración 73 - Detalle de la maquina ec2

ii. Despliegue de la plataforma en la máquina ec2:

Una vez accedido a la máquina, se procede a instalar y levantar la plataforma mediante Docker. Para que el sistema funcione correctamente, se deben indicar los puertos:

Reglas de entrada (13)

Tipo	Protocolo	Intervalo de puertos
HTTP	TCP	80
HTTP	TCP	80
PostgreSQL	TCP	5432
TCP personalizado	TCP	8668
SSH	TCP	22
TCP personalizado	TCP	27017
TCP personalizado	TCP	7899
TCP personalizado	TCP	3000
HTTPS	TCP	443
HTTPS	TCP	443
TCP personalizado	TCP	5433
TCP personalizado	TCP	4061
TCP personalizado	TCP	1026

Ilustración 74 - Reglas de conexiones en AWS

Para realizar pruebas con este servicio, se ha repetido el proceso explicado en el desarrollo de la plataforma, acomodando la dirección ip, a la ip pública de la máquina ec2.

Ilustración 75 - Ejemplo de provisionamiento de un dispositivo mediante la plataforma levantada en una maquina ec2 con ip pública 3.15.20.44

Finalmente, después de probar este servicio, se considera que es una buena alternativa si no se posee un servidor. Sin embargo, al disponer de un servidor, no se ha avanzado en su uso.

b. Servicio IoT Core [41]

AWS IoT Core permite conectar dispositivos de IoT a la nube de AWS sin la necesidad de aprovisionar o administrar servidores. AWS IoT Core admite miles de millones de dispositivos y billones de mensajes, y es capaz de procesarlos y direccionarlos a puntos de enlace de AWS y a otros dispositivos de manera confiable y segura. Con AWS IoT Core, las aplicaciones pueden realizar un seguimiento de todos los dispositivos y comunicarse con ellos en todo momento, incluso cuando no están conectados.

Sabiendo la existencia de un servicio orientado a IoT, se han realizado pruebas y búsquedas para el posible control de los dispositivos mediante IoT Core, sustituyendo así a la plataforma.

No se ha podido probar en profundidad este servicio, debido a que no se dispone de los dispositivos reales.

Sin embargo, las pruebas realizadas auguran la futura adaptación de ese servicio al proyecto. Estas pruebas han sido extraídas de la documentación oficial de AWS. [42]

Para probar este servicio, se ha seguido un tutorial ofrecido en la documentación mencionada, que consiste en crear tu primer objeto, conectar el dispositivo, y enviar mensajes MQTT.

MQTT, es un protocolo de transporte para IoT. Esta designado para suscribir mensajes pequeños, perfecto para conectar dispositivos remotos. [43]

Para realizar estas pruebas se ha utilizado como dispositivo, la Raspberry Pi. Primero se debe preparar la máquina, así como, instalar múltiples librerías, git, python, aws iot sdk para python, …

A continuación, se instalan los certificados que autentican el dispositivo. Para crear estos certificados, suponiendo que se tiene una cuenta AWS, se debe crear una política como la que se visualiza en la siguiente imagen:

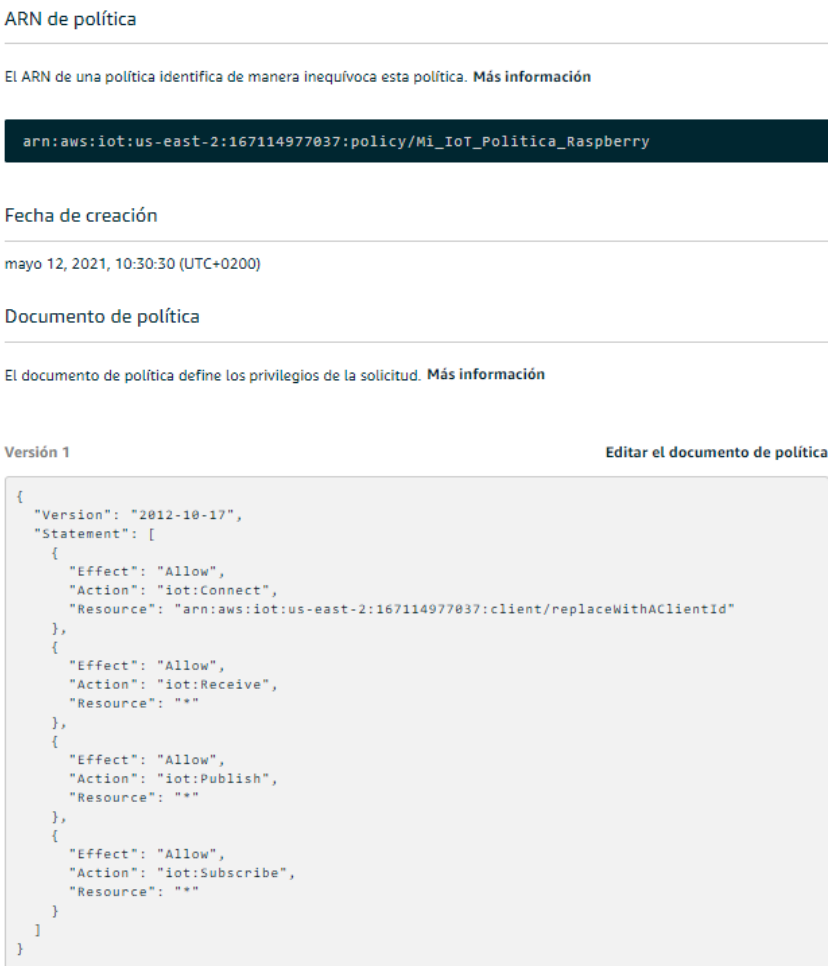


Ilustración 76 - Política de objeto en AWS

En esta política, se especifica que se permite conectar, recibir, publicar y suscribir. Son acciones que necesitan el permiso para ejecutarse al poner en marcha la conexión.

Además, se debe crear un objeto “thing”, que representa al dispositivo físico que queremos conectar:

Ilustración 77 - Objeto en IoT Core

En el proceso de creación del objeto, se deben descargar los certificados creados los cuales se instalan en la Raspberry Pi. En este proceso, también se relaciona la política, anteriormente creada, al objeto.

Por último, se ejecuta la conexión en la RaspberryPi, donde si el proceso ha sido el correcto devuelve algo similar a lo que se muestra a continuación:

```
Connected!
Subscribing to topic 'topic_1'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'topic_1': Hello World! [1]
Received message from topic 'topic_1': b'Hello World! [1]'
Publishing message to topic 'topic_1': Hello World! [2]
Received message from topic 'topic_1': b'Hello World! [2]'
Publishing message to topic 'topic_1': Hello World! [3]
Received message from topic 'topic_1': b'Hello World! [3]'
Publishing message to topic 'topic_1': Hello World! [4]
Received message from topic 'topic_1': b'Hello World! [4]'
Publishing message to topic 'topic_1': Hello World! [5]
Received message from topic 'topic_1': b'Hello World! [5]'
Publishing message to topic 'topic_1': Hello World! [6]
Received message from topic 'topic_1': b'Hello World! [6]'
Publishing message to topic 'topic_1': Hello World! [7]
Received message from topic 'topic_1': b'Hello World! [7]'
Publishing message to topic 'topic_1': Hello World! [8]
Received message from topic 'topic_1': b'Hello World! [8]'
Publishing message to topic 'topic_1': Hello World! [9]
Received message from topic 'topic_1': b'Hello World! [9]'
Publishing message to topic 'topic_1': Hello World! [10]
```

Ilustración 78 - Mensajes de conexión

Después de ver el futuro de esta plataforma y en concreto de este servicio, la empresa plantea adaptar el control de los dispositivos instalados a la AWS.

5. Pruebas de integración y validación

Pruebas de los componentes desarrollados

Las pruebas realizadas a los componentes desarrollados individualmente han consistido en comprobar que cumplen con todos los requisitos y que su funcionamiento es el que se espera.

Pruebas de la plataforma

Para verificar que la plataforma funciona correctamente, se han realizado diversas pruebas de envío de datos.

Cada uno de los atributos definidos en los objetos, tienen el tipo definido, por lo tanto, se debe comprobar que Quantumleap funciona correctamente, no aceptando tipos erróneos.

Además, se quiere comprobar que el envío de datos se puede realizar insertando únicamente un atributo y múltiples atributos.

Para comprobar que inserta correctamente el envío de un único atributo, se realiza un POST como el que se muestra en la imagen:

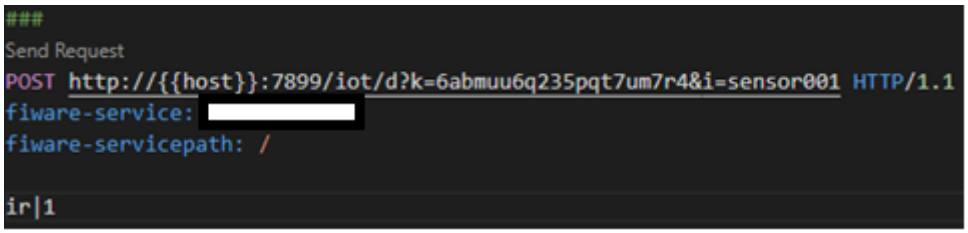


Ilustración 79 - Insertar un atributo

Inicialmente la respuesta que se debe recibir es la cabecera con el código 200 OK, como la que se ha indicado con anterioridad.

A continuación, si ha sido correcto, se comprueba la inserción en la base de datos:

Ilustración 80 - Inserción en la base de datos

Como todo es correcto, se asume que funciona como debe.

Para comprobar la correcta inserción de múltiples atributos, se realizó la misma prueba y se comprobó que la inserción también era correcta.

Para comprobar que no se admiten datos de tipo diferente al definido se prueba a insertar el dato ir de tipo double en vez de entero:

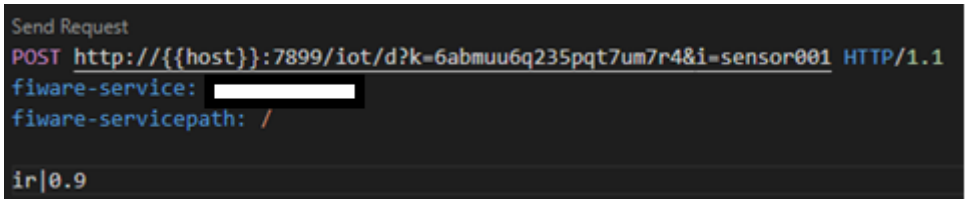


Ilustración 81 - Insertar atributo erróneo

A continuación, comprobamos que la cabecera devuelve código 200 OK, lo que indica que es correcto.

Pero cuando se comprueba la inserción en la base de datos, se ve que Quantum Leap no ha sido capaz de manejar correctamente los datos, y aunque sí que realiza una inserción, de manera que almacena el json original, no es una inserción valida.

fiware_servicepath	__original_ngsi_entity__	timeinstant	distancia	ir
[NULL]	{"data": {"id": "urn:ngsi-ld:sensor:001", "ir": {"type": "Inte	[NULL]	[NULL]	[NULL]

Ilustración 82 - Inserción del atributo erróneo

Por último, comprobamos que no es posible crear un nuevo objeto con mismo id. Para ello, partiendo de que existe un objeto con id “cerrdura001”, se procede a crear un nuevo objeto con el mismo id.

Al intentar realizar esta petición, la respuesta recibida indica que ha habido un conflicto:

Ilustración 83 - Código 409

Tras analizar y completar las pruebas descritas, se asume que el comportamiento de la plataforma es correcto.

Pruebas de la aplicación

Para comprobar el correcto funcionamiento de la aplicación, se han listado una serie de pruebas a realizar y se han anotado los resultados de dichas pruebas.

Pruebas Inicio de sesión y Cerrar sesión	**Suceso**
Introducir credenciales incorrectas	No permite el acceso
Dejar campos vacíos en el formulario	No permite el acceso y comunica que faltan campos por rellenar
Introducir credenciales que no existen	No permite el acceso
Introducir credenciales correctas	Permite el acceso a la aplicación
Seleccionar botón cerrar sesión	Vuelve a la página inicio de sesión
Pruebas del menú	
Seleccionar todos los links y botones	Muestra las vistas necesarias
Pruebas modificar atributo estático	
Introducir campo vacío	Comunica que el campo no tiene valor
Introducir el mismo valor	Comunica que no se ha introducido un valor nuevo
Introducir tipo diferente	Realiza la modificación
Introducir valor correcto	Modifica el valor en el atributo correcto
Un usuario sin privilegios de modificación intenta modificar valor	No le aparece la opción de modificar atributos
Pruebas de seguridad	
Intentar entrar en la aplicación vía url	No permite el acceso
Pruebas de diseño	
Comprobar que la pagina es responsive	La página se acomoda al tamaño de la pagina
Preguntar a personas externas sobre el diseño	Respuestas que indican que el diseño es agradable

Tabla 10 - Pruebas en la aplicación

En la siguiente imagen se muestra que la pagina es responsive, es decir que la imagen de la aplicación se acomoda adecuadamente al tamaño de la página.

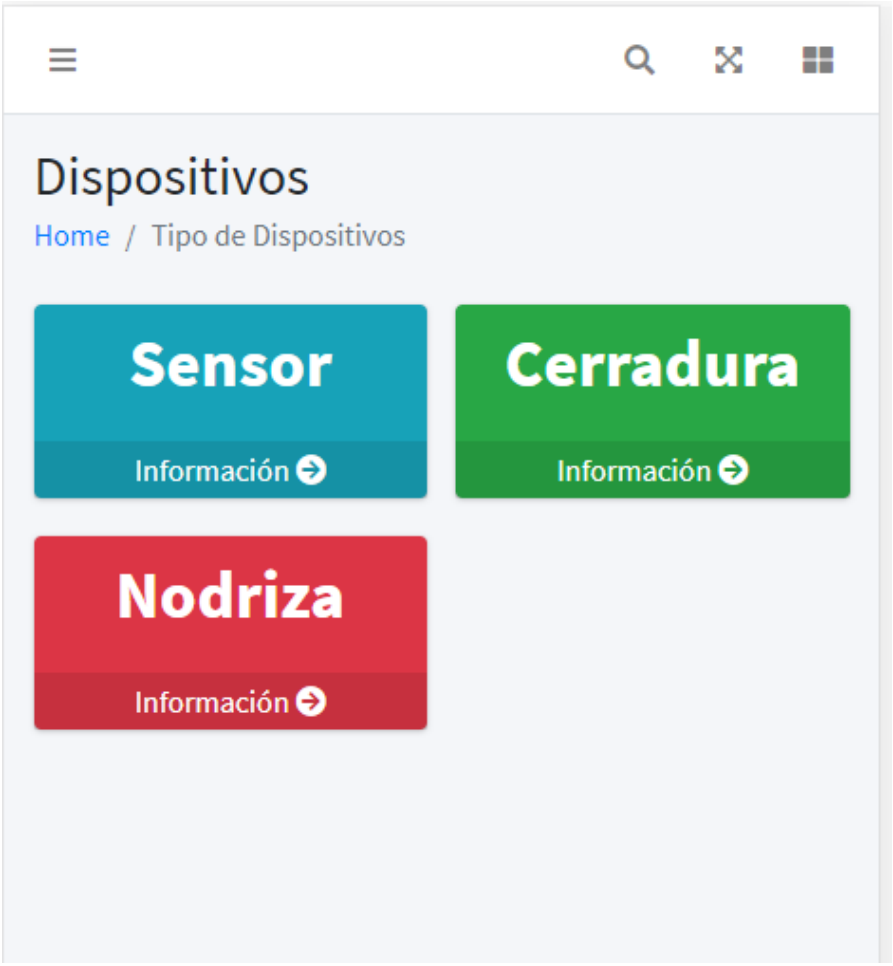


Ilustración 84 - Página principal responsive

Integración de todos los componentes

Para integrar los componentes desarrollados, se ha aplicado integración incremental, que consiste en combinar el componente recién desarrollado y que ya se ha probado, con el conjunto de componentes previamente integrados y probados.

Esta práctica de integración aplicada al proyecto ha servido para identificar y localizar errores más rápidamente. Ha resultado ser una práctica lenta pero muy eficaz.

En concreto, la aplicación de este método de integración ha sido sencilla ya que el proyecto está dividido en fases o bloques independientes, que se han ido desarrollado y probando individualmente. A la finalización de cada bloque, se ha dedicado tiempo a integrarlo al resto de componentes que ya se habían desarrollado.

Validación y rendimiento del sistema

Para realizar pruebas de estrés contra la plataforma estructurada se ha utilizado la herramienta Apache JMeter. [44]

Apache JMeter, es un proyecto de apache, diseñado para probar el comportamiento funcional de pruebas y medir el rendimiento. Aunque originalmente fue diseñado para probar aplicaciones web, se ha expandido a otras funciones de prueba.

Ilustración 85 - Apache JMeter

Las pruebas consisten en simular múltiples usuarios, enviando múltiples peticiones. Lo que se desea probar con esta prueba es que el sistema soporta una gran carga de peticiones y que las soluciona correctamente.

JMeter es capaz de realizar estas pruebas con gran rapidez y con un proceso muy sencillo el cual se describe a continuación:

primero, se seleccionan el contador del bucle (usuarios) y la cantidad de hilos (peticiones) que se deben generar.

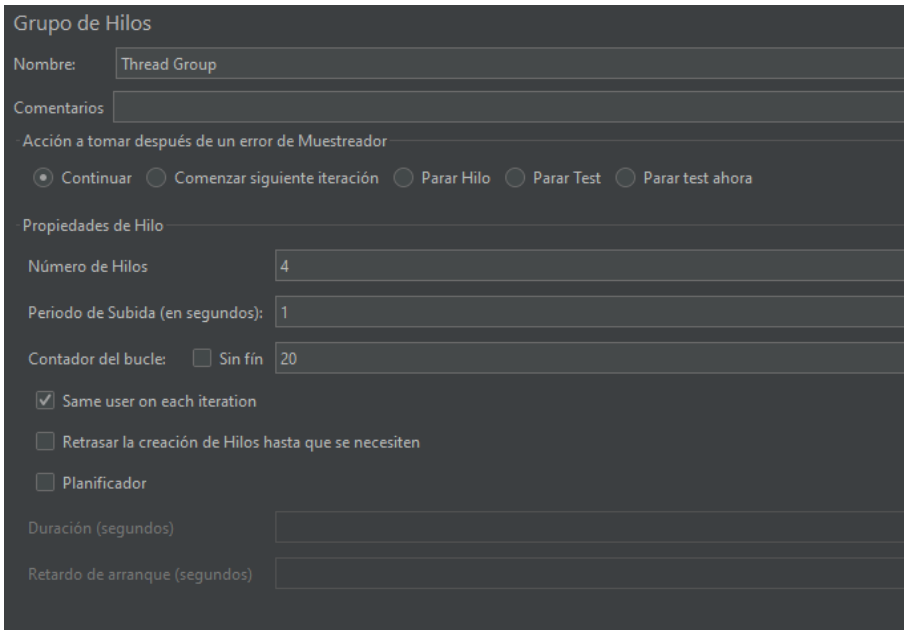


Ilustración 86 - definición de hilos en JMeter

A continuación, se definen múltiples peticiones GET por cada uno de los atributos dinámicos de los dispositivos. En este caso se han definido los atributos de los sensores.

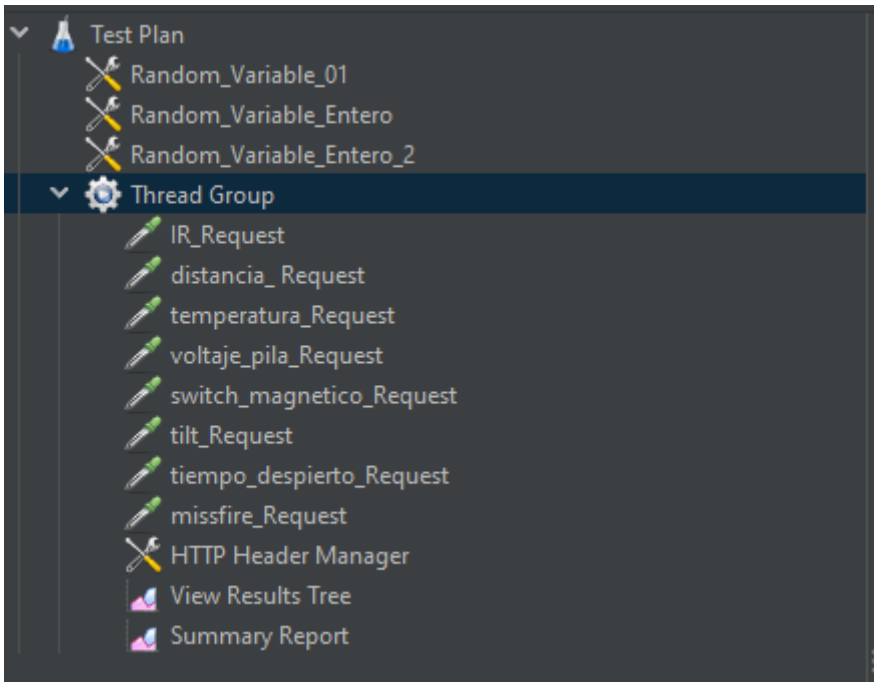


Ilustración 87 - Grupo de peticiones

Además, JMeter tiene la opción de generar números aleatorios con diferentes características. Por ejemplo, se puede crear un número aleatorio entero que tenga un rango del 0 al 1. Este número aleatorio se utilizará para asignar un valor aleatorio al atributo ir.

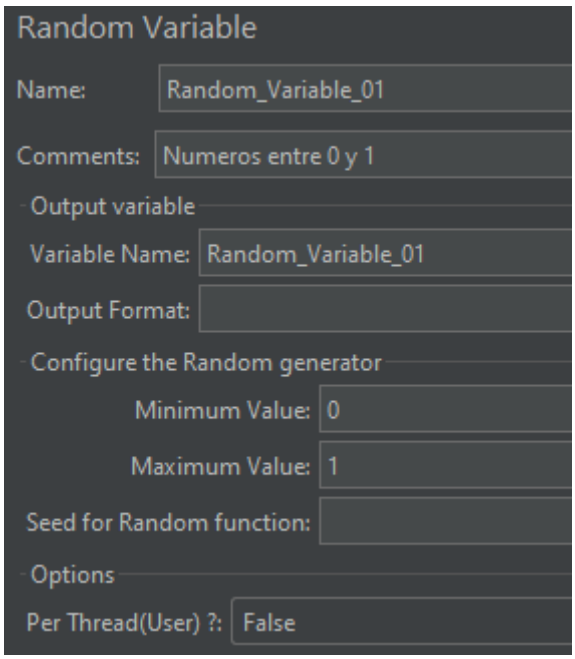


Ilustración 88 - Creación de números aleatorios en JMeter

La definición de las peticiones se realiza de la siguiente manera. Se indica el nombre con el que se identifica la petición, y se selecciona el tipo de petición, en este caso GET, la dirección IP del servidor y el puerto al que se quiere enviar la petición.

Además, se debe indicar la ruta y los parámetros que se quieren enviar en la petición.

Los parámetros que hay que especificar son k, i, d:

- K: k es el parámetro que identifica el apykey.
- I: i es el parámetro que identifica el id del dispositivo.
- D: d es el parámetro que identifica el parámetro que se desea enviar. Aquí es donde se hace referencia a las variables aleatorias definidas anteriormente.

Se ha indicado que el contenido de estos parámetros es texto plano.

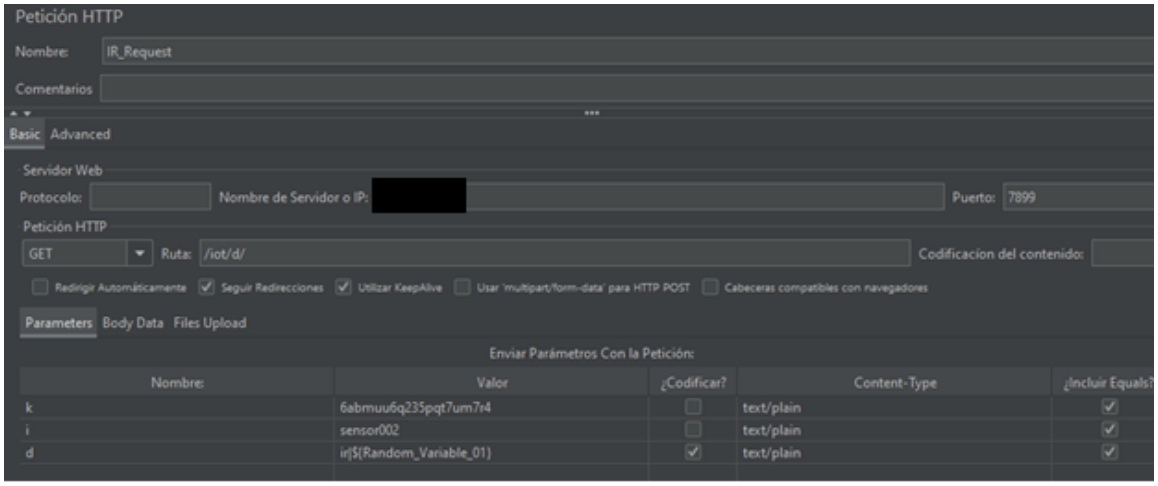


Ilustración 89 - Petición en JMeter

Para realizar estas pruebas se ha simulado el comportamiento masivo, mediante diferentes pruebas con múltiples hilos y múltiples envíos por cada hilo. Se ha comenzado realizando una prueba pequeña, con pocos hilos, y a medida que ha resultado satisfactoria, se han ido ampliando los valores.

Peticiones	Usuarios	Resultados
30	10	100% OK
50	30	100% OK
100	70	100% OK
150	150	96.16% OK

Tabla 11 - Pruebas de carga

Tras el lanzamiento, hay diversas maneras de visualizar los resultados. En este caso se ha utilizado el método visual "View Results Tree", el cual nos indica que peticiones han resultado satisfactorias y cuáles no.

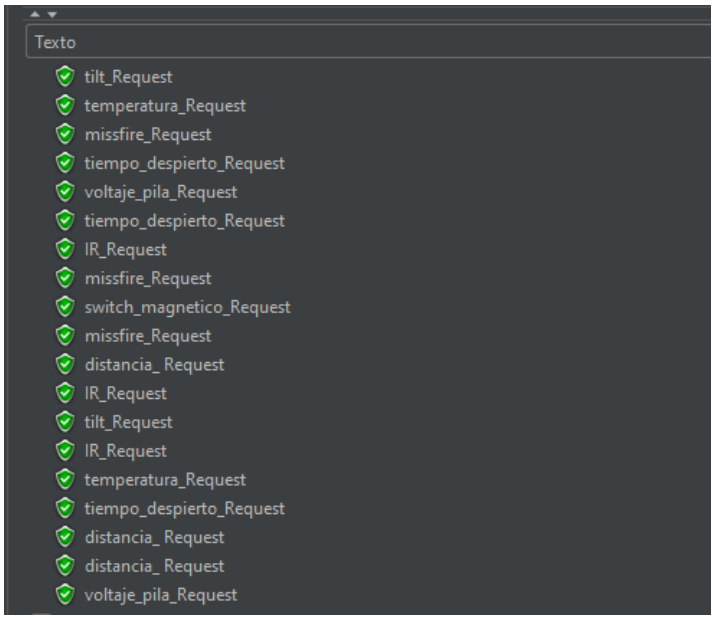


Ilustración 90 - Resultados en árbol JMeter

Y el método de visualización "Summary Report" el cual hace un sumario de todas las peticiones, indicando el número de peticiones, el tiempo medio, el porcentaje de error, y más propiedades que indican como ha resultado la ejecución de las peticiones.

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
IR_Request	31300	2264	57	26926	1536.60	3.61%	2.3/sec	0.70	0.47	314.9
distancia_ Requ...	31300	2247	66	26913	1483.49	3.76%	2.3/sec	0.71	0.52	318.6
temperatura_Re...	31300	2214	138	26917	1399.70	3.84%	2.3/sec	0.71	0.53	320.8
voltaje_pila_Re...	31300	2169	270	22844	1241.99	3.87%	2.3/sec	0.71	0.53	321.7
switch_magneti...	31300	2151	275	19149	1114.66	3.89%	2.3/sec	0.71	0.50	322.5
tilt_Request	31300	2160	358	20866	1098.40	3.89%	2.3/sec	0.71	0.48	322.5
tiempo_despier...	31300	2202	409	26849	1266.73	3.89%	2.3/sec	0.71	0.51	322.5
missfire_Request	31300	2235	121	26918	1428.61	3.96%	2.3/sec	0.72	0.49	324.0
TOTAL	250400	2205	57	26926	1330.90	3.84%	18.1/sec	5.68	4.02	320.9

Ilustración 91 - Resultados en tabla JMeter

Teniendo en cuenta que las pruebas realizadas han resultado casi satisfactorias, se da por hecho que el rendimiento de la plataforma es correcto y que soporta una gran cantidad de carga de datos. Por lo tanto, se da por validada la plataforma desarrollada.

6. Operación y mantenimiento

Planes de mantenimiento y desarrollo de nuevas funcionalidades

Plantear la aplicación de NB-IoT

Narrowband-IoT es la primera tecnología centrada en conectar a Internet objetos cotidianos que requieren pequeñas cantidades de datos en periodos de tiempos largos. [45]

Esta tecnología ha sido desarrollada para permitir comunicaciones eficientes y una alta durabilidad de la batería, para dispositivos distribuidos masivamente. Utiliza la ya existente red móvil para conectar todos esos objetos. NB-IoT está diseñado para ampliar el futuro de la conectividad IoT de una manera más segura y fiable. Es ideal para dispositivos que generan un tráfico de datos no muy alto y tienen un ciclo de vida largo.

Vodafone ofrece una solución NB-IoT, complementaria a los servicios IoT, proporcionando diferentes tipos de conectividades a diferentes casos. [46]

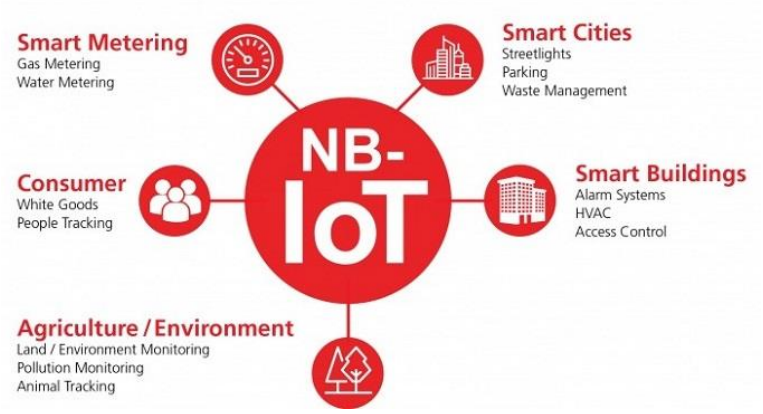


Ilustración 92 - NB-IoT Vodafone

Posibilitar al usuario privilegiado añadir y eliminar atributos estáticos

Para añadir una gestión más profunda, una vez se pruebe la demo, se plantea permitir a los usuarios privilegiados, la posibilidad de añadir y eliminar atributos estáticos de los dispositivos.

Al igual que la modificación del valor de los atributos estáticos, esta opción se haría mediante mongodb.

Buscando información sobre ello, se ha visto que la función que realiza inserciones se define como, bulk ->Write.

Despliegue real de los objetos en IoT Core de AWS

Por último, una vez trabajadas las posibilidades que ofrece AWS y una vez estén operativos los dispositivos a instalar, se plantea sustituir el uso de la plataforma con los componentes Fiware, por los servicios que ofrece IoT Core de AWS.

7. Conclusiones y líneas futuras

Conclusiones

El proyecto sobre el que ha tratado este documento se ha realizado íntegramente en la empresa I3I ingeniería avanzada. Para poder completarlo, he seguido las indicaciones que me han dado los responsables de la empresa. Aunque me han ido indicando las herramientas y tecnologías que podía utilizar, siempre me han dado la libertad para buscar alternativas. Sin embargo, el desconocimiento previo de los conceptos importantes del proyecto y el límite de tiempo, no me ha permitido buscar estas alternativas.

Uno de los aspectos más importantes del proyecto, ha sido el desarrollo de la plataforma, formado por una gran variedad de componentes, de los cuales no tenía conocimiento alguno. Esto ha conllevado a que el tiempo dedicado a la investigación y solución de errores haya sido una parte importante de mi proyecto. Los puntos que más me han costado, son los siguientes:

- Conectar la base de datos temporal TimeScale a la plataforma: aunque sí que existe documentación referente a TimeScale, no había una guía pautada de como conectar mediante Docker-Compose la componente Orion con TimeScale.
- Errores al conectar la base de datos MongoDB: la plataforma desarrollada, requería de una versión específica de mongoDB y el desconocimiento de esto ha provocado pequeños retrasos.
- Entender el procedimiento del provisionamiento de los dispositivos a realizar.

En referencia a la aplicación web que he desarrollado, aunque me siento satisfecha del aspecto final y su funcionamiento, para la programación del backend he aplicado los conocimientos adquiridos en la carrera utilizando el lenguaje de programación php y los patrones ya mencionados. Sin embargo, considero que esta práctica se está quedado anticuada, y me hubiese gustado aplicar nuevas herramientas y técnicas como por ejemplo JavaScript.

Desde el punto de vista personal, partiendo del punto de que no conocía la mayoría de los conceptos, estoy satisfecha con el desarrollo llevado a cabo. Considero que he abarcado una gran cantidad de conceptos y herramientas nuevas que me han enriquecido profesionalmente.

Líneas futuras

El tipo de proyecto que he realizado puede conllevar una gran cantidad de mejoras y avances, tanto por el rápido avance tecnológico (hardware y software) como el negocio al que va a ser dedicado (mundo IoT).

Una línea futura de desarrollo que se plantea es la sustitución de la plataforma con componentes FIWARE por algo más novedoso y en pleno auge como AWS.

Considero también, que la aplicación de gestión podría ser mejorada mediante la incorporación de alertas y avisos a usuarios a través del envío de mensajes. Esto facilitaría aún más la gestión y organización de los contenedores.

Bibliografía

[1] https://www.fiware.org/

[2] https://www.docker.com/

[3] https://es.wikipedia.org/wiki/Objetivos_de_Desarrollo_Sostenible#Objetivos_de_Desarrollo_Sostenible

[4] https://www.un.org/sustainabledevelopment/es/cities/

[5] Solución inteligente de gestión de residuos para la ciudad - SENSONEO

[6] Soluciones SMART: Cerraduras electrónicas | CONTENUR

[7] https://ticnegocios.camaravalencia.com/servicios/tendencias/caminar-con-exito-hacia-la-industria-4-0-capitulo-14-dispositivos-i-internet-de-las-cosas-iot/

[8] https://fiware-tutorials.readthedocs.io/en/latest/time-series-data/index.html#architecture

[9] https://fiware-tutorials.readthedocs.io/en/latest/iot-agent/index.html

[10] https://fiware-orion.readthedocs.io/en/master/

[11] https://www.mongodb.com/es

[12] https://www.timescale.com/

[13] https://www.postgresql.org/

[14] https://quantumleap.readthedocs.io/en/latest/

[15] https://grafana.com/

[16] https://www.fiware.org/developers/data-models/

https://es.wikipedia.org/wiki/NGSI-LD

[17] https://www.raspberrypi.org/

[18] https://www.proxmox.com/en/

[19] https://code.visualstudio.com/

[20] https://dbeaver.io/

[21] https://robomongo.org/

[22] https://www.google.com/

[23] https://www.putty.org/

[24] https://httpd.apache.org/

https://www.webempresa.com/hosting/que-es-servidor-web.html

[25] https://www.nginx.com/

https://es.wikipedia.org/wiki/Nginx

[26] https://letsencrypt.org/es/

https://es.wikipedia.org/wiki/Let's_Encrypt

[27] https://certbot.eff.org/

[28] https://aws.amazon.com/es/

[29] https://softwarelab.org/es/lte-4g/

[30] https://es.wikipedia.org/wiki/Pppd

[31] https://es.wikipedia.org/wiki/Point-to-Point_Protocol

[32] https://www.welivesecurity.com/la-es/2012/09/10/vpn-funcionamiento-privacidad-informacion/

[33] http://www.amitwireless.com/jp/upload/products/download/Quectel_EC25&EC21_PPP_Application_Note_V1.0.pdf

[34] https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units-es

https://wiki.archlinux.org/title/Systemd_(Espa%C3%B1ol)

[35] https://www.zeppelinux.es/como-ejecutar-un-script-o-programa-al-iniciar-linux-con-rc-local/

[36] https://quantumleap.readthedocs.io/en/latest/admin/timescale/

[37] https://www.postgresql.org/docs/8.3/pgcrypto.html

[38] https://adminlte.io/

[39] https://account.dyn.com/

[40] https://aws.amazon.com/es/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

https://es.wikipedia.org/wiki/Amazon_EC2

[41] https://aws.amazon.com/es/iot-core/

[42] https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

[43] https://mqtt.org/

[44] https://jmeter.apache.org/

[45] https://accent-systems.com/es/nb-iot/?v=04c19fa1e772

[46] https://www.vodafone.es/c/empresas/grandes-clientes/es/iot-big-data/iot/narrowband-iot/

[47] https://azure.microsoft.com/es-es/

https://es.wikipedia.org/wiki/Microsoft_Azure

Anexos

Comparando sistemas de virtualización

Con el propósito de almacenar el sistema, se plantea adecuarlo al entorno nube. Para llevar a cabo esta tarea, se plantearon dos opciones, Azure y AWS.

La empresa desde el principio optaba por la opción de AWS, pero se hicieron comparaciones, de manera que la decisión tuviese más criterio.

Microsoft Azure es un servicio de computación en la nube creado por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. [47]

Amazon Web Services (AWS abreviado) es una colección de servicios de computación en la nube pública (también llamados servicios web) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com.

En la siguiente tabla se muestra la comparación realizada:

PROPIEDADES	**AZURE**  *Ilustración 93 - Azure*	**AWS**  *Ilustración 94 - AWS*
Precios	La creación de una cuenta es gratuita. Una vez te das de alta, Azure ofrece el uso de todos sus servicios gratuitos durante 30 días. Tiene muchos elementos convincentes, como el desglose de precios dependiendo de las necesidades del proyecto, sin embargo, sus costos son más agresivos.	La creación de una cuenta es gratuita. Una vez te das de alta, Amazon ofrece servicios gratuitos, con ciertos límites. Los precios se ajusten a las necesidades del proyecto, no es necesario comprar costosas soluciones de nube privada que se vuelven obsoleta en poco tiempo. Con Amazon Web Services solo necesita ajustar su máquina virtual y la nube de AWS se adaptará según los requerimientos, para solo pagar por el poder de cómputo usado.

Servicios	Almacenamiento Bases de datos Migraciones Herramientas de desarrollo Seguridad Analítica Inteligencia Artificial Servicios Móviles Servicios de aplicaciones Mensajería Internet de las cosas Desarrollo de juegos Software Aplicaciones de escritorio Y muchos mas

Tabla 12 - Comparativa Azure y AWS

Finalizada esta comparación, aunque los servicios ofrecidos sean muy similares, lo que AWS ofrece en IoT se acomoda más a lo que la empresa quiere aplicar.

Por otro lado, los servicios que AWS ofrece gratuitamente son suficientes para las pruebas que se desean realizar.

Por todo esto y por el gran futuro que se augura de este entorno, se ha optado por probar la colección de servicios ofrecidos por Amazon Web Services.