MEMORIA RA

Gabriel Gutierrez Fuentes
David Pavo Puertas
Alejandro García Hidalgo
Pedro Angel Guzmán Torres

ÍNDICE

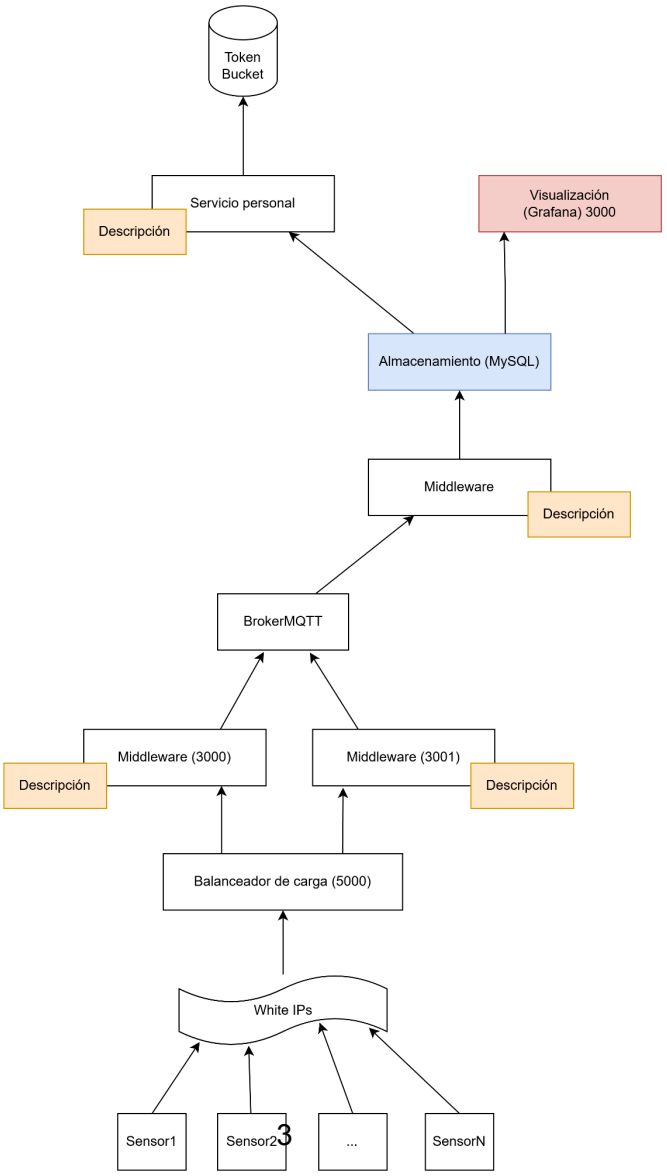
ARQUITECTURA

En esta red, el balanceador de carga recibe las peticiones POST's de los sensores que a través de la Whitelist de IPs permite el paso a la red que tengas configurada. Esas peticiones se reparten con el método Round Robin a los middlewares a los puertos 3000 y 3001 respectivamente. Los middleware, como se explica posteriormente, transforman las peticiones HTTP a MQTT.

Cada sensor va a tener un canal donde publicar los datos recogidos, pero MySQL no tiene soporte con MQTT y por tanto tenemos que transformarlo para insertarlo en la tabla.MySQL se comunica con Grafana mediante el puerto 3000, para poder visualizar unos gráficos específicos de forma más visual.

También, tenemos conectado a la BBDD un servicio personal el cual nos permite hacer un GET de la media de CO2 y la fecha del valor más alto de CO2. Las peticiones que hace el cliente pasan por un leaky bucket para tener limitadas las peticiones al servicio y aumentar la seguridad.

Teniendo en cuenta lo explicado así quedaría el esquema de la arquitectura del proyecto:



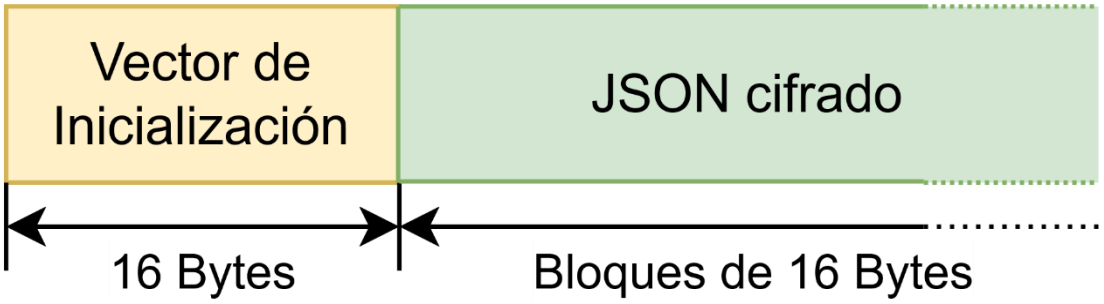
Implementación de los servicios

SENSORES

Los sensores se comunican con el sistema a través de peticiones POST de HTTP. Estos recogen valores de temperatura, humedad, CO_2 y sustancias volátiles, y los envían al Middleware en formato JSON, junto con su identificador único, al endpoint `/record`.

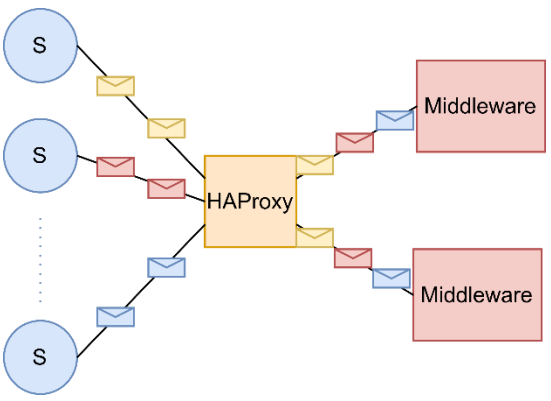
```
{
    "id": "1",
    "temperatura": 20.1,
    "humedad": 10,
    "co2": 1253,
    "volatiles": 45
}
```

La comunicación entre el Middleware y los sensores está cifrada utilizando un esquema personalizado, y no usa HTTPS. Todos los mensajes se cifran usando AES128 en modo CTR, con un vector de inicialización aleatorio y único para cada mensaje. Una vez cifrado, se concatenan el IV y el mensaje cifrado y se envían al Middleware. Cabe destacar que entre el Middleware y los sensores no hay mecanismo de negociación de clave, sino que esta es fija. Los datos se rellenan usando PKCS7 (relleno de `0x00` hasta alcanzar un múltiplo del tamaño de bloque).



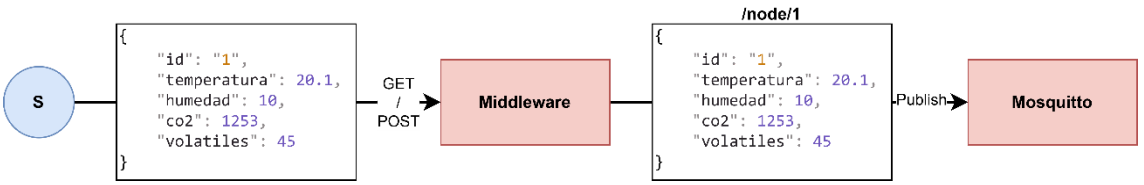
BALANCEADOR DE CARGA

Para balancear el tráfico entre los sensores y el Middleware se ha utilizado HAProxy. Los sensores no se comunican directamente con el Middleware en ningún momento, lo hacen siempre a través del balanceador. HAProxy se ha configurado en modo Round-Robin, seleccionando servidores de manera secuencial hasta llegar al último, para luego volver al primero.



MIDDLEWARE DE SENSORES

Dado que los sensores no pueden comunicarse con el broker MQTT directamente, se precisa un Middleware capaz de traducir las peticiones de HTTP a MQTT. Este Middleware es capaz de recibir datos cifrados a través del método POST y datos sin cifrar a través del método GET para luego retransmitirlos al broker MQTT en el topic `/node/{id nodo}`. El Middleware también cuenta con una lista blanca de IP, de manera que solo los sensores puedan enviar datos al sistema.



MIDDLEWARE DE BASE DE DATOS

El servidor de la base de datos MySQL tampoco tiene soporte para MQTT, por lo que es necesario otro Middleware. El servicio se suscribe al topic `/node/#` en el broker MQTT, de manera que pueda recibir todos los datos de los sensores. Al recibir una publicación, genera el código SQL necesario para insertar los datos en la base de datos.

```sql
INSERT INTO info
    (id, temperatura, humedad, co2, volatiles, fecha)
    VALUES
    ("1", 20.1, 10, 1253, 45, "2025-05-16 16:20:58");
```

SERVICIO PERSONAL

El servicio personal acepta peticiones GET a dos endpoints: `/media_co2` y `/fecha_co2_alto`. El endpoint `/media_co2` devuelve la media de CO_2 de un periodo de tiempo y el endpoint `/fecha_co2_alto` devuelve el valor más alto de CO_2 en otro periodo de tiempo. Para ello, el servicio utiliza los datos guardados en la base de datos.

Para proteger la base de datos y el sistema, se limita el acceso al servicio utilizando una implementación de leaky bucket.

Guía de uso

Para poder usar el proyecto hay que seguir unos sencillos pasos.

- Paso 1: Descargar el proyecto usando git clone https://gitlab.etsisi.upm.es/bt0106/ra.git en el directorio donde se quiera tener el proyecto.
- Paso 2: Acceder al directorio ra utilizando cd ra.
- Paso 3: Una vez en esta carpeta ejecutar el comando sudo sh start.sh .

El proyecto ya está ejecutado, se podrá observar una terminal TMux dividida en cuatro pantallas. Las dos situadas en la izquierda pertenecen al middleware de los sensores, en los puertos 3000 y 3001.
La ventana situada arriba a la derecha pertenece al middleware de la base de datos.
Por último, la situada en la parte inferior derecha ejecuta el servicio que conecta al cliente con la base de datos y le deja usar las peticiones implementadas.

Todo esto se puede probar también de manera individual habiendo iniciado todos los servicios.

- Para probar los middlewares de los sensores habría que poner:
  curl
  “http://10.100.0.119:5000/record?id_nodo={ID}&temperatura={NUM}&humedad={NUM}&co2={NUM}&volatiles={NUM}”
- Para suscribirse a un “canal”
  mosquitto_sub -h 10.100.0.119:1883 -t node/#
- Para publicar en un “canal”
  mosquitto_pub -h 10.100.0.119:1883 -t node/{ID} -m {MENSAJE}
- Para probar el servicio personal:
  curl “http://10.100.0.119:4000/media_co2?fecha_inicio={UNIX}&fecha_fin={UNIX}”

  curl “http://10.100.0.119:4000/fecha_co2_alto?fecha_inicio={UNIX}&fecha_fin={UNIX}”

Conclusión

A lo largo del desarrollo de este proyecto hemos podido poner en práctica conceptos muy importantes relacionados con la comunicación entre servicios, el procesamiento de datos en sistemas distribuidos y la visualización de información en tiempo real. Hemos podido trabajar con tecnologías muy actuales y también hemos comprendido cómo se integran entre sí para formar una arquitectura robusta y funcional.

Durante la implementación hemos aprendido a desplegar y coordinar servicios variados, a gestionar peticiones de forma segura, y a manejar los datos eficientemente desde sensores hasta visualización final. Este tipo de tareas son características del trabajo que realiza un ingeniero DevOps, puesto muy presente y valorado en el entorno laboral.

En definitiva, este proyecto ha sido una aproximación muy práctica al mundo real del desarrollo de sistemas distribuidos y nos ha proporcionado una base sólida para enfrentarnos a entornos similares en el ámbito profesional.