CARRERA DE ESPECIALIZACIÓN EN INTERNET DE LAS COSAS

MEMORIA DEL TRABAJO FINAL

Diseño e implementación de una central operativa para el control y monitoreo en el material rodante

Autor:
Fernando Julio Iglesias

Director:
Fernando Lichtschein (FI-UBA)

Jurados:
Nombre del jurado 1 (pertenencia)
Nombre del jurado 2 (pertenencia)
Nombre del jurado 3 (pertenencia)

Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires, entre marzo de 2022 y agosto de 2023.

Resumen

Ante la falla de sistemas como el control de puertas o la cámara frontal de registro de accidentes viales, las formaciones ferroviarias activan automáticamente el corte de tracción y el freno de emergencia. En esas circunstancias la formación queda detenida y los pasajeros deben descender a las vías si la detención no se produce en una estación ferroviaria. En este trabajo se presenta el desarrollo para Trenes Argentinos de una arquitectura modular basada en el paradigma de Internet de las Cosas (IoT) que permite visualizar y gestionar los sistemas involucrados en esas situaciones.

Agradecimientos

Esta sección es para agradecimientos personales y es totalmente **OPCIONAL**.

Índice general

Resumen I

1. Introducción general 1
1.1. Introducción . . . 1
1.1.1. La Internet de las Cosas y las formaciones ferroviarias . . . 1
1.1.2. Motivación . . . 1
1.1.3. Estado del arte . . . 3
1.1.4. Alcance y objetivos . . . 3

2. Introducción específica 5
2.1. Protocolos de comununicación utilizados . . . 5
2.1.1. Protocolos de comunicación . . . 5
Ethernet . . . 6
Address Resolution Protocol . . . 6
Internet Protocol . . . 6
User Datagram Protocol . . . 7
Message Queuing Protocol . . . 7
Hypertext Transfer Protocol Secure . . . 7
Protocolos por analizar si estan involucrados en algun punto de la arquitectura . . . 8
2.2. Tecnologías *full-stack* . . . 8
2.2.1. Tecnologías del *front-end* . . . 8
TypeScript . . . 8
React . . . 9
Parcel . . . 9
Apollo . . . 10
Material UI . . . 10
JWT Decode . . . 10
2.2.2. Tecnologías del *backend* . . . 11
Kotlin . . . 11
Ktor . . . 11
Kafka . . . 11
Hasura . . . 12
PostgreSQL . . . 12
Mosquitto Broker . . . 12
2.2.3. Tecnologías del *firmware* . . . 13
C lang . . . 13
FreeRTOS . . . 13
Paho MQTT *client* . . . 13
2.3. Herramientas utilizadas . . . 14
Docker . . . 14
Docker Compose . . . 14

Redpanda . . . 14
Git . . . 14
MQTT.fx . . . 15
CLion . . . 15
Wireshark . . . 15
Herramientas del navegador de internet . . . 15
JTAG . . . 15
ST-Link . . . 16
STM32CubeMX . . . 16
STM32CubeProgrammer - SWV . . . 16

3. Diseño e implementación **17**
3.1. Descripción del sistema . . . 17
3.1.1. Comunicación device backend . . . 18
3.1.2. Comunicación backend frontend . . . 18
3.2. Arquitectura del sistema . . . 18
3.3. Arquitectura propuesta . . . 18
3.3.1. Capa de dispositivos . . . 20
3.3.2. Capa de conectividad . . . 20
3.3.3. Capa de almacenamiento . . . 20
3.3.4. Capa de interacción . . . 21
3.3.5. Capa de visualización . . . 21
3.4. Arquitectura de datos . . . 21
3.4.1. Entidades del modelo de datos . . . 22
Schema auth_service . . . 22
Schema enums . . . 22
Schema public . . . 22
3.5. Seguridad del sistema . . . 23
3.5.1. Protocolo HTTPS . . . 23
3.5.2. MQTT v5.0 . . . 23
3.6. Desarrollo del frontend . . . 23
3.6.1. Estructura del cliente web . . . 23
3.6.2. Páginas . . . 24
Login . . . 24
Tabla de trenes+salt . . . 24
Tabla de usuarios . . . 24
dashboard . . . 24
3.6.3. Modales . . . 24
3.7. Desarrollo del backend . . . 24
Mosquitto Broker . . . 25
Hasura . . . 25
Kafka . . . 25
3.8. Desarrollo del firmware . . . 25
3.9. Desarrollo de la API GraphQL . . . 25

Índice de figuras

1.1. Panel de control de la Central Operativa SAL/T para la visualización de datos en tiempo real. 2

2.1. Arquitectura de la Central Operativa SAL/T. 9

3.1. Diagrama en bloques del sistema SALT. 19

Índice de tablas

Dedicado a... [OPCIONAL]

Capítulo 1

Introducción general

1.1. Introducción

1.1.1. La Internet de las Cosas y las formaciones ferroviarias

La Internet de las Cosas (*IoT*, por sus siglas en inglés) es una tecnología que se ha convertido en una tendencia en la actualidad. Este concepto se estableció en el mercado en la década de 1990, aunque su verdadero auge comenzó en los últimos años gracias a la aparición de dispositivos interconectados y al aumento en el uso de internet. La IoT se enfoca en la interconexión de objetos cotidianos con la red, lo que permite recopilar datos en tiempo real, optimizar procesos y tomar decisiones basadas en la información recolectada. Algunos beneficios que la IoT ofrece son una mayor eficiencia, una reducción de costos, un mejor monitoreo y control, y una mejor calidad de vida para las personas.

En el sector ferroviario, la IoT ha demostrado ser una tecnología clave para mejorar la eficiencia y la seguridad de las formaciones ferroviarias. Las formaciones ferroviarias pueden ser equipadas con sensores y dispositivos interconectados que permiten recopilar información sobre la ubicación, velocidad, consumo de energía, estado mecánico y otros aspectos importantes. Esta información se puede enviar en tiempo real a la red, lo que permite una mejor gestión del tráfico ferroviario y una toma de decisiones más rápida y eficaz. Además, la IoT también puede ayudar a prevenir accidentes ferroviarios al detectar problemas mecánicos antes de que se conviertan en un riesgo para la seguridad.

1.1.2. Motivación

El sistema ferroviario de la República Argentina cuenta con una gran cantidad de formaciones ferroviarias en las que se encuentran diferentes sistemas de seguridad a bordo. Estos equipos se encargan de supervisar el correcto funcionamiento de los subsistemas críticos. Ante una falla en uno de los subsistemas, una formación ferroviaria se detiene inmediatamente por la activación automática de las señales de corte de tracción (*CT*) y frenado de emergencia (*FE*). En esta situación, el conductor debe llevar la formación a un lugar seguro para que los pasajeros puedan descender y, posteriormente, trasladarla a un taller para que pueda ser reparada.

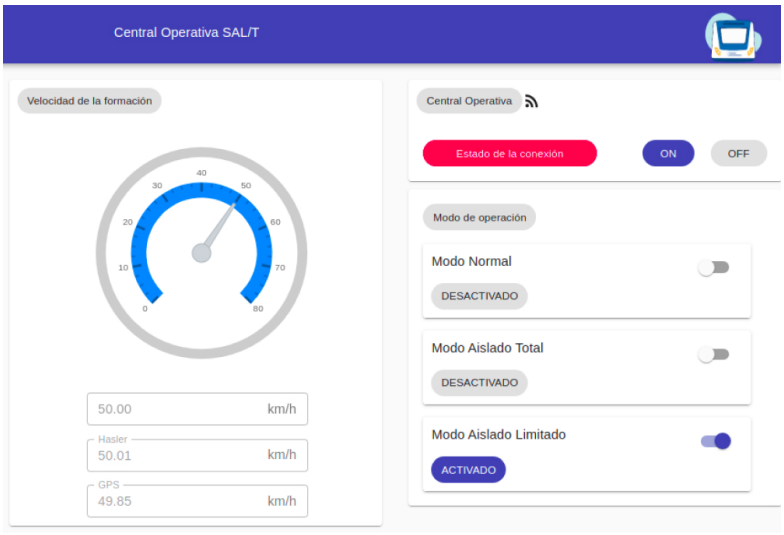

FIGURA 1.1. Panel de control de la Central Operativa SAL/T para la visualización de datos en tiempo real.

El *SAL/T*, según sus siglas, Sistema de Aislamiento Limitado y Total [1], es un dispositivo del cual se cuenta con una primera versión prototipada; que se presenta como solución a las contingencias descritas anteriormente. De esta manera, el maquinista de una formación ferroviaria cunta con la posibilidad de activar y desactivar el *modo aislado limitado*. En este modo, el equipo permite la circulación de la formación al desactivar las señales de corte de tracción y freno de emergencia generadas por los subsistemas críticos. Para que esta operación se complete de forma segura, se debe monitorear la velocidad de la formación tal que sea posible garantizar que no se supere cierto valor máximo.

A partir del desarrollo previo del prototipo, se presentan los avances en la implementación de una central operativa que centraliza los dispositivos *SAL/T* para su respectiva administración, configuración y monitoreo en tiempo real de la información recibida y transmitida desde una plataforma digital. El proyecto se desarrolla por *CONICET-GICSAFe* [b2] para la empresa Trenes Argentinos [3].

Los subsistemas asociados al *SAL/T*, como la seguridad de puertas, el sistema de hombre vivo y la protección de coche a la deriva, son críticos debido a que, en caso de fallar, pueden ocasionar lesiones o muertes de personas e incluso generar pérdidas materiales.

La central operativa permite la administración y configuración en forma remota de los dispositivos de supervisión de seguridad de cada formación ferroviaria, la visualización de los diferentes parámetros de interés involucrados por las personas asignadas dentro de una entidad y de este modo es posible optimizar la toma de decisiones.

1.1.3. Estado del arte

A partir del análisis en las últimas tendencias referidas a la Central Operativa SAL/T, se listan aquellas herramientas, de gestión y control de dispositivos de seguridad en el sector ferroviario, que presentan características similares al producto propuesto,

1. *Indra*: ofrece una solución integral de seguridad para el sector ferroviario, que incluye una plataforma de gestión centralizada y un sistema de supervisión remota de dispositivos de seguridad.
2. *Thales Group*: desarrolla soluciones de seguridad para el sector ferroviario, que incluyen una aplicación de gestión y control centralizada para supervisar y configurar dispositivos de seguridad de forma remota.
3. *Alstom*: ofrece una plataforma de gestión y control para el sector ferroviario, que permite la supervisión y configuración de dispositivos de seguridad de forma remota, así como el análisis en tiempo real de los datos recopilados por estos dispositivos.
4. Siemens Mobility: ofrece una solución de gestión y control para el sector ferroviario, que incluye una aplicación web de central operativa para supervisar y configurar dispositivos de seguridad de forma remota, así como un sistema de análisis de datos en tiempo real.

1.1.4. Alcance y objetivos

El sistema resultante de este trabajo se encuentra destinado al desarrollo del software para una central operativa que permite administrar y configurar de forma remota dispositivos de supervisión de seguridad de formaciones ferroviarias denominados SAL/T (Sistema de Aislamiento Limitado/Total).

El alcance del proyecto comprende los siguientes puntos,

- Servicio de monitoreo y control: visualizar en tiempo real los datos recibidos y enviar comandos de control a los dispositivos activos.
- Gestión de configuración de dispositivos: visualizar y modificar los parámetros configurables de los dispositivos.
- Base de datos: registrar la información recibida desde todos los dispositivos.
- Componentes de seguridad: brindar seguridad a las comunicaciones y a la información almacenada.
- Gestión de usuarios y perfiles: administrar roles y permisos de acceso.
- Realizar la migración de un microcontrolador de la familia Cortex tipo M a uno de la misa familia que disponga de características destinadas a la seguridad funcional del sistema.

Capítulo 2

Introducción específica

2.1. Protocolos de comununicación utilizados

Comenzar escribiendo que aqui se detallaran los protocolos utilizados segun el modelo OSI y hablar sobre el diagrama

AGREGAR UNA IMAGEN CON LOS 3 ENTES INTERCONECTADOS BAJO EL MODELO OSI Nucleo F429ZI Board <—->Dedicated Server <—->Web Client

2.1.1. Protocolos de comunicación

Se presentan los protocolos de comunicación que se utilizan para el monitoreo y control de las formaciones ferroviarias a través de la central operativa según el modelo *TCP/IP*. En particular, el listado se expone en un formato *uplink* y se distinguen al microcontrolador, que se encuentra en el kit de desarrollo y se configura bajo la pila de protocolos *LwIP*, del microprocesador, que se utiliza tanto en el servidor dedicado como en el ordenador.

1. Capa de red:
   - Microcontrolador: LwIP utiliza el protocolo ARP (Address Resolution Protocol) para mapear direcciones IP a direcciones físicas (MAC).
   - Microprocesador: al igual que en la capa física se emplea el protocolo *Ethernet*.
2. Capa de internet: en ambos casos se usa el protocolo *IP* (*Internet Protocol*).
3. Capa de transporte:
   - Microcontrolador: LwIP admite varios protocolos de transporte, incluyendo *TCP* (*Transmission Control Protocol*) y *UDP* (*User Datagram Protocol*), que se utilizan para proporcionar una conexión confiable o no confiable, respectivamente. En especial, se pone en práctica el protocolo *UDP*.
   - Microprocesador: al igual que en el caso del microcontrolador, se utiliza el protocolo *UDP*.
4. Capa de aplicación:
   - Microcontrolador: en esta capa no se utiliza alguno de los protocolos dispuestos por el *stack LwIP* sino que se opta por el protocolo *MQTT* (*Message Queuing Telemetry Transport*).

- Microprocesador: se utiliza el protocolo *HTTPS* (*Hypertext Transfer Protocol Secure*).

Ethernet

Ethernet es un protocolo de la capa de enlace de datos (Link layer) en la pila de protocolos TCP/IP. En la capa de enlace de datos, Ethernet utiliza una subcapa de Control de Acceso al Medio (MAC) para permitir que varias máquinas compartan el mismo medio físico de transmisión. La subcapa MAC controla el acceso al medio físico para asegurar que un dispositivo no intente transmitir datos al mismo tiempo que otro dispositivo. Para prevenir las colisiones, Ethernet utiliza un esquema de detección de colisiones.

Ethernet es el protocolo más comúnmente utilizado en redes LAN debido a su bajo costo y facilidad de implementación.

Address Resolution Protocol

El protocolo ARP (Address Resolution Protocol) es un protocolo de la capa de enlace de datos del modelo TCP/IP. Su función principal es mapear una dirección de protocolo de red (como una dirección IP) a una dirección física (como una dirección MAC).

Cuando un host quiere comunicarse con otro host en la misma red, necesita conocer la dirección física del otro host. Para hacer esto, el host emisor envía un paquete ARP a la red preguntando quién tiene la dirección IP deseada. El host receptor responderá con su dirección física y el host emisor almacenará esta información en su caché ARP para futuras comunicaciones.

En resumen, ARP ayuda a los hosts a comunicarse entre sí en una red TCP/IP al proporcionar una manera de encontrar la dirección física de un host a partir de su dirección IP.

Internet Protocol

El Protocolo de Internet (IP) es uno de los protocolos más importantes en el modelo de referencia TCP/IP. Es responsable de enrutar y entregar paquetes de datos en la red, identificando y direccionando los nodos y subredes.

Cuando un paquete de datos se envía a través de la red, se divide en fragmentos y se le asigna una dirección IP de origen y destino. La capa IP se encarga de enrutar los paquetes hacia su destino final, utilizando información de la dirección IP para identificar y localizar los nodos y redes.

En resumen, IP es el protocolo responsable de la comunicación de extremo a extremo en la red, y es esencial para la conectividad de Internet y la comunicación de datos en general.

User Datagram Protocol

El protocolo UDP (User Datagram Protocol) es un protocolo de transporte en la capa de transporte del modelo TCP/IP. UDP es un protocolo sin conexión, lo que significa que no establece una sesión entre el emisor y el receptor antes de enviar los datos. En lugar de eso, los datos se envían como datagramas independientes, cada uno con su propia dirección de destino. UDP no garantiza la entrega de los datagramas y no tiene mecanismos de control de flujo o corrección de errores incorporados, lo que lo hace ideal para aplicaciones que requieren transmisiones de datos rápidas y eficientes, pero que pueden tolerar la pérdida ocasional de datos, como en videojuegos o transmisiones en vivo.

Message Queuing Protocol

El protocolo MQTT (Message Queuing Telemetry Transport) es un protocolo de aplicación utilizado en el nivel de transporte del modelo TCP/IP. MQTT es un protocolo de mensajería ligero diseñado para enviar mensajes en situaciones de ancho de banda limitado y conexiones de red inestables. Es utilizado en aplicaciones de IoT para comunicar dispositivos con servidores.

El protocolo MQTT utiliza el protocolo TCP como capa de transporte, y se centra en la transferencia de mensajes en lugar de la conexión y autenticación. MQTT también utiliza el modelo de publicación/suscripción, donde los clientes pueden publicar mensajes a un tema y los suscriptores pueden recibir los mensajes de ese tema.

En resumen, MQTT es un protocolo de mensajería ligero utilizado en el nivel de transporte del modelo TCP/IP, que se enfoca en la transferencia de mensajes en situaciones de ancho de banda limitado y conexiones de red inestables, utilizando el modelo de publicación/suscripción y el protocolo TCP como capa de transporte.

Hypertext Transfer Protocol Secure

HTTP (Hypertext Transfer Protocol) es un protocolo de capa de aplicación que se ejecuta en la cima del modelo TCP/IP. Se utiliza para transmitir información en la World Wide Web (WWW). HTTP define cómo los mensajes son formulados y transmitidos, y cómo los servidores y navegadores responden a los diversos comandos.

Cuando un cliente, como un navegador web, quiere acceder a una página web, envía una solicitud HTTP al servidor web que aloja esa página. El servidor web luego responde con un mensaje HTTP que contiene la página web solicitada. HTTP es un protocolo sin estado, lo que significa que cada solicitud es independiente y no se mantiene información sobre la sesión entre las solicitudes.

HTTP utiliza los puertos 80 para las comunicaciones HTTP normales y el puerto 443 para las comunicaciones HTTPS seguras. HTTPS (HTTP seguro) es una extensión del protocolo HTTP que utiliza SSL/TLS para proporcionar una capa de seguridad adicional al cifrar las comunicaciones entre el cliente y el servidor.

Protocolos por analizar si estan involucrados en algun punto de la arquitectura

WebSocket: WebSocket is a communication protocol designed for bi-directional, real-time communication between IoT devices and a dashboard. It can be used to provide real-time updates to the dashboard and enable real-time control of IoT devices.

RESTful API: Representational State Transfer (REST) is an architectural style used for creating web services. RESTful APIs can be used to transfer data between IoT devices and a web dashboard.

GraphQL API https://hub.qovery.com/guides/tutorial/deploy-fullstack-application-composed-of-hasura-postgresql-angular

2.2. Tecnologías *full-stack*

A partir de los modelos más relevantes que se encuentran establecidos para el desarrollo e implementación de aplicaciones de software se consideran el *stack web* y el *stack IoT*. A pesar de las similitudes entre ambos, en la aplicación el *stack web* emplea el protocolo *HTTP* mientras que el *stack IoT* utiliza el protocolo *MQTT*; siendo este último el más adecuado para escenarios donde son limitados los recursos como el ancho de banda y el consumo de energía.

En especial, el protocolo *MQTT* dispone de mensajes más livianos y además es posible transmitir y/o recibir datos en formato binario sin la necesidad de una codificación previa. También, este protocolo permite asignar niveles de calidad de servicio (*QoS*) a los mensajes transmitidos, resultando una característica primordial en aplicaciones donde la probabilidad de pérdidas de paquetes es considerable.

La figura 1 presenta la estratificación de las capas, según el *stack IoT*, que son detalladas a lo largo de esta sección.

2.2.1. Tecnologías del *front-end*

TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft que se basa en JavaScript. Es una extensión de JavaScript que agrega tipos estáticos opcionales y características de programación orientada a objetos avanzadas. TypeScript se compila a JavaScript y se puede usar en cualquier navegador o entorno de servidor que admita JavaScript. Permite una mayor seguridad y mantenibilidad en el código, lo que lo convierte en una opción popular para proyectos de gran escala y equipos de desarrollo. Además, se integra con muchos editores de código populares y frameworks, lo que facilita su adopción en proyectos existentes.

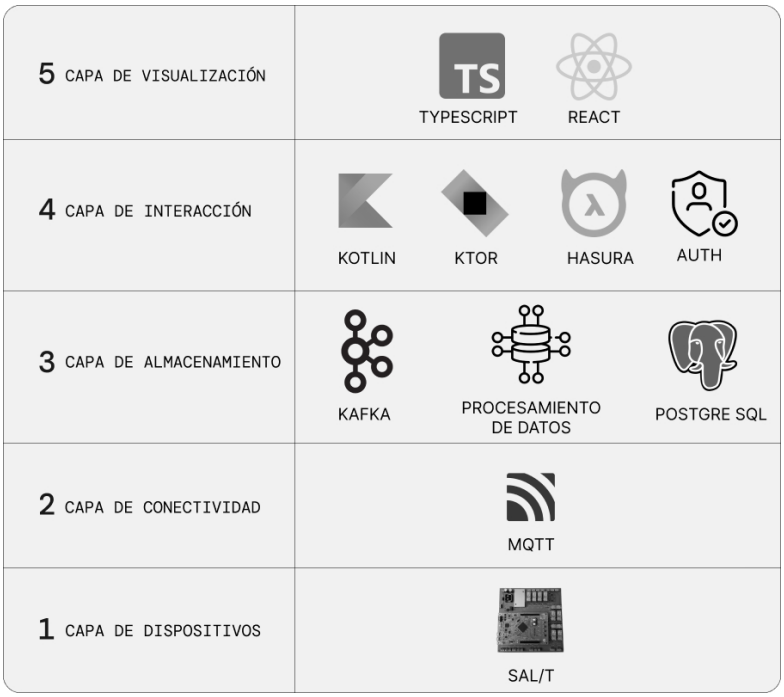

FIGURA 2.1. Arquitectura de la Central Operativa SAL/T.

React

React es una biblioteca de JavaScript de código abierto para construir interfaces de usuario (UI) interactivas y reutilizables. Fue desarrollado por Facebook y se ha convertido en una de las bibliotecas más populares para el desarrollo de aplicaciones web. React utiliza un enfoque basado en componentes para construir UI, lo que significa que los desarrolladores pueden crear piezas de UI reutilizables y componerlas para crear interfaces más complejas. React utiliza el DOM virtual para mejorar el rendimiento, lo que significa que solo actualiza las partes de la UI que necesitan ser actualizadas, en lugar de volver a dibujar toda la UI. React también se integra bien con otras bibliotecas y frameworks, lo que lo hace muy flexible y extensible.

Parcel

Parcel es una herramienta de construcción de código abierto que se utiliza para empaquetar y optimizar módulos de JavaScript y otros tipos de archivos. Ofrece una amplia gama de características útiles, como integración con hot module replacement (HMR), optimización de imágenes y capacidad de construcción paralela. Una de las principales ventajas de Parcel es su estrategia de çero configuración", que hace que sea fácil de usar sin necesidad de una configuración adicional. Parcel también es altamente configurable y tiene una amplia compatibilidad con diferentes tecnologías y frameworks, como React, Vue y TypeScript. En general, Parcel es una herramienta de construcción rápida, fácil de usar y altamente personalizable que puede mejorar significativamente el flujo de trabajo del desarrollador.

Apollo

Apollo es una plataforma de código abierto para el desarrollo de aplicaciones GraphQL. Ofrece una amplia gama de herramientas y servicios para ayudar a los desarrolladores a construir y mantener aplicaciones GraphQL de manera efectiva. Con Apollo, los desarrolladores pueden crear fácilmente clientes y servidores GraphQL, lo que les permite interactuar con diferentes fuentes de datos de manera eficiente. Además, Apollo tiene una amplia compatibilidad con diferentes frameworks y tecnologías, como React, Angular, Vue y Node.js. Apollo también ofrece características útiles, como la administración de caché, que pueden mejorar significativamente el rendimiento de las aplicaciones. En general, Apollo es una plataforma completa y fácil de usar que puede facilitar el proceso de desarrollo de aplicaciones GraphQL.

Material UI

Material UI es una biblioteca de componentes de interfaz de usuario de código abierto basada en el lenguaje de diseño Material Design de Google. Ofrece una amplia gama de componentes preconstruidos que se pueden utilizar para construir aplicaciones web modernas y atractivas. Los componentes de Material UI están altamente personalizados y pueden ser fácilmente adaptados a diferentes estilos y necesidades de diseño. Además, Material UI tiene una amplia compatibilidad con diferentes tecnologías y frameworks, como React, Angular y Vue. Material UI también ofrece características útiles, como soporte para temas personalizados y una interfaz de usuario responsiva que se adapta automáticamente a diferentes dispositivos. En general, Material UI es una biblioteca de componentes completa y fácil de usar que puede mejorar significativamente el proceso de desarrollo de aplicaciones web.

JWT Decode

JWT Decode es una biblioteca JavaScript de código abierto que se utiliza para decodificar tokens JSON Web Token (JWT). JWT Decode se puede utilizar para leer la información codificada en un token JWT, como el nombre de usuario y los roles de acceso. La biblioteca es fácil de usar y se puede integrar fácilmente en diferentes aplicaciones web y frameworks de JavaScript, como React y Angular. JWT Decode también ofrece características útiles, como la validación de tokens y la verificación de la firma, que pueden ayudar a garantizar la seguridad de la aplicación. En general, JWT Decode es una herramienta útil para decodificar y leer tokens JWT en aplicaciones de JavaScript.

2.2.2. Tecnologías del *backend*

Kotlin

Kotlin es un lenguaje de programación de tipado estático, desarrollado por JetBrains, que puede correr sobre la máquina virtual de Java y también ser compilado en JavaScript y LLVM. Se caracteriza por ser un lenguaje moderno, conciso y seguro, que simplifica la programación en comparación con Java. Kotlin es utilizado en diversas aplicaciones, como Android, backend y frontend web, procesamiento de datos y desarrollo de aplicaciones para escritorio. Además, Kotlin cuenta con una gran comunidad de desarrolladores que lo respaldan, lo que ha llevado a que sea adoptado por empresas como Google, Netflix y Amazon. Otras de las características de Kotlin son su interoperabilidad con Java, soporte para programación funcional, orientación a objetos y manejo de nulos seguro. Kotlin ha ganado popularidad en los últimos años debido a su facilidad de uso, legibilidad y capacidad de reducir la cantidad de código necesaria para realizar tareas complejas.

Ktor

Ktor es un framework web de Kotlin que permite crear aplicaciones web y API REST de manera fácil y eficiente, con un enfoque en la programación funcional. Ktor es altamente modular y personalizable, lo que significa que los desarrolladores pueden seleccionar solo los componentes que necesitan para su aplicación, reduciendo así la complejidad y el tamaño del código. Algunas de las características de Ktor son su enfoque en la concurrencia y la eficiencia, soporte para diferentes tipos de autenticación y autorización, integración con librerías como Jackson y Koin, y facilidad para trabajar con bases de datos. Ktor es utilizado por empresas como JetBrains, Netflix y Pinterest, y ha ganado popularidad en la comunidad de desarrolladores de Kotlin debido a su facilidad de uso, alto rendimiento y capacidad de adaptarse a diferentes tipos de aplicaciones. Ktor es también compatible con diferentes plataformas, como Android y iOS, lo que lo hace ideal para proyectos que necesitan un backend escalable y eficiente.

Kafka

Kafka es una plataforma de streaming de datos, desarrollada por Apache, que permite a las aplicaciones enviar y recibir datos en tiempo real, de manera escalable, confiable y eficiente. Kafka se basa en el modelo de publicación/suscripción, en el que los productores envían mensajes a un "temaz los consumidores reciben los mensajes de ese tema. Kafka se caracteriza por su alta velocidad y rendimiento, lo que lo hace ideal para aplicaciones que necesitan procesar grandes cantidades de datos en tiempo real, como análisis de datos, procesamiento de eventos y monitoreo de aplicaciones. Otras de las características de Kafka son su alta disponibilidad, escalabilidad horizontal, seguridad y facilidad para integrarse con otras herramientas y sistemas.

Hasura

Hasura es una plataforma de desarrollo de aplicaciones que permite a los desarrolladores crear, ejecutar y escalar aplicaciones con GraphQL de manera rápida y sencilla. Hasura se basa en una arquitectura sin servidor, lo que significa que los desarrolladores no tienen que preocuparse por el aprovisionamiento y la gestión de servidores, lo que reduce significativamente el tiempo de desarrollo. Hasura también ofrece una capa de seguridad de extremo a extremo, lo que permite a los desarrolladores proteger fácilmente sus datos y aplicaciones. Otras características de Hasura incluyen su facilidad para integrarse con diferentes bases de datos, incluyendo Postgres, MySQL y MongoDB, y su capacidad para generar automáticamente una API de GraphQL a partir de una base de datos existente.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto y gratuito, utilizado por miles de empresas y organizaciones en todo el mundo. PostgreSQL se caracteriza por su fiabilidad, escalabilidad y seguridad, y es conocido por ser una de las bases de datos más poderosas y versátiles disponibles en la actualidad. PostgreSQL es compatible con una amplia gama de sistemas operativos y lenguajes de programación, y cuenta con una amplia variedad de herramientas y extensiones que hacen que sea fácil de usar y personalizar. Entre las características más destacadas de PostgreSQL se encuentran su soporte para transacciones ACID, su capacidad para manejar grandes volúmenes de datos y su integración con lenguajes de programación populares como Java, Python y PHP.

Mosquitto Broker

Mosquitto es un broker MQTT de código abierto que se utiliza para transmitir mensajes entre dispositivos de Internet de las cosas (IoT) y otras aplicaciones. Mosquitto permite a los dispositivos conectarse y publicar o suscribirse a temas específicos, lo que permite una comunicación eficiente y en tiempo real. Es una herramienta popular en la industria del IoT debido a su escalabilidad y facilidad de uso. Mosquitto es compatible con una amplia variedad de plataformas, incluyendo Linux, Windows y macOS, y cuenta con una API de programación en C que permite a los desarrolladores integrar fácilmente la funcionalidad del broker MQTT en sus aplicaciones. Además, Mosquitto también tiene características de seguridad como autenticación y cifrado de datos, lo que lo hace adecuado para entornos de producción. Mosquitto es mantenido por la Eclipse Foundation y es utilizado por empresas como IBM, Amazon y Google en proyectos de IoT y M2M (machine-to-machine).

2.2.3. Tecnologías del *firmware*

C lang

C es un lenguaje de programación de propósito general que se utiliza ampliamente en el desarrollo de sistemas operativos, aplicaciones de bajo nivel y dispositivos embebidos. Fue desarrollado originalmente en la década de 1970 y ha sido una de las lenguas de programación más influyentes en la historia de la informática. C es conocido por ser un lenguaje de programación estructurado que permite a los desarrolladores escribir código claro y legible, con una sintaxis simple y directa. Ofrece un gran control sobre la memoria y la capacidad de acceder directamente al hardware. C es también un lenguaje de programación portátil, lo que significa que el código escrito en C puede ser compilado en diferentes plataformas y sistemas operativos. C se ha utilizado para crear algunos de los sistemas operativos y programas más importantes del mundo, como Unix, el kernel de Linux y la mayoría de los sistemas operativos de Apple. Es un lenguaje de programación esencial para cualquier persona que busque una carrera en la programación de sistemas y aplicaciones de bajo nivel.

FreeRTOS

FreeRTOS es un sistema operativo en tiempo real (RTOS) de código abierto y gratuito que se utiliza para controlar sistemas embebidos y microcontroladores. Es ampliamente utilizado en aplicaciones de control industrial, automoción, electrónica de consumo y aeroespacial, debido a su capacidad para proporcionar un rendimiento confiable y predecible en tiempo real. FreeRTOS es conocido por ser altamente portátil, lo que permite a los desarrolladores escribir código que se puede compilar para diferentes plataformas y arquitecturas de hardware. Ofrece una variedad de funcionalidades, como gestión de tareas, semáforos, colas y temporizadores, lo que permite a los desarrolladores crear sistemas complejos con una gestión eficiente de los recursos. FreeRTOS es fácil de usar y cuenta con una gran comunidad de desarrolladores que ofrecen soporte y recursos. Es una opción popular para proyectos de sistemas embebidos que requieren una solución RTOS de alto rendimiento y confiabilidad.

Paho MQTT *client*

Paho MQTT es una biblioteca de código abierto para implementar el protocolo MQTT (Message Queuing Telemetry Transport) en aplicaciones de IoT y sistemas embebidos. La biblioteca Paho proporciona una API consistente para programar clientes MQTT en diferentes plataformas, como Java, C, C++, Python, Javascript, entre otras. Paho MQTT permite la conexión y la comunicación bidireccional entre dispositivos IoT y aplicaciones, lo que facilita la implementación de soluciones de IoT en diferentes contextos. La biblioteca es altamente configurable y permite ajustar varios parámetros, como la calidad de servicio, la retención de mensajes y los tiempos de espera. Paho MQTT también proporciona herramientas para monitorear y depurar las comunicaciones MQTT, lo que facilita la identificación y solución de problemas en el sistema. La biblioteca se mantiene activamente y cuenta con una gran comunidad de desarrolladores que contribuyen a su mejora y ofrecen soporte.

2.3. Herramientas utilizadas

Docker

Docker es una plataforma de software libre que permite crear, distribuir y ejecutar aplicaciones en contenedores de software. Los contenedores de Docker encapsulan aplicaciones y todas sus dependencias en un paquete independiente que se puede ejecutar en cualquier entorno, independientemente del sistema operativo o la infraestructura subyacente. Docker utiliza tecnologías como el kernel de Linux y las redes virtuales para garantizar que los contenedores sean livianos y portables. Los contenedores de Docker se ejecutan en un entorno aislado y seguro, lo que ayuda a prevenir conflictos entre aplicaciones y mejora la seguridad de los sistemas. Docker es una herramienta esencial para la creación y el despliegue de aplicaciones en la nube, ya que facilita el proceso de implementación y mejora la escalabilidad y la eficiencia de los sistemas. La plataforma cuenta con una gran comunidad de desarrolladores que contribuyen con plugins y herramientas adicionales, y que ofrecen soporte y documentación para su uso.

Docker Compose

Docker Compose es la herramienta que permite ejecutar aplicaciones multi-contenedor en Docker se utiliza para definir las relaciones entre los contenedores de la aplicación. Con ella, es posible crear un archivo YAML que especifica la configuración de cada servicio, las redes y volúmenes que se utilizarán para comunicar y almacenar datos entre los contenedores. Gracias a esto, podemos coordinar eficientemente múltiples servicios como bases de datos, servidores web, balanceadores de carga, etc. en una o varias máquinas.

Redpanda

Redpanda es una plataforma de streaming de datos de alta velocidad que ofrece una alternativa escalable y confiable a los sistemas de streaming de datos existentes. Permite el procesamiento en tiempo real de grandes cantidades de datos, lo que lo hace ideal para su uso en aplicaciones de transmisión de datos en vivo, análisis de datos, aprendizaje automático y más. Además, Redpanda es altamente resistente a fallos, lo que significa que puede manejar situaciones inesperadas y garantizar la continuidad del servicio en todo momento. También ofrece herramientas integradas de monitoreo y gestión, lo que facilita la administración de la plataforma de manera eficiente.

Git

Git es un sistema de control de versiones de software de código abierto que permite a los desarrolladores rastrear y administrar los cambios en el código fuente de sus proyectos. Con Git, los desarrolladores pueden colaborar en proyectos de software de manera efectiva, ya que pueden trabajar en diferentes versiones del código simultáneamente y fusionar los cambios de manera transparente. Git utiliza una estructura de árbol para registrar los cambios en el código y tiene una gran cantidad de características avanzadas, como la creación de ramas, el etiquetado y la reversión de cambios. Git es una herramienta esencial para la gestión de proyectos de software y es utilizado por millones de desarrolladores en todo el mundo.

MQTT.fx

MQTT.fx es una herramienta de escritorio gratuita y de código abierto para la visualización y la prueba de clientes MQTT. Esta aplicación permite a los usuarios conectarse a brokers MQTT y suscribirse a diferentes temas para recibir mensajes en tiempo real. Además, MQTT.fx proporciona una interfaz gráfica de usuario intuitiva para la publicación de mensajes MQTT y la configuración de parámetros avanzados de MQTT. Es una herramienta útil para los desarrolladores y administradores de sistemas que necesitan depurar y probar aplicaciones MQTT, así como para aquellos que desean explorar y aprender sobre el protocolo MQTT.

CLion

CLion es un entorno de desarrollo integrado (IDE) multiplataforma y de alta calidad para el lenguaje de programación C y C++. Ofrece una amplia gama de características avanzadas, como la integración con herramientas externas, el depurador avanzado, la finalización automática de código y la refactorización de código inteligente. CLion también admite múltiples sistemas de construcción y está diseñado para trabajar con una variedad de herramientas populares, como CMake, Makefile y otros.

Wireshark

Wireshark es una herramienta de análisis de redes de código abierto y gratuita que permite capturar y analizar el tráfico de red en tiempo real. Con Wireshark, los usuarios pueden ver el tráfico de red en detalle y analizar protocolos en profundidad, lo que facilita la resolución de problemas en redes de datos y la detección de vulnerabilidades de seguridad. Ofrece una interfaz de usuario gráfica y es compatible con diferentes plataformas, como Windows, Mac y Linux. Es una herramienta útil para los profesionales de redes y seguridad, así como para los entusiastas de la tecnología que desean profundizar en el análisis de redes.

Herramientas del navegador de internet

Las herramientas del navegador de Internet son fundamentales para el desarrollo web. La consola es una herramienta útil para ver y depurar errores en el código JavaScript, y también para interactuar con la página en tiempo real. La pestaña de red es esencial para el análisis de la carga de recursos de la página, permitiendo ver las solicitudes HTTP y el tiempo de carga de cada recurso. Por último, la pestaña de aplicación permite ver y modificar los datos almacenados en el almacenamiento local y las cookies de la página web.

JTAG

JTAG (Joint Test Action Group) es un estándar para la depuración y programación de dispositivos electrónicos. Permite acceder a los pines de prueba de un dispositivo para realizar pruebas de hardware y depuración a nivel de circuito. JTAG se utiliza en una amplia variedad de dispositivos, desde chips integrados en dispositivos móviles hasta procesadores de computadoras de escritorio. También se utiliza en la programación de dispositivos de Internet de las cosas (IoT) y en la depuración de tarjetas electrónicas. JTAG se ha convertido en un estándar común

para la depuración y programación de dispositivos, y muchas herramientas de desarrollo de software y hardware lo admiten.

ST-Link

ST-Link es una herramienta de programación y depuración de microcontroladores fabricada por STMicroelectronics. Se utiliza para programar y depurar microcontroladores STM32 y otros dispositivos compatibles con JTAG o SWD. La herramienta se conecta al ordenador mediante USB y se puede usar con una variedad de entornos de desarrollo integrados (IDE) y herramientas de depuración. ST-Link es una opción popular y accesible para los desarrolladores que trabajan con microcontroladores STM32.

STM32CubeMX

STM32CubeMX es una herramienta de software que permite la configuración rápida y fácil de los microcontroladores STM32 de STMicroelectronics. Con esta herramienta se puede generar código inicial para proyectos y configurar periféricos de manera visual. Además, incluye un amplio conjunto de ejemplos de código y documentación para ayudar en el desarrollo de proyectos. STM32CubeMX también ofrece una integración sencilla con el entorno de desarrollo integrado (IDE) de STM32, lo que facilita el flujo de trabajo para los desarrolladores.

STM32CubeProgrammer - SWV

STM32CubeProgrammer es una herramienta de programación y depuración de microcontroladores STM32 de STMicroelectronics. Además de la programación y el borrado de la memoria flash, también es capaz de programar la memoria RAM y de configurar las opciones de protección de la memoria. SWV (Serial Wire Viewer) es una de las características más destacadas de STM32CubeProgrammer, que permite el monitoreo en tiempo real del comportamiento del microcontrolador a través de la línea de depuración SWD (Serial Wire Debug). SWV es útil para depurar y optimizar el código, ya que permite visualizar la ejecución de instrucciones, la utilización de la memoria y el uso de interrupciones en tiempo real.

Capítulo 3

Diseño e implementación

3.1. Descripción del sistema

PARAFRASEAR EL PRIMER PARRAFO PORQUE SE REPITE CON LA MOTIVACION DEL PROYECTO !

El *SAL/T*, según sus siglas, Sistema de Aislamiento Limitado o Total, es un sistema que le permitirá al conductor la activación y desactivación del modo aislado limitado. En este modo, el equipo permite la circulación de la formación al desactivar las señales de corte de tracción y freno de emergencia generadas por los otros subsistemas. Para que esta operación se realice de forma segura, se debe monitorear la velocidad de la formación tal que sea posible evitar que supere cierto valor máximo. Se considera un sistema crítico debido a que, en caso de fallar, puede ocasionar lesiones o muertes de personas, dañar el medio ambiente y/o generar grandes pérdidas materiales.

En trabajos anteriores se desarrollaron las primeras cinco fases de catorce que componen el ciclo de vida según la norma *UNE-EN 50126*, centradas en el relevamiento de la necesidad y la obtención de los requerimientos técnicos del sistema.

En la actualidad, existe un prototipo de este sistema que fue finalizado en el año 2019, del cual, en lo que respecta al *firmware*, se propone realizar la migración a un microcontrolador de la familia *Cortex* tipo M con mayor soporte al actual y que cuente con funcionalidades de seguridad.

Por otro lado, tal como se puede observar en el margen inferior derecho de la figura 1, se plantea el desarrollo de una central operativa. Se trata de una plataforma web que cuenta con una unidad lógica de compartición y empaquetado de software posibilitando la administación, la configuración y el monitoreo en tiempo real de la información recibida y transmitida por parte de cada dispositivo SAL/T.

En caso de ser necesario, a partir de la activación de una señal crítica provista por el artefacto *Hasler* [1] que se encuentra instalado en cada formación ferroviaria, un operario puede accionar alguno de los comandos que se listan a continuación:

- **Modo aislado total**: se habilita la tracción y se libera el freno de emergencia, independientemente de cualquier condición.
- **Modo coche en deriva**: se corta la tracción y se libera el freno de emergencia, independientemente de cualquier condición.
- **Modo parada total**: se corta la tracción y se aplica el freno de emergencia, independientemente de cualquier condición.
- **Modo intermitente**: se habilita la tracción y se aplica el freno de emergencia en ciclos de tiempo configurables.
- Anulación de comandos remotos vigentes.

Para el seguimiento de las variables, publicadas por cada dispositivo SAL/T, que permiten la visualización en la plataforma, se realiza la suscripción a un tópico del *broker MQTT* y se registra la información indexada en la base de datos de un servidor. En consecuencia, resulta de extrema importancia que la comunicación a través de *MQTT* opere sobre el protocolo de encriptación *SSL/TSL*, de modo que se exponga un considerable nivel de seguridad; de igual manera, es necesaria la utilización de certificados para el acceso a la lectura y la escritura en la base de datos.

Así mismo, es necesario que la página web ofrezca la configuración de los parámetros de cada dispositivo conectado a la formación ferroviaria, la administración de los roles y la autenticación de los usuarios.

3.1.1. Comunicación device backend

MQTT

Ver cuanto se repite con el ultimo parrafo de arriba.

3.1.2. Comunicación backend frontend

HTTP

Ver cuanto se repite con el ultimo parrafo de arriba.

3.2. Arquitectura del sistema

PARAFRASEAR EL PRIMER PARRAFO PORQUE SE REPITE CON LA MOTIVACION DEL PROYECTO !

A partir de los modelos más relevantes que se encuentran establecidos para el desarrollo e implementación de aplicaciones de software se consideran el *stack web* y el *stack IoT*. A pesar de las similitudes entre ambos, en la aplicación el *stack*

[1] es un sistema electromecánico que registra los eventos de velocidad del material rodante, la señal de anuncio y la señal de frenado automático.

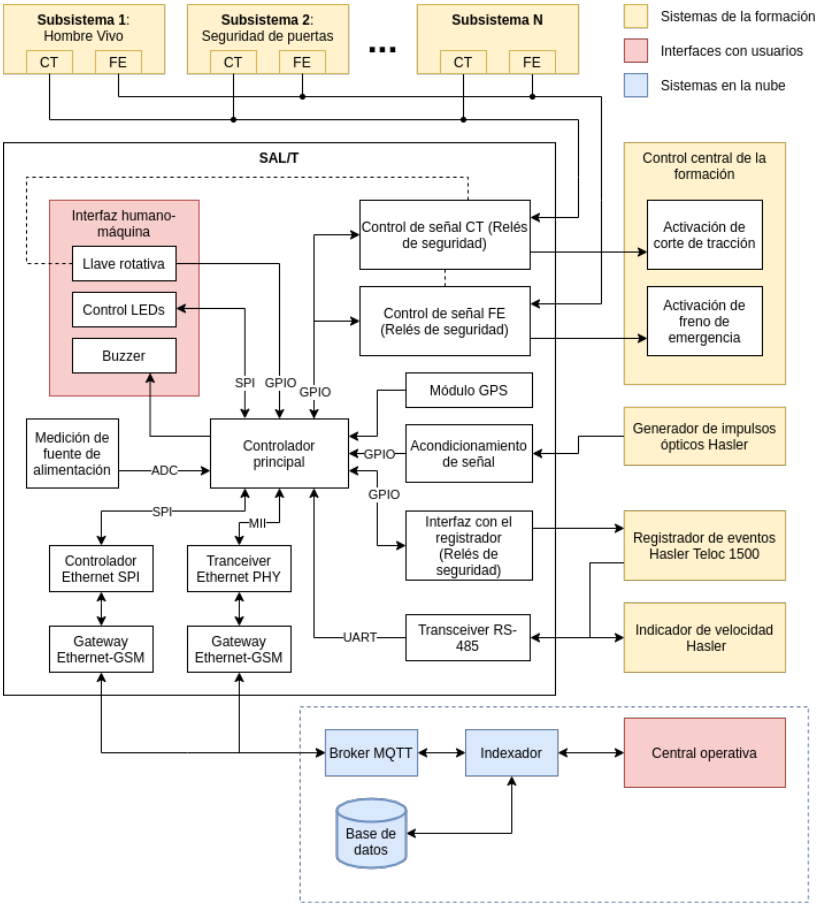

FIGURA 3.1. Diagrama en bloques del sistema SALT.

web emplea el protocolo *HTTP* [**b5**] mientras que el *stack IoT* utiliza el protocolo *MQTT* [**b6**]; siendo este último el más adecuado para escenarios donde son limitados los recursos como el ancho de banda y el consumo de energía.

En especial, el protocolo *MQTT* dispone de mensajes más livianos y además es posible transmitir y/o recibir datos en formato binario sin la necesidad de una codificación previa. También, este protocolo permite asignar niveles de calidad de servicio (*QoS*) [**b7**] a los mensajes transmitidos, resultando una característica primordial en aplicaciones donde la probabilidad de pérdidas de paquetes es considerable.

La **figura X** presenta la estratificación de las capas, según el *stack IoT*, que son detalladas a lo largo de esta sección.

AGREGAR NUEVA FIGURA SUPER GENERAL DEL STACK IOT CON SUS CAPAS

3.2.1. Capa de dispositivos

Sobre la base de un prototipo del Sistema de Aislamiento Limitado o Total, *SAL/T* [**b1**], se desarrolló mediante el kit de desarrollo *Nucleo F429* [**b8**], un cliente *MQTT* que permite la publicación de la velocidad de la formación y de los parámetros provistos por los subsistemas de falla, entre otras variables; y llegado el caso, el dispositivo pueda realizar la lectura de diferentes parámetros configurables.

3.2.2. Capa de conectividad

Para la comunicación entre los dispositivos *SAL/T* y la capa de almacenamiento se emplea un *Broker MQTT* que oficia de orquestador entre el cliente, el dispositivo *SAL/T* que publica los mensajes y *Apache Kafka* [**b9**], un sistema distribuido que efectúa la lectura de la información y su posterior procesamiento.

Dado que los mensajes intercambiados entre las partes contienen información sensible, se ha optado por agregar la capa de seguridad *TLS* [**b10**]. En este sentido, resulta indispensable utilizar el certificado *X509* [**b11**] para prevenir ataques del tipo *adversary-in-the-middle* [**b12**] y los certificados emitidos por autoridades reconocidas.

3.2.3. Capa de almacenamiento

La capa de almacenamiento se encuentra constituida por el sistema distribuido *Kafka* que se encarga de almacenar los datos de la aplicación. Entre ellos se destacan los mensajes MQTT enviados por los dispositivos SAL/T, como también la información referida a las formaciones y a los SAL/T que las ocupan. Además, se tienen servicios de procesamiento de datos que se encargan de adaptar la información de los tópicos en datos que posteriormente serán consumidos por los servicios que integran la capa superior.

Por otro lado, se tiene una base de datos relacional *PostgreSQL* [**b13**] para el almacenamiento de los datos provenientes del servicio de autenticación y el motor que facilita la interacción con la plataforma web. Los conectores se encargan de insertar los datos que se quieran disponer a la capa de visualización.

En la figura **devicesTrainEntity** se observa el modelo de datos que representa a los dispositivos SAL/T, las formaciones y las entidades. El primero cuenta con un identificador del dispositivo en el sistema, el estado de operación y una referencia al tren en el que se desplegó el artefacto. El modelo de la formación está enriquecido con la entidad a la que pertenece y el número de serie del tren. Por último, las entidades están compuestas por nombre y descripción. En la figura **devicesSubsystems** se aprecian los subsistemas de fallas relacionados con el dispositivo SAL/T. Su modelo está integrado con el nombre del subsistema, el identificador del dispositivo al que está asociado y el estado actual del mismo.

3.2.4. Capa de interacción

La capa de interacción se encuentra conformada por tres unidades. Por un lado, se ha desarrollado en el lenguaje *Kotlin* [**b14**] en conjunto con el *framework Ktor* [**b15**], un microservicio de autenticación donde es posible la gestión de los usuarios con sus respectivos roles y también el *backend*. Entre sus tareas principales se encuentran las relacionadas con el protocolo *MQTT* para la actualización de los certificados que brindan seguridad a las comunicaciones, mediante el uso de la capa de seguridad *TLS*, y la indexación de los eventos que se publiquen en tiempo real; como la configuración general que permite el funcionamiento integral de los servicios y módulos dispuestos en el sistema.

Además, se dispone del motor *Hasura* [**b16**], el cual se encuentra conectado a una base de datos relacional (*PostgreSQL*) basado en el lenguaje de consulta estructurado *SQL* [**b17**], que permite exponer una *API GraphQL* [**b18**] a aquellos clientes que deseen obtener una visualización del modelo de datos propuesto en que se conjuga la información de cada formación ferroviaria con su correspondiente dispositivo *SAL/T* y diseño destinado al *frontend*. Más aún, el enlace permite realizar modificaciones y hasta remociones de las columnas propuestas en las tablas de cada esquema dentro de la base de datos.

3.2.5. Capa de visualización

En esta capa se ha utilizado el lenguaje de programación *TypeScript* [**b19**] junto con la biblioteca gráfica *React* [**b20**] para brindar una web reactiva en la que se elaboraron los formularios, las tablas y los paneles a partir de la explotación de la información almacenada en la base de datos. Cabe destacar que la plataforma web se encuentra condicionada por el rol y la entidad a la que pertenece cada usuario, de modo que, cada formación ferroviaria tendrá un operador asignado, el cual podrá modificar los parámetros correspondientes con el dispositivo asociado.

En la figura **dashboard** se puede apreciar el panel de control de la central operativa al que tendrán acceso los operadores de cada entidad. En el mismo se observan distintas tarjetas con el estado de cada subsistema de fallas y del tren, como así también un velocímetro que informa la velocidad de la formación.

3.3. Arquitectura de datos

Para crear la estructura de la base de datos SQL, se realiza un análisis detallado del sistema y se identifican las principales entidades: la formación ferroviaria, los

subsistemas de falla y las áreas. Toda la información de estas entidades se nuclea en la entidad tren, la cual representa una formación ferroviaria con todos sus subsistemas y áreas asociadas. A partir del modelado de estas entidades y sus atributos correspondientes, se definen las relaciones entre ellas y se establecen las claves primarias y foráneas.

Para visualizar de manera más clara la estructura de la base de datos y sus relaciones, se utiliza el diagrama UML (Lenguaje de Modelado Unificado). En la siguiente figura se muestra el diagrama UML de la base de datos, donde se pueden observar las diferentes entidades, sus atributos y relaciones. A partir de este diagrama se procede a crear las tablas en la base de datos SQL y a definir sus columnas y restricciones, garantizando así la integridad de los datos y el correcto funcionamiento del sistema.

CONTINUAR CON EL DIAGRAMA UML CONECTANDO LAS ENTIDADES
VER Figures/salt_uml_drawio https://app.diagrams.net/

3.3.1. Entidades del modelo de datos

COMENTAR SI ES QUE LAS HAY, PARTICULARIDADES SOBRE CADA UNA DE LAS ETIDADES DEL SISTEMA Y CADA TABLA DEL DIAGRAMA UML

Schema auth_service

- **credentials**: Esta tabla almacena las credenciales de los usuarios para acceder al sistema. Tiene una clave foránea a la tabla **auth_service users** y es referenciada por la tabla **auth_service roles**.
- **roles**: Esta tabla almacena los roles de los usuarios. Tiene una clave foránea a la tabla **auth_service users** y es referenciada por la tabla **enums roles**.
- **users**: Esta tabla almacena la información básica de los usuarios del sistema. Es referenciada por la tabla **auth_service credentials** y la tabla **public users**.

Schema enums

- **device status**: Esta tabla almacena los estados posibles para los dispositivos. Es referenciada por la tabla **public devices**.
- **events**: Esta tabla almacena los tipos de eventos posibles. Es referenciada por la tabla **public internal events**.
- **fail system**: Esta tabla almacena los sistemas de falla posibles. Es referenciada por la tabla **public remote events** y la tabla **public subsystems**.
- **roles**: Esta tabla almacena los roles posibles para los usuarios del sistema. Es referenciada por la tabla **auth_service roles**.

Schema public

- **devices**: Esta tabla almacena información sobre los dispositivos del sistema. Tiene una clave foránea a la tabla **public train** y es referenciada por la tabla **public remote events** y la tabla **public internal events**.

- **entities**: Esta tabla almacena información sobre las entidades del sistema. Es referenciada por la tabla **public train**.
- **internal events**: Esta tabla almacena información sobre los eventos internos del sistema. Tiene claves foráneas a las tablas **public devices**, **enums events** y **public users**.
- **remote events**: Esta tabla almacena información sobre los eventos remotos del sistema. Tiene una clave foránea a la tabla **public devices** y es referenciada por la tabla **enums fail system**.
- **subsystems**: Esta tabla almacena información sobre los subsistemas del sistema. Tiene una clave foránea a la tabla **public devices** y es referenciada por la tabla **enums fail system**.
- **train**: Esta tabla almacena información sobre los trenes del sistema. Tiene una clave foránea a la tabla **public entities** y es referenciada por la tabla **public devices**.
- **user data**: Esta tabla almacena información adicional de los usuarios del sistema. Es referenciada por la tabla **public users**.
- **users**: tabla que contiene información de los usuarios. Tiene una relación de clave externa con la tabla entities.

3.4. Seguridad del sistema

3.4.1. Protocolo HTTPS

security layer

jwt

Referencia - pag 44 https://lse-posgrados-files.fi.uba.ar/tesis/LSE-FIUBA-Trabajo-Final-CEIoT-Pedro-Rosito-2021.pdf

3.4.2. MQTT v5.0

payload encryption

3.5. Desarrollo del frontend

AGREGAR UNA BREVE INTRO SOBRE QUE SE EXPONDRA AQUI *tipos de clientes, apollo client, codegen,*

3.5.1. Estructura del cliente web

1. index html
2. main tsx
3. react router ->pages

3.5.2. Páginas

HABLAR DE CADA PAGINA A PARTIR DE UNA IMAGEN DONDE SE MUESTRE EL SIDEBAR

dashboard, sign in, configuration, entities, trains, subsystems of each train, logs, etc

Login

hablar sobre jwt *Referecia - bastian pag 48*

add figure

Tabla de trenes+salt

add figure

Tabla de usuarios

add figure

dashboard

add figure

3.5.3. Modales

crear: usuario, tren, dispositivo ; configuracion: sub sistemas de seguridad, etc

3.6. Desarrollo del backend

A partir del uso de archivo docker-compose, que se ejecuta en un entorno Docker, para proporcionar el conjunto completo de servicios para enviar, recibir y almacenar los datos a través de diferentes protocolos. Cada servicio se ejecuta en su propio contenedor y se conecta a los otros servicios según el siguiente formato:

AGREGAR IMAGEN DEL ENTORNO DOCKER CON SUS CONTENEDORES Y CONEXIONES https://www.gotoiot.com/pages/articles/docker_intro/index.html https://www.gotoiot.com/pages/articles/docker_intro/images/image2.png

- mosquitto: es un servidor MQTT de código abierto que se utiliza para enviar y recibir mensajes. Se utiliza la imagen .eclipse-mosquittoz se mapea el puerto 1883 en el host al puerto 1883 en el contenedor. También se montan dos volúmenes, uno para la configuración del servidor y otro para los datos.
- hasura: es un motor de GraphQL que proporciona una API para interactuar con una base de datos. Se utiliza la imagen "hasura/graphql-engine:v2.1.0z se mapea el puerto 8080 en el host al puerto 8080 en el contenedor. Se configuran varias variables de entorno, incluyendo la URL de la base de datos y el secreto de administrador. También depende del servicio "postgres".
- postgres: es una base de datos relacional que se utiliza para almacenar datos. Se utiliza la imagen "postgres:13z se mapea el puerto 5432 en el host al puerto 5432 en el contenedor. Se configuran varias variables de entorno,

incluyendo el nombre de la base de datos, el nombre de usuario y la contraseña. También se monta un volumen para almacenar los datos de la base de datos.

- zookeeper: es un servicio que se utiliza para coordinar los nodos de Kafka. Se utiliza la imagen çonfluentinc/cp-zookeeper:6.2.4z se configura el puerto de cliente en 2181.
- kafka: es un servicio de mensajería de código abierto que se utiliza para enviar y recibir mensajes. Se utiliza la imagen çonfluentinc/cp-kafka:6.2.4z se mapea el puerto 9092 en el host al puerto 9092 en el contenedor. Se configuran varias variables de entorno, incluyendo la conexión de Zookeeper y el número de réplicas de las particiones. También depende del servicio "zookeeper".
- kafka-ui: es una interfaz de usuario para administrar y monitorear clústeres de Kafka. Se utiliza la imagen "provectuslabs/kafka-uiz se mapea el puerto 8085 en el host al puerto 8080 en el contenedor. Se configuran varias variables de entorno, incluyendo la conexión al clúster de Kafka, el esquema de registro y el servidor de KSQLDB.

Mosquitto Broker

TALK ABOUT THE MOSQUITTO CONFIG FILE AND THE CLI INSTALLATION, USAGE, ETC.

Hasura

TALK ABOUT HASURA GUI, CLI AND USAGE. ADD AN EXAMPLE QUERY

Kafka

TALK ABOUT KAFKA AND REDPANDA. ADD AN EXAMPLE QUERY

3.7. Desarrollo del firmware

Explicar como se combinan todas las tecnologias y herramientas en el firmware. basarlo en el stack lwip

Referecia - pagina 43 *https://lse-posgrados-files.fi.uba.ar/tesis/LSE-FIUBA-Trabajo-Final-CEIoT-Pedro-Rosito-2021.pdf*

- based on: https://blog.naver.com/eziya76/221938551688
- nucleo F429ZI kit development
- free rtos
- tools: cube mx, cube programmer, mosquitto cli

3.8. Desarrollo de la API GraphQL

Referencias

- https://github.com/nandroidj/app-fullstack-base-2021-1c
  Detalles de implementación ->Ver los endpoints disponibles
- https://www.notion.so/briken/KoyweTransaction-8d9ba1eda4664819a30378c3548ebacf?pvs=4
- https://www.notion.so/briken/User-2ad1013d829548d0a7e992e31e53f7df?pvs=4

Bibliografía

[1] 'Sistema de supervisión de la seguridad del material ferroviario utilizando patrones de diseño', Ivan Mariano Di Vito, Pablo Gomez, Ariel Lutenberg, Libro de trabajos del CASE2019, Congreso Argetino de Sistemas Embebidos, Santa Fe, Argentina. (2019).

[2] CONICET-GICSAFe (June 2022). [Online]. Disponible: https://sites.google.com/view/conicet-gicsafe/inicio

[3] Trenes Argentinos (June 2022). [Online]. Disponible: https://www.argentina.gob.ar/transporte/trenes