



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:

Ing. 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

En la presente memoria se describe el desarrollo, para Trenes Argentinos, de una arquitectura modular basada en el paradigma de Internet de las Cosas (IoT) que permite visualizar y gestionar, por un supervisor u operario, las formaciones ferroviarias de forma remota desde una central operativa. De esta manera, se logra mejorar la seguridad, la eficiencia y aumentar la flexibilidad del sistema ferroviario.

El sistema se trata de una solución fullstack embebida en donde se expone cada uno de los componentes de hardware y software involucrados. Entre los conocimientos aplicados se destacan el diseño de una arquitectura cliente-servidor, el diseño de una API GraphQL, la interacción con un broker de mensajes y la seguridad en las comunicaciones.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general

En este capítulo se presentará una breve introducción sobre la inserción de la Internet de las cosas (*IoT*)¹ en el ámbito ferroviario, abarcando las motivaciones que llevaron a la realización de este trabajo y el estado del arte en cuanto a esta temática. Además, se establecerán los alcances y objetivos del trabajo, los cuales se enfocan en diseñar e implementar un sistema IoT que permita monitorear y controlar los componentes de una formación ferroviaria, con el fin de mejorar la eficiencia y seguridad en el transporte ferroviario.

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.

1.2. 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

¹<https://www.investopedia.com/terms/i/internet-things.asp>

señales de corte de tracción (CT) y freno 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.

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 cuenta 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 trabajo se desarrolla por CONICET-GICSAFe [2] 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.

1.3. 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*²: 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*³: 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*⁴: 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*⁵: 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.

1.4. 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.

²[urlhttps://www.indracompany.com](https://www.indracompany.com)

³[urlhttps://www.thalesgroup.com](https://www.thalesgroup.com)

⁴[urlhttps://www.alstom.com/](https://www.alstom.com/)

⁵[urlhttps://www.mobility.siemens.com/](https://www.mobility.siemens.com/)

- Realizar la migración de un microcontrolador de la familia Cortex tipo M⁶ a uno de la misma familia que disponga de características destinadas a la seguridad funcional del sistema.

⁶<https://developer.arm.com/Processors/Cortex-M4>

Capítulo 2

Introducción específica

2.1. Protocolos de comunicación utilizados

Comenzar escribiendo que aqui se detallaran los protocolos utilizados segun el modelo TCP/IP 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

De los modelos *stack IoT* y *lwIP* [4], basados en el stack *TCP/IP*¹, se emplean los siguientes protocolos de comunicación tal como se puede observar en la tabla ??.

TABLA 2.1. Protocolos de comunicación utilizados en cada capa del *stack TCP/IP*

Capa	Protocolo
Capa de Aplicación	HTTP [5], DHCP [6] y MQTT [7]
Capa de Transporte	UDP [8]
Capa de Red	IPv4 [9], ARP [10]
Capa de Enlace de Datos	Ethernet [11]

Cabe destacar que ambos modelos se encuentran diseñados para sistemas con recursos limitados y buscan garantizar una correcta comunicación entre los dispositivos conectados en la red.

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

¹<https://www.ibm.com/docs/en/aix/7.2?topic=protocol-tcpip-protocols>

de servicio (QoS^2) 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.

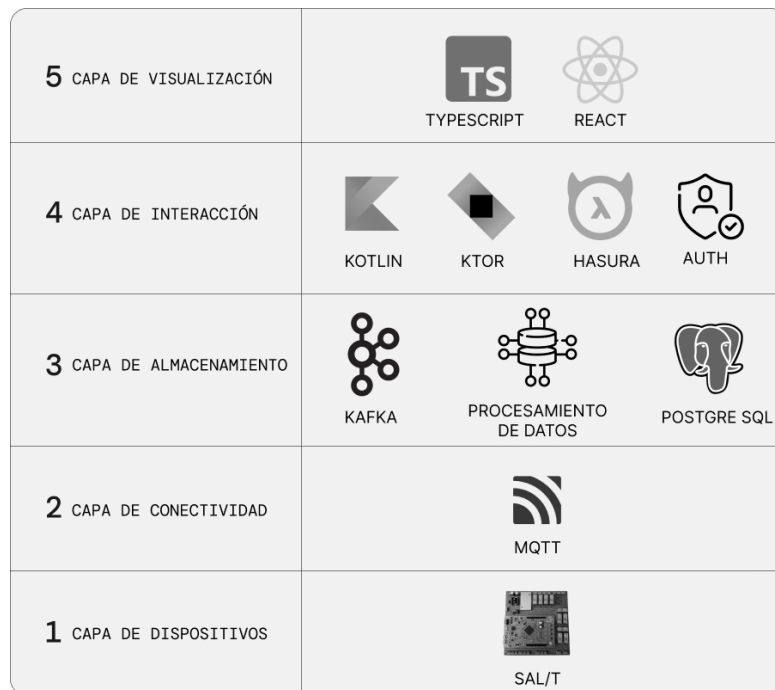


FIGURA 2.1. Arquitectura de la Central Operativa SAL/T.

²<https://www.ibm.com/docs/en/i/7.2?topic=services-quality-service>

2.2.1. Tecnologías del *front-end*

TypeScript

TypeScript [12] es una extensión de código abierto de JavaScript que agrega tipos estáticos opcionales y características de programación orientada a objetos avanzadas, lo que permite una mayor seguridad y mantenibilidad en el código.

React

React [13] es una biblioteca de JavaScript para construir UI reutilizables e interactivas, desarrollada por Facebook y ampliamente utilizada en el desarrollo web. Utiliza un enfoque basado en componentes y el DOM virtual para mejorar el rendimiento y es altamente integrable con otras bibliotecas y frameworks.

Parcel

Parcel [14] es una herramienta de construcción de código abierto con una estrategia de "cero configuración" que optimiza y empaqueta módulos de JavaScript y otros archivos. En particular, mejora significativamente el flujo de trabajo del desarrollador al ser rápido, fácil de usar y altamente personalizable.

Apollo GraphQL

Apollo GraphQL [15] es una plataforma de código abierto para el desarrollo de aplicaciones GraphQL, con una amplia gama de herramientas y servicios para construir y mantener aplicaciones GraphQL de manera efectiva, ofreciendo una administración de caché y una amplia compatibilidad con diferentes frameworks y tecnologías.

Material UI

Material UI [16] es una biblioteca de componentes de interfaz de usuario de código abierto basada en Material Design de Google, que ofrece componentes pre-construidos altamente personalizados y una amplia compatibilidad con diferentes tecnologías y frameworks para construir aplicaciones web modernas y atractivas.

JWT Decode

JWT Decode [17] es una biblioteca JavaScript que decodifica tokens JWT, con características útiles como validación de tokens y verificación de firma. Se integra fácilmente en diferentes frameworks de JavaScript como React y Angular.

2.2.2. Tecnologías del *backend*

Kotlin

Kotlin [18] es un lenguaje de programación de tipado estático que puede correr sobre la máquina virtual de Java y ser compilado en JavaScript y LLVM. También ofrece interoperabilidad con Java, programación funcional, orientación a objetos y manejo de nulos seguro, lo que ha llevado a su popularidad debido a su facilidad de uso y legibilidad.

Ktor

Ktor [19] 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.

Kafka

Kafka [20] es una plataforma de streaming de datos que permite a las aplicaciones enviar y recibir datos en tiempo real. Basada en el modelo de publicación/-suscripción, se caracteriza por su alta velocidad y rendimiento, haciéndola ideal para aplicaciones que necesitan procesar grandes cantidades de datos en tiempo real.

Hasura

Hasura [21] es una plataforma sin servidor que permite a los desarrolladores crear y escalar aplicaciones con GraphQL rápidamente. Ofrece seguridad de extremo a extremo y se integra fácilmente con diferentes bases de datos, generando automáticamente una API de GraphQL.

PostgreSQL

PostgreSQL [22] es un sistema de gestión de bases de datos de código abierto y gratuito, conocido por su fiabilidad, escalabilidad y seguridad. Ofrece soporte para transacciones ACID, integración con lenguajes de programación populares y una amplia variedad de herramientas y extensiones.

Mosquitto Broker

Mosquitto [23] es un broker MQTT de código abierto utilizado en IoT para transmitir mensajes entre dispositivos. Es popular por su escalabilidad, facilidad de uso y características de seguridad.

2.2.3. Tecnologías del *firmware*

C lang

C [24] es un lenguaje de programación estructurado y de propósito general, ampliamente utilizado en sistemas operativos, aplicaciones de bajo nivel y dispositivos embebidos. Su sintaxis simple y directa permite escribir código claro y legible, y ofrece gran control sobre la memoria y acceso directo al hardware.

FreeRTOS

FreeRTOS [25] es un RTOS de código abierto y gratuito que controla sistemas embebidos y microcontroladores, utilizado en control industrial, automoción y electrónica de consumo, con capacidad de rendimiento confiable y predecible. Es altamente portátil y ofrece gestión de tareas, semáforos, colas y temporizadores para sistemas complejos y eficientes en recursos.

Paho MQTT *client*

Paho [26] es una biblioteca cliente MQTT de código abierto que se utiliza para conectar aplicaciones a un broker MQTT. Paho es compatible con una amplia variedad de plataformas y lenguajes de programación, y ofrece una API sencilla para publicar y suscribir mensajes. Es ampliamente utilizado en aplicaciones de IoT y M2M para la transmisión eficiente de datos.

2.2.4. Herramientas utilizadas

Docker

Docker [27] es una plataforma de software que permite a los desarrolladores crear, desplegar y ejecutar aplicaciones en contenedores. Estos contenedores permiten que las aplicaciones se ejecuten de manera aislada del sistema operativo y otras aplicaciones, lo que facilita su portabilidad y escalabilidad.

Docker Compose

Docker Compose [28] es una herramienta que permite definir y ejecutar aplicaciones de múltiples contenedores Docker. Permite a los desarrolladores especificar los servicios y la configuración de cada contenedor en un archivo YAML para simplificar la creación, ejecución y gestión de aplicaciones complejas.

Redpanda

Redpanda [29] es una plataforma de streaming de datos distribuida y de alto rendimiento que combina las funcionalidades de Kafka y Redis. Es una solución escalable y confiable para el procesamiento de datos en tiempo real en entornos empresariales, y es compatible con una amplia variedad de casos de uso, desde el análisis de datos hasta la inteligencia artificial y el aprendizaje automático.

Git

Git [30] es un sistema de control de versiones distribuido que se utiliza para rastrear los cambios en el código fuente de un proyecto de software. Permite a los

desarrolladores trabajar en colaboración en el mismo código fuente y hacer un seguimiento de las diferentes versiones y ramas del proyecto.

MQTT.fx

MQTT.fx [31] es una herramienta de escritorio de código abierto que se utiliza para probar y depurar conexiones MQTT. Ofrece una interfaz gráfica de usuario fácil de usar para interactuar con brokers MQTT y suscribirse a temas y mensajes. MQTT.fx es compatible con una variedad de características de seguridad, como TLS/SSL y autenticación, lo que lo hace adecuado para su uso en entornos de producción.

CLion

CLion [32] es un entorno de desarrollo integrado (IDE) para programar en C y C++, que ofrece herramientas para la edición de código, depuración, refactorización y gestión de proyectos. Es conocido por su alta capacidad de análisis estático de código, lo que permite a los desarrolladores encontrar errores de forma eficiente.

Wireshark

Wireshark [33] es una herramienta de análisis de redes de código abierto y gratuita, utilizada para capturar y analizar paquetes de datos en tiempo real. Permite examinar el tráfico de red para identificar problemas de rendimiento, seguridad y configuración, y es compatible con una amplia variedad de protocolos de red.

Herramientas del navegador de internet

Las herramientas del navegador de Internet son características incorporadas en los navegadores web que permiten a los usuarios analizar y modificar elementos de una página web, como su estructura HTML, CSS y JavaScript. Algunas herramientas comunes incluyen la consola de desarrollador, el inspector de elementos, el depurador de JavaScript y el analizador de red, que ayudan a los desarrolladores a depurar problemas y mejorar la calidad de sus sitios web.

JTAG

JTAG (Joint Test Action Group) [34] 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 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 [35] 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.

STM32CubeMX

STM32CubeMX [36] es una herramienta de configuración gráfica para dispositivos STM32 que permite a los desarrolladores generar automáticamente código inicial para su proyecto. Ofrece una interfaz de usuario intuitiva que simplifica la configuración de periféricos y pines, y admite una variedad de opciones de generación de código.

STM32CubeProgrammer

STM32CubeProgrammer [37] es una herramienta de programación y actualización de firmware para dispositivos STM32 que permite programar y depurar dispositivos STM32, así como actualizar su firmware en campo. Es compatible con una variedad de interfaces de programación, como JTAG, SWD y UART, y es fácil de usar gracias a su interfaz gráfica de usuario.

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 administración, 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 y backend

MQTT

Ver cuanto se repite con el ultimo parrafo de arriba.

3.1.2. Comunicación backend y 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 web* emplea el protocolo *HTTP* mientras que el *stack IoT* utiliza el protocolo *MQTT*;

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

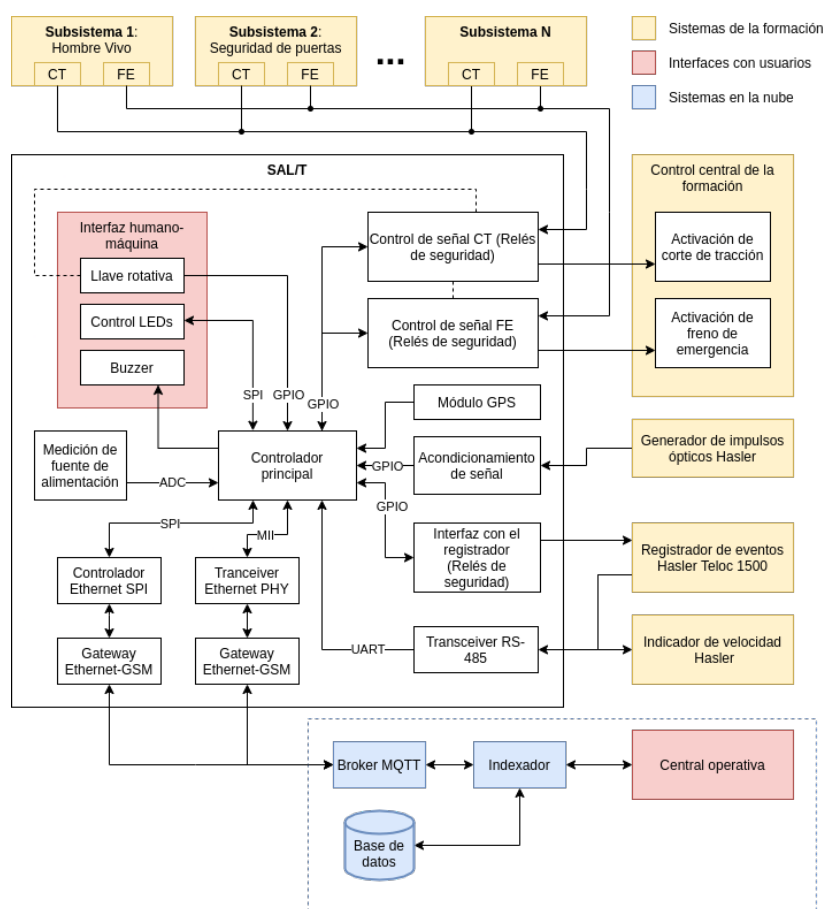


FIGURA 3.1. Diagrama en bloques del sistema SALT.

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 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*, se desarrolló mediante el kit de desarrollo *Nucleo F429 [nucleo-kit]*, 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*, 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*. En este sentido, resulta indispensable utilizar el certificado *X509* para prevenir ataques del tipo *adversary-in-the-middle* 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* 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 **devices-Subsystems** 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* en conjunto con el *framework Ktor*, 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*, el cual se encuentra conectado a una base de datos relacional (*PostgreSQL*) basado en el lenguaje de consulta estructurado *SQL*, que permite exponer una *API GraphQL* 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* junto con la biblioteca gráfica *React* 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 ENTIDADES 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 EXPONDRÁ 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-mosquitto` 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:v2.1.0` 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 confluentinc/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 confluentinc/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

Referencia - 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] Ariel Lutenberg Ivan Mariano Di Vito Pablo Gomez. «Sistema de supervisión de la seguridad del material ferroviario utilizando patrones de diseño». En: *Journal* (2019).
- [2] CONICET-GICSAFe. Online. 2023. URL: <https://sites.google.com/view/conicet-gicsafe/inicio>.
- [3] Trenes Argentinos. Online. 2023. URL: <https://www.argentina.gob.ar/transporte/trenes>.
- [4] lwIP. Online. 2023. URL: <https://savannah.nongnu.org/projects/lwip/>.
- [5] HTTP. Online. 2023. URL: <https://www.ietf.org/rfc/rfc2616.txt>.
- [6] DHCP. Online. 2023. URL: <https://www.rfc-editor.org/rfc/rfc2131>.
- [7] MQTT. Online. 2023. URL: <https://mqtt.org/>.
- [8] UDP. Online. 2023. URL: <https://www.rfc-es.org/rfc/rfc0768-es.txt>.
- [9] IPv4. Online. 2023. URL: <https://www.rfc-editor.org/rfc/rfc791>.
- [10] ARP. Online. 2023. URL: <https://tools.ietf.org/html/rfc826>.
- [11] Ethernet. Online. 2023. URL: <https://www.rfc-editor.org/rfc/rfc894>.
- [12] TypeScript. Online. 2023. URL: <https://www.typescriptlang.org/>.
- [13] React. Online. 2023. URL: <https://reactjs.org/>.
- [14] Parcel. Online. 2023. URL: <https://parceljs.org/>.
- [15] Apollo GraphQL. Online. 2023. URL: <https://www.apollographql.com/>.
- [16] Material UI. Online. 2023. URL: <https://mui.com/>.
- [17] JWT Decode. Online. 2023. URL: <https://jwt.io/>.
- [18] Kotlin. Online. 2023. URL: <https://kotlinlang.org/>.
- [19] Ktor. Online. 2023. URL: <https://ktor.io/>.
- [20] Kafka. Online. 2023. URL: <https://kafka.apache.org/>.
- [21] Hasura. Online. 2023. URL: <https://hasura.io/>.
- [22] PostgreSQL. Online. 2023. URL: <https://www.postgresql.org/>.
- [23] Eclipse Mosquitto. Online. 2023. URL: <https://mosquitto.org/>.
- [24] Lenguaje C. Online. 2023. URL: <https://www.iso.org/standard/74528.html>.
- [25] FreeRTOS. Online. 2023. URL: <https://www.freertos.org/>.
- [26] Paho MQTT Client. Online. 2023. URL: <https://www.eclipse.org/paho/>.
- [27] Docker. Online. 2023. URL: <https://www.docker.com/>.
- [28] Docker Compose. Online. 2023. URL: <https://docs.docker.com/compose/>.
- [29] Redpanda. Online. 2023. URL: <https://vectorized.io/>.
- [30] Git. Online. 2023. URL: <https://git-scm.com/>.
- [31] MQTT.fx. Online. 2023. URL: <https://softblade.de/en/mqtt-fx/>.
- [32] CLion. Online. 2023. URL: <https://www.jetbrains.com/es-es/clion/>.
- [33] Wireshark. Online. 2023. URL: <https://www.wireshark.org/>.
- [34] JTAG. Online. 2023. URL: <https://www.jtag.com/>.
- [35] ST-Link. Online. 2023. URL: <https://www.st.com/en/development-tools/st-link-v2.html>.
- [36] STM32CubeMX. Online. 2023. URL: <https://www.st.com/en/development-tools/stm32cubemx.html>.

- [37] STM32CubeProgrammer. Online. 2023. URL:
<https://www.st.com/en/development-tools/stm32cubeprog.html>.