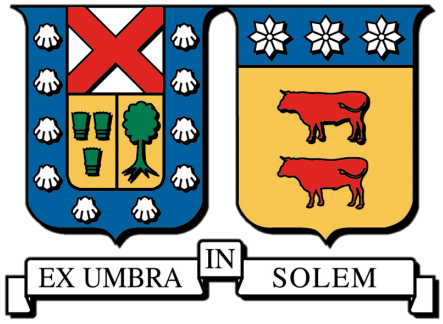
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"DESARROLLO DEL BACKEND PARA UN SISTEMA DE TRAZABILIDAD DE ACTIVOS DE TRANSMISIÓN ELÉCTRICA"

Felipe Vicente Garay Jarufe

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA: **Nicolás Jara**

PROFESOR CORREFERENTE: **Patricio Olivares**

MAYO 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Desarrollo de Backend para un sistema de trazabilidad de activos de transmisión eléctrica

Nombre del candidato(a): Felipe Vicente Garay

Carrera / Grado: Ingeniería Civil Telemática

Campus: Casa Central Valparaiso **;** **Departamento**: Electrónica / Telemática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nicolás Alonso Jara Carvallo, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 12/05/2026 ; **Firma:** _____

Estudiante o Candidato(a):

Fecha: 11/05/2026 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Para presentar esta memoria, es necesario dar las gracias a toda la gente que ha sido parte de este proceso, desde los más lejanos hasta los más cercanos. Debido a que cada persona ha aportado a lo que ha sido esta experiencia universitaria, la cual me ha reforzado una actitud de seguir adelante y aprender tanto de conocimientos académicos como del día a día.

Quiero partir dándole las gracias a todos los funcionarios de la universidad que han hecho esta experiencia más grata, en los cuales se nota la determinación de ayudarnos a crecer tanto como profesionales como personas. Los cuales reciben con puertas abiertas a los alumnos y permiten segundas oportunidades cuando los tiempos son difíciles. Unas gracias especiales a los profesores que han sido parte de este trabajo, Nicolás Jara y Patricio Olivares, ambos docentes ejemplares.

Darles las gracias a mis compañeros, los cuales saben perfectamente lo que es esta experiencia y los que han sido un gran apoyo, ya sea cuando se trata de trabajar en equipo como cuando se está estudiando para salir adelante por separado. Los que se toman su tiempo libre para explicarte las dudas y los que acompañan hasta tarde a terminar los trabajos.

Finalmente, dar un montón de gracias a mi red de apoyo más cercana. Toda mi familia con la que he vivido esta experiencia y más: especialmente a mi mamá, papá, Pedro y mi querida pareja, Kristel, a la cual gracias a esta universidad conocí. Los cuales no solo me han ayudado a completar esta meta, sino mucho más en la vida; me empujan a crecer como persona y apuntar a un mejor futuro, por lo cual nunca dejaré de estar agradecido por todo lo que han significado.

Abstract

En el contexto del rubro energético, específicamente en el área de activos de transmisión eléctrica, siempre ha existido la necesidad de gestionar eficientemente los recursos. Esto debido a lo esencial que es el servicio y la gran variedad de activos involucrados.

En este escrito, se discute la problemática y las herramientas utilizadas en este ámbito. A pesar de que existen sistemas que proveen solución a esta problemática, como el software de Meridian en el caso del cliente estudiado, estos aun así presentan faltas o funcionalidades con potencial de mejora. Por lo tanto, los puntos principales que se buscan resolver con la solución presente en este trabajo son la falta de una fuente de información histórica centralizada, la cual permita llevar la trazabilidad del ciclo de vida de los activos, y la falta de adaptabilidad de las herramientas actuales para poder utilizarlas bajo múltiples ambientes laborales, como lo es trabajar en terreno.

En respuesta a estas limitaciones, se propone y desarrolla una solución basada en una aplicación web para la gestión de activos. Este trabajo se enfoca en el backend de esta y su diseño contempla una API tipo REST, una base de datos relacional y herramientas de migración de datos a través de scripts tipo ETL. El sistema propuesto tiene como objetivo centralizar la información, facilitar el acceso desde distintos entornos de trabajo y mejorar la gestión de registros, permitiendo así una toma de decisiones más confiable y una reducción en los tiempos de búsqueda de información histórica.

Índice de figuras

Índice de cuadros

Índice general

Capítulo 1

Introducción

La gestión de activos es un proceso fundamental en organizaciones que constantemente dependen de sus bienes físicos para entregar continuidad operacional y seguridad, especialmente en industrias de gran escala como energía, minería y transporte. En estos contextos, los activos pueden corresponder, por ejemplo, a transformadores de potencia, interruptores, maquinaria pesada u otras infraestructuras críticas. Mantener información confiable sobre el ciclo de vida de estos activos, como su ubicación, estado operativo, intervenciones de mantenimiento, inspecciones y eventos relevantes, es clave para planificar, priorizar recursos y tomar decisiones informadas.

Asimismo, distintos ámbitos normativos y de buenas prácticas impulsan la formalización de estos procesos. Entre ellos se encuentran estándares internacionales como ISO-55001, además de regulaciones locales que promueven la trazabilidad y una gestión responsable de los recursos y residuos.

A pesar de su importancia, en la práctica, la gestión de activos suele enfrentar dificultades operacionales. En muchos casos, la información se encuentra distribuida entre múltiples sistemas y documentos, lo que dificulta su consolidación. Además, numerosos registros se generan de forma manual y en distintos momentos del ciclo operativo, aumentando la posibilidad de inconsistencias. A esto se suma que los cambios en el estado de los activos no siempre quedan trazados de manera consistente a lo largo

del tiempo. Como consecuencia, responder preguntas para la operación puede volverse complejo, incrementando los tiempos de búsqueda y el riesgo de tomar decisiones basadas en información incompleta.

En el marco del Programa de Memoria Multidisciplinaria (PMM), este trabajo utiliza como caso de estudio a una empresa del sector eléctrico, la cual proporcionó acceso a un entorno de prueba y a su proceso actual de gestión de información. Este contexto permitió observar de primera fuente las limitaciones típicas de la trazabilidad de activos y orientar el desarrollo de una solución de apoyo.

1.1. Problema a resolver

En organizaciones intensivas en activos, un problema recurrente es que el historial de cambios asociado a cada activo no queda registrado de forma centralizada y consistente. Es común que los sistemas disponibles prioricen el almacenamiento documental o la visualización del estado actual, pero no aseguren la retención de estados previos cuando ocurren modificaciones (por ejemplo, cambios de ubicación, condición o estado operativo). Esto dificulta la trazabilidad y obliga a reconstruir eventos históricos a partir de múltiples fuentes, aumentando tiempos de búsqueda y riesgo de errores.

Adicionalmente, la captura de información en terreno suele verse limitada por barreras de usabilidad y accesibilidad. Cuando los mecanismos de registro no están disponibles en dispositivos móviles o requieren procesos complejos, parte de los cambios y eventos asociados a los activos pueden quedar sin documentar. En conjunto, estas dificultades se enmarcan en el ámbito de la gestión del ciclo de vida de activos (Asset Lifecycle Management), donde se requiere mantener registros históricos confiables que acompañen al activo durante toda su vida útil.

1.2. Acercamiento a la solución

En este proyecto se propone el diseño e implementación de un sistema de apoyo para la trazabilidad de activos. Principalmente, orientado a la consulta y conservación de cambios históricos asociados a cada activo durante su ciclo de vida. La propuesta apunta a facilitar los registros manuales y a centralizar la información relevante para operación y mantenimiento. Esto resulta en una mayor disponibilidad de datos en ambientes diversos como oficina y terreno.

La solución consiste en una aplicación web enfocada en Asset Lifecycle Management y trazabilidad histórica. Para su diseño se consideran, además, mecanismos de integración y visualización que permitan explorar la información registrada mediante herramientas analíticas (por ejemplo, tableros de control, mapas, tablas). Esto se lograría a través de integración con plataformas existentes o mediante vistas dedicadas en la propia aplicación.

Los principales componentes de esta solución software son los siguientes:

- Una base de datos (backend) para almacenar activos y su historial de eventos, estados y atributos relevantes.
- Una API (backend) que actúe como puente entre la base de datos y la interfaz de usuario, permitiendo operaciones eficientes y seguras de consulta y actualización.
- Una interfaz de usuario (frontend) diseñada para múltiples roles y adaptable a distintos dispositivos, con énfasis en el uso en terreno.

1.3. Hipótesis

Se plantea que el desarrollo de una interfaz de usuario intuitiva, orientada a la gestión y trazabilidad de activos con adaptabilidad a múltiples dispositivos, mejoraría la accesibilidad a información histórica. Esto resultaría en tiempos reducidos de búsqueda de registros y favorecería la adopción del sistema por parte de personal técnico y

administrativo, incluso en condiciones de trabajo en terreno.

Asimismo, se propone el diseño e implementación de una base de datos relacional con un modelo lógico adecuado y mecanismos de trazabilidad incluidos. De esta manera se permitiría el manejo correcto de la información asociada a los activos, garantizando integridad, consistencia y trazabilidad histórica de los registros a lo largo de su ciclo de vida. Este enfoque contribuye a la toma de decisiones informadas y a la reducción del tiempo necesario para la búsqueda y gestión de información histórica.

La creación de la base de datos y la interfaz de usuario deben ir acompañadas de una API diseñada con eficiencia y seguridad en mente. Al permitir un flujo de datos fluido y seguro, esta aportaría a reducir los tiempos de gestión de registros y habilitaría el uso simultáneo por múltiples usuarios mediante mecanismos de acceso controlado.

Por último, se plantea que la implementación de un proceso ETL (Extracción, Transformación y Carga) facilitaría la migración ordenada y confiable de los datos provenientes de sistemas previos. Así asegurando su correcta integración al nuevo sistema y demostrando que el uso de este tipo de procesos resulta beneficioso para escenarios de transición entre plataformas.

Concretando, se pueden tener 4 hipótesis de lo mencionado anteriormente:

- **H1:** Una interfaz de usuario dedicada, intuitiva y adaptativa mejora la accesibilidad y eficiencia de los usuarios en comparación con las herramientas actualmente empleadas.
- **H2:** Un diseño lógico de base de datos relacional permite una gestión correcta y trazable de los datos históricos de los activos.
- **H3:** Una API segura y eficiente, que permita flujos de datos controlados de múltiples usuarios a la vez, resultaría en operaciones de gestión de manera más rápida y sin errores.
- **H4:** La implementación de un proceso ETL mejora la calidad y confiabilidad de la migración de datos desde fuentes externas hacia el nuevo sistema.

1.4. Objetivos

1.4.1. Objetivo General

El objetivo de este trabajo consiste en mejorar el proceso de trazabilidad de activos a través del desarrollo de una herramienta de software. Esta sería compuesta por un sistema de registro histórico que conserve estados previos de los activos. También sería acompañado de una interfaz de usuario adaptable al dispositivo y mecanismos de importación de datos para migración. Los resultados esperados son reducir el tiempo dedicado a la búsqueda de información histórica, mejorar la trazabilidad de los activos, facilitar la toma de decisiones operativas y contribuir al cumplimiento de estándares como ISO 55001. La propuesta se valida mediante un caso de estudio provisto en el marco del PMM.

1.4.2. Objetivos Específicos

Este trabajo se enmarca en el contexto del Programa de Memoria Multidisciplinaria (PMM), desarrollado por un equipo compuesto por dos integrantes. Para ello, se definieron los siguientes objetivos específicos generales del equipo:

- **Objetivo Específico 1 - Analizar proceso actual de búsqueda y gestión de registros históricos de activos:** Se busca, a partir del análisis, identificar sus principales limitaciones, oportunidades de mejora y funciones esenciales que el sistema debiese cumplir.
- **Objetivo Específico 2 - Diseñar e implementar la arquitectura del sistema:** Diseñar la arquitectura del sistema que permita una gestión eficiente, segura y organizada de los registros históricos de los activos.
- **Objetivo específico 3 - Implementar componente para el almacenamiento de los datos de activos:** Desarrollar una base de datos que facilite el almacenamiento, acceso y actualización de la información histórica relacionada con los activos.

- **Objetivo Específico 4 - Implementar componente intermediario del sistema:** Crear una API que permita realizar transacciones entre el almacenamiento de datos y los usuarios, tales como creación, eliminación o actualización de registros.
- **Objetivo Específico 5 - Diseñar e implementar una interfaz de usuario:** Crear una interfaz de usuario accesible tanto en computadoras como en dispositivos móviles. El objetivo es permitir operaciones del día a día de manera eficiente, optimizando el proceso de ingresar o visualizar registros mientras el usuario se encuentra trabajando en terreno o en oficina.
- **Objetivo Específico 6 - Elaborar un manual de usuario:** Crear la documentación técnica necesaria para que los posibles usuarios del sistema, como personal de mantenimiento y administradores, puedan hacer uso del sistema a través de la interfaz de usuario.
- **Objetivo Específico 7 - Integrar con sistemas existentes:** Analizar posibles opciones de integración con herramientas y sistemas ya utilizados en la organización del caso de estudio, determinando cuáles se adecúan mejor al escenario. Finalmente, documentar un plan para que el sistema pueda operar como alternativa o apoyo en paralelo.

En el marco de estos objetivos específicos, el enfoque particular de esta memoria se centra en los siguientes aspectos:

- **Respecto al Objetivo Específico 1:** Analizar el proceso actual observado en el caso de estudio para gestionar los registros históricos. El fin de esto es identificar las principales falencias del sistema vigente y proponer mejoras funcionales y de accesibilidad que orientan el diseño de la solución.
- **Respecto al Objetivo Específico 2:** Diseñar la arquitectura general del sistema, definiendo sus componentes principales —frontend, API backend, base de datos y módulo de importación de datos— junto con las tecnologías más adecuadas para cada uno, considerando escalabilidad, seguridad y compatibilidad.

- **Respecto al Objetivo Específico 3:** Aportar en la elaboración del modelo conceptual y lógico de la base de datos relacional, definiendo las entidades, relaciones y mecanismos de versionado necesarios para conservar los estados históricos de los activos, levantar una implementación física para el desarrollo del prototipo, la cual se fue modificando de manera iterativa para la adición de cambios necesarios durante el trabajo.

- **Respecto al Objetivo Específico 4:** Diseñar y desarrollar la API con su documentación estandarizada, incluyendo todas sus operaciones del tipo CRUD (Create, Read, Update, Delete) necesarias para llevar los registros de los activos y el uso de la interfaz de usuario de manera eficiente.

- **Respecto al Objetivo Específico 5:** Aportar al diseño preliminar inicial de las vistas de la interfaz de usuario y al desarrollo de la interfaz, priorizando mayormente la conexión de esta con la API del sistema.

- **Respecto al Objetivo Específico 7:** Integración del sistema con herramientas utilizadas en el caso de estudio, como Power BI.

- **Objetivo adicional:** Aporté al desarrollo de un componente auxiliar para la carga inicial de datos, permitiendo importar registros históricos desde archivos CSV o Excel, facilitando así la transición de datos hacia el nuevo sistema.

De esta forma, el trabajo presentado en esta memoria abarca el análisis del proceso actual, el diseño arquitectónico y de base de datos, la documentación e implementación de una API y el desarrollo de mecanismos de importación de datos, constituyendo la base fundamental para la trazabilidad de los activos.

1.5. Estructura

El presente documento se encuentra organizado en cuatro capítulos principales, diseñados para exponer de manera ordenada el proceso completo de investigación, di-

seño, desarrollo e implementación de la solución propuesta en forma de prototipo. Estos capítulos son:

- **Capítulo 2, Estado del arte:** Detalla la investigación necesaria para el desarrollo, incluyendo una revisión del estado del arte de la gestión de activos y las plataformas existentes. Se presenta la fundamentación teórica y se justifica la selección del stack tecnológico que se utilizará para el desarrollo del sistema.

- **Capítulo 3, Diseño y Desarrollo de la Solución:** Se enfoca en la arquitectura y el diseño técnico del sistema. Incluye la definición de la arquitectura lógica, el diseño detallado del modelo de base de datos relacional y la explicación de los componentes clave de la aplicación, como la API y la metodología de implementación de proceso Extraer, Transformar y Cargar (ETL) para la migración de datos.

- **Capítulo 4, Resultados:** Presenta los resultados de los múltiples módulos incorporados y todo lo relacionado a la validación. Este capítulo demuestra el cumplimiento de los objetivos y la confirmación o refutación de las hipótesis planteadas.

- **Capítulo 5, Conclusiones y Trabajo a Futuro:** Se presentan las conclusiones obtenidas tras el desarrollo e implementación del sistema, evaluando el grado en que se cumplieron los objetivos y se validaron las hipótesis. Se proponen posibles mejoras, tales como la integración con nuevas herramientas, uso de otras tecnologías, optimizaciones de rendimiento o incorporación de nuevas funcionalidades orientadas a fortalecer la trazabilidad y la gestión de activos.

Capítulo 2

Estado del arte

El concepto de 'Asset Lifecycle Management', hoy en día, está altamente presente en todo tipo de industrias o áreas, debido a que con el pasar del tiempo se prioriza realizar la toma de decisiones de la manera más informada posible. Aquí es donde es esencial llevar los eventos de la vida de los activos, en este contexto, de los activos de transmisión eléctrica.

2.1. Soluciones Existentes

Debido a lo previo, actualmente existen diversas soluciones enfocadas en la gestión y trazabilidad de activos, cada una con fortalezas y limitaciones dependiendo del contexto en el que se implementen.

Una de estas herramientas es IBM Maximo[1], plataforma orientada a la gestión del ciclo de vida de activos y mantenimiento empresarial. Este software destaca por sus capacidades de administración de información y monitoreo de activos, además de permitir acceso desde dispositivos móviles, facilitando el trabajo del personal técnico en terreno. Asimismo, incorpora funcionalidades avanzadas para la planificación de mantenimiento y análisis operacional. Sin embargo, una de sus principales desventajas corresponde al alto costo de implementación, licenciamiento y capacitación requerido para su correcta utilización.

Por otro lado, AVEVA Asset Information Management[2] corresponde a una solución enfocada en la centralización y visualización de información asociada a activos industriales. Entre sus ventajas destaca la capacidad de integración con plataformas como AutoCAD y SAP, permitiendo consolidar información proveniente de múltiples fuentes y visualizar detalles avanzados de los activos, incluyendo modelos tridimensionales relacionados con su ubicación física. Este enfoque se aproxima al concepto de "digital twin", entendido como una representación virtual de un activo físico utilizada para apoyar simulaciones y toma de decisiones. No obstante, al igual que otras soluciones empresariales, presenta elevados costos de implementación y una alta complejidad de integración.

Otra plataforma utilizada en este ámbito es Hitachi Energy Lumada[3], solución enfocada en el monitoreo y análisis avanzado de activos eléctricos. Esta herramienta incorpora capacidades de análisis predictivo e inteligencia artificial para estimar el estado operacional de los activos y anticipar posibles fallas. Sin embargo, su adopción puede verse limitada por los requerimientos de infraestructura y especialización técnica necesarios para su implementación.

Asimismo, IFS Cloud[4] corresponde a un sistema empresarial con funcionalidades de Enterprise Asset Management (EAM), permitiendo gestionar información de activos, mantenimiento y operaciones dentro de una misma plataforma. Su principal ventaja radica en la integración de múltiples procesos organizacionales; sin embargo, debido a su amplitud funcional, puede resultar una solución sobredimensionada para problemáticas específicas de trazabilidad de activos.

Finalmente, otro software ampliamente utilizado es SAP[5], mencionado anteriormente. A pesar de poseer capacidades robustas para la gestión empresarial y administración de activos, no logró adaptarse completamente a la problemática abordada en este trabajo. Esto se debe a que su enfoque generalista orientado a Enterprise Asset Management cubre un espectro funcional considerablemente más amplio que el requerido para la solución propuesta.

2.2. Marco teórico

Estudios han abordado enfoques más integrales y personalizados para la gestión de activos, como es el caso de la transmisión eléctrica. La revisión sistemática de Gavrikova (2020)[6] destaca que la gestión estratégica de activos ha evolucionado desde una orientación meramente técnica hacia un enfoque holístico y continuo, que considera el alineamiento con los objetivos estratégicos de la organización, la sostenibilidad a largo plazo y la ventaja competitiva.

Este enfoque estratégico es fuertemente impulsado por la serie de normas internacionales ISO 55000, particularmente la ISO 55001, que establece requisitos para un sistema de gestión de activos. Esta norma proporciona un marco para alinear la gestión de los activos físicos. Su implementación permite abordar no solo el ciclo de vida técnico de los activos, sino también su impacto económico y social, lo cual es especialmente relevante para empresas de servicios públicos sujetas a regulaciones ambientales y de servicio[6].

Chile es un país que tiene estas regulaciones ambientales, como la ley REP, la cual empuja a tener una economía circular, donde los activos deben terminar sus ciclos de vida de manera adecuada, y para esto se desarrollan sistemas de Asset Lifecycle Management para consolidar la toma de decisiones para la vida de estos activos.[7]

En el ámbito técnico, estudios como el de Khuntia (2015)[8] destacan la importancia de clasificar la gestión de activos según horizontes de tiempo (largo, mediano y corto plazo), y dominios de actividad (técnico, económico y social). El tener estas clasificaciones otorga una ayuda al momento de clasificar factores importantes a tener en cuenta en el diseño y las metas que se están buscando cumplir.

Un caso real de estudio es el del sistema de transmisión de Java-Bali en Indonesia, documentado por Habibie (2020) [9], donde se implementaron técnicas de gestión de activos para enfrentarse a problemas de crecimiento de infraestructura de transmisión. En este caso, la transición hacia un sistema de gestión de activos basado en la norma ISO 55000 permitió una mejor toma de decisiones a nivel organizacional y de manera

más estructurada en torno al riesgo, integrando criterios estratégicos y técnicos. Esta experiencia ilustra cómo una gestión de activos bien estructurada puede responder eficazmente a desafíos operacionales complejos y escalables.

En el siguiente capítulo, se mostrará el proceso completo del desarrollo, tanto los requisitos de la problemática que han impulsado la elección de las tecnologías como la implementación de estas para solventar el problema presentado.

Capítulo 3

Desarrollo de la solución

A continuación se detalla el proceso completo de creación de la solución, desde la identificación de los requisitos del sistema hasta el diseño de la arquitectura y el desarrollo de los componentes clave. El enfoque se centra en los módulos de API, almacenamiento e importación de datos, que constituyen el núcleo de esta memoria de titulación.

3.1. Análisis de Requerimientos del Sistema

El análisis de requerimientos del sistema se realizó en conjunto con el equipo de trabajo, con el propósito de definir las funcionalidades esenciales y las restricciones técnicas del sistema a desarrollar. Para asegurar que la solución responda eficazmente a las necesidades del caso, se realizó un levantamiento detallado de los requisitos, clasificándolos en funcionales, no funcionales y de interfaz.

3.1.1. Requisitos Funcionales

Estos definen las operaciones que el sistema debe ser capaz de realizar:

- **RF1**: El sistema debe permitir la creación y edición de los registros de los activos,

estos registros pueden ser tanto de activos nuevos o que ya estuvieran anteriormente ingresados.

- **RF2**: El sistema debe ser accesible desde múltiples tipos de dispositivos, debe permitir realizar las diferentes acciones disponibles en el dispositivo que se decida usar.
- **RF3**: El sistema debe permitir una visualización de los registros pertinentes a los activos a través de una hoja de vida.
- **RF4**: Dentro del sistema debe estar la funcionalidad de importar datos ya existentes a la base de datos que se implemente.
- **RF5**: Se debe poder almacenar archivos relacionados con los activos de transmisión.

El enfoque de esta memoria recae principalmente en RF1, RF4 y RF5.

3.1.2. Requisitos no Funcionales

Estos describen los atributos de calidad del sistema:

- **RNF1**: Se necesita que el sistema sea seguro, esto dado por la sensibilidad de los datos con los que se trabajan, hay que hacer que el acceso a esta sea solo para personal autorizado.
- **RNF2**: Que el sistema permita agregar nuevos tipos de activos para una escalabilidad a futuro.
- **RNF3**: La visualización de datos de los activos debe ser rápidamente accesible para la gente que se encuentra trabajando tanto en oficina como en terreno.
- **RNF4**: Existencia de filtros intuitivos para el uso de los múltiples perfiles de usuarios.

- **RNF5**: Que el uso en dispositivos móviles con poca cobertura de red tenga persistencia cuando se acceda o realicen cambios en los registros.

- **RNF6:** Que todos los datos guardados en el sistema, sean suficientes para cumplir con normativas legales.

De estos el trabajo desarrollado se enfoca en RNF1, RNF4 y RNF.

3.1.3. Requisitos de Interfaces

En el marco del trabajo individual se participó activamente en la definición de los requisitos de interfaz y de accesibilidad, los cuales orientaron el desarrollo del frontend y la adaptación del sistema a distintos dispositivos.

- **RI1**: Se requiere que el sistema permita la búsqueda de activos para visualización o actualización de sus datos.

- **RI2**: Se requiere de visualización de hojas de vida de los activos que contengan la información de sus estados previos.

- **RI3**: La interfaz debe tener herramientas tales como barras y filtros de búsqueda que permiten agilizar los procesos para hallar información de los activos.

- **RI4**: Es necesario que la interfaz del sistema tenga dashboards para mostrar los datos y tener mejor análisis de estos.

- **RI5**: Deben existir plantilla de ingreso de datos para cada familia de activo participe.

- **RI6**: Las plantillas de ingreso o edición de datos deben poseer sus campos con formatos estandarizados a través de herramientas como listas de opciones tipo dropdown.

- **RI7**: Se requiere una interfaz que haga de ”Login”para el sistema para permitir un acceso controlado a los datos debido la privacidad de estos.

- **RI8**: Se debe tener una página principal dentro del sistema, su interfaz debe presentar una herramienta para hacer la búsqueda de algún activo y mostrar las otras vistas disponibles.

- **RI9**: Es necesario que cada vista sea adaptable para una correcta visualización en dispositivos móviles.

3.1.4. Perfiles de Usuario

Se identificaron los siguientes perfiles de usuarios clave para el sistema, considerando tanto sus necesidades operativas como los contextos de uso (oficina y terreno):

Perfil	**Ocupacional**	**Socioeconómico y cultural**	**Etario**	**Características físicas, fisiológicas, psicológicas**	**Otros**
Administradores, supervisores y gestores de activos.	Actualizar estados de activos, generar reportes, asegurar cumplimiento de normativas y coordinar recursos.	Técnico o profesional. Familiarizados con sistemas digitales.	18-65 años	Uso predominante de computadores	Requieren acceso a datos históricos completos y filtros avanzados
Técnicos electricistas, operarios de mantenimiento y equipos de inspección	Registrar fallas o mantenimientos, inspeccionar activos y seguir órdenes de trabajo	Técnico medio o certificaciones especializadas. Cultura práctica y orientación a resultados inmediatos.	18-65 años	Exposición a trabajos en terreno. Preferencia por flujos simples y rápidos	Necesitan interfaz móvil y datos concisos asociados al trabajo realizado

Cuadro 3.1: Características de los perfiles de usuarios

3.2. Diseño de la Solución

El diseño general de la solución contempla una arquitectura modular compuesta por los siguientes elementos: frontend, backend, base de datos y módulo ETL.

3.2.1. Diagrama de Contexto

El sistema se diseñó para interactuar con dos tipos de actores principales: el personal técnico en terreno y los administradores en oficina. El personal de terreno utiliza principalmente dispositivos móviles para actualizar y revisar datos de activos, mientras que los administradores usan computadores de escritorio para una gestión más completa de usuarios y registros.

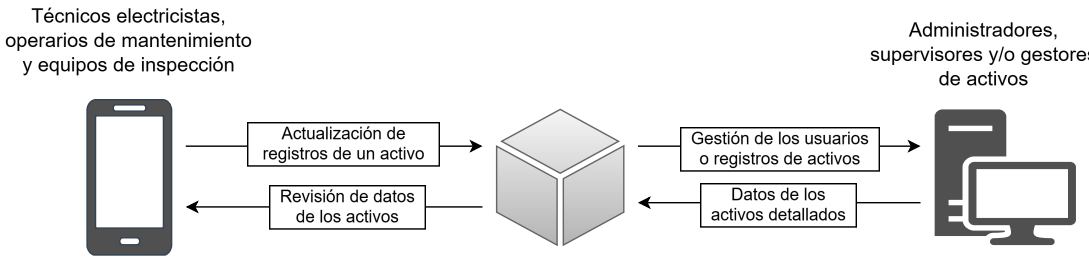


Figura 3.1: Diagrama de Contexto.

3.2.2. Arquitectura del sistema

Para el desarrollo de la herramienta fue necesaria la creación de 4 módulos, los cuales se encargan del frontend, el backend, el almacenamiento de los datos y por último un módulo de importación de datos, de modo de hacer una primera carga para el sistema o hacer una migración a este. Los módulos importantes para este escrito son los correspondientes al backend (API), el almacenamiento de datos y la importación de datos. En este documento se describe en profundidad del desarrollo de estos. De igual manera, a forma de dar un contexto completo del trabajo del equipo, los módulos fueron definidos de la siguiente manera:

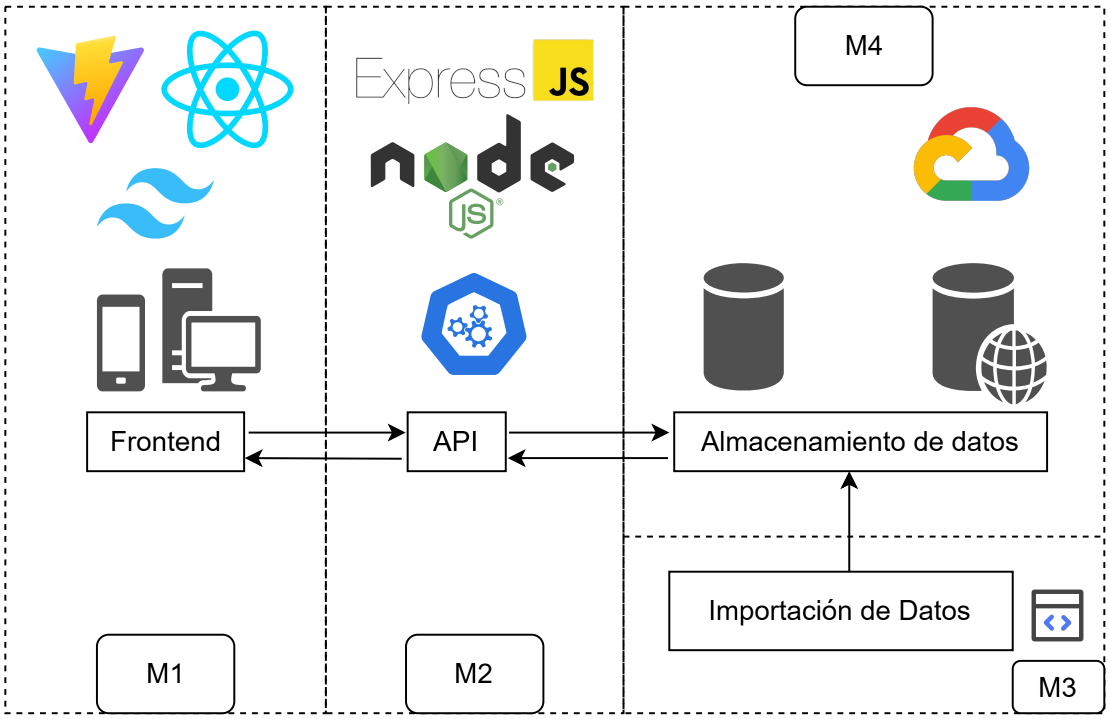


Figura 3.2: Diagrama de Contexto.

Módulo	**Propósito**
Frontend (M1)	Se necesita de un frontend que sea intuitivo, fácil de usar y adaptable para diferentes tipos de dispositivos y usuarios. Teniendo en cuenta el último punto, es importante el tener persistencia en la carga de datos desde el frontend a la API.
Backend (M2)	Este componente estaría formado por una API central, la cual debe seguir los mismos principios de la base de datos, y además debe realizar transacciones de datos de manera segura, tanto en control de acceso como en la integridad de estos.
Importación de Datos (M3)	Dado que ya existen numerosos datos que serán entregados a través de archivos CSV o XLSX, se requiere de un componente capaz de realizar un *parsing* apropiado de los datos y así poder brindar una carga inicial de datos al sistema a desarrollar.
Almacenamiento de Datos (M4)	La creación de al menos una base de datos desde cero, es integral a la solución. Por un lado, se requiere de un lugar de almacenamiento completamente moldeable, para así tener un sistema escalable y personalizado al manejo de activos de transmisión eléctrica. Por otro lado, la arquitectura y sistemas del caso de estudio actuales no han permitido acceso para trabajar con aquellos, por lo cual se buscará realizar un diseño en base a los datos puros entregados respecto a los activos. Dentro de los datos puros se encuentran plantillas CSV con múltiples columnas de datos y documentos PDF de los activos.

Cuadro 3.3: Módulos de la arquitectura del sistema

3.2.3. Herramientas Tecnológicas Relevantes

El diseño e implementación de una herramienta de trazabilidad de activos requiere seleccionar cuidadosamente el conjunto de tecnologías que sustentarán su funcionamiento. Dicha selección debe basarse en criterios de desempeño, escalabilidad, mantenibilidad, facilidad de uso y compatibilidad con el contexto operacional.

En este caso, se buscó desarrollar un sistema moderno, modular y extensible, capaz de integrarse con herramientas corporativas ya existentes, como Power BI, y que pueda ser utilizado tanto en entornos de oficina como en terreno, mediante dispositivos móviles o de escritorio.

API: Node.js + Express

Como parte esencial del *backend* se decide diseñar e implementar la API con **Node.js** [10], acompañado del framework **Express** [11]. Esta elección se basa en la necesidad de desarrollar una API RESTful eficiente, asíncrona y fácil de mantener, ideal para integrarse con el *frontend* moderno de una aplicación web y manejar múltiples solicitudes simultáneas de usuarios en localidades distribuidas.

El objetivo principal de la API es establecer un flujo de datos robusto entre el frontend y la base de datos relacional; gracias a Node.js parte de esto se puede realizar con las bibliotecas disponibles que hay para este lenguaje. Por el lado de la base de datos, se pueden utilizar paquetes de biblioteca como **node-postgres (pg)** [12], el cual permite establecer tanto la conexión como las consultas hacia la base de datos en lenguaje natural de PostgreSQL, entregando un grado alto de control respecto a cómo y qué datos se acceden. Adicionalmente, si se estima necesario, se puede complementar con bibliotecas adicionales del tipo Object Relational Mapping (ORM), que facilitan la implementación de consultas hacia la base de datos a través de abstracciones con funciones predeterminadas de los paquetes. Algunos ejemplos de estas son **Sequelize** [13] o **Prisma** [14]. Finalmente, Node.js requiere de pocos recursos (*lightweight*) lo cual es ideal, ya que la aplicación web será concisa en cuanto a las funcionalidades y objetivos

que busca lograr.

¿Por qué no otras alternativas como Django [15] (Python) o Spring Boot [16] (Java)?

- Node.js tiene una arquitectura basada en eventos, ideal para construir APIs altamente escalables que respondan en tiempo real, algo importante dado que múltiples usuarios pueden consultar o modificar datos de activos simultáneamente desde distintas sucursales de trabajo.
- En comparación con Django, Node.js ofrece mayor rendimiento bajo carga a menor costo de recursos y menor complejidad para configurar consultas simultáneas hacia la API.
- Comparándolo con las otras alternativas, Node.js es más apropiado y ligero para los objetivos actuales, además de permitir compartir lógica con el frontend, al estar ambos escritos en JavaScript/TypeScript.
- A la vez, permite mantener el equipo de desarrollo del frontend y backend unificado, ya que ambos programarán en JavaScript/TypeScript, permitiendo un desarrollo más fluido y colaborativo.

El framework de Express es minimalista, brinda lo justo y necesario para resolver la problemática sin tomar mas recursos o configuraciones adicionales de lo que se deba. Uno de sus puntos claves es permitir implementar *middleware*, lo cual ayuda a reforzar tanto la autenticación como autorización hacia los datos. En cuanto la autenticación, se complementa con el uso de **JsonWebToken** [17] y **bcrypt** [18] para la implementación de un ingreso de sesión tradicional.

Para consolidar las capacidades de la API, se decidió utilizar un estándar para la documentación, este siendo **OpenAPI** [19]. Este estándar permite brindar un método ordenado de documentar toda ruta con sus detalles como parámetros y posibles salidas. Adicionalmente, se puede acompañar de herramientas como **SwaggerUI** [20] que brinda una manera visual y mas amigable para comprender los mecanismos de la API,

esta inclusive se puede emparejar con un ambiente de pruebas para validar el trabajo en desarrollo. En este caso, la herramienta preferida para las pruebas fue **Postman** [21] el cual permite crear una colección de pruebas de manera eficiente.

Finalmente, el stack de tecnología elegido para esta parte del trabajo se estima adecuado para cumplir con el desarrollo de una API robusta, la cual sea capaz de responder a múltiples consultas a la vez y permita un control eficiente sobre los datos.

Herramienta de Importación de Datos: Python + Pandas + SQLAlchemy

Para la carga inicial de registros históricos al sistema se hizo necesaria la creación de una herramienta de importación usando **Python**[22] y las bibliotecas **Pandas**[23] y **SQLAlchemy**[24], por sus siguientes características:

- **Python:** lenguaje de alto nivel, de sintaxis clara y amplia disponibilidad de bibliotecas para procesamiento de datos y conexión con bases de datos.
- **Pandas:** biblioteca especializada en manejo de datos tabulares. Permite leer archivos CSV o Excel, limpiar y transformar los datos antes de su inserción.
- **SQLAlchemy:** ORM (Object Relational Mapper) que simplifica la interacción con la base de datos, ofreciendo validaciones automáticas y abstracción del modelo lógico.

Esto permite implementar un proceso *ETL* (Extracción, Transformación y Carga) que facilita la migración de datos desde archivos en formato *CSV* o *Excel* hacia la base de datos PostgreSQL.

Aunque existen alternativas más rápidas (como el uso directo de `psycopg2` con la instrucción `COPY`[25]), se optó por **SQLAlchemy** debido a su carácter de *ORM*, que permite un manejo de datos más estructurado, validado y adaptable a futuras modificaciones en el modelo.

Estas tecnologías garantizan una carga inicial de datos confiable en este escenario que los datos provienen de fuentes no normalizadas y requieren limpieza previa. La so-

lución implementada garantiza una migración ordenada y confiable de datos históricos, reduciendo errores humanos y facilitando la adopción del sistema mediante la integración automatizada de registros existentes.

Base de Datos: PostgreSQL

El sistema requiere una base de datos capaz de almacenar registros históricos de los activos de transmisión, manteniendo la integridad de la información, las relaciones entre entidades y la trazabilidad de los cambios a lo largo del tiempo. Dado que la información presenta una estructura fuertemente relacional, se optó por un sistema de gestión de bases de datos relacional (RDBMS). Frente a esta necesidad, se evaluaron distintas alternativas:

- **MySQL[26]:** Aunque popular y eficiente en aplicaciones simples, presenta limitaciones en integridad referencial y manejo de transacciones complejas, además de menor soporte para extensiones como PostGIS.
- **SQL Server[27]:** Ofrece un alto rendimiento, pero requiere licencias propietarias y está más orientado a entornos corporativos con infraestructura Microsoft.
- **NoSQL (MongoDB)[28]:** Adecuado para datos no estructurados, pero menos eficiente para consultas relacionales y trazabilidad histórica basada en relaciones.

En este contexto, **PostgreSQL**[29] fue la tecnología seleccionada, ya que ofrece un equilibrio entre potencia, flexibilidad y costo. Esta es una base de datos relacional de código abierto ampliamente utilizada en entornos empresariales, reconocida por su estabilidad, seguridad y soporte a operaciones complejas, aspectos esenciales para garantizar la integridad de los registros históricos. A continuación, se mencionan características importantes para el proyecto:

- **Cumplimiento de estándares SQL y transacciones ACID:** garantiza la consistencia y confiabilidad de las operaciones, lo cual es esencial en un sistema donde los datos deben reflejar con exactitud el historial de los activos.

- **Soporte de integridad referencial avanzada:** permite definir claves foráneas, restricciones y reglas de cascada que aseguran la coherencia de las relaciones entre entidades.
- **Funciones de almacenado:** presenta la posibilidad de implementar Triggers o Change Data Capture (CDC), los que posibilitan automatizar procesos como la creación de eventos o registros históricos al modificar un activo, reduciendo errores humanos.
- **Extensiones avanzadas:**
  - **PostGIS**, para manejar coordenadas geográficas de los activos, útil si se requiere visualización espacial.
  - **JSONB**, para almacenar datos semiestructurados que varían según el tipo de activo.
- **Alto rendimiento y escalabilidad:** PostgreSQL soporta grandes volúmenes de datos y consultas complejas con eficiencia.
- **Código abierto y multiplataforma:** su naturaleza *open source* elimina costos de licenciamiento y facilita su despliegue en distintos entornos.

Además, esta base de datos nos permite el uso de herramientas como **pgAdmin** y **DBeaver**, que facilitan el modelado y la administración de la base de datos durante las etapas de diseño y validación. Finalmente, su soporte a transacciones ACID garantiza confiabilidad, mientras que su capacidad de definir **triggers** permite automatizar el registro histórico sin depender de la lógica de aplicación.

3.3. Backend

La API es el componente intermediario del sistema que, a pesar de que los usuarios no interactúan con este directamente, les permite la transferencia de datos fluida entre

la interfaz y el almacenamiento. Para esto se desarrolló con Express como framework base, el cual rápidamente permitió estructurar esta parte del trabajo en múltiples categorías: documentación, rutas, middlewares, controladores, servicios, modelos y configuraciones. Cada una de las anteriores son partes esenciales para el desarrollo de una API tipo RESTful, la cual permite operaciones dentro del umbral *Create, Read, Update* y *Delete* (CRUD), dentro de otras necesarias para que el sistema opere de manera robusta, como operaciones de autenticación. Las categorías y parte de lo que encapsulan se detallan más adelante en esta sección. A continuación, se muestran los requisitos establecidos en base a la investigación de este trabajo, que luego son utilizados para diseñar la API y sus capacidades.

3.3.1. Diseño Inicial API

Gran parte de lo que define una API REST, son las rutas a las cuales se puede acceder a través de otros servicios, tales como la interfaz de usuario del frontend. Para saber a dónde dirigir este diseño, primero se estableció una base de historias de usuarios, las cuales consideran las funciones esenciales del sistema. Estas fueron formadas en conjunto con el equipo de trabajo a través de comunicación recurrente con el representante de la empresa del caso de estudio e investigaciones a fondo.

Historias de Usuario

Esta sección corresponde a las historias de usuario que permiten establecer puntos iniciales a las funcionalidades de la API.

UH1
- **Titulo:** Inicio de sesión.
- **Historia de Usuario:** Como trabajador (técnico o administrativo) necesito ingresar al sistema para uso de este.
- **Criterio de aceptación:** Al ingresar los datos se da acceso al sistema y se es redirigido a la página principal.

UH2
- **Titulo:** Página Principal de Búsqueda de Activos.
- **Historia de Usuario:** El trabajador (técnico o administrativo) necesita que, al ingresar al sistema, pueda realizar búsqueda de datos de activos.
- **Criterio de aceptación:** El trabajador, posteriormente al realizar el inicio de sesión, se encuentra una página capaz de permitir la búsqueda de los activos a través de herramientas como barras de búsqueda y filtros avanzados.

UH3
- **Titulo:** Visualización de Hoja de Vida de un Activo.
- **Historia de Usuario:** Como trabajador en terreno necesito los estados previos del activo para ver en qué ha fallado últimamente.
- **Criterio de aceptación:** El trabajador al poder haber hallado el activo a través de la funcionalidad de búsqueda, ahora es capaz de visualizar una lista con los estados más recientes del activo que incluye datos tales como estado (código definido arbitrariamente en caso de estudio), observaciones y fecha de registro.

UH4
- **Titulo:** Gestión de Permisos de Usuarios.
- **Historia de Usuario:** Como administrador quiero poder brindar, o revocar permisos de visualización, o edición de los activo, a los usuarios para supervisar el flujo de trabajo de los trabajadores.
- **Criterio de aceptación:** Una vista que permita modificar permisos para cada usuario.

UH5
- **Titulo:** Gestión de Acceso de Usuarios.
- **Historia de Usuario:** Como administrador quiero poder manejar quiénes pueden acceder al sistema, para mantener seguridad sobre la integridad de los datos.
- **Criterio de aceptación:** Debe existir una vista que permita tanto la eliminación de cuentas de acceso al sistema y el otorgar el acceso para gestionar

los trabajadores que interactúen con él.

UH6
- **Titulo:** Descarga de Archivos de un Activo.
- **Historia de Usuario:** Como trabajador necesito tener acceso a archivos relacionados con cierto activo en específico, siendo estos archivos, planos e informes, entre otros, para realizar la mantención correspondiente.
- **Criterio de aceptación:** Dentro de la visualización de los datos de un activo se deben poder ver todos los archivos relacionados con este activo.

UH7
- **Titulo:** Ingreso de un Nuevo Activo al Sistema.
- **Historia de Usuario:** Como trabajador necesito rellenar un formulario de los activos para registrar un nuevo activo en el sistema.
- **Criterio de aceptación:** En la vista se debe tener una sección donde se selecciona la familia del activo a ingresar y donde luego se debe poder proceder a rellenar todos los datos relacionados a ese activo para que se cree bien el registro del activo.

UH8
- **Titulo:** Edición de Registros para Correcciones.
- **Historia de Usuario:** Como administrador de activos, debo cambiar un registro de estado de un activo para corregir un error que hubo al ingresarlo.
- **Criterio de aceptación:** El trabajador debe poder encontrar el registro específico en cuestión, y debe poder editar los campos necesarios para la corrección de datos en el registro histórico.

UH9
- **Titulo:** Actualización del Estado Actual de un Activo.
- **Historia de Usuario:** Como trabajador administrativo, necesito agregar un evento al registro histórico de un activo, para actualizar la información de este en el sistema y así tener un seguimiento apropiado.
- **Criterio de aceptación:** El trabajador al haber buscado un activo y seleccionarlo para actualizarlo, se debe mostrar un formulario a llenar con los

datos a actualizar tales como subestación de instalación, estado (código definido por el cliente), encargado del registro y observaciones.

3.3.2. Documentación OpenAPI

Las historias de usuario previamente establecidas permiten tener claridad sobre que funciones debiese cumplir la API. Considerando lo anterior, uno de los primeros pasos post investigación fue la creación de un archivo YAML, el cual sigue el estándar de **OpenAPI 3.0.4**, este cumple la función de ser el archivo de documentación de la API principal. Este archivo no solo se creó con el propósito de contener el diseño inicial, sino para irse actualizando a lo largo del trabajo hasta su conclusión.

El archivo tipo OpenAPI permite ordenadamente mantener el registro de todas las rutas bajo las entidades (tags) que se consideran para esta API. Las entidades son las siguientes: autenticación, activos, eventos, usuarios, documentos y compañías. Igualmente, con este archivo se incluyen posibles respuestas y *schemas* involucrados de las rutas.

Finalmente, esta documentación fue esencial para establecer una base para comenzar con la implementación. Luego, en el proceso de desarrollo, funcionó como un registro de todo cambio y evolución que se fue agregando a la API. Esto es esencial para establecer el puente con el frontend e igualmente tener un documento de referencia, el cual se puede complementar con herramientas visuales (Swagger UI) que faciliten todo trabajo futuro.

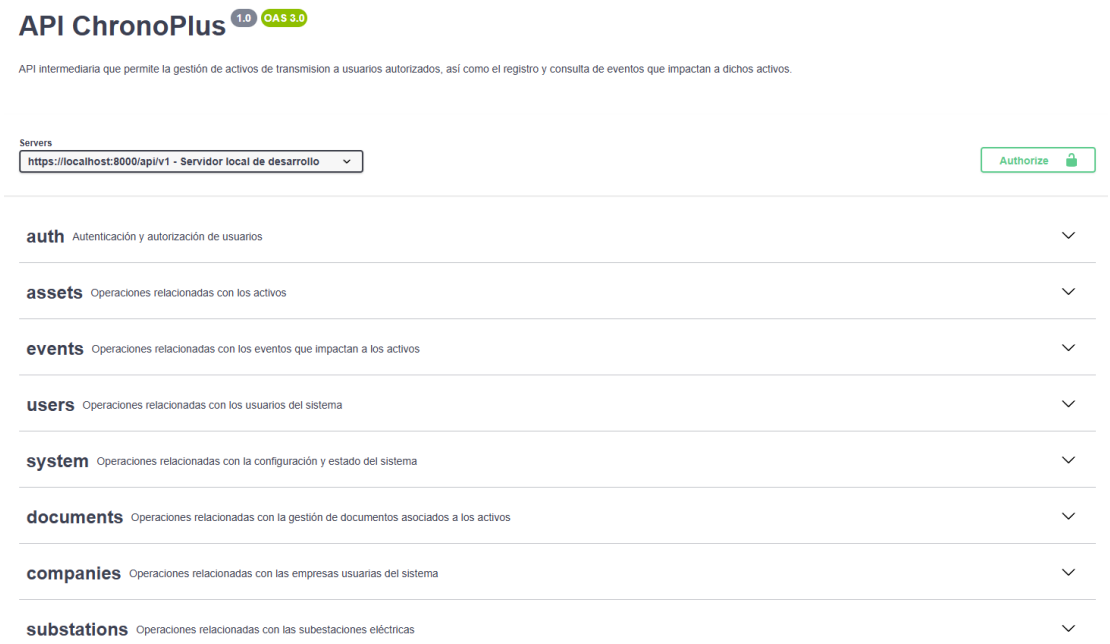


Figura 3.3: Archivo OpenAPI YAML visualizado a través de Swagger Editor.

3.3.3. Estructura Repositorio API

Previamente, se hace la mención de múltiples categorías que componen la API y a la vez representan el repositorio del código de este componente. Estas categorías son importantes para mantener una escalabilidad y facilidad de trabajo a largo plazo. A continuación se describe cada una de estas para profundizar.

Configuraciones

Esta parte es esencial para establecer las variables para conectarse a ambas bases de datos, la de tipo PostgreSQL y la de Google Cloud Storage. La conexión hacia la base de datos PostgreSQL, se logra gracias a los paquetes de *pg* y variables de ambiente (paquete dotenv), lo cual permite configurar la conexión como se estime necesario, como con ciertos grados de seguridad SSL.

Adicionalmente, se encuentran funciones relacionadas a **JSON Web Tokens (JWT)**, las cuales permiten la generación de tokens para los mecanismos de seguridad.

Middlewares

En cuanto al middleware, el cual permite robustamente implementar Express, se mencionan sus dos partes importantes: funciones de autenticación y de autorización hacia las rutas. En cuanto la autenticación, la cual ocurre en primer lugar, se apoya en la generación de tokens (JWT) y una clave secreta para asegurarse de que el usuario que intenta acceder a las rutas ha ingresado al sistema recientemente y con credenciales válidas. Luego, en las rutas correspondientes, se avanza al paso de autorización, esto para las funciones donde el rol de administrador o técnico es relevante al escenario. De esta manera, se incorpora un grado de seguridad y accesibilidad a los datos correspondientes.

Rutas

El núcleo de la API se puede representar a través de sus rutas, donde los otros servicios acceden a estas para poder manipular los datos como deseen. Cada ruta tiene su nombre o URL con el cual se distinguen, pero todas siguen la misma lógica cuando se trata de acceder a estas: primero se debe pasar por los middlewares (autenticación y autorización), y después a la función con los parámetros o datos enviados a través del llamado hacia la ruta. Es importante recalcar que estas funciones igualmente se encuentran divididas en tres capas para una organización de trabajo eficiente, estas son: controlador, servicio y modelo. Las cuales se pueden describir de la siguiente manera:

- **Controlador:** Se determinan los parámetros esperados de las llamadas y las posibles respuestas de cada ruta, conjunto sus códigos correspondientes de HTTP. Luego se pasa a la capa de servicio.
- **Servicio:** Acá es donde se procesan las variables desde un punto de vista de negocio, como por ejemplo, la validación de los parámetros, reglas del ambiente de trabajo o el armado de la respuesta en base a lógica preestablecida. En algunos casos, se establece la consulta hacia la base de datos que se invoca desde la capa modelo.

- **Modelo:** Finalmente, esta es la capa donde se encuentran ya las consultas directas hacia las bases de datos, en su lenguaje de PostgreSQL. En algunos casos, las consultas reciben parámetros de la capa anterior y, si no, vienen escritas para el caso correspondiente.

Finalmente, hay rutas para cada entidad que se había mencionado previamente: activos, autenticación, compañía, documentos, eventos, subestaciones y usuarios. Estas son las entidades más relevantes para este trabajo y de esta manera se mantienen escalables a futuro. Un listado de estas se presentará más adelante.

Utilidad

En todo componente siempre habrá una categoría de utilidad para mantener repositorios de código limpios y eficaces. En este caso, la utilidad encontrada bajo esta sección se encuentra enfocada en la seguridad y el *mapping* entre las variables de la API y los nombres de las variables en la base de datos. En cuanto a la seguridad, múltiples consultas PostgreSQL se arman indicante, para evitar los riesgos que conlleva esto, se utiliza un listado de filtros tipo whitelist. Esto existe tanto para operaciones con los eventos como con los activos; así, solo lo que se encuentre en estas listas va a poder ser interactuado a través de las consultas de la API.

3.3.4. Consideraciones de la Problemática en el Diseño

Dentro de la problemática surgen necesidades como la seguridad y eficiencia del sistema para los múltiples usuarios que lo estén simultáneamente. Esto incluye características como que la herramienta solo esté disponible para ciertos usuarios con credenciales al día, que el sistema se encuentre con la información necesaria para la trazabilidad y que, finalmente, pueda ser utilizado por ambos tipos de usuarios que fueron identificados previamente, administrativo y técnico.

Considerando lo anterior se establecieron algunos elementos como los siguientes para alcanzar los requisitos:

- **Autenticación** para toda ruta, menos la de ingreso al sistema, esto llevado a cabo través de **JSON Web Tokens**, los cuales permiten establecer sesiones con los usuarios las cuales expiran luego de un tiempo pre establecido, como por ejemplo 15 a 30 minutos.

- **Autorización Role Based Access Control (RBAC):** La autenticación debe ir acompañada de una autorización robusta, la cual se logró con la implementación de roles para los usuarios, esto basado en la identificación de tipos de usuarios ya establecida. Esto es esencial para asegurarse de que no haya errores u operaciones indebidas por parte de algunos de los usuarios ya autenticados en el sistema debido a una falta de permisos administrativos. Por ejemplo, en el sistema, la creación de más cuentas de usuarios solo se encuentra habilitada para los que tengan el rol de administrador. Adicionalmente, se dejó la base para a futuro continuar este trabajo con **Attribute Based Access Control (ABAC)**, debido a que la empresa de la problemática es extensa, por lo cual se beneficiaría de un control más granular hacia los datos. Por ejemplo, si un usuario pertenece a la subestación *A* solo puede registrar eventos de un activo perteneciente a esta misma subestación.

- **Field Projection:** Este concepto le permite a algunas de las rutas de lectura, como la de los activos, solo consultar por los datos requeridos en cada caso. Esto se realiza a través de un **query parameter**, llamado **fields**. La importancia de este parámetro, es que permite reducir y precisar la cantidad de datos a mostrar, no solo en las múltiples vistas de la interfaz, sino también en los distintos tipos de dispositivos. Esto con el fin de aumentar la eficiencia y parte de la seguridad sin tener que enviar datos que realmente no sean necesarios al caso.

3.3.5. Funciones API

Una vez ya mencionadas las consideraciones y estructura de este componente detalladamente, se lista a continuación cada función capaz a través de las rutas existentes. Esto incluye su tipo y descripciones de forma concisa, para más detalle, se recomienda

la lectura de la documentación OpenAPI. Estarán categorizadas bajo cada una de las entidades o tags a los cuales pertenecen.

Autenticación (auth)

Rutas relacionadas con la autenticación de los usuarios al sistema, a pesar de no ser del tipo REST, son esenciales para brindar parte de la capa de seguridad.

RT-A 1
- **Tipo:** POST
- **Ruta:** /auth/login
- **Descripción:** Autentica a un usuario y devuelve un JWT.

RT-A 2
- **Tipo:** POST
- **Ruta:** /auth/logout
- **Descripción:** Invalida el token JWT, cerrando la sesión del usuario.

RT-A 3
- **Tipo:** POST
- **Ruta:** /auth/change-password
- **Descripción:** Cambia la contraseña del usuario autenticado actualmente.

Activos (assets)

Rutas relacionadas con los activos, esto incluye mayormente operaciones de lectura o registro de datos relacionados con estos.

RT-B 1
- **Tipo:** GET
- **Ruta:** /assets
- **Descripción:** Obtiene todos los activos, con opciones de filtrado, ordenamiento y límite.

RT-B 2
- **Tipo:** POST

- **Ruta:** /assets
- **Descripción:** Crea un nuevo activo y en base a los datos de entrada les asigna un ID único (TAG).

RT-B 3
- **Tipo:** GET
- **Ruta:** /assets/count
- **Descripción:** Obtiene el número total de activos registrados.

RT-B 4
- **Tipo:** GET
- **Ruta:** /assets/export
- **Descripción:** Exporta los activos filtrados en formato Excel (.xlsx).

RT-B 5
- **Tipo:** GET
- **Ruta:** /assets/id
- **Descripción:** Obtiene un activo por su ID (TAG).

RT-B 6
- **Tipo:** PUT
- **Ruta:** /assets/id
- **Descripción:** Actualiza un activo existente por su ID.

RT-B 7
- **Tipo:** GET
- **Ruta:** /assets/id/documents
- **Descripción:** Obtiene los documentos asociados a un activo.

RT-B 8
- **Tipo:** POST
- **Ruta:** /assets/id/documents
- **Descripción:** Sube un nuevo documento asociado a un activo.

RT-B 9
- **Tipo:** DELETE
- **Ruta:** /assets/id/documents

- **Descripción:** Borra un documento asociado a un activo.

RT-B 10
- **Tipo:** GET
- **Ruta:** /assets/id/events/export
- **Descripción:** Exporta todos los eventos relacionados con un activo específico.

Eventos (events)

Rutas relacionadas con los eventos, el concepto principal con el cual se lleva la trazabilidad de los activos.

RT-C 1
- **Tipo:** GET
- **Ruta:** /events
- **Descripción:** Obtiene todos los eventos, por defecto retorna los últimos 20 eventos recientemente registrados, puede ser modificado con el parámetro *limit*.

RT-C 2
- **Tipo:** POST
- **Ruta:** /events
- **Descripción:** Crea un nuevo evento asociado a un activo, por un usuario autorizado.

RT-C 3
- **Tipo:** GET
- **Ruta:** /events/id
- **Descripción:** Obtiene un evento por su ID (TAG).

RT-C 4
- **Tipo:** PUT
- **Ruta:** /events/id
- **Descripción:** Actualiza un evento existente por su ID (TAG).

RT-C 5
- **Tipo:** DELETE
- **Ruta:** /events/id
- **Descripción:** Elimina un evento por su ID.

RT-C 6
- **Tipo:** GET
- **Ruta:** /assets/id/events
- **Descripción:** Obtiene todos los eventos asociados a un **activo específico**, por defecto retorna los últimos 20 eventos, puede ser modificado con el parámetro *limit*.

Usuarios (users)

Rutas relacionadas con los usuarios registrados en el sistema, incluye operaciones como la creación de más cuentas y el acceso a los datos de estos.

RT-D 1
- **Tipo:** GET
- **Ruta:** /users
- **Descripción:** Obtiene todos los usuarios del sistema. Por defecto retorna los últimos 20 usuarios. Se puede filtrar por rol y/o empresa, y soporta paginación y ordenamiento.

RT-D 2
- **Tipo:** POST
- **Ruta:** /users
- **Descripción:** Crea un nuevo usuario en el sistema.

RT-D 3
- **Tipo:** GET
- **Ruta:** /users/me
- **Descripción:** Obtiene la información del usuario autenticado.

RT-D 4
- **Tipo:** GET

- **Ruta:** /users/id
- **Descripción:** Obtiene un usuario por su ID.

RT-D 5
- **Tipo:** PUT
- **Ruta:** /users/id
- **Descripción:**Actualiza un usuario existente por su ID.

Documentos (documents)

Algunas de las rutas relacionadas con los documentos se encuentran bajo activos; sin embargo, aun así se encuentra la siguiente bajo esta categoría.

RT-E 1
- **Tipo:** GET
- **Ruta:** /documents/id/download
- **Descripción:** Permite la descarga de uno de los archivos a través de su ID.

Compañías (companies)

Operaciones de lectura de datos de las compañías (empreas).

RT-F 1
- **Tipo:** GET
- **Ruta:** /companies
- **Descripción:** Obtiene la lista de empresas registradas

Subestaciones (substations)

Operaciones de lectura de datos de las subestaciones.

RT-G 1
- **Tipo:** GET
- **Ruta:** /substations
- **Descripción:** Obtiene la lista de todas las subestaciones registradas.

Todo lo anterior resume las rutas existentes de la API que permiten que el resto de los componentes interactúen para llevar la trazabilidad de los activos.

3.4. Importación de Datos

El proceso de importación de datos fue una etapa clave dentro del desarrollo de la solución, ya que permitió incorporar al sistema los registros históricos entregados por el cliente, los cuales se encontraban en un archivo Excel con una gran cantidad de información asociada a los distintos activos de transmisión eléctrica.

Durante la revisión inicial del archivo, se identificó que los datos no estaban normalizados ni estructurados de manera uniforme. Existían tipos de datos inconsistentes (por ejemplo, campos numéricos almacenados como texto o fechas con diferentes formatos), duplicidades y omisiones. Esta falta de estandarización generaba un riesgo elevado de errores al momento de la carga hacia la base de datos, además de dificultar la trazabilidad y análisis posterior.

Para abordar esta problemática, se implementó, dentro del módulo de importación de datos, un proceso ETL (Extracción, Transformación y Carga), que consta de las siguientes etapas:

- **Extracción**: lectura del archivo Excel proporcionado por la empresa del caso de uso, mediante la biblioteca *Pandas* de *Python*.
- **Transformación**: limpieza y estandarización de los datos. En esta etapa se realizaron tareas como:
  - Normalización de nombres de columnas.
  - Conversión de formatos de fecha a un estándar (datetime).
  - Validación de campos obligatorios.
  - Conversión de tipos de datos según los requerimientos de la base de datos.
- **Carga**: inserción de los registros procesados en la base de datos PostgreSQL, utilizando SQLAlchemy como ORM para mapear las entidades y asegurar la integridad referencial.

Además, se observó que cada familia de activos requería distintos campos y estructuras de datos. Por esta razón, se diseñaron archivos auxiliares de configuración que definieron los tipos de datos requeridos para cada familia, permitiendo validar y adaptar el proceso de importación según el tipo de activo. Este enfoque modular facilitó la escalabilidad del sistema, ya que nuevas familias de activos pueden ser incorporadas simplemente actualizando los archivos de configuración.

Como resultado de este proceso, los datos fueron estandarizados y cargados de manera controlada, reduciendo la posibilidad de errores humanos y asegurando la consistencia de los registros en la base de datos. Este paso fue fundamental para establecer una base sólida sobre la cual se construye la trazabilidad histórica de los activos dentro del sistema.

3.5. Base de datos

El diseño del modelo de datos constituye una parte fundamental del sistema, ya que es el encargado de almacenar toda la información relacionada con los activos de transmisión eléctrica, sus características, estados, y los eventos que ocurren a lo largo de su ciclo de vida. El objetivo principal de este diseño fue desarrollar una estructura que permitiera mantener un registro histórico completo de los cambios realizados sobre los activos, garantizando al mismo tiempo consistencia, escalabilidad y eficiencia en las operaciones.

3.5.1. Proceso de diseño

El desarrollo del modelo comenzó con un análisis de los datos entregados por el cliente y de los requerimientos de trazabilidad definidos durante la etapa de levantamiento. A partir de esta revisión, se identificó la necesidad de diferenciar entre dos tipos de información asociada a cada activo:

- **Datos estáticos:** corresponden a atributos que no cambian a lo largo del tiempo,

como el identificador del activo, su tipo, fabricante o fecha de fabricación.

- **Datos dinámicos:** representan atributos susceptibles de modificación durante la vida útil del activo, como su ubicación, estado operativo o la empresa a la que pertenece.

Esta separación se hizo pensando en el mantener un historial de los datos que irán cambiando con el tiempo de mejor manera, todo esto a través de funcionalidades incluidas en PostgreSQL como los Triggers o CDC.

3.5.2. Estructura del modelo

El modelo de base de datos diseñado está compuesto por las siguientes entidades principales:

- **Usuarios:** tabla que almacena la información de los distintos perfiles que interactúan con el sistema, manteniendo la diferencia de roles presentes y de qué empresa son estos.
- **Datos estáticos:** tiene información que no debería cambiar con el tiempo. Estos datos son la marca, modelo, la fecha de adquisición del activo entre otros.
- **Datos dinámicos:** contiene la información sujeta a cambios, como ubicación, estado operativo o responsable.
- **Eventos:** tabla que registra acciones relevantes sobre los activos, tales como mantenciones, inspecciones o traslados. Cada evento se asocia a un usuario y a un activo, permitiendo reconstruir la secuencia de actividades asociadas a cada uno.
- **Historial de dinámica:** tabla especialmente diseñada para almacenar los cambios históricos realizados sobre los datos dinámicos de los activos. Esta tabla es utilizada cuando se necesita actualizar el dato de un activo, pero no en base a un evento. Cada vez que se modifica un registro en la tabla de datos dinámicos, un

trigger inserta automáticamente una copia del estado anterior en esta tabla, junto con información sobre la fecha y el usuario responsable del cambio.

- **Familia de activo:** entidad la cual contiene los formularios de datos específicos correspondientes a cada familia de activo. Esto permite poder implementar los formularios necesarios para el ingreso de nuevos activos al sistema.

Esta estructura permite mantener un registro completo y automático de la evolución de cada activo, sin requerir mayor intervención manual.

3.5.3. Mecanismo de trazabilidad

La trazabilidad, como se mencionó anteriormente, se implementó mediante un trigger definido sobre la tabla de datos dinámicos y con una tabla de eventos.

Cada vez que se realiza una operación de actualización, el trigger:

- Captura el estado previo del registro.
- Inserta una nueva entrada en la tabla historial_dinámica.
- Registra información complementaria como:
  - El identificador del activo afectado.
  - El usuario que ejecutó la modificación.
  - La fecha y hora exacta del cambio.

Este mecanismo asegura la persistencia de todos los estados anteriores de cada activo, permitiendo reconstruir en cualquier momento su evolución temporal y cumplir con los requisitos de trazabilidad definidos por normas como la ISO 55001.

El uso específico de una tabla de eventos es para tener un mejor manejo en cuanto a como se ha administrado el activo. Al ingresar un evento se actualizarán los datos que están en la tabla dinámica, así manteniendo la consistencia de estos.

Capítulo 4

Resultados

Los resultados de este trabajo se ilustran en este capítulo a través de una muestra de las vistas de la interfaz y cómo estas se incorporan con el backend, especialmente con la API, para así brindar las funcionalidades necesarias para resolver la problemática descrita. Esto acompañado de una discusión de la validación de sus componentes y casos de uso.

4.1. Módulos Incorporados

Esta sección busca demostrar cómo la API está incorporada en las múltiples vistas de la interfaz del sistema, y cómo estas aportan a cumplir con los múltiples requisitos establecidos en el proceso.

Vista 1: Inicio de Sesión

En esta vista tradicional de inicio de sesión, logra funcionar a través de la ruta */auth/login* (RT-A 1). Adicionalmente, los usuarios pueden cerrar sesión en cualquiera de las otras vistas que se mostrarán, gracias a */auth/logout* (RT-A 2) y se ven redirigidos a esta vista inicial. Importante que esta parte del sistema ayuda a cumplir principalmente con el RNF1, cuyo punto principal es que el sistema sea seguro, y a la vez con UH1,

historia de usuario que describe la necesidad de un ingreso de sesión con credenciales.

Figura 4.1: Vista Inicio de Sesión

Vista 2: Buscador de Activos Tradicional

Esta es la primera vista de tres, de las cuales están enfocadas a la búsqueda de activos, esta siendo la más tradicional con tablas de múltiples columnas la cual se actualiza con barra de búsqueda y filtros adicionales. Acá la ruta principal relevante es la */assets* (RT-B 1), la cual permite la búsqueda con todos los filtros presentes en pantalla e incluye una respuesta con paginación para no sobrecargar el sistema en casos particulares. Igualmente, en esta vista se permite la exportación de los datos filtrados con la barra de búsqueda, esto a través de un archivo tipo Excel. Esta función de exportación es posible a */assets/export* (RT-B 4), el cual toma la búsqueda filtrada anterior y genera el archivo para que el usuario lo descargue a través de su navegador.

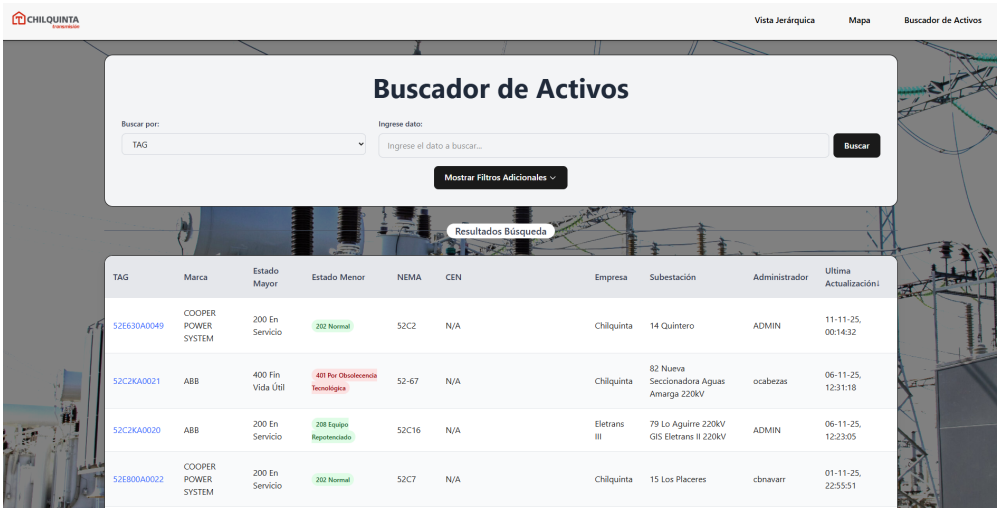


Figura 4.2: Vista Buscador de Activos Tradicional

La incorporación de los módulos en esta vista ayudan a aportar a completar requisitos tales como el RF3, el cual destaca la necesidad de la visualización de registros, especialmente el de una hoja de vida. Por el lado no funcional, se aporta hacia los siguientes RNF3 y RNF4, visualización rápida de los datos de los activos y existencia de filtros intuitivos. Finalmente, esta vista y su implementación están directamente relacionados con la historia de usuario 2 (UH2), la cual describe la necesidad de una vista como esta.

Vista 3: Buscador de Activos Geográfico

En el proceso del desarrollo se profundizaron las maneras de poder buscar los activos y sus datos, de esto resultaron dos buscadores adicionales que buscan lograr el mismo objetivo que en la vista 2 anterior. La vista a continuación corresponde al segundo buscador, el cual incorpora un mapa geográfico en conjunto la barra de búsqueda y filtros. Esto es posible debido a que los activos tienen datos de localización como latitud y longitud, y se aprovechan para ubicarlos en el mapa. Las rutas utilizadas en este caso son las mismas que la vista anterior, siendo RT-B 1 la principal, sin embargo, también se utiliza */assets/count* (RT-B 3), debido a que entrega el número total de activos existentes para inicialmente hacer una carga más completa de todos los datos GPS de

estos.

Figura 4.3: Vista Buscador de Activos Geográfico

En cuanto a que requisitos e historias de usuarios aporta a cumplir esta vista, son RNF3, RNF4 e igualmente UH2, como en el caso de la vista 2 mostrada anteriormente.

Vista 4: Buscador de Activos Jerárquico

Finalmente, este es el tercer buscador de activos, el cual utiliza una búsqueda jerárquica. Este buscador surgió, no solo de la necesidad de búsquedas más robustas, sino de la familiaridad del cliente con buscadores de este tipo. La idea de esto es que permita bajar la barra de entrada a aprender a utilizar la aplicación web, ya que habría elementos con los cuales los usuarios ya tendrían un grado de familiaridad. En esta vista, las rutas utilizadas y el aporte hacia la completitud son exactamente los mismos que en la vista 3.

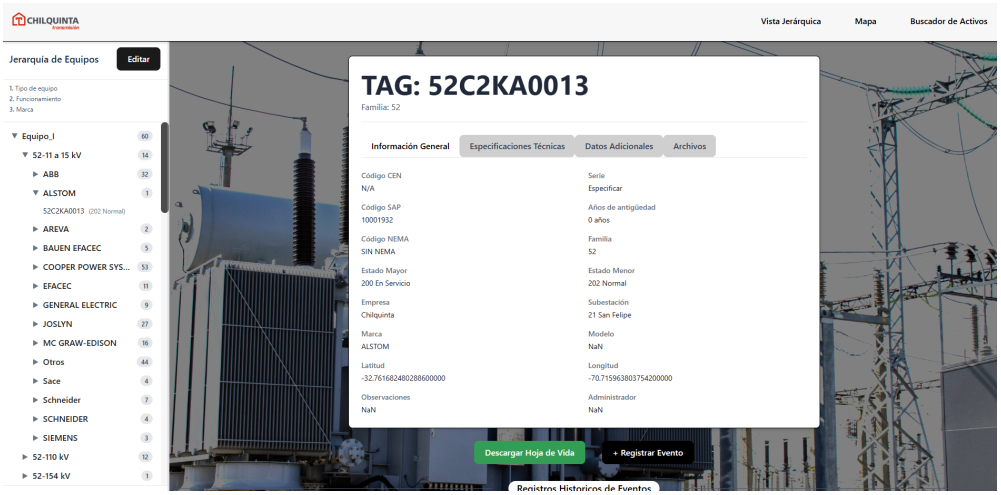


Figura 4.4: Vista Buscador de Activos Jerárquico

Vista 5: Detalles Activo

Para los detalles de los datos de cada activo se tiene la siguiente vista. Donde se encuentran múltiples categorías para la lectura de estos, y a la vez se muestran en pantallas todos los eventos que se van registrando bajo el activo. La idea de los eventos es que mientras se vayan creando, irán actualizando los campos correspondientes del activo como tal y guardando los datos en forma de historial. Aparte de servir como una vista de los datos, se puede optar por exportar todo evento en un archivo Excel, el cual representa la hoja de vida que se busca crear en esta solución. Adicionalmente, se da la opción para ir registrando más eventos o cargando, descargando documentos del activo en pantalla.

Acá se utilizan múltiples rutas para permitir las funciones de esta página, partiendo por la obtención de todo dato del activo a través de */assets/id* (RT-B 5), sin embargo, esto no incluye los eventos y los documentos relacionados con este. Para estos se utilizan */assets/id/events* (RT-C 6) y */assets/id/documents* (RT-B 7), respectivamente.

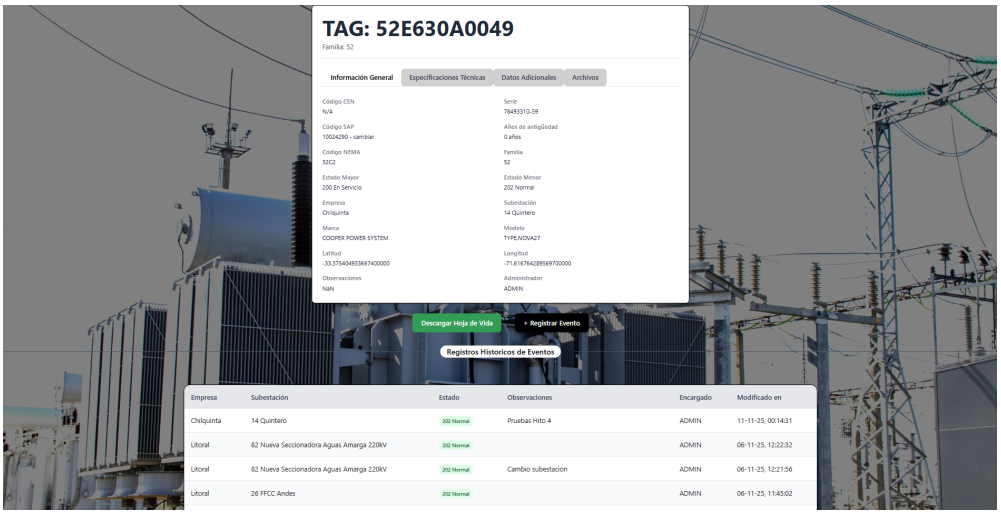


Figura 4.5: Vista Detalles Activo

Finalmente, esta vista es central a la solución ya que acá se aporta a múltiples metas, tales como RF1, RF3 y RF5. En cuanto a las historias de usuarios de las cuales se consideran para esta vista son UH3 y UH6, ya que se permite visualizar la hoja de vida en la tabla inferior y a la vez permite la descarga de esta. Por el otro lado, también se da la opción de poder descargar los archivos ya registrados bajo un activo.

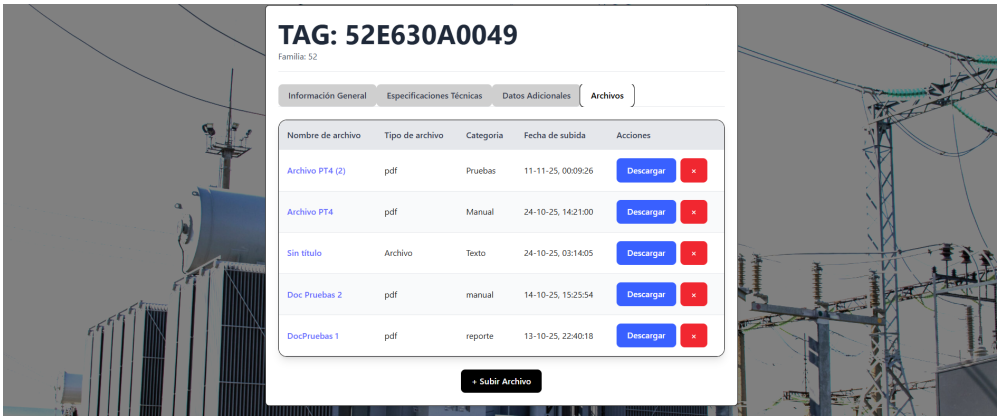


Figura 4.6: Vista Lista Documentos Activo

Vista 6: Formulario Registro de Eventos

Para llevar a cabo la creación de nuevos eventos y la trazabilidad de activos, existe este formulario el cual contiene los campos de datos más relevantes para la traza. Ini-

cialmente cuando carga este formulario, los campos se encuentran con los datos más recientes del activo, esto con el fin de aumentar la accesibilidad de la herramienta. Algunos de los campos incluyen menús desplegables con las opciones que deben existir según la problemática y también otros campos donde se puede explayar más al respecto.

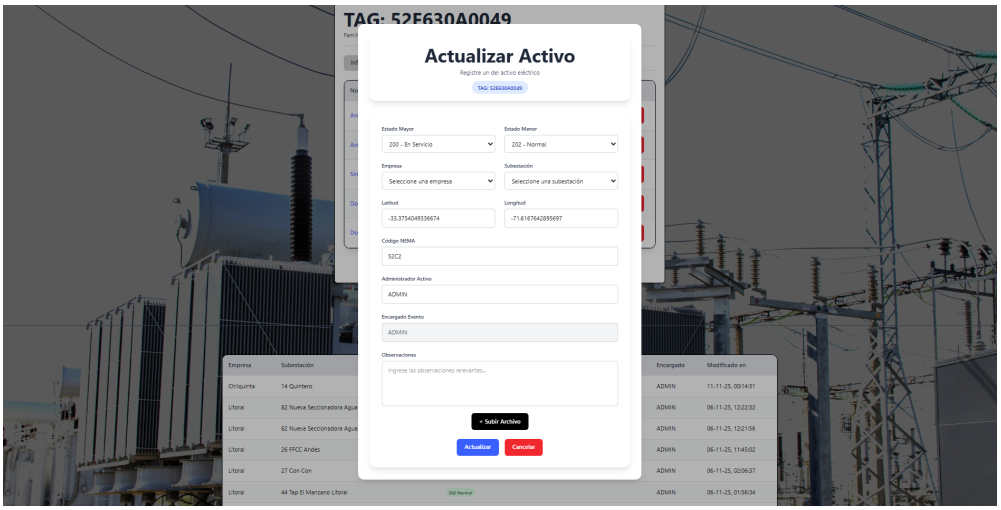


Figura 4.7: Vista Formulario Registro de Eventos

En esta vista se hace uso de más de una ruta, primero se tiene la más esencial */events* (RT-C 2), la cual toma todos los datos del formulario y crea el evento correspondiente. Luego también se utilizan las rutas como */companies* (RT-F 1) y */substations* (RT-G 1) para brindar los datos a los menús desplegables de manera dinámica. Finalmente también se utiliza */assets/id/documents* (RT-B 8) ya que también se encuentra la opción de poder subir archivos al activo.

Funciones Adicionales

A pesar de que la mayoría de las rutas de las API se utilizan a través del frontend, no todas tienen alguna manera para utilizarse de momento, a continuación se describen algunas de estas y las funcionalidades que permiten:

- **Modificar Datos Usuario:** La ruta de */users/id* (RT-D 5) permite actualizar valores relacionados al usuario esto puede ser valores tales como rol o empresa.

Esta función fue necesaria y es la base para escalar a futuro ya que cada usuario va a tener múltiples atributos como los anteriores los cuales pueden cambiar con el tiempo. A la vez, en este trabajo fue utilizado mayormente para cambiar roles y asegurarse de que las autorizaciones a los datos estén bajo un funcionamiento correcto.

- **Crear Cuenta Usuario**: Bajo la ruta */users* (RT-D 2) se encuentra la capacidad para crear nuevas cuentas para los usuarios que tengan el rol de administrador. Esto es necesario para escalabilidad a futuro e igualmente para la creación de cuentas de pruebas como se realizó en este trabajo.

- **Edición de Eventos:** Los eventos eventualmente podrían permitirse ser actualizados a través de esta ruta, */events/id* (RT-C 4), esto permitiría corregir algún evento en el caso de un error, ya que actualmente la manera de realizar esto es a través de la creación de otro evento.

4.2. Casos de Uso

En la siguiente sección, se documentan casos de uso con el fin de poder evidenciar con detalle el funcionamiento del sistema; para cada uno, se incluyen elementos como actores, flujo principal, precondiciones, postcondiciones, diagrama del flujo e imágenes de la interfaz de usuario durante el flujo.

4.2.1. Actualización del estado de un activo

Descripción: El usuario administrativo debe actualizar el activo de una familia en particular que se encuentra considerado para un proyecto en una subestación en específico para poder indicar que se ha instalado.

Actor: Usuario administrativo

Precondiciones: El usuario ha ingresado al sistema a través del Log In con sus credenciales.

Flujo Principal:

1. El usuario se encuentra en la página de inicio (buscador tradicional) después del ingreso a la plataforma.
2. Decide quedarse en la página de inicio para utilizar el buscador tradicional.
3. Despliega el panel superior con el botón de Filtros Adicionales.
4. Utiliza los filtros de Estado Mayor, Subestación y Familia del Activo; confirma la búsqueda con el botón de Aplicar Filtros.
5. Luego, dentro de la tabla con la búsqueda resultante, hace clic en el activo de su interés a través de la columna TAG.
6. Ahora, en la vista de detalles de un activo, se dirige al botón de Registrar Evento.
7. Cambia los estados mayor y menor, escribe el administrador a cargo y escribe una observación para indicar de qué trató el evento.
8. Revisa que el evento se haya registrado correctamente en la página de detalles del activo bajo la tabla de Registros Históricos de Eventos

Postcondiciones:

- Los datos anteriores del activo, se guardan en la base de datos.
- Nuevo evento registrado queda visible en la hoja de vida del activo.

Diagrama:

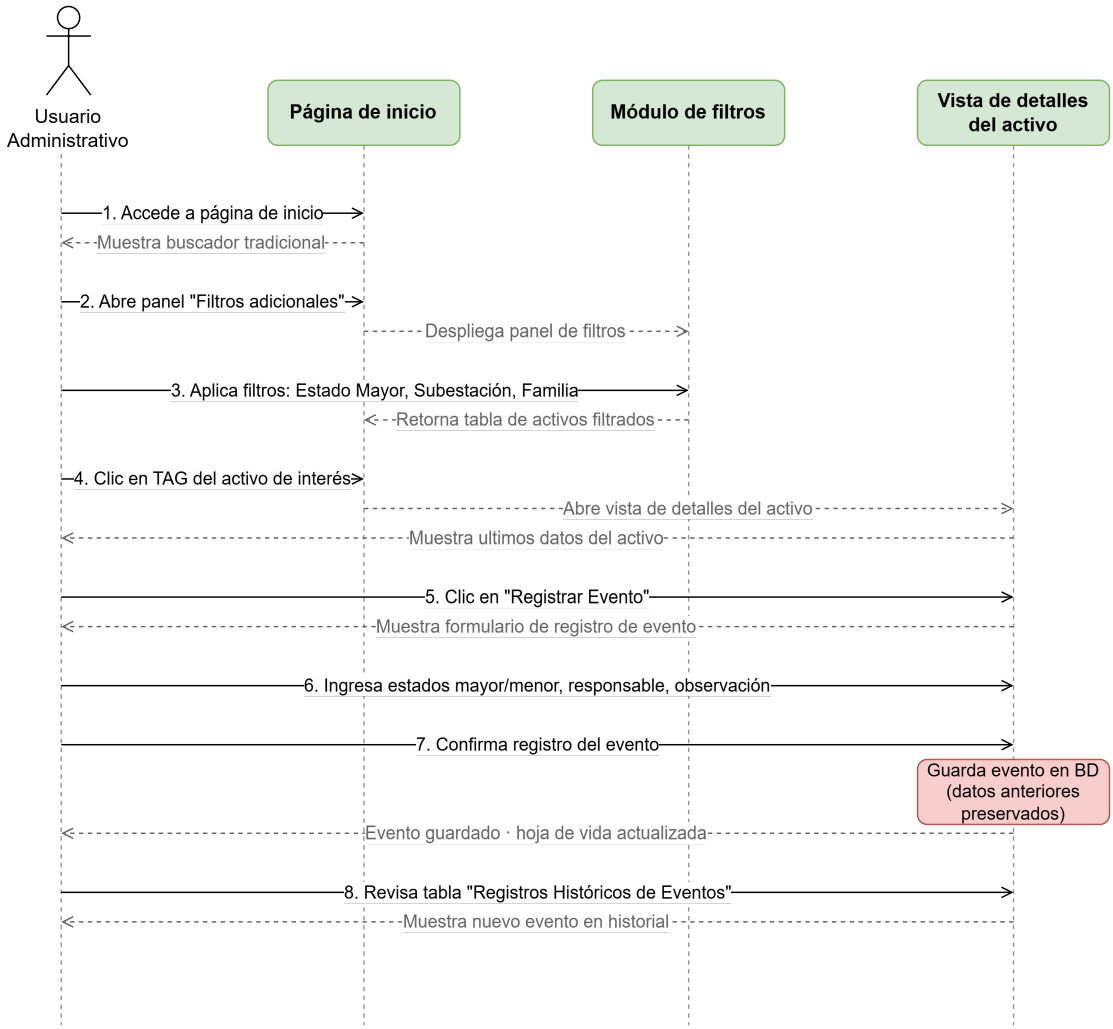


Figura 4.8: Diagrama de Flujo Caso de Uso 1

Imágenes UI del Flujo:

1. Inicialmente, el usuario está en el buscador tradicional del inicio, donde luego aprieta el botón de Mostrar Filtros Adicionales.

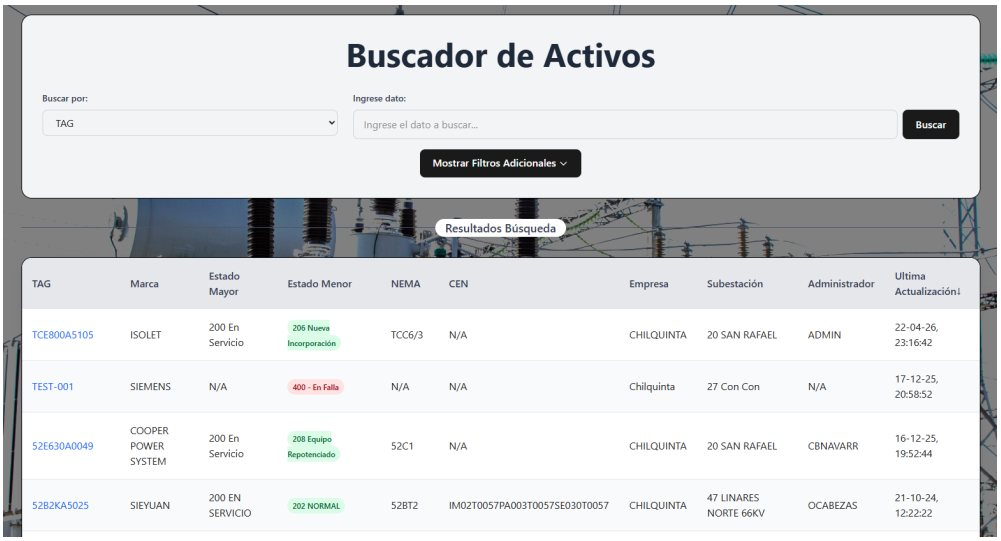


Figura 4.9: Vista Inicio - Buscador Tradicional

2. Al apretar el botón, termina con el panel desplegado con los múltiples filtros donde selecciona o rellena los de estado mayor, familia y subestación, para luego apretar el botón de Aplicar Filtros, el cual realiza la búsqueda.

Figura 4.10: Panel de Filtros - Buscador Tradicional

3. Los resultados de la búsqueda se muestran al inferior del panel de filtros, donde se puede notar que se filtró por los activos de la familia TC (TAG parte por las

iniciales del código), la subestación 20 San Rafael y los que se encuentran en estado mayor de 100 En Proyecto. Luego, el usuario va al activo que necesite para actualizarlo, apretando en el TAG correspondiente.

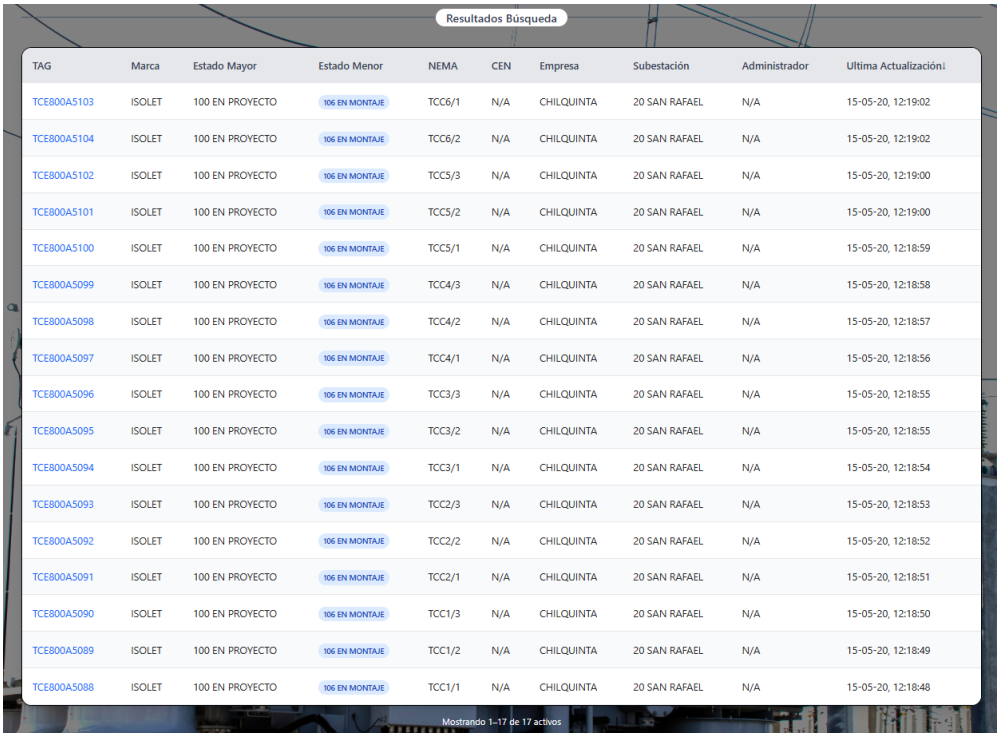


Figura 4.11: Resultados Búsqueda Filtrada - Buscador Tradicional

4. Una vez en la vista de detalles del activo, el usuario es presentado con el botón de Registrar Evento, el cual necesita apretar para actualizar el activo. Adicionalmente, el Registro Histórico de Eventos se encuentra vacío, debido a que no se han realizado registros a través de la plataforma para este activo.

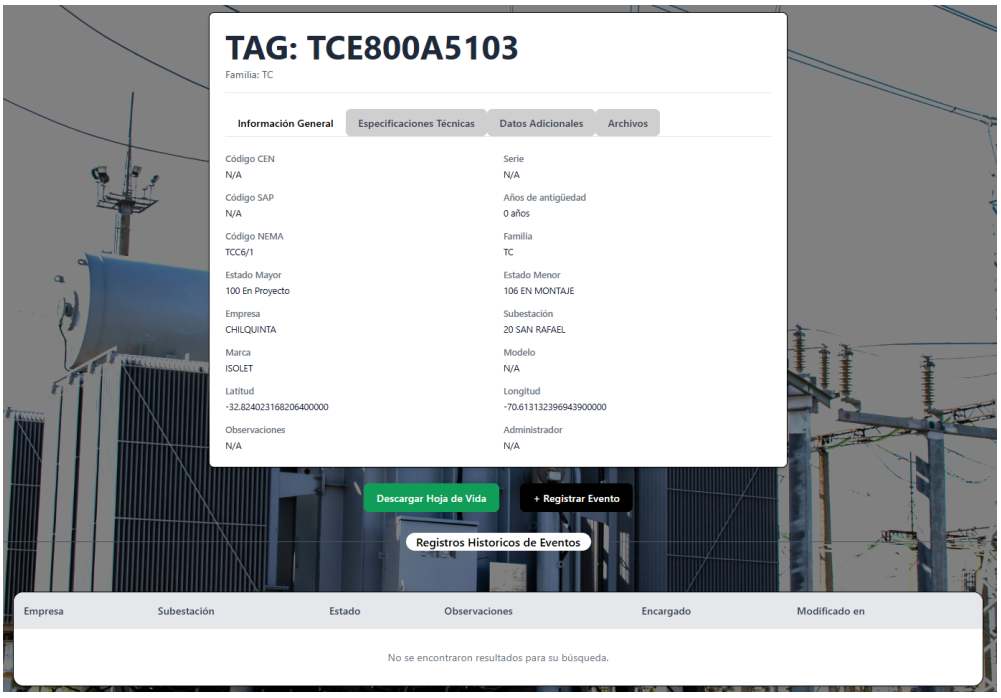


Figura 4.12: Detalles de un Activo

5. El botón de Registrar Evento permite acceder al formulario de registro de estos. El cual se encuentra precargado con los últimos datos del activo. Acá el usuario cambia estado mayor, estado menor y anota tanto el administrador del activo como una observación de este. Confirma el registro con el botón de Actualizar.

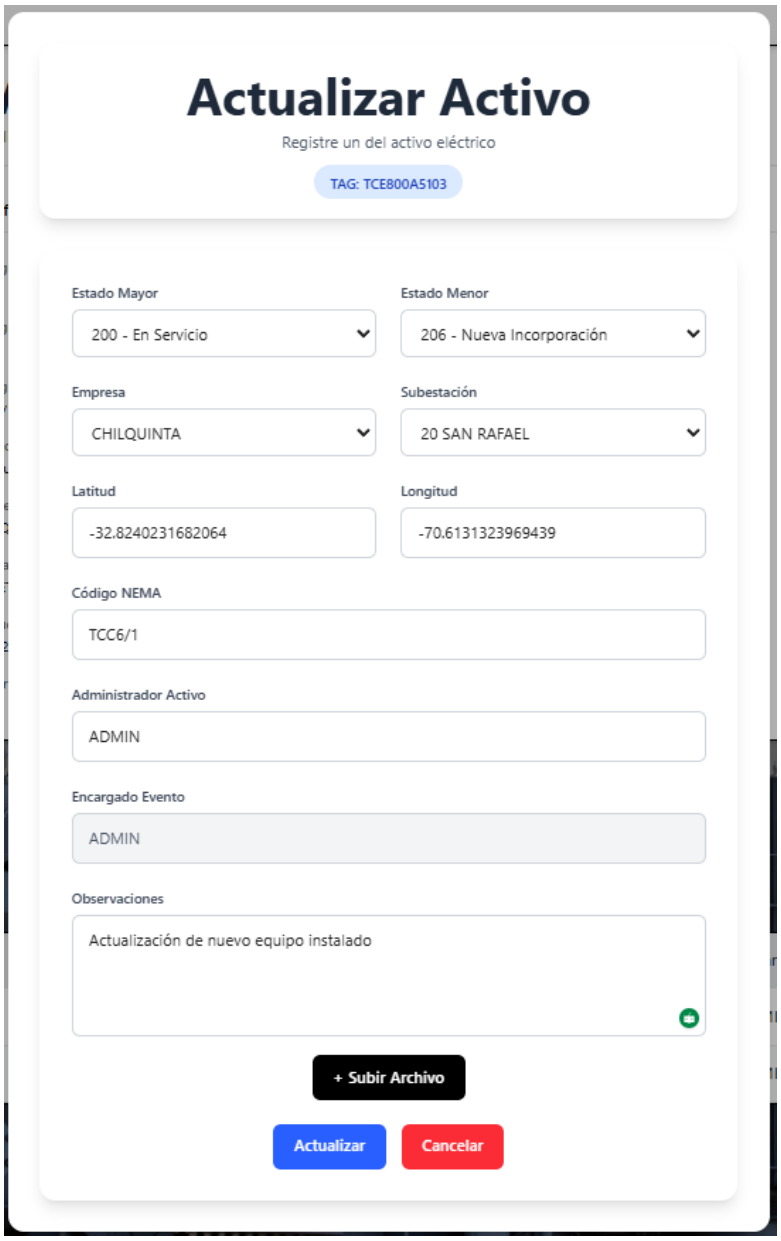


Figura 4.13: Formulario Registro de Eventos de un Activo

6. Finalmente, una vez ya hecho el registro, el usuario se encuentra en la vista de detalles y puede observar los cambios tanto en el panel superior (el cual contiene siempre la última información del activo) como en la tabla de Registros Históricos de Eventos.

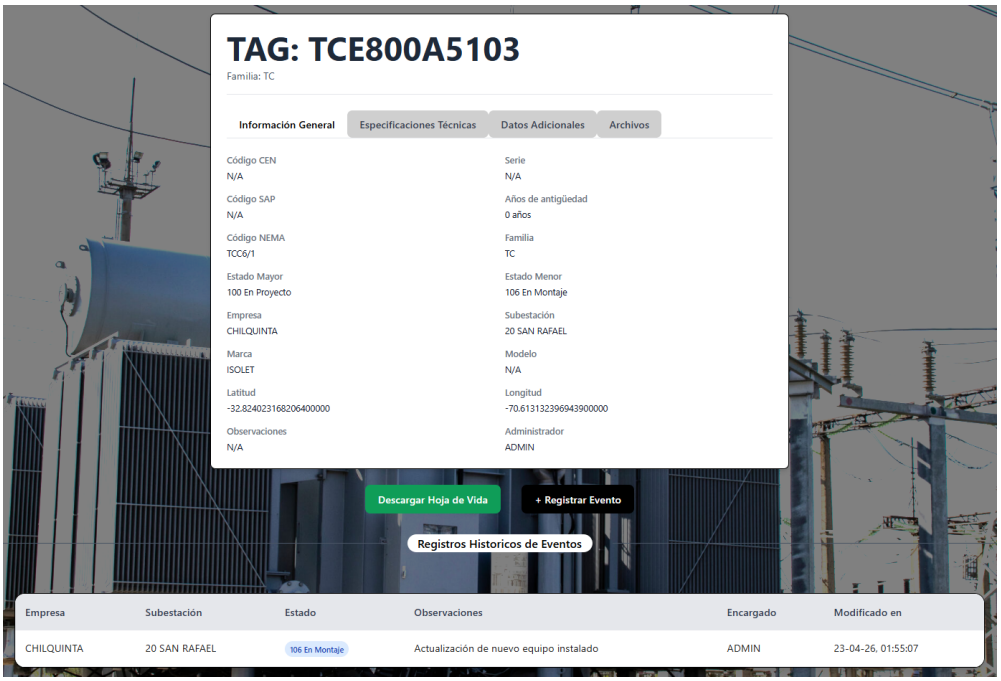


Figura 4.14: Registro Histórico con Cambio Reciente

7. Adicionalmente, se muestra el caso en que se registre otro evento para ilustrar el crecimiento de la hoja de vida del activo. Por ejemplo, si el activo empieza a mostrar fallas y se registra, la tabla resulta como a continuación.

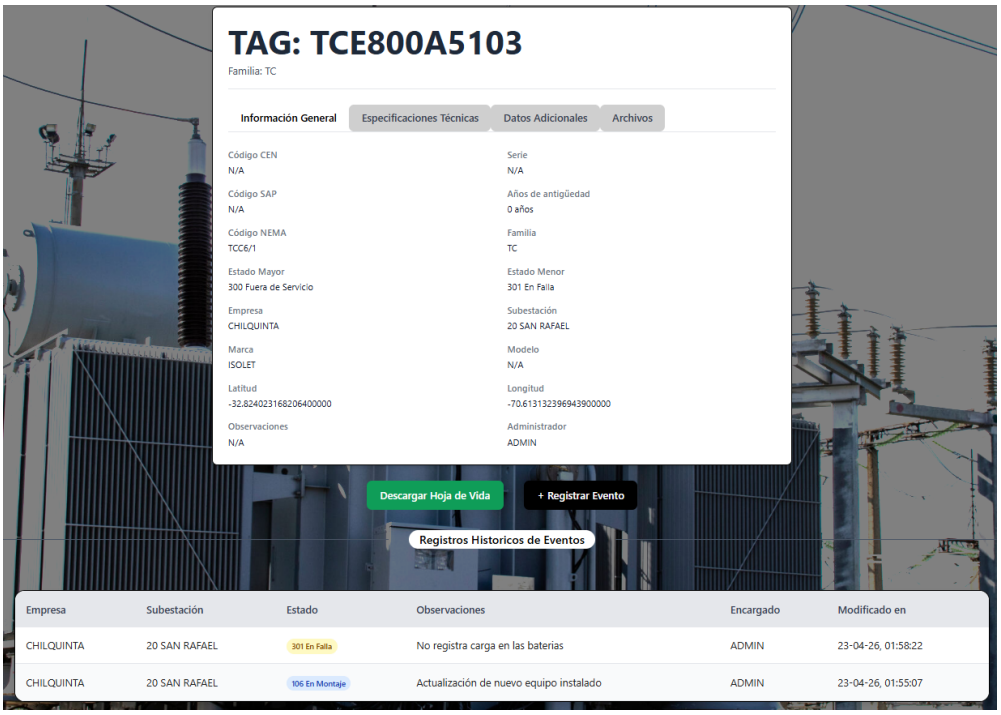


Figura 4.15: Registro Histórico con Múltiples Eventos

4.2.2. Subir documentos pertenecientes a un activo

Descripción: El usuario administrador (o técnico) quiere subir la última ficha técnica, la cual no se encuentra relacionada con un evento en particular del activo.

Actor: Usuario Administrador o Usuario Técnico

Precondiciones:

- El usuario ingresó al sistema recientemente con sus credenciales.
- Usuario utilizó uno de los buscadores para llegar a la vista de detalles de un activo.

Flujo Principal:

1. El usuario se encuentra en la vista de detalles del activo de interés, donde aprieta la categoría de Archivos que se encuentra en el panel superior.

2. Al usuario se le presenta una lista de los archivos existentes del activo y el botón de Subir Archivo, el cual aprieta.

3. Aparece un formulario de tres campos: uno para el archivo como tal, otro para la categoría de este y uno final para describir el archivo. Procede a llenarnos y confirmar la subida con el botón de Subir Archivo.

4. Finalmente, el usuario ve el archivo subido a través del panel superior.

Postcondiciones:

- El archivo se sube al servicio de Google Cloud Storage, y su metadata para acceder e informar en la base de datos de PostgreSQL.
- El archivo queda enlazado al activo a través de la metadata.

Diagrama:

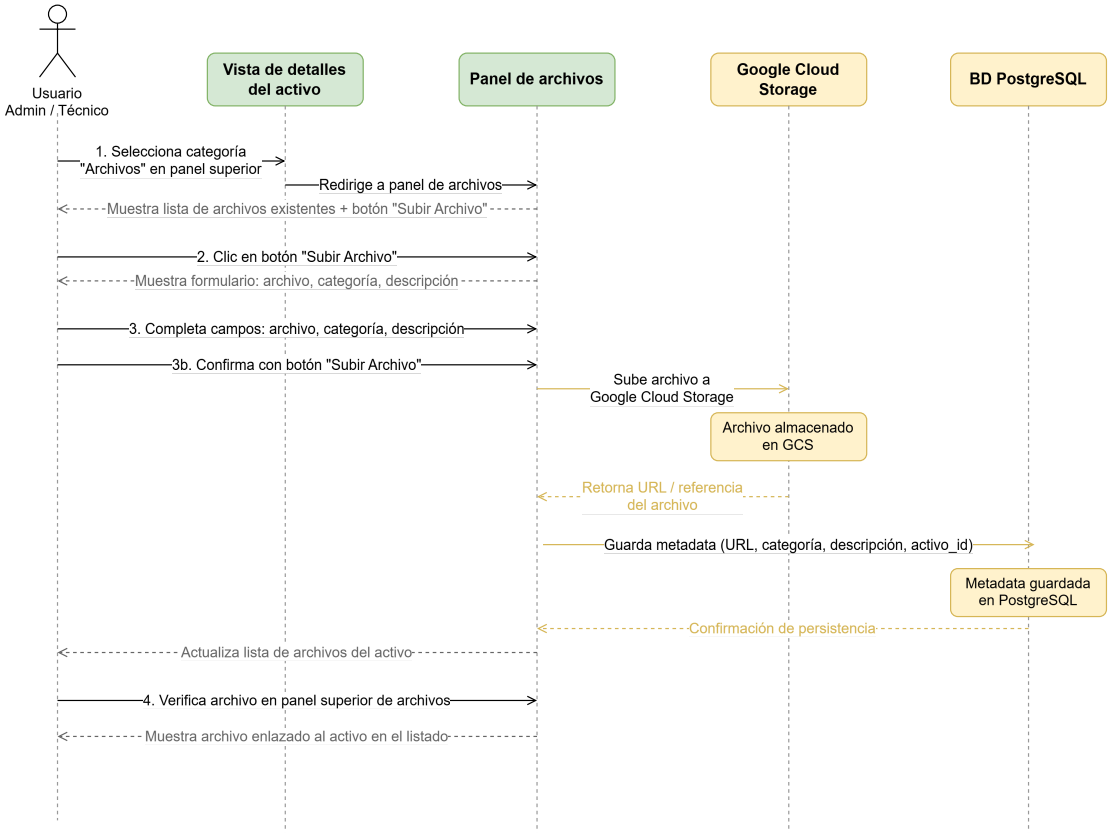


Figura 4.16: Diagrama de Flujo Caso de Uso 2

Imágenes UI del Flujo:

1. El usuario en la vista de detalles aprieta la categoría de Archivos y se encuentra en la siguiente vista.

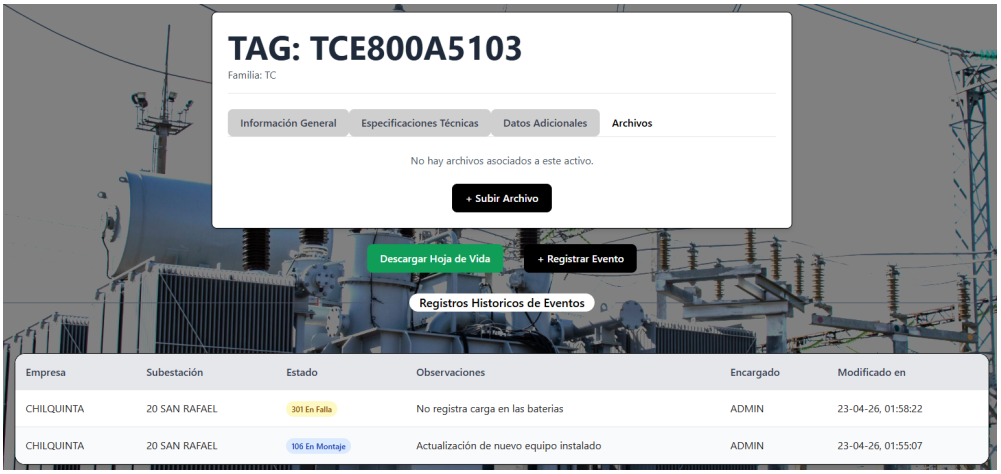


Figura 4.17: Categoría Archivos - Detalles de Activo

2. Al apretar el botón de Subir Archivo, se le presenta el formulario, donde puede describir el archivo, la categoría y subir el documento como tal. Confirma la subida con el botón de Subir Archivo.

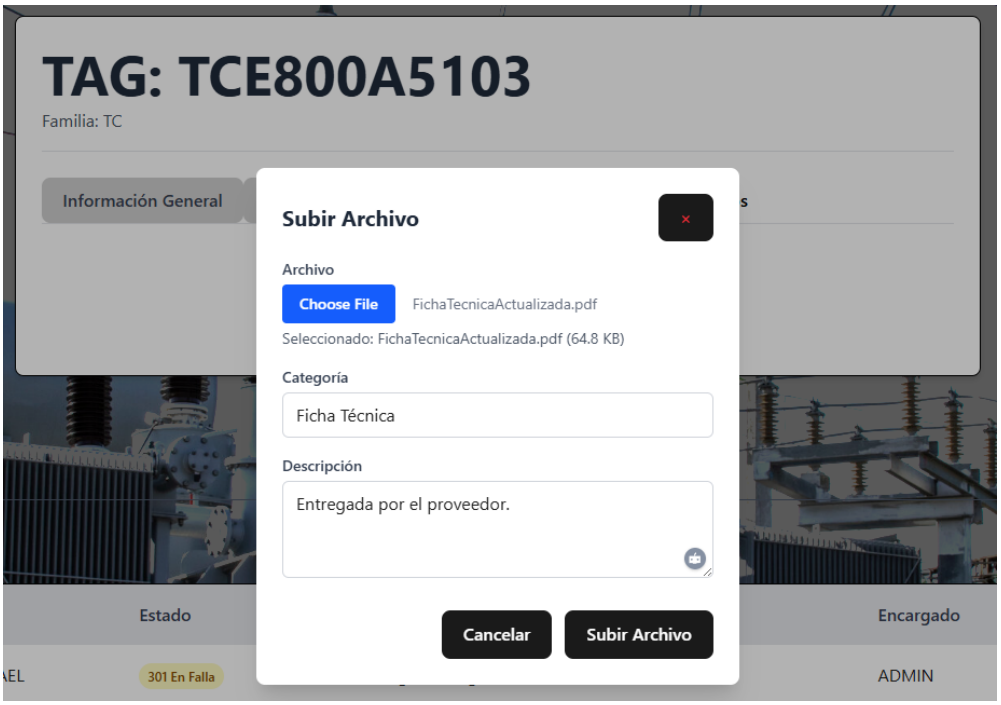


Figura 4.18: Formulario de Subida de Archivo

3. Luego el enfoque vuelve a los detalles del activo y su panel de Archivos donde se puede ver el archivo mas reciente agregado.

Figura 4.19: Categoría Archivos Actualizada

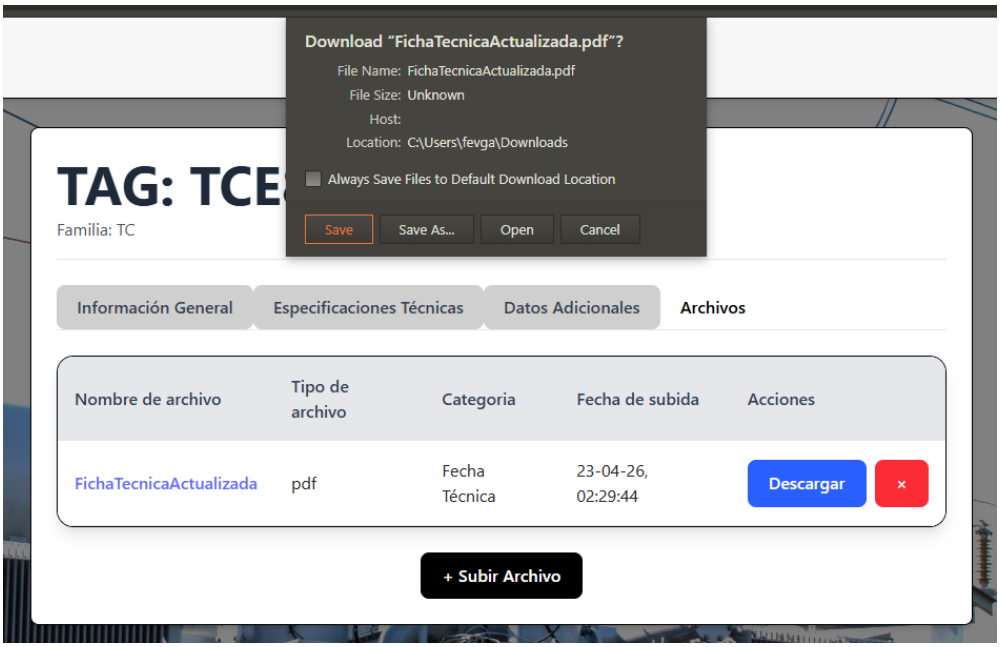


Figura 4.20: Descarga de un Archivo

4.2.3. Consultar historial de trazabilidad de activo

Descripción: Un usuario administrador desea ver el historial de un activo y descargarlo para compartirlo con otras personas para tomar decisiones.

Actor: Usuario administrador

Precondiciones:

- El usuario ingresó al sistema recientemente con sus credenciales.
- Usuario utilizó uno de los buscadores para llegar a la vista de detalles de un activo.

Flujo Principal:

1. El usuario se encuentra en los detalles de un activo, donde navega a la parte inferior de Registros Históricos de Eventos.
2. Observa el historial y decide descargarlo para ver los eventos mas a detalle y compartirlos a través del botón de Descargar Hoja de Vida.

3. Ahora el usuario tiene acceso a la hoja de vida en un archivo Excel, el cual puede compartir como desee.

Postcondiciones:

- Hoja de vida descargada para el activo de interés.
- Usuario puede visualizar datos del activo en la plataforma o en archivos Excel.

Diagrama:

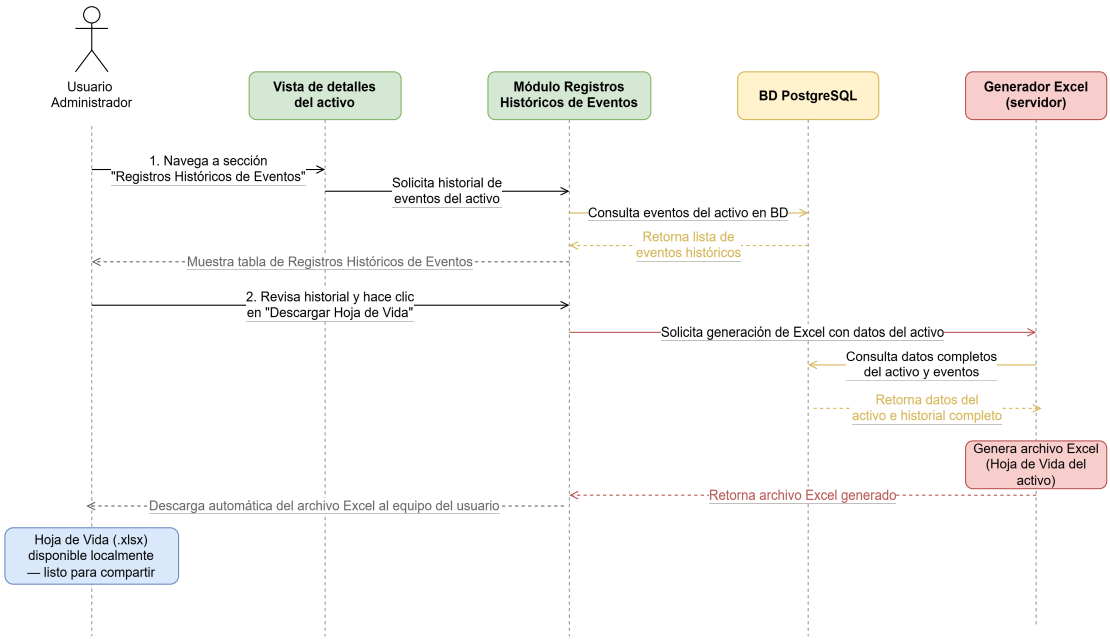


Figura 4.21: Diagrama de Flujo Caso de Uso 3

Imágenes UI del Flujo:

1. Ya el usuario en la vista de detalles de un activo es capaz de ver el historial de este en la parte inferior. Sin embargo, para ver más a detalle y compartir los datos, decide descargar el archivo Excel.

Figura 4.22: Registro Histórico - Detalles de Activo

2. Una vez descargado, se puede apreciar que en un programa como Excel, se pueden ver más columnas indicando detalles del activo respecto a los eventos que han sido registrados.

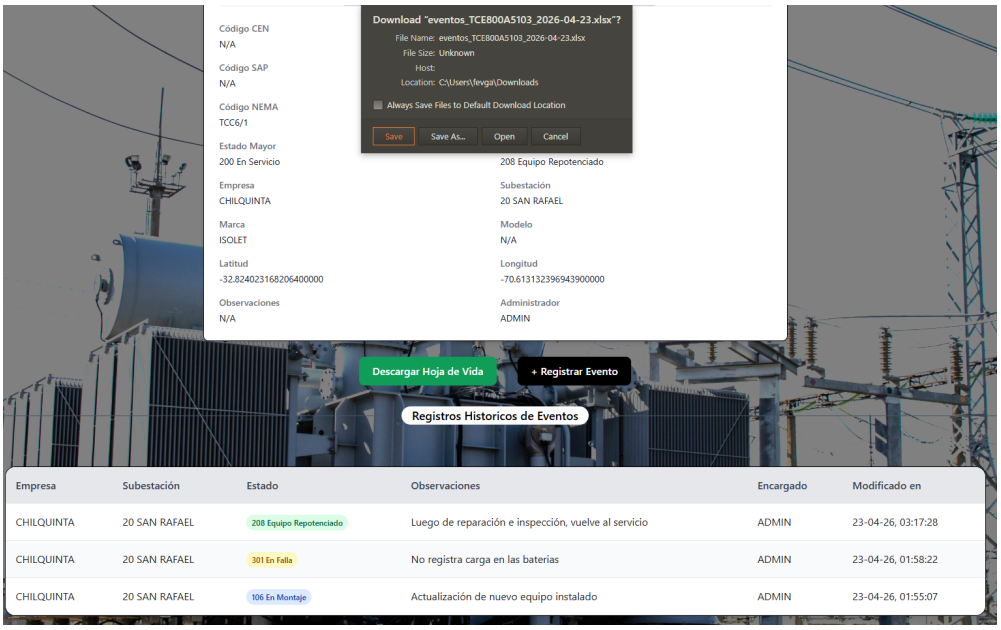


Figura 4.23: Descarga en Proceso - Hoja de Vida

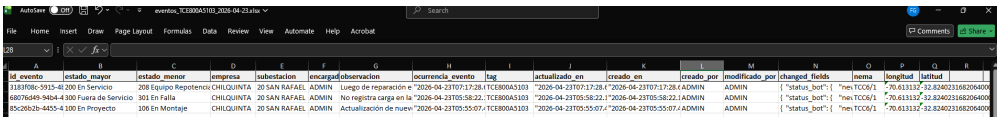


Figura 4.24: Visualización en Excel - Hoja de Vida

4.2.4. Localizar geográficamente un conjunto de activos

Descripción: Un usuario administrativo debe analizar la concentración de un tipo de activos a lo largo del país. Por lo tanto, debe informarse de cuántos hay en algunas subestaciones.

Actor: Usuario Administrativo

Precondiciones: El usuario recientemente ha ingresado al sistema con sus credenciales.

Flujo Principal:

1. Desde la página de inicio, el usuario se pasa al buscador geográfico llamado Mapa en la barra superior de navegación.

2. Selecciona el Buscar Por del buscador con TAG, e ingresa el código del activo que desea investigar.

3. Adicionalmente, a través del botón Mostrar Filtros Adicionales, agrega el estado de interés.

4. El mapa se actualiza, mostrando todo activo y su geolocalización.

Postcondiciones: El usuario pudo recopilar el número de activos presentes en múltiples lugares geográficos de un tipo en particular.

Diagrama:

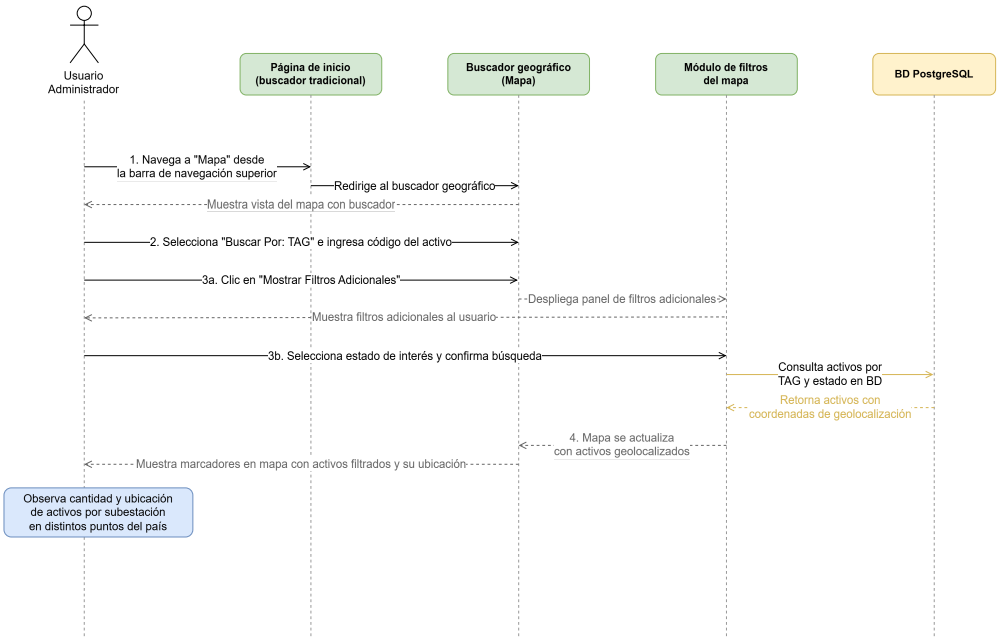


Figura 4.25: Diagrama de Flujo Caso de Uso 4

Imágenes UI del Flujo:

1. Usuario ubica el buscador geográfico a través de la barra de navegación en el botón Mapa. Lo cual lo lleva a las siguientes vistas.

Figura 4.26: Barra de Navegación

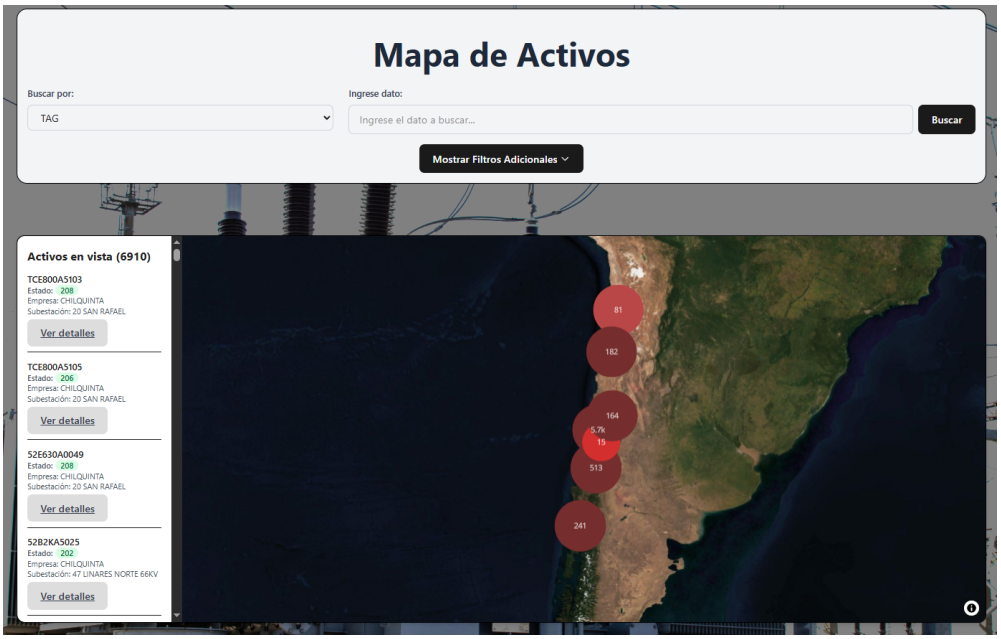


Figura 4.27: Buscador Geográfico de Activos

2. Inicialmente, el usuario puede ver el total de los activos que están presentes en el sistema, en el menú lateral del mapa. Para buscar solo los de interés, primero utiliza el buscador con TAG, y logra obtener el total de una familia de activos.

Figura 4.28: Busqueda por TAG

3. Luego se ayuda con los filtros adicionales para poder obtener el número de esos activos en servicio. Finalmente, está interesado en la región de Valparaíso, por lo que a través del mapa se acerca a la región para ver el número de activos en esa localidad.

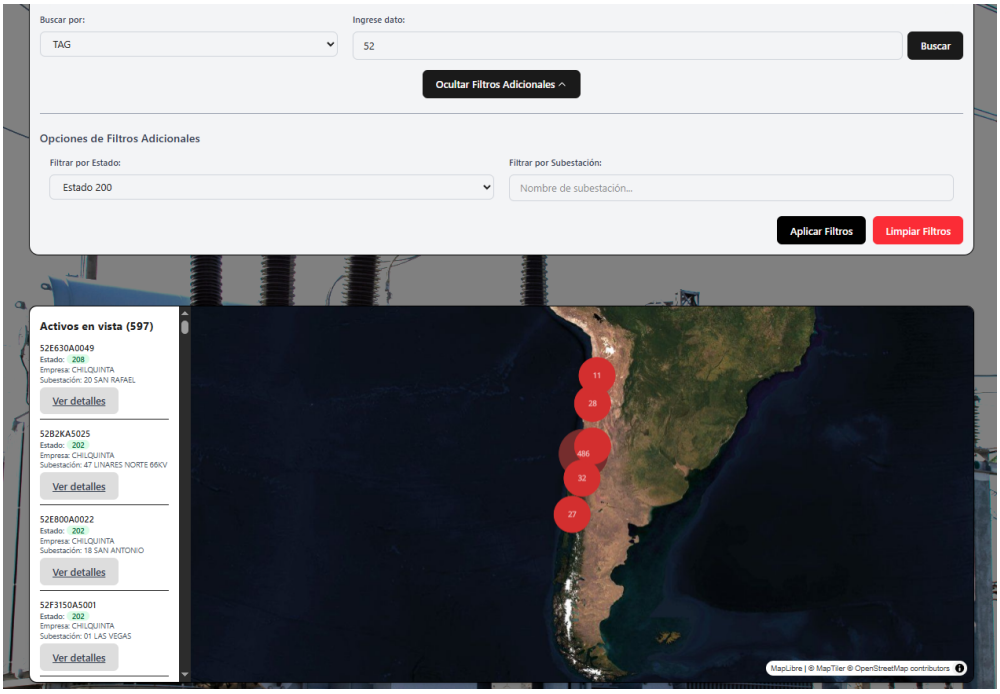


Figura 4.29: Uso de Filtros Adicionales

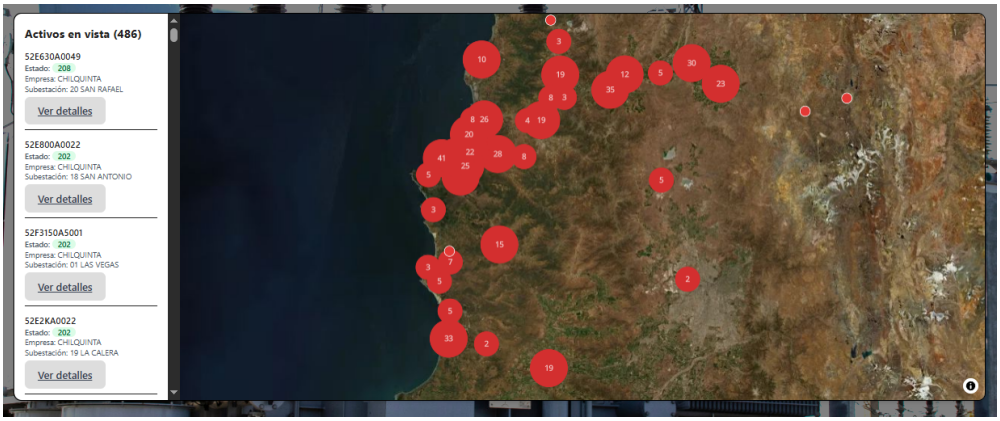


Figura 4.30: Uso de Mapa para Búsqueda

4.2.5. Búsquedas con filtros jerárquicos para encontrar múltiples activos a la vez

Descripción: El usuario administrador o técnico utiliza el buscador jerárquico para buscar los activos de una familia y de dos subestaciones a la vez, para buscar un posible reemplazo en base al estado de los activos.

Actor: Usuario administrador o técnico

Precondiciones: El usuario ha ingresado recientemente al sistema con sus credenciales. **Flujo Principal**:

1. Usuario accede al buscador jerárquico a través del botón Vista Jerárquica en la barra de navegación.

2. Usuario se encuentra en el buscador jerárquico, donde puede ver el orden de búsqueda actual de los activos y procede a cambiarlo con el botón de Editar.

3. Remueve los filtros actuales y arrastra los deseados a la parte superior, primero por subestación y luego por familia de equipos.

4. La barra lateral de búsqueda se actualiza con el nuevo orden de filtrado.

5. Usuario despliega las subestaciones que desea para poder ver la cantidad de activos por familia que hay en cada una.

6. Luego modifica nuevamente los filtros y agrega los de estado mayor y menor.

7. Vuelve al buscador actualizado y visualiza los detalles de los activos para escoger un activo de reemplazo en el panel central de la vista.

Postcondiciones: El usuario tiene acceso a datos de múltiples activos a la vez, los cuales comparten características. **Diagrama de flujo**:

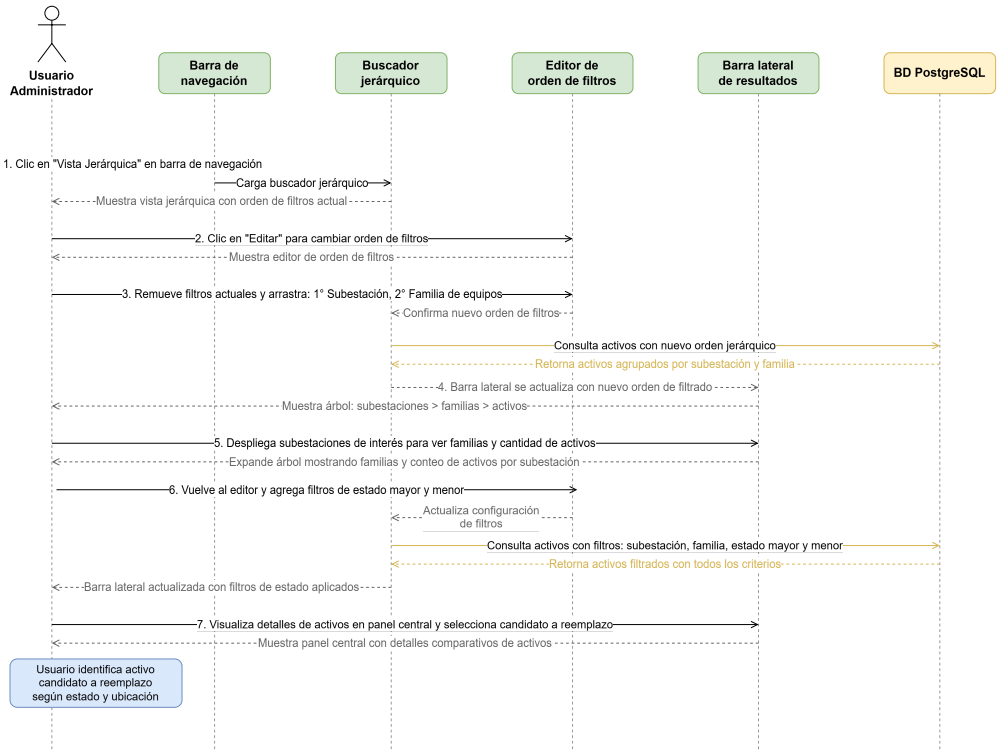


Figura 4.31: Diagrama de Flujo Caso de Uso 5

Imágenes UI del Flujo:

1. El usuario, apenas ingresa al buscador jerárquico, se encuentra con los activos filtrados por un orden por defecto. Por lo cual editar el buscador a traves del boton superior en el panel lateral.

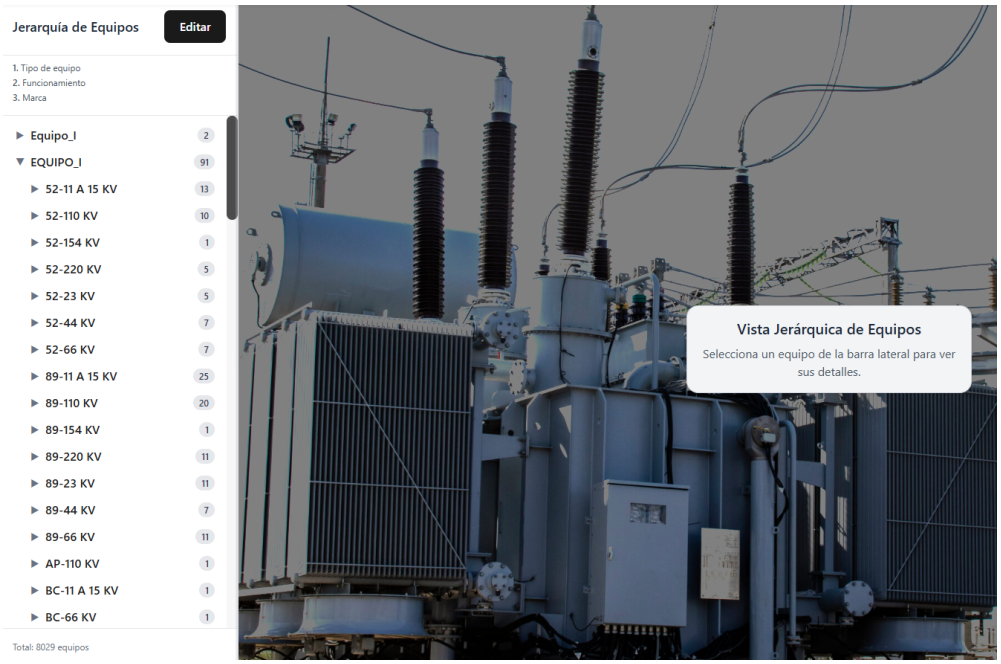


Figura 4.32: Buscador Jerárquico por Defecto

2. El editor del buscador le permite reordenar los activos con las opciones que se encuentran en este. Pasa de las opciones por defecto a subestacion y familia de equipo.

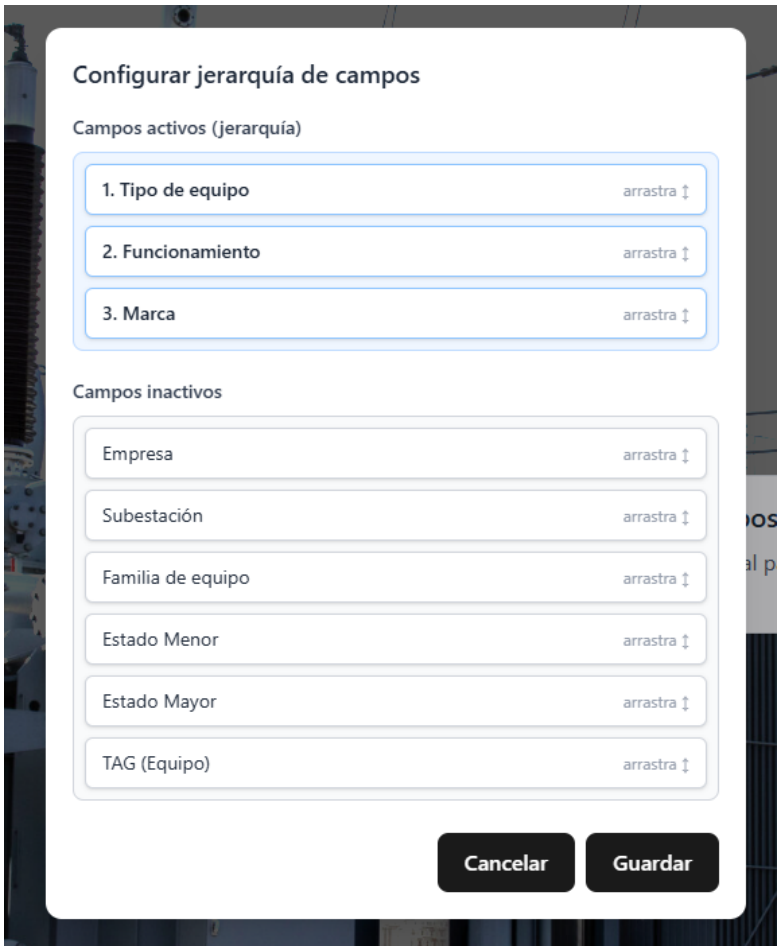


Figura 4.33: Editor Buscador Jerárquico por Defecto

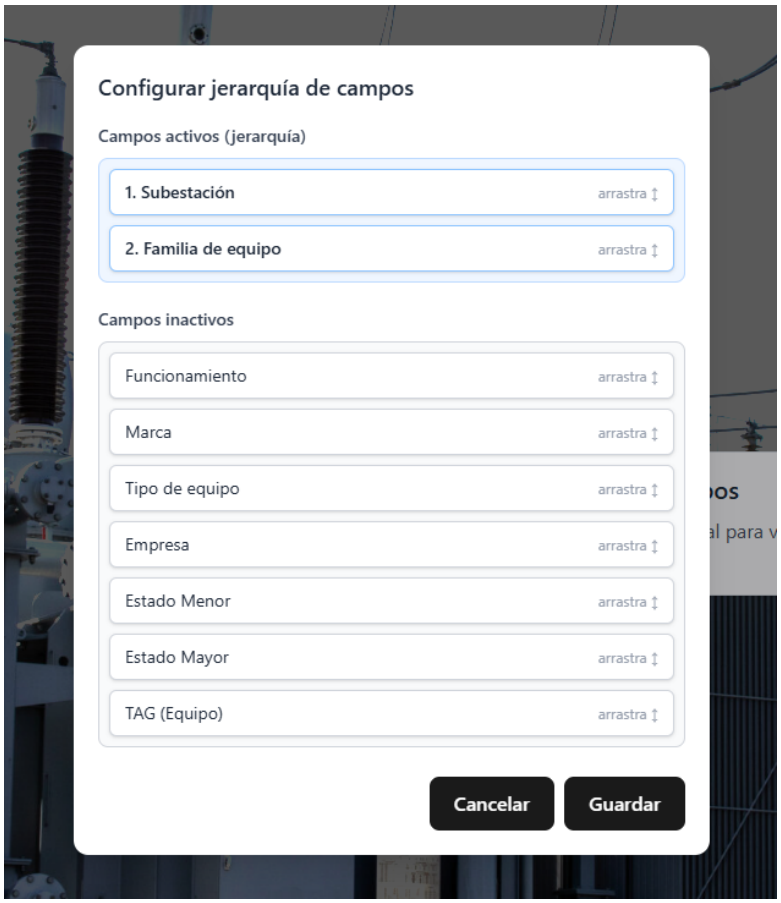


Figura 4.34: Editor Buscador Jerárquico Modificado

3. Una vez hechos los cambios y guardados. El panel lateral del buscador se ve actualizado con su nuevo orden y se muestra la navegación de este.

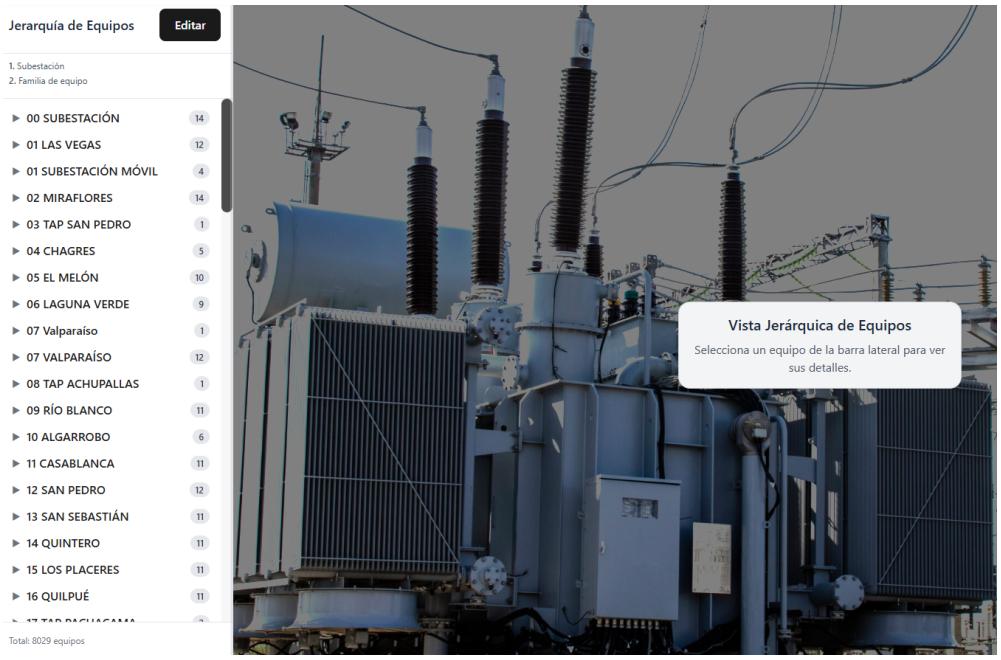


Figura 4.35: Navegación Resultados Buscador Jerárquico - 1

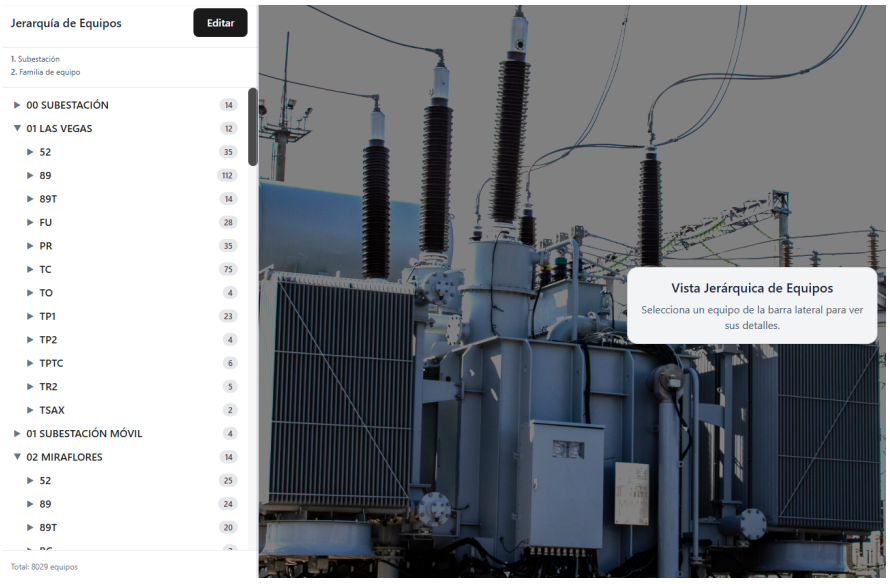


Figura 4.36: Navegación Resultados Buscador Jerárquico - 2

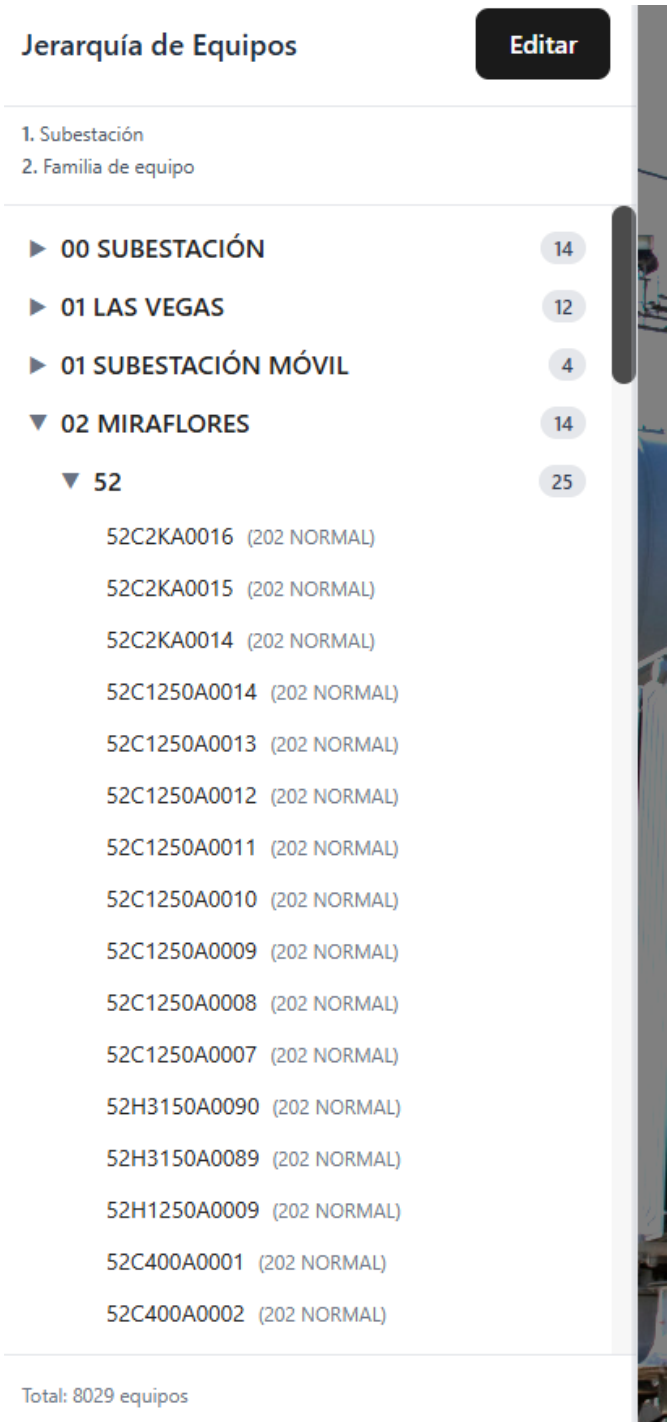


Figura 4.37: Navegación Resultados Buscador Jerárquico - 3

4. El usuario, para precisar más su búsqueda, añade dos filtros más al buscador y llega a un filtrado adecuado, el cual le permite ver el detalle de los activos en el

panel central de cada activo del filtrado.

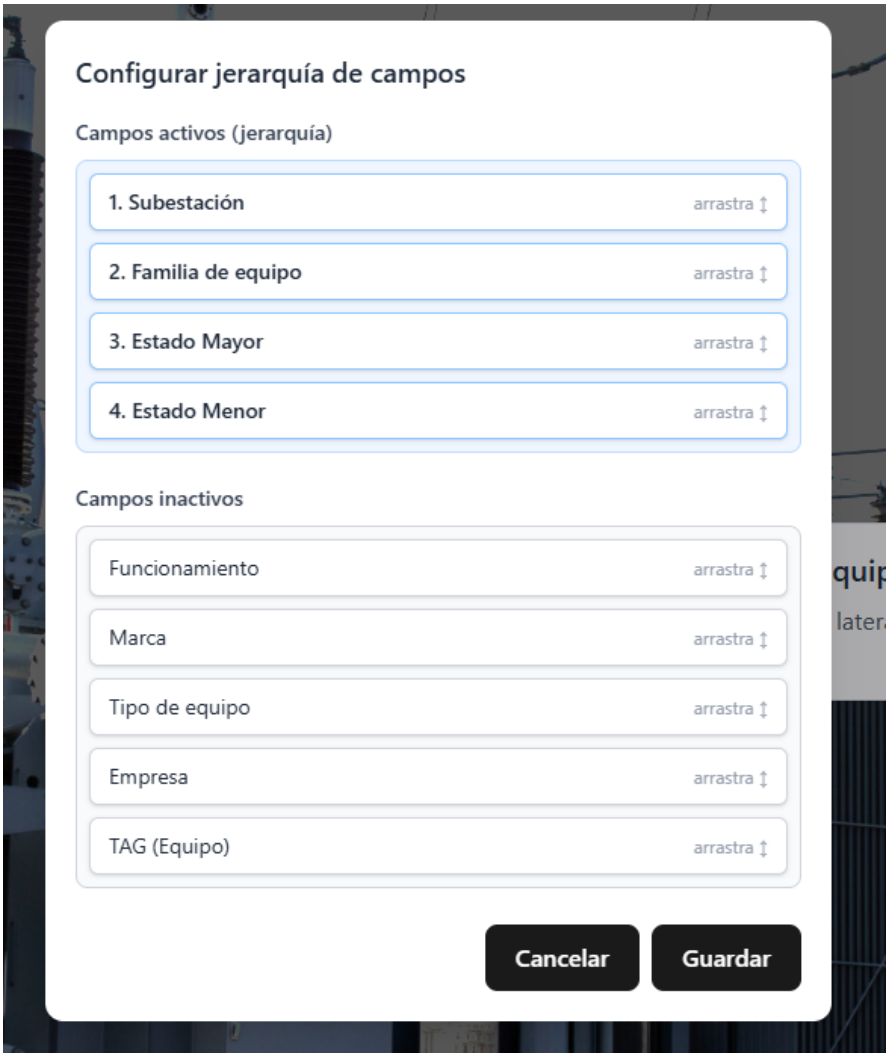


Figura 4.38: Navegación Resultados Buscador Jerárquico - 4

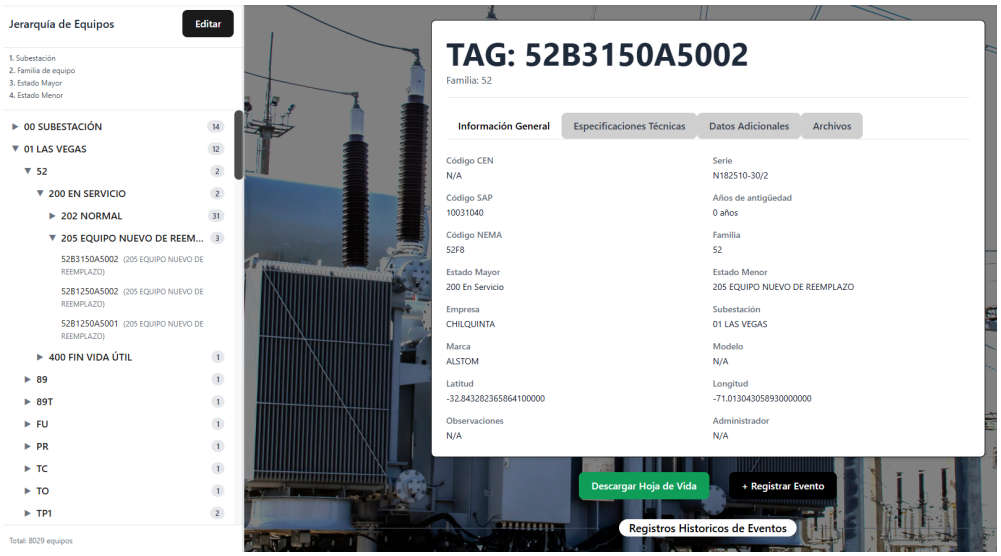


Figura 4.39: Vista Detalles de Activos dentro de Buscador Jerárquico

4.2.6. Búsqueda y exportación de activos fuera de servicio

Descripción: El usuario administrativo busca la información de todos los activos fuera de servicio y requiere exportarlos a un archivo Excel para una reunión.

Actor: Usuario administrativo

Precondiciones: El usuario ha ingresado al sistema recientemente con sus credenciales. **Flujo Principal**:

1. El usuario se encuentra en el buscador tradicional, donde procede a abrir los filtros adicionales.

2. Escoge el estado mayor y el estado menor adecuados para encontrar los activos fuera de servicio.

3. Exporta los activos con los filtros aplicados a través del botón de Generar Excel, el cual procede a descargar el archivo.

Postcondiciones: El usuario está en posesión de un archivo Excel con la última información de cada activo filtrado por el sistema. **Diagrama de flujo**:

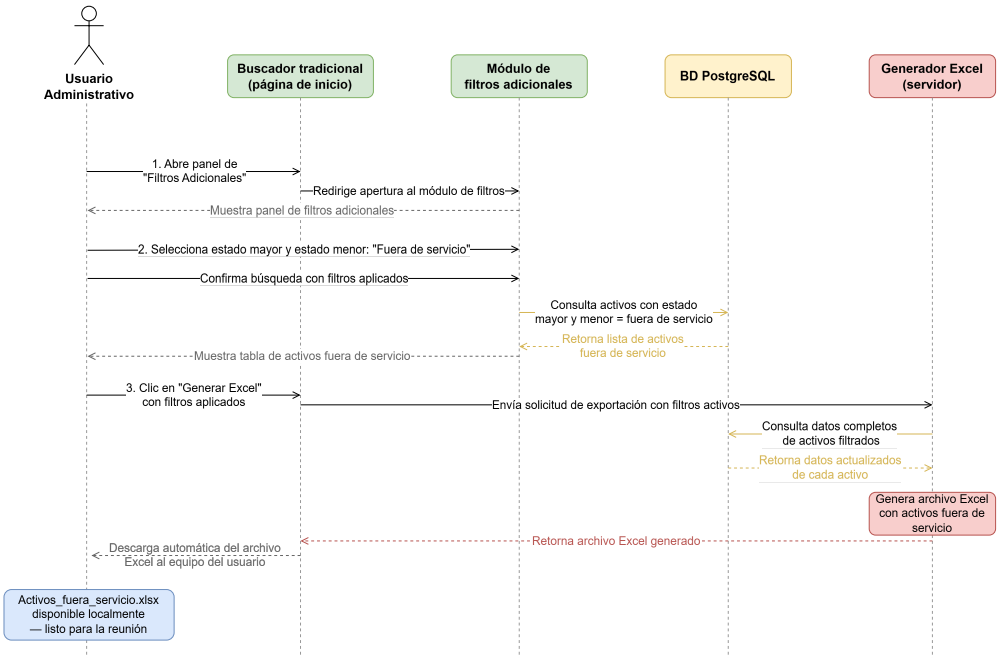


Figura 4.40: Diagrama de Flujo Caso de Uso 6

Imágenes UI del Flujo:

1. En el buscador tradicional, con el panel de filtros adicionales abierto, el usuario se asegura de modificar los estados (mayor y menor) de tal manera de filtrar por los activos deseados. Luego exporta con el boton de Generar Excel.

Figura 4.41: Exportación de Activos Filtrados

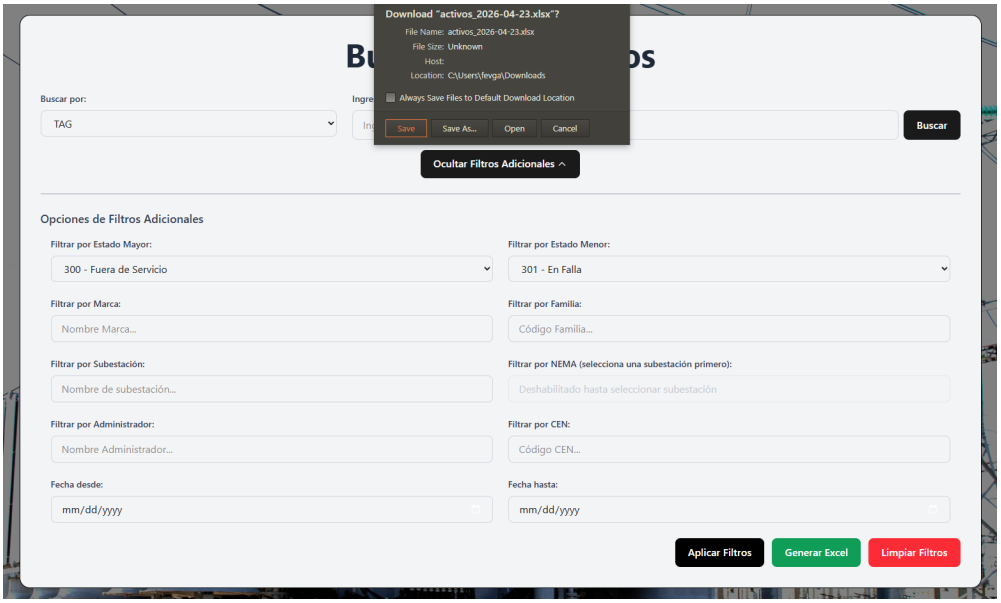
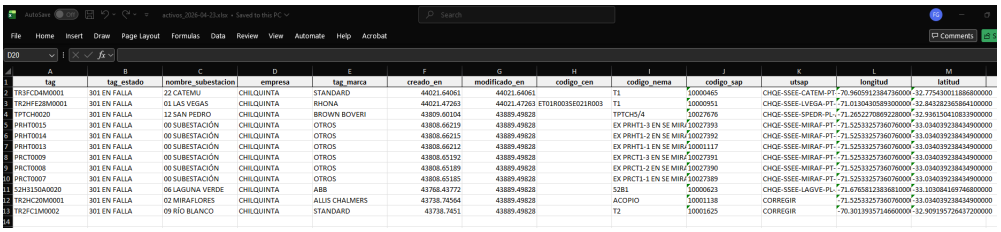


Figura 4.42: Descarga de Archivo con Activos Filtrados

2. Una vez ya descargado el archivo, el usuario puede abrirlo en un programa como Excel para ver todos los activos filtrados con sus últimos datos registrados para cada uno.



tag	tag_estado	nombre_subestacion	empresa	tag_marca	creado_en	modificado_en	codigo_cen	codigo_nema	codigo_sap	utsap	longitud	latitud
TR3FCD4M0001	301 EN FALLA	22 CATEMU	CHILQUINTA	STANDARD	44021.64061	44021.64061		T1	10000465	CHQE-SSEE-CATEM-PT	-70.96059123847360000	-32.77543001188680000
TR2HFE28M0001	301 EN FALLA	01 LAS VEGAS	CHILQUINTA	RHONA	44021.47263	44021.47263	ET01R003SE021R003	T1	10000951	CHQE-SSEE-LVEGA-PT	-71.01304305893500000	-32.84328236586410000
TPTCH0020	301 EN FALLA	12 SAN PEDRO	CHILQUINTA	BROWN BOVERI	43809.60104	43889.49828		TPTCH5/4	10027676	CHQE-SSEE-SPEDR-PL	-71.26522708692280000	-32.93615041083390000
PRHT0015	301 EN FALLA	00 SUBESTACIÓN	CHILQUINTA	OTROS	43808.66219	43889.49828		EX PRHT1-3 EN SE MIRA	10027393	CHQE-SSEE-MIRAF-PT	-71.52533257360760000	-33.03403923843490000
PRHT0014	301 EN FALLA	00 SUBESTACIÓN	CHILQUINTA	OTROS	43808.66215	43889.49828		EX PRHT1-2 EN SE MIRA	10027392	CHQE-SSEE-MIRAF-PT	-71.52533257360760000	-33.03403923843490000
PRHT0013	301 EN FALLA	00 SUBESTACIÓN	CHILQUINTA	OTROS	43808.66212	43889.49828		EX PRHT1-1 EN SE MIRA	10001117	CHQE-SSEE-MIRAF-PT	-71.52533257360760000	-33.03403923843490000
PRCT0009	301 EN FALLA	00 SUBESTACIÓN	CHILQUINTA	OTROS	43808.65192	43889.49828		EX PRCT1-3 EN SE MIRA	10027391	CHQE-SSEE-MIRAF-PT	-71.52533257360760000	-33.03403923843490000
PRCT0008	301 EN FALLA	00 SUBESTACIÓN	CHILQUINTA	OTROS	43808.65189	43889.49828		EX PRCT1-2 EN SE MIRA	10027390	CHQE-SSEE-MIRAF-PT	-71.52533257360760000	-33.03403923843490000
PRCT0007	301 EN FALLA	00 SUBESTACIÓN	CHILQUINTA	OTROS	43808.65185	43889.49828		EX PRCT1-1 EN SE MIRA	10027389	CHQE-SSEE-MIRAF-PT	-71.52533257360760000	-33.03403923843490000
52H3150A0020	301 EN FALLA	06 LAGUNA VERDE	CHILQUINTA	ABB	43768.43772	43889.49828		52B1	10000623	CHQE-SSEE-LAGVE-PL	-71.67658123836810000	-33.10308416974680000
TR2HC20M0001	301 EN FALLA	02 MIRAFLORES	CHILQUINTA	ALLIS CHALMERS	43738.74564	43889.49828		ACOPIO	10001138	CORREGIR	-71.52533257360760000	-33.03403923843490000
TR2FC1XM0002	301 EN FALLA	09 RÍO BLANCO	CHILQUINTA	STANDARD	43738.7451	43889.49828		T2	10001625	CORREGIR	-70.30139357146600000	-32.90919572643720000

Figura 4.43: Visualización de Archivo Excel

4.3. Validación

La validación del sistema propuesto se realizará mediante pruebas técnicas para confirmar la integridad del diseño lógico de la API con el resto de componentes como base de datos, migración de datos e interfaz de usuario.

Previo a los planes de validación a discutir, se deben tener algunas consideraciones respecto al proceso y cómo se llevó a cabo. Todo progreso de validación donde se requería contacto con el cliente para obtener retroalimentación sobre el trabajo, se pudo lograr gracias a reuniones bisemanales donde se mostraba progreso y/o se discutían tópicos del trabajo. Igualmente, se intentó llevar a cabo un proceso de encuestas hacia los potenciales usuarios dentro de la empresa, sin embargo, este formulario no se pudo llevar a cabo dentro de los márgenes de tiempo correspondientes.

Otro punto esencial del ambiente de desarrollo del prototipo, es que los componentes de interfaz de usuario, API y base de datos fueron levantados en un ambiente de nube a través del servicio de **Railway**, el cual permite despliegues eficientes a través de una integración con los repositorios de **GitHub**. El objetivo de este levantamiento fue optimizar el desarrollo en equipo, ya que todos tenían acceso a los componentes andando de manera simultánea, e igualmente tratar de brindar un acercamiento a un ambiente de trabajo más realista y de acuerdo a los tiempos de hoy en día, como son todos los servicios que se levantan en la nube. Por ejemplo, toda prueba de análisis de rendimiento o símil fue ejecutada con los servicios desplegados en este ambiente.

A continuación, se detalla el plan de validación para cada hipótesis.

4.3.1. Validación del diseño de la base de datos (H2)

La validación de esta hipótesis fue de carácter técnico, donde se inspeccionan las capacidades de rendimientos, escalabilidad y completitud de la base de datos. La parte lograda se lleva a cabo por el equipo de desarrollo y se plantean pasos de validación futura.

Instrumentos de Evaluación

Se emplearon las siguientes herramientas para verificar el comportamiento de la base de datos:

1. **Inspecciones Manuales de Datos Almacenados:** Una parte importante del proceso fue revisar que cada columna de cada tabla contuviera los valores importantes de trazabilidad que fueron entregados por el cliente en sus formatos correspondientes.

2. **Explain Analyze con DBeaver:** Se utilizó comandos de **EXPLAIN ANALYZE** a través de DBeaver, de las consultas comunes que se estarían ejecutando regularmente para así ver posibles tiempos de espera y sus costos de ejecución.

Métricas de Evaluación

Con los instrumentos definidos, se consideran los siguientes indicadores:

- **Integridad de Datos:** Consiste en verificar que todos los datos estén en sus formatos correspondientes y que a través del proceso de ETL no se hayan transformado drásticamente hasta perder sus estados debidos.

- **Completitud y Precisión de la Trazabilidad:** A través de un proceso de pruebas de registros de cambios o eventos conocidos, se espera llevar un porcentaje de cuántos de estos se ven luego presentes en las hojas de vida con todo dato correspondiente.

- **Rendimiento de Consultas:** A través de la herramienta de *Explain Analyze* se pueden obtener tanto costos de ejecución como tiempo de las consultas, sin otro elemento intermediario como la API.

Resultados

- **Integridad de Datos:** En base a inspecciones realizadas, la cantidad de datos que mantiene su integridad se mantiene sobre el 90 %. Sin embargo, esto requiere de pruebas automatizadas con el fin de poder verificar que el 100 % de los datos se encuentran como corresponden luego del proceso de ETL.

- **Completitud y Precisión de la Trazabilidad:** Usando las consultas SQL de forma manual y a través de la interfaz, se confirma que el 100 % de las modificaciones generaron un registro de historial preciso y completo, ya sea a través de los *triggers* ejecutados al crear eventos o editando los datos dinámicos directamente.

- **Rendimiento de Consultas:** Mayoría de las consultas toman tiempos menores a los 4 segundos.

Trabajo Futuro

1. **Scripts de Pruebas Automatizadas Gran Escala:** Se estima necesario de scripts para simular la creación, actualización y eliminación masiva de activos, verificando el comportamiento de las restricciones y la robustez frente operaciones de alta escala.

2. **Consultas SQL de Verificación:** Con el fin de asegurar que los mecanismos historiales se mantengan robustos bajo alta carga, se propone la creación de scripts que prueben la durabilidad de la base datos bajo numerosas operaciones simultáneas a la vez.

4.3.2. Validación de la API (H3)

La validación de esta hipótesis fue de carácter técnico para ver principalmente la eficiencia y seguridad que brinda la API para todas las operaciones del sistema relevantes.

Instrumentos de Evaluación

Se emplearon las siguientes herramientas o métodos para verificar el comportamiento de la base de datos:

1. **Postman:** La herramienta principal para ir verificando que las rutas estén bajo funcionamiento, permitiendo verificar elementos como autenticación, velocidad, tamaño y formato de las respuestas.

2. **Comparación OpenAPI:** El documento de diseño de la API contiene los mecanismos de funcionamiento de esta incluyendo parámetros, respuestas, cuerpos, y schemas. Por lo tanto, es esencial ir comparando el funcionamiento actual con la documentación de este para asegurarse de un trabajo correctamente documentado.

3. **Pruebas con Interfaz de Usuario:** El otro método de pruebas para asegurarse de que la API esté cumpliendo con lo que se requiere es a través de la API misma.

Métricas de Evaluación

Con los instrumentos definidos, se midieron los siguientes indicadores:

1. **Seguridad:** En cuanto la seguridad se consideran elementos como la autenticación y autorización de ciertas rutas. Adicionalmente, se considera que no se sobre expongan datos bajo ninguna circunstancia.

2. **Rapidez Respuestas:** Este consiste en el tiempo de respuestas que toma cada respuesta, los cuales fueron analizados a través de Postman.

Resultados

1. **Seguridad:** Cada ruta a excepción de la de inicio de sesión se encuentra asegurada con autenticación, y en casos relevantes donde existen operaciones solo reservadas para los administradores se encuentran mecanismos de autorización.

2. **Rapidez Respuestas:** Se tomaron los tiempos de cada respuesta de las rutas y la gran mayoría se encuentra en el rango de 0.5 a 5 segundos, para la problemática en cuestión, estos tiempos de respuesta son más que ideales. Los casos más altos corresponden a la exportación de datos en archivos Excel, donde se toman todos los datos de los activos, sin incluir documentos o eventos, esta función toma alrededor de 11 segundos. Aun así, termina siendo mucho más eficiente en cuanto a tiempo en comparación con el sistema existente del caso de uso, el cual tomó sobre 1 hora para entregarnos los mismos datos.

Trabajo Futuro

Algunos de los puntos a reforzar de la validación son los siguientes:

- **Escalabilidad:** A pesar de tener una API en ambiente desplegado en nube y con cada una de sus rutas con pruebas realizadas, no es suficiente para asegurar una escalabilidad a números más altos de usuarios. Acá es donde sería esencial establecer scripts de pruebas para simular conexiones simultáneas de la cantidad de usuarios esperados, especialmente realizando operaciones las cuales podrían colisionar unas con otras. En este trabajo, la cantidad de usuarios existentes entregados por el cliente fue poca; no obstante, este número siempre puede ir en alza si la herramienta finalmente se estima eficiente para todos los tipos de usuarios y se impulsa el uso de esta.

4.3.3. Validación del proceso ETL (H4)

Al igual que en la H2, la validación fue realizada por el equipo de desarrollo. El objeto de estudio fue el script de importación desarrollado en Python con Pandas y SQLAlchemy.

Instrumentos de Evaluación

Para evaluar la robustez del proceso de migración se utilizaron:

1. **Consultas PostgreSQL:** Se utilizaron consultas para ir verificando que la cantidad de activos que se importaron calzara con los datos originales.
2. **Logs de Ejecución:** El script ETL fue diseñado para generar un log detallado que documentara el número de registros procesados, cargados con éxito y rechazados.
3. **Prueba de Re-ejecución:** Consistió en ejecutar el proceso de importación múltiples veces con el mismo archivo fuente para verificar su comportamiento ante datos ya existentes.

Métricas de Evaluación

El éxito del proceso ETL se cuantificó a través de las siguientes métricas:

- **Tasa de Éxito y Calidad de Datos:** A partir del log de ejecución y el script de validación, se midió el porcentaje de registros cargados exitosamente y el porcentaje de datos que cumplían con los formatos correctos.
- **Manejo de Errores:** Se analiza el log para confirmar que todos los registros con errores fueran identificados y omitidos sin detener el proceso.
- **Idempotencia:** Mediante la prueba de re-ejecución, se verifica que no se generen registros duplicados.

Resultados

- **Tasa de Éxito y Calidad de Datos:** El script de ETL fue ejecutado múltiples veces, y la cantidad de registros que se alcanzó a ingresar a la base de datos fue 8024 del total original que eran 8029. Esto es una tasa alta para la cantidad de registros que se manejan de alrededor del 99.9 %. Si el script se sigue utilizando a futuro, se espera que la cantidad de registros a migrar/importar reduzca con el tiempo y la cantidad de errores disminuya con el tiempo.

- **Idempotencia:** Al ejecutar el script de ETL en múltiples etapas del desarrollo, no solo se logró importar los datos en su completitud, sino que también se verificó que no se sobrescribiera ninguno de los registros que ya existen en la base de datos, así disminuyendo los posibles errores que pueden causar una sincronización o migración con el script.

Trabajo Futuro

Algunos de los puntos a reforzar de la validación son los siguientes:

- **Recursos Utilizados:** A pesar de que el script logró importar todos los datos sin problema, el tiempo promedio de ejecución fue de alrededor de 40 minutos, para la cantidad de datos y las circunstancias, esto resulta no problemático. Sin embargo, si se desea ejecutarlo a menudo, se deberían realizar tanto pruebas como revisiones para ver su rendimiento con una menor cantidad de datos.

- **Modificaciones al Script:** Con el fin de poder asegurar que todos los registros realmente sean ingresados sin problemas al correr el script una vez, se estima necesaria una revisión y posteriores cambios basados en esta.

Capítulo 5

Conclusiones y Trabajo a Futuro

Los resultados del trabajo de este escrito, los cuales mayormente estuvieron enfocados en el backend del sistema, tuvieron un grado alto de éxito en cuanto a los objetivos inicialmente establecidos. Se demuestra que cada uno de los componentes aporta a tener un sistema que ya sea capaz de reemplazar o complementar uno existente, como en el caso de esta problemática. Este caso específico de trazabilidad de activos de transmisión eléctrica tiene como posible solución incorporar un sistema como el presentado en este trabajo, lo cual resultaría en flujos de trabajo más fluidos y eficaces para quienes deseen utilizarlos. Cada componente se realizó con escalabilidad y replicabilidad en mente, por lo cual se podría adaptar a casos de trazabilidad de otros tipos de activos e ir evolucionando con el tiempo.

A pesar de ser un sistema conciso que cumple con los objetivos de manera precisa, este sistema tiene espacio para crecer. Hay múltiples ideas que se pueden seguir añadiendo o mejorando en esta solución como el tema de seguridad y accesibilidad, los cuales requieren de más feedback por parte de los futuros usuarios, el cual fue dificultoso obtener durante este proceso. El acceso a las funciones del sistema contiene un grado de autorización el cual funciona con RBAC, pero esta puede ser mejorada a ABAC, el cual toma más tiempo de implementar pero puede resultar en interfaces más dinámicas y seguras, tomando en consideración múltiples variables como roles, subestaciones o

compañías del usuario.

Otro ámbito que debe ser reforzado es el ingreso y borrado de activos, debido a que de momento funcionan de manera bastante directa con poca afinidad. Lo cual no es ideal para un sistema que busca reforzar la trazabilidad, se deben incorporar sistemas de *soft deletion*, lo cual permitiría quitar los activos o registros de la vista de los usuarios, pero los datos permanecerían respaldados en caso de cualquier necesidad futura como una auditoría.

El último punto a reforzar es la disponibilidad y persistencia de los datos, este fue un punto clave que se estableció como parte de esta solución pero durante el desarrollo no fue trabajado extensamente. Es necesario incorporar mecanismos que permitan tanto la carga como la descarga de datos en caso de conexiones poco estables al servicio, ya sea a través de cargas de datos previas que se realicen regularmente u otras medidas.

El sistema tiene áreas donde se puede reforzar pero, igualmente, es importante destacar hoy en día que hay conceptos para profundizar más en una herramienta como esta. Algunas ideas tales como un *Chat Box* el cual esté acompañado de algún modelo de IA para realmente dar una experiencia más de acuerdo a los tiempos, este podría resultar en una experiencia más transparente, donde los usuarios ya sabrán cómo utilizar la herramienta debido al extenso uso que hay hoy en día de herramientas de esta índole. Finalmente, lo ideal sería pasar la versión móvil de esta aplicación web a una aplicación móvil completa, para así también aprovechar estos dispositivos con todo su potencial. Un ejemplo de esto sería utilizar los componentes del móvil, como su GPS, para así brindar un posicionamiento hacia los activos en tiempo real y preciso para los usuarios. Estos tipos de sistemas tienen alto potencial y una alta presencia, la cual se espera que solo aumente debido a la alta cantidad de datos que hoy en día se monitorizan.

Bibliografía

[1] IBM. IBM Maximo Application Suite. https://www.ibm.com/products/maximo, 2025.

[2] AVEVA. Aveva asset information management. https://www.aveva.com/en/products/asset-information-management/, 2025.

[3] HITACHI. Lumada asset perfomance management. https://www.hitachienergy.com/products-and-solutions/asset-and-work-management/lumada-apm, 2025.

[4] IFS. Ifs cloud enterprise asset management. https://www.ifs.com/solutions/ifs-cloud-eam, 2025.

[5] SAP. What is sap? https://www.sap.com/about/what-is-sap.html, 2025.

[6] Elizaveta Gavrikova, Irina Volkova, and Yegor Burda. Strategic aspects of asset management: An overview of current research. *Sustainability*, 12(15), 2020.

[7] Ministerio del Medio Ambiente. Ley marco 20.920 para la gestión de residuos. https://economiacircular.mma.gob.cl/ley-rep/, 2025.

[8] S. R. Khuntia, José L. Rueda, S. Bouwman, and M. A. M. M. van der Meijden. Classification, domains and risk assessment in asset management: A literature study. In *2015 50th International Universities Power Engineering Conference (UPEC)*, pages 1–5, 2015.

[9] A. S. Habibie, A. P. Purnomoadi, B. B. S. D. A. Harsono, N. W. Priambodo, A. S. Surya, H. B. Tambunan, and K. M. Tofani. A journey of asset management in java-bali transmission system. In *2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, pages 27–30, 2020.

[10] OpenJS Foundation. Node.js. `https://nodejs.org/en`, 2025.

[11] OpenJS Foundation. Express. `https://expressjs.com`, 2025.

[12] Brian Carlson. node-postgress. `https://node-postgres.com`, 2025.

[13] Open Source Collective. Sequelize. `https://sequelize.org`, 2025.

[14] Prisma Data, Inc. Prisma. `https://www.prisma.io`, 2025.

[15] Django Software Foundation. django. `https://www.djangoproject.com`, 2025.

[16] VMware Tanzu. Spring Boot 4.0. `https://spring.io/projects/spring-boot`, 2025.

[17] Autho0. jsonwebtoken. `https://www.npmjs.com/package/jsonwebtoken`, 2023.

[18] kelektiv. node.bcrypt.js. `https://www.npmjs.com/package/bcrypt`, 2025.

[19] Smartbear. openAPI Specification. `https://swagger.io/specification/`, 2025.

[20] Smartbear. Swagger UI. `https://swagger.io/tools/swagger-ui/`, 2025.

[21] Postman, Inc. Postman. `https://www.postman.com`, 2025.

[22] Python Software Foundation. Welcome to Python.org. https://www.python.org/, 2025.

[23] NumFOCUS, Inc. pandas - Python Data Analysis Library. https://pandas.pydata.org/, 2025.

[24] SQLAlchemy authors and contributors. Features - SQLAlchemy. https://www.sqlalchemy.org/features.html, 2025.

[25] Daniele Varrazzo and The Psycopg Team. Using COPY TO and COPY FROM. https://www.psycopg.org/psycopg3/docs/basic/copy.html, 2025.

[26] Oracle. The Main Features of MySQL. https://dev.mysql.com/doc/refman/8.4/en/features.html, 2025.

[27] Microsoft. Editions and supported features of SQL Server 2022. https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022?view=sql-server-ver17, 2025.

[28] MongoDB, Inc. MongoDB Features. https://www.mongodb.com/resources/products/capabilities/features, 2025.

[29] The PostgreSQL Global Development Group. PostgreSQL: About. https://www.postgresql.org/about/, 2025.