



UNIVERSIDAD TÉCNICA  
FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA  
INGENIERÍA EN INFORMÁTICA

# Geocuadrilla: Integración de servicios para plataforma de monitoreo en tiempo real en Chilquinta Transmisión

Roberto Romero Vergara

Roberto.romerov@sansano.usm.cl

Óscar Carrasco Vera  
Profesor Guía

Hernán Saavedra González  
Profesor Correferente

## Resumen:

El trabajo de título se enmarca en el proyecto Geocuadrilla, desarrollado junto a Chilquinta Transmisión, con el propósito de mejorar la coordinación y supervisión de cuadrillas en terreno. Actualmente, la empresa no cuenta con un sistema que permita conocer la ubicación y estado operativo de las cuadrillas en tiempo real, lo que genera demoras en la respuesta ante incidentes eléctricos y aumenta los costos operativos.

La propuesta consiste en el diseño e implementación de un sistema de integración que conecte la aplicación móvil utilizada por las cuadrillas con el panel web del centro de control, permitiendo el intercambio de datos en tiempo real.

El sistema fue implementado y validado bajo la metodología ágil Scrum en un ciclo de cuatro *Sprints*. La validación funcional confirmó el correcto procesamiento y difusión de actualizaciones de ubicación con una latencia inferior a 1 segundo, y la arquitectura basada en mensajería asíncrona soportó una carga de 30.000 mensajes sin registrar errores, con un *throughput* de 655 mensajes por segundo. El sistema fue presentado ante Chilquinta Transmisión y exhibido en la Feria de Software de la Universidad Técnica Federico Santa María.

**Palabras Clave:** integración de sistemas, mensajería asíncrona, API REST, *WebSocket*, Monitoreo en tiempo real, datos geoespaciales.



## 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:** Geocuadrilla: Integración de servicios para plataforma de monitoreo en tiempo real en Chilquinta Transmisión

**Nombre del candidato(a):** Roberto Patricio Romero Vergara

**Carrera / Grado:** Ingeniería Informática

**Campus:** Viña del Mar **Departamento:** Electrotecnia e informática

### 2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Oscar Carrasco Vera, 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 (marcar una opción)

☒ 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 (**marcar una opción**):

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

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

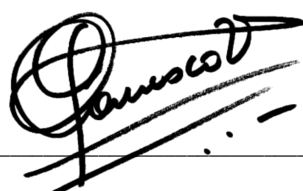
---

---

---

### 4.- FIRMAS

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

**Fecha:** 11-03-2026 **Firma:** 

**Estudiante o Candidato(a):**

**Fecha:** 11 de Marzo del 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.*

# 1 Introducción

## 1.1 Contexto y antecedentes.

En las operaciones de transmisión eléctrica, la coordinación eficiente de cuadrillas en terreno es fundamental para mantener la continuidad del servicio y responder oportunamente ante fallas o emergencias. Chilquinta Transmisión actualmente enfrenta una limitación crítica: no dispone de un sistema que permita conocer en tiempo real la ubicación y el estado operativo de sus cuadrillas. Esta falta de visibilidad dificulta la asignación de tareas, incrementa los tiempos de respuesta ante incidentes y genera costos adicionales asociados a la movilización de recursos.

Las cuadrillas utilizan teléfonos corporativos para registrar actividades y manejar su agenda, sin embargo, la información geográfica generada por estos dispositivos no es integrada de manera centralizada. Esto impide contar con una trazabilidad histórica, análisis posterior de eventos y una supervisión operativa continua, elementos esenciales para una gestión de operaciones de campo efectiva.

Chilquinta Transmisión actualmente gestiona la agenda de sus cuadrillas mediante un sistema propietario pero que no permite un monitoreo en tiempo real de la ubicación de estas.

## 1.2 Planteamiento del problema.

La ausencia de un sistema centralizado de monitoreo en tiempo real provoca:

- Demoras en la supervisión y asignación de cuadrillas.
- Falta de trazabilidad de las actividades.
- Dificultades para identificar la cuadrilla más cercana ante un incidente.

Formulamos el problema central de la siguiente manera: *Chilquinta Transmisión carece de un sistema que integre, procese y distribuya en tiempo real la información operativa generada por sus cuadrillas en terreno, dificultando la toma de decisiones y reduciendo la eficiencia de sus operaciones.*

### 1.3 Propuesta de solución

La propuesta general del proyecto Geocuadrilla consiste en el desarrollo de una plataforma que permita a Chilquinta Transmisión visualizar y supervisar en tiempo real las operaciones de sus cuadrillas en terreno. Con este propósito, el sistema integra diversos componentes tecnológicos que centralizarán la información operativa y geográfica relevante para el centro de control.

La plataforma contempla una aplicación móvil, utilizada por las cuadrillas en sus teléfonos corporativos para reportar periódicamente su ubicación, así como también un panel web destinado al centro de control, cuyo objetivo es mostrar un mapa en tiempo real con las posiciones de las cuadrillas en terreno. Este panel incorpora además la visualización de archivos *KMZ*, proporcionados por Chilquinta Transmisión, que contienen información geográfica de las líneas y torres de alta tensión.

Dentro de esta arquitectura, el sistema de integración constituye el núcleo del flujo de información y corresponde al alcance del presente trabajo de título. Este componente es responsable de:

- Recibir y procesar los datos enviados por la aplicación móvil.
- Almacenar estos datos para futuras auditorías.
- Permitir la gestión de archivos *KMZ* en el sistema.
- Entregar la información procesada al panel web en tiempo real.
- Integrar la agenda ya existente de las cuadrillas con el resto del sistema.

La solución propuesta permite dotar al centro de control de una herramienta integral para la supervisión operativa, favoreciendo la toma de decisiones informada, la reducción de tiempos de respuesta ante incidentes y la optimización de la planificación de las cuadrillas en terreno.

### 1.4 Alcance del proyecto y aporte del autor

El proyecto Geocuadrilla involucra el desarrollo conjunto de múltiples componentes: la aplicación móvil utilizada por las cuadrillas, el sistema de integración y el panel web del centro de control.

En este contexto, el presente trabajo de título se enfoca exclusivamente en el diseño, implementación y despliegue del sistema de integración encargado de habilitar el flujo

de datos entre la aplicación móvil, el *bróker* de mensajería y el panel web. Este componente incluye tanto la construcción del *backend* como la puesta en marcha de la infraestructura necesaria para su operación, considerando servicios como *RabbitMQ* para la recepción de datos desde la aplicación, PostgreSQL para el almacenamiento persistente y Minio para la administración de objetos tales como archivos KMZ.

El aporte del autor comprende:

- La implementación del procesamiento de los datos enviados por la aplicación móvil.
- La integración de la agenda operativa de cuadrillas entregada por Chilquinta, permitiendo asociar personal, vehículo y trabajo asignado a cada posición.
- La implementación de un sistema CRUD para la gestión de archivos KMZ utilizados por el panel web.
- El diseño y despliegue de la infraestructura base (RabbitMQ, PostgreSQL, Minio).
- La provisión de servicios que permiten entregar información operativa en tiempo real al panel web.
- El mantenimiento de registros históricos y consistentes para auditoría y análisis operativo.

### **1.5 Limitaciones del trabajo**

El alcance de este trabajo no incluye:

- El desarrollo de la aplicación móvil utilizada por las cuadrillas.
- La implementación del panel web de monitoreo.
- La integración directa o desarrollo con la API de Google Maps.
- La visualización geográfica de líneas, torres o posiciones.
- Algoritmos de optimización de rutas

### **1.6 Objetivo general**

Diseñar, implementar y desplegar, en un periodo de 15 semanas, un sistema de integración capaz de recibir, procesar, almacenar y entregar en tiempo real la

información operativa proveniente de las cuadrillas, asegurando una latencia de actualización menor a 2 segundos.

### 1.7 Objetivos específicos

1. Analizar los requerimientos operativos del centro de control y de las cuadrillas durante la primera semana del proyecto, generando la documentación inicial.
2. Diseñar la arquitectura del sistema de integración en un plazo máximo de dos semanas definiendo modelos de datos, flujos de información y mecanismos de comunicación documentados.
3. Configurar y poner en funcionamiento la infraestructura requerida (RabbitMQ, PostgreSQL y MinIO), asegurando su correcto funcionamiento mediante pruebas funcionales del sistema integrado.
4. Desarrollar e implementar, en un plazo de 3 semanas, los servicios responsables de recibir, procesar y almacenar en PostgreSQL los datos enviados por la aplicación móvil, verificando su funcionamiento mediante pruebas funcionales y de carga.
5. Implementar, en un tiempo de 1 semana, la importación y normalización de la agenda operativa entregada por Chilquinta, permitiendo asociar el personal, vehículos y trabajos de cada ubicación, validando el proceso con una prueba de campo.
6. Realizar durante un periodo de 3 semanas pruebas de funcionamiento y rendimiento que verifiquen el correcto procesamiento de datos y la persistencia de los datos.

### 1.8 Metodología utilizada

El desarrollo del presente trabajo se realizó utilizando la metodología ágil *Scrum*, adaptada a un ciclo de quince semanas de trabajo distribuidas en cuatro *Sprints*. Este enfoque permitió organizar el proyecto en iteraciones con objetivos definidos y entregables verificables al término de cada ciclo, facilitando la retroalimentación temprana y el ajuste progresivo de la solución.

- *Sprint 1* se centró en el levantamiento de requerimientos, análisis del dominio operativo y diseño preliminar de la arquitectura del sistema de integración.

- *Sprint 2* abordó la implementación de la recepción y procesamiento de datos desde la aplicación móvil, junto con el inicio del trabajo en el almacenamiento persistente. Al término de este *Sprint* se realizó una reunión de revisión con el patrocinador Chilquinta Transmisión, instancia en la cual se evaluaron los avances y se validó la línea de desarrollo. Esta retroalimentación permitió ajustar prioridades y refinar los flujos de integración y procesamiento de datos.
- *Sprint 3* se enfocó en la integración de la agenda operativa de cuadrillas, la incorporación del sistema CRUD para la gestión de archivos KMZ y la estabilización de la infraestructura base. Nuevamente, al cierre del *Sprint* se llevó a cabo una revisión formal con Chilquinta Transmisión, donde se presentaron los avances funcionales y se recibieron observaciones que derivaron en un ajuste de curso en la priorización y funcionamiento del sistema.
- *Sprint 4* estuvo destinado a la validación del sistema, ejecución de pruebas funcionales y pruebas de consistencia. Durante este *Sprint* se consolidó la operación del sistema en un entorno real y culminó en la presentación del sistema durante la Feria de Software.

Este proceso iterativo permitió mantener la trazabilidad del avance y asegurar que el sistema evolucionara de manera alineada con las necesidades operativas del patrocinador.

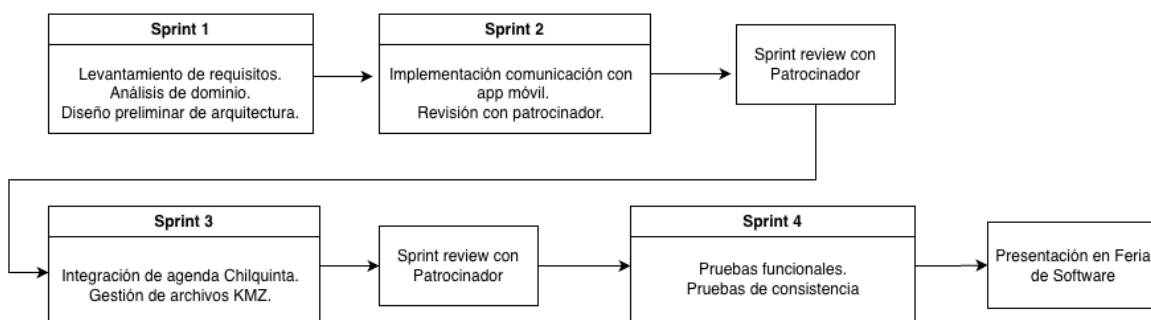


Figura 1. Diagrama del ciclo de desarrollo Scrum aplicado al proyecto Geocuadrilla

Fuente: Elaboración propia

## **1.9 Estructura del documento**

El presente documento se organiza en seis capítulos.

En el Capítulo 1 se introduce el contexto del problema, se plantea la motivación del proyecto, se presenta la propuesta de solución, el alcance del trabajo, los objetivos y la metodología utilizada.

El Capítulo 2 desarrolla el marco teórico, abordando los conceptos, tecnologías y enfoques relevantes para la comprensión del sistema de integración propuesto.

En el Capítulo 3 se describe la metodología de desarrollo y el diseño del sistema, detallando la arquitectura general, los componentes principales y los flujos de información que conforman la solución.

El Capítulo 4 presenta la implementación del sistema de integración, incluyendo la descripción de los módulos desarrollados, la infraestructura desplegada y los mecanismos utilizados para el procesamiento y almacenamiento de datos.

El Capítulo 5 expone las pruebas realizadas y la validación del sistema, analizando los resultados obtenidos en función de los objetivos definidos y evaluando el desempeño de la solución en un entorno real.

Finalmente, el Capítulo 6 presenta las conclusiones del trabajo, junto con las principales lecciones aprendidas, limitaciones identificadas y posibles líneas de trabajo futuro.

## **2 Marco teórico**

En este capítulo se presentan los fundamentos conceptuales, arquitectónicos y tecnológicos necesarios para comprender la solución desarrollada en el proyecto Geocuadrilla. Los elementos descritos a continuación sustentan las decisiones de diseño.

### **2.1 Integración de sistemas**

La integración de sistemas corresponde al conjunto de mecanismos mediante los cuales aplicaciones independientes pueden interoperar, compartiendo información de forma confiable y consistente. Según Hohpe y Woolf [1], la integración de aplicaciones empresariales constituye uno de los desafíos centrales en arquitecturas de software modernas, dado que los sistemas raramente operan de forma aislada en entornos productivos reales.

En el contexto del presente trabajo, la integración involucra la recepción de datos desde una aplicación móvil, su procesamiento en un componente centralizado y su distribución hacia un panel web de supervisión, consolidando información proveniente de dispositivos móviles, sistemas corporativos internos y fuentes geoespaciales.

## 2.2 Arquitecturas basadas en eventos

Una arquitectura basada en eventos es un estilo arquitectónico en el que los componentes del sistema se comunican mediante la producción y consumo de eventos de manera desacoplada. A diferencia de los modelos basados en llamadas síncronas directas, este enfoque permite que los productores de información operen con independencia de los consumidores, reduciendo el acoplamiento entre componentes y mejorando la resiliencia del sistema ante fallos parciales.

En sistemas de monitoreo en tiempo real, este patrón resulta especialmente apropiado, ya que permite que múltiples componentes reaccionen ante actualizaciones de estado sin requerir coordinación explícita entre ellos. En el sistema desarrollado, la recepción de actualizaciones de ubicación y su posterior procesamiento asíncrono se organizan bajo este principio. [2]

## 2.3 Mensajería AMQP y RabbitMQ

AMQP (*Advanced Message Queuing Protocol*) es un protocolo estándar abierto para la mensajería orientada a mensajes, que define mecanismos para el enrutamiento, entrega garantizada y confirmación de recepción de mensajes entre componentes distribuidos. A diferencia de protocolos más simples, AMQP provee semántica de entrega configurable —incluyendo *at-least-once* y *at-most-once*— lo que permite adaptar el comportamiento del sistema según los requisitos de confiabilidad de cada caso de uso.

RabbitMQ es una implementación ampliamente adoptada de este protocolo, que actúa como *bróker* de mensajes intermediando la comunicación entre productores y consumidores. Su modelo de colas y exchanges permite configurar topologías de mensajería flexibles, desacoplando los componentes emisores de los receptores tanto en el tiempo como en el espacio. En el sistema desarrollado, RabbitMQ opera como componente central del flujo de procesamiento asíncrono de actualizaciones de ubicación.

## 2.4 APIs REST

REST (*Representational State Transfer*) es un estilo arquitectónico para el diseño de interfaces de comunicación entre sistemas, definido por Fielding en su tesis doctoral como un conjunto de restricciones que, aplicadas sobre el protocolo HTTP, permiten construir servicios web escalables, sin estado y con una interfaz uniforme [3]. Bajo este modelo, los recursos del sistema se identifican mediante URIs y se manipulan a través de los métodos estándar del protocolo HTTP (GET, POST, PUT, DELETE), favoreciendo la interoperabilidad y la independencia tecnológica entre cliente y servidor.

En el sistema desarrollado, la API REST constituye el mecanismo principal mediante el cual la aplicación móvil envía actualizaciones de ubicación y los usuarios administradores gestionan los recursos del sistema, incluyendo archivos geoespaciales, usuarios y agenda operativa.

## 2.5 Bases de datos relacionales y PostgreSQL

Las bases de datos relacionales organizan la información en tablas estructuradas, vinculadas mediante claves primarias y foráneas, garantizando la integridad referencial de los datos a través de restricciones declarativas. Este modelo, fundamentado en el álgebra relacional, permite expresar consultas complejas de manera declarativa y asegurar la consistencia transaccional mediante las propiedades ACID (*Atomicity, Consistency, Isolation, Durability*) [4].

PostgreSQL es un sistema gestor de bases de datos relacionales de código abierto con más de treinta años de desarrollo activo, reconocido por su robustez, extensibilidad y cumplimiento estricto del estándar SQL. En el contexto del sistema desarrollado, PostgreSQL actúa como repositorio principal de la información operativa, almacenando el historial de actualizaciones de ubicación, los metadatos de archivos geoespaciales y la información de usuarios y agenda.

## 2.6 Almacenamiento de objetos y Minio

El almacenamiento de objetos es un paradigma de gestión de datos que organiza la información en unidades discretas denominadas objetos, cada una compuesta por los datos en sí, metadatos descriptivos y un identificador único. A diferencia de los sistemas de archivos jerárquicos tradicionales, el almacenamiento de objetos expone sus recursos mediante APIs HTTP, lo que facilita su integración en arquitecturas distribuidas y su escalabilidad horizontal.

MinIO es una implementación de código abierto compatible con la API de Amazon S3, ampliamente utilizada para el despliegue de infraestructura de almacenamiento de objetos en entornos locales o privados. En el sistema desarrollado, MinIO gestiona el almacenamiento físico de los archivos geoespaciales en formato KMZ, mientras que sus metadatos se persisten en la base de datos relacional, manteniendo el *backend* como único punto de acceso controlado a estos recursos.

## 2.7 Datos geoespaciales y formato KMZ

Los sistemas de información geográfica (SIG) permiten capturar, almacenar, procesar y visualizar datos asociados a posiciones geográficas, constituyendo una herramienta fundamental en la gestión de operaciones de campo y la toma de decisiones espaciales. Los datos geoespaciales pueden representar elementos puntuales, lineales o poligonales, y se estructuran mediante formatos estándar que facilitan su intercambio entre sistemas.

KML (*Keyhole Markup Language*) es un formato basado en XML, estandarizado por el Open Geospatial Consortium (OGC), utilizado para la representación de información geográfica incluyendo puntos, rutas, polígonos y capas de visualización. KMZ corresponde a la versión comprimida de KML y es ampliamente soportado por plataformas de visualización geográfica. En el contexto del proyecto, los archivos KMZ proporcionados por Chilquinta Transmisión contienen la representación georreferenciada de líneas y torres de alta tensión.

## 2.8 Procesamiento y monitoreo en tiempo real

Los sistemas de procesamiento en tiempo real se caracterizan por la capacidad de ingerir, procesar y entregar información dentro de ventanas de tiempo acotadas desde su generación, a diferencia de los sistemas de procesamiento por lotes que operan sobre conjuntos de datos acumulados. En aplicaciones de monitoreo operativo, este requisito es crítico, ya que la utilidad de la información está directamente condicionada por su latencia: una actualización de ubicación con varios minutos de retraso puede resultar irrelevante para la toma de decisiones en tiempo real.

Los mecanismos habituales para implementar comunicación en tiempo real entre servidor y cliente incluyen el sondeo periódico (*polling*), los eventos enviados por el servidor (*Server-Sent Events*) y los *WebSockets*.

## 2.9 WebSocket y comunicación en tiempo real

Entre los mecanismos descritos anteriormente, WebSocket fue seleccionado para la comunicación en tiempo real del sistema por establecer un canal bidireccional y persistente sobre una única conexión TCP, eliminando la sobrecarga del sondeo periódico y minimizando la latencia de notificación. El protocolo se inicia mediante un handshake HTTP estándar que es promovido a una conexión WebSocket, manteniendo compatibilidad con la infraestructura de red existente.

En el sistema desarrollado, esta comunicación se implementa mediante la librería SignalR de Microsoft, que abstrae el uso de WebSockets y provee mecanismos adicionales como la gestión de grupos de suscriptores y la reconexión automática ante interrupciones de conectividad.

## 2.10 Patrón Outbox

El patrón *Outbox*, o buzón de salida, es una técnica de diseño utilizada en sistemas distribuidos para garantizar la consistencia entre la persistencia de datos y la publicación de mensajes hacia un *bróker*, sin requerir transacciones distribuidas entre ambos sistemas. El problema que resuelve es el siguiente: si un componente persiste un dato en la base de datos y luego intenta publicar un mensaje al *bróker* de mensajería, un fallo entre ambas operaciones puede dejar el sistema en un estado inconsistente, con el dato almacenado pero el mensaje nunca enviado. La solución clásica consiste en almacenar el mensaje a publicar en una tabla dedicada dentro de la misma transacción que persiste la entidad de negocio, garantizando atomicidad completa. Un proceso independiente denominado publicador lee periódicamente los mensajes pendientes y los envía al *bróker*, marcándolos como procesados una vez confirmada su entrega.

En el sistema desarrollado se adoptaron los principios centrales de este patrón con una variación deliberada: el controlador persiste el mensaje en la tabla *outbox* de forma inmediata, y un proceso publicador independiente lo envía a RabbitMQ, manteniendo el desacoplamiento entre recepción y procesamiento. La entidad de negocio LocationUpdate es creada posteriormente por el consumer de RabbitMQ al momento del procesamiento, en una transacción separada. Esta decisión prioriza la baja latencia de confirmación hacia el dispositivo móvil sobre la atomicidad estricta del patrón clásico, aceptando como contrapartida que ante un fallo del consumer, el mensaje podría quedar en la cola sin generar el registro histórico correspondiente. Este riesgo se considera aceptable en el contexto operativo del sistema, dado que los dispositivos móviles reportan su ubicación de forma periódica: la pérdida aislada de un mensaje no

interrumpe la continuidad del monitoreo ni genera una brecha significativa en el historial de posiciones. [5]

## 2.11 Clean Architecture

*Clean Architecture* es un estilo de organización del código propuesto por Robert C. Martin que establece una separación estricta de responsabilidades mediante capas con dependencias dirigidas exclusivamente hacia el núcleo del sistema. Las capas más internas contienen las entidades y reglas de negocio del dominio, independientes de cualquier framework o tecnología, mientras que las capas externas contienen los detalles de implementación como bases de datos, interfaces de usuario y servicios externos.

Un principio central de este enfoque es la Regla de Dependencia: el código fuente solo puede depender de capas más internas, nunca al revés. Esto se implementa mediante el uso de interfaces definidas en la capa de aplicación, implementadas por adaptadores en las capas externas. El resultado es un sistema cuya lógica de negocio puede evolucionar de forma independiente a los detalles tecnológicos, facilitando el mantenimiento y la extensión del sistema a largo plazo. En el sistema desarrollado, esta arquitectura estructura el *backend* en capas de Dominio, Aplicación, Adaptadores e Infraestructura. [6]

## 2.12 Throughput y métricas de rendimiento

En el contexto de sistemas de procesamiento de datos, el *throughput* se define como el volumen de trabajo que un sistema es capaz de completar por unidad de tiempo, típicamente expresado en solicitudes por segundo o mensajes por segundo. Esta métrica es fundamental para evaluar la capacidad de un sistema ante escenarios de carga variable y constituye un indicador complementario a la latencia: mientras la latencia mide el tiempo de respuesta de una operación individual, el *throughput* mide la capacidad agregada del sistema para procesar múltiples operaciones de forma simultánea.

En sistemas de monitoreo de flotas con múltiples dispositivos reportando de manera concurrente, el *throughput* es especialmente relevante, ya que el sistema debe absorber ráfagas de actualizaciones simultáneas sin degradar los tiempos de respuesta ni producir pérdida de mensajes. En el sistema desarrollado, esta capacidad se aborda mediante el patrón *Outbox* y el procesamiento asíncrono, que desacoplan la recepción de mensajes de su procesamiento posterior, manteniendo un alto *throughput* de ingesta sin comprometer la latencia de confirmación hacia el dispositivo móvil.

### 2.13 Autenticación basada en tokens JWT

*JSON Web Token* (JWT) es un estándar abierto que define un formato compacto y autocontenido para la transmisión segura de información de autenticación entre partes. Un token JWT se compone de tres partes codificadas en Base64 separadas por puntos: el encabezado (*header*), que especifica el algoritmo de firma; la carga útil (*payload*), que contiene las afirmaciones sobre el usuario y sus permisos; y la firma (*signature*), que permite verificar la integridad del token.

Esta característica de ser autocontenido hace de JWT un mecanismo especialmente adecuado para APIs REST, donde la validación de identidad sin necesidad de consultar un almacén de sesiones centralizado reduce la complejidad y mejora la escalabilidad. En el sistema desarrollado, los tokens JWT se emiten al momento de la autenticación e incluyen el identificador de usuario y sus roles, permitiendo que el *backend* valide las solicitudes de forma independiente sin acceder a la base de datos en cada operación.

### 2.14 Patrón Repositorio

El patrón Repositorio es un patrón de diseño que actúa como capa de abstracción entre la lógica de dominio de una aplicación y la capa de acceso a datos. Su objetivo es encapsular la lógica de consulta y persistencia detrás de una interfaz orientada al dominio, de modo que los servicios de aplicación interactúen con colecciones de objetos sin conocer los detalles del mecanismo de almacenamiento subyacente.

Esta separación permite que la lógica de negocio permanezca independiente de la tecnología de persistencia utilizada, facilitando tanto la mantenibilidad del sistema como su capacidad de prueba. En el sistema desarrollado, el patrón Repositorio se implementa mediante Entity Framework Core sobre PostgreSQL, exponiendo interfaces de acceso a datos que son consumidas por los servicios de aplicación sin acoplamiento directo al ORM. [7]

### 3 Metodología y Diseño del Sistema

#### 3.1 Metodología de desarrollo

La metodología *Scrum* fue seleccionada como enfoque de desarrollo debido a la naturaleza iterativa del proyecto y a la necesidad de incorporar retroalimentación temprana del patrocinador durante el proceso de construcción del sistema de integración. Dado que los requerimientos operativos podían ajustarse en función de las validaciones progresivas, resultaba fundamental contar con un marco metodológico que permitiera adaptar tanto el diseño como la implementación de manera incremental.

La aplicación de *Scrum* permitió estructurar el desarrollo en iteraciones controladas, facilitando la validación temprana de los flujos de integración, el procesamiento de datos y la arquitectura propuesta antes de su consolidación final. El uso de entregables parciales al término de cada *Sprint* contribuyó a identificar ajustes necesarios en los requisitos funcionales y no funcionales, reduciendo el riesgo asociado a decisiones de diseño tomadas en etapas iniciales.

Desde el punto de vista técnico, este enfoque influyó directamente en la definición de una arquitectura modular y desacoplada, priorizando componentes que pudieran evolucionar de forma independiente. En particular, la adopción de mensajería asíncrona y de mecanismos de comunicación en tiempo real respondió a la necesidad de validar tempranamente el comportamiento del sistema bajo escenarios reales de operación.

De esta forma, la metodología utilizada no solo estructuró el proceso de desarrollo, sino que también condicionó las decisiones de diseño adoptadas para el sistema de integración descrito en las secciones siguientes.

#### 3.2 Justificación de tecnologías y herramientas seleccionadas

La selección de las tecnologías utilizadas en el sistema respondió a criterios técnicos específicos evaluados en función de los requisitos del proyecto.

**.NET** fue seleccionado como plataforma de desarrollo del *backend* por tres razones principales. Primero, la familiaridad del equipo con el ecosistema, con más de cuatro años de experiencia trabajando con la plataforma, lo que reduce significativamente la curva de aprendizaje y el riesgo técnico del proyecto. Segundo, la disponibilidad de Entity Framework Core como ORM nativo, que simplifica el acceso y la gestión de datos relacionales mediante un modelo orientado a objetos, reduciendo el código repetitivo

asociado a la capa de persistencia. Tercero, su portabilidad multiplataforma: desde la versión .NET Core, la plataforma es compatible con Windows, Linux y macOS, lo que permite desplegar el sistema en el servidor del patrocinador independientemente del sistema operativo utilizado. Adicionalmente, .NET es de código abierto desde 2014, lo que elimina restricciones de licenciamiento para su uso en proyectos institucionales.

**RabbitMQ** fue seleccionado como *bróker* de mensajería con el objetivo de desacoplar la recepción de actualizaciones de ubicación de su procesamiento posterior. Este desacoplamiento permite que el componente receptor confirme la recepción al dispositivo móvil de forma inmediata, sin esperar a que la lógica de procesamiento y persistencia se complete, reduciendo la latencia percibida por el cliente. Alternativas como Apache Kafka fueron descartadas por dos razones principales: su modelo de retención de eventos y complejidad operativa están orientados a escenarios de mayor volumen y procesamiento analítico, lo que excede los requisitos del presente sistema; y RabbitMQ es una solución consolidada para mensajería en sistemas IoT y comunicación con dispositivos en campo, caso de uso que se alinea directamente con el presente proyecto.

**PostgreSQL** fue elegido como sistema gestor de base de datos relacional por ser la alternativa de código abierto más consolidada del ecosistema, con soporte nativo para tipos de datos geoespaciales mediante la extensión PostGIS, cumplimiento estricto del estándar SQL y amplia compatibilidad con los frameworks ORM utilizados en el proyecto. Su robustez y madurez lo hacen adecuado para garantizar la integridad referencial requerida por el modelo de datos del sistema.

**MinIO** fue incorporado como solución de almacenamiento de objetos por ofrecer una API completamente compatible con Amazon S3, lo que permite utilizar las mismas abstracciones e interfaces que el estándar de la industria sin depender de infraestructura de nube propietaria. Esta compatibilidad facilita una eventual migración hacia S3 en caso de que el patrocinador requiera escalar la infraestructura, manteniendo el código del sistema sin modificaciones.

### 3.3 Requisitos de sistema

#### 3.3.1 Requisitos funcionales

- **RF-01 Recepción de actualizaciones de ubicación:**

El sistema deberá recibir actualizaciones de ubicación que incluyan, como mínimo, latitud, longitud, identificador del dispositivo y marca de tiempo, provenientes de la aplicación móvil mediante un mecanismo de mensajería asíncrona.

- **RF-02 Registro histórico de ubicaciones:**

El sistema deberá almacenar de forma persistente las actualizaciones de ubicación recibidas, manteniendo un historial que permite la consulta posterior de la información registrada.

- **RF-03 Notificación automática de actualizaciones:**

El sistema deberá notificar automáticamente las nuevas actualizaciones de ubicación al panel web del centro de control sin requerir solicitudes explícitas por parte del cliente.

- **RF-04 Gestión de archivos geospaciales:**

El sistema deberá permitir la carga, actualización, eliminación y listado de archivos geospaciales en formato KMZ asociados a la operación.

- **RF-05 Importación y normalización de agenda operativa:**

El sistema deberá permitir la importación y normalización de la agenda operativa de las cuadrillas, de modo que sea posible asociar vehículos y personal a los dispositivos monitoreados.

#### 3.3.2 Requisitos no funcionales

- **RNF-01 Rendimiento (latencia de actualización):**

La latencia entre la recepción de una actualización de ubicación por parte del sistema y su disponibilidad en el panel web no deberá exceder los 2 segundos, medido bajo condiciones normales de operación.

- **RNF-02 Capacidad de procesamiento:**

El sistema deberá ser capaz de ingerir ráfagas de actualizaciones de ubicación de forma continua sin bloquear el procesamiento de nuevas entradas ni degradar el cumplimiento del requisito RNF-01.

- **RNF-03 Disponibilidad operativa:**

El sistema deberá encontrarse disponible para la recepción y procesamiento de actualizaciones de ubicación durante el horario operativo definido por el centro de control.

- **RNF-04 Seguridad de acceso:**

El sistema deberá restringir el acceso a sus funcionalidades y recursos mediante mecanismo de autenticación y autorización, garantizando que solo los usuarios autorizados puedan acceder a la información operativa.

### 3.4 Casos de uso del sistema

#### 3.4.1 Actores

- **Técnico de terreno:** usuario que utiliza la aplicación móvil y genera actualizaciones de ubicación.
- **Operador de centro de control:** usuario que supervisa la operación en tiempo real mediante el panel web.
- **Administrador del sistema:** usuario responsable de la gestión de recursos, accesos y configuración operativa.

#### 3.4.2 Casos de uso identificados

- Recibir actualización de ubicación.
- Sincronizar ubicaciones más recientes por cuadrilla.
- Reflejar actualización en el panel web.
- Listar archivos geoespaciales en el panel web.
- Visualizar archivo geoespacial en el panel web.
- Cargar/Actualizar archivos geoespaciales KMZ.
- Eliminar archivo geoespacial KMZ.
- Importar y normalizar agenda operativa.
- Autenticación y autorización de usuarios.
- Gestionar usuarios.
- Asignar roles a usuarios.
- Cerrar sesión.

## Casos de uso

Tabla 1. Caso de uso narrativo: CU01 – Recibir actualización de ubicación.

|                                                                                                                      |                                                                                                                                                                                                                                                                                                         |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                                                       | CU01 – Recibir actualización de ubicación.                                                                                                                                                                                                                                                              |
| <b>Descripción:</b>                                                                                                  | Registrar actualizaciones periódicas de ubicación asociadas a un técnico en terreno, enviadas automáticamente por la aplicación móvil, y habilitar su posterior difusión al panel web                                                                                                                   |
| <b>Ref. Cruzadas:</b>                                                                                                | CU03 – Reflejar actualización en el panel web.                                                                                                                                                                                                                                                          |
| <b>Actores:</b>                                                                                                      | Técnico en terreno.                                                                                                                                                                                                                                                                                     |
| <b>Precondiciones:</b>                                                                                               | <ul style="list-style-type: none"> <li>– El técnico se encuentra autenticado y autorizado.</li> <li>– El mensaje contiene los datos mínimos requeridos (identificador de cuadrilla, coordenadas y marca de tiempo).</li> <li>– La aplicación móvil cuenta con conectividad hacia el sistema.</li> </ul> |
| <b>Postcondiciones:</b>                                                                                              | <ul style="list-style-type: none"> <li>– La actualización quedó persistida en el sistema.</li> <li>– La información quedó disponible para notificación al panel web.</li> </ul>                                                                                                                         |
| <b>FLUJO NORMAL</b>                                                                                                  |                                                                                                                                                                                                                                                                                                         |
| <b>Acción del Usuario</b>                                                                                            | <b>Respuesta del sistema</b>                                                                                                                                                                                                                                                                            |
| 1. La aplicación móvil envía automáticamente una actualización de ubicación asociada al técnico, de forma periódica. | 2. El sistema recibe la información y valida la autenticación y el formato de los datos.                                                                                                                                                                                                                |
|                                                                                                                      | 3. El sistema registra la actualización para su procesamiento asíncrono.                                                                                                                                                                                                                                |
|                                                                                                                      | 4. El sistema procesa la información y la almacena en el repositorio persistente.                                                                                                                                                                                                                       |
|                                                                                                                      | 5. El sistema envía la actualización procesada al panel web mediante el canal de comunicación en tiempo real.                                                                                                                                                                                           |
| <b>FLUJO ALTERNATIVO</b>                                                                                             |                                                                                                                                                                                                                                                                                                         |
| 1a. La aplicación móvil envía una actualización con datos incompletos o inválidos.                                   | 2a. El sistema descarta la actualización.                                                                                                                                                                                                                                                               |

Fuente: Elaboración propia

Tabla 2. Caso de uso narrativo: CU02 – Sincronizar ubicaciones más recientes por cuadrilla.

|                                                                                                             |                                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                                              | CU02 – Sincronizar ubicaciones más recientes por cuadrilla.                                                                                                                       |
| <b>Descripción:</b>                                                                                         | Proveer automáticamente al panel web las ubicaciones más recientes de las cuadrillas operativas al momento de establecer una suscripción al canal de comunicación en tiempo real. |
| <b>Ref. Cruzadas:</b>                                                                                       | CU-01 – Recibir actualización de ubicación.<br>CU-03 – Notificar actualización en el panel web.<br>CU-08 – Importar y normalizar agenda operativa.                                |
| <b>Actores:</b>                                                                                             | Operador del centro de control.                                                                                                                                                   |
| <b>Precondiciones:</b>                                                                                      | <ul style="list-style-type: none"> <li>– El operador se encuentra autenticado y autorizado.</li> <li>– El panel web cuenta con conectividad hacia el sistema.</li> </ul>          |
| <b>Postcondiciones:</b>                                                                                     | <ul style="list-style-type: none"> <li>– El panel web recibió las ubicaciones más recientes de las cuadrillas para su visualización inicial.</li> </ul>                           |
| <b>FLUJO NORMAL</b>                                                                                         |                                                                                                                                                                                   |
| <b>Acción del Usuario</b>                                                                                   | <b>Respuesta del sistema</b>                                                                                                                                                      |
| 1. El operador accede al panel web, estableciendo una conexión con el canal de comunicación en tiempo real. |                                                                                                                                                                                   |
|                                                                                                             | 2. El sistema valida la autenticación y autorización del operador.                                                                                                                |
|                                                                                                             | 3. El sistema obtiene las ubicaciones más recientes asociadas a las cuadrillas operativas, incluyendo la agenda de la cuadrilla asociada.                                         |
|                                                                                                             | 4. El sistema envía automáticamente la información al panel web a través del canal de comunicación en tiempo real.                                                                |
| <b>FLUJO ALTERNATIVO</b>                                                                                    |                                                                                                                                                                                   |
| 1a. El operador se conecta al panel web, pero no existen ubicaciones registradas para una o más cuadrillas. | 2a. El sistema establece la conexión al canal y envía una lista de ubicaciones vacía al panel web.                                                                                |
| 1b. El operador intenta acceder al panel web sin autorización suficiente.                                   | 2b. El sistema deniega el acceso y la conexión es rechazada.                                                                                                                      |

Fuente: Elaboración propia

Tabla 3. Caso de uso narrativo: CU03 – Reflejar actualización en el panel web.

|                                                                                         |                                                                                                                                                                                                                                                                            |
|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                          | CU03 – Reflejar actualización en el panel web.                                                                                                                                                                                                                             |
| <b>Descripción:</b>                                                                     | Notificar automáticamente al panel web de las nuevas actualizaciones de ubicación procesadas, permitiendo la visualización en tiempo real de los movimientos de las cuadrillas operativas.                                                                                 |
| <b>Ref. Cruzadas:</b>                                                                   | CU-01 – Recibir actualización de ubicación.<br>CU-02 – Sincronizar ubicaciones más recientes por cuadrilla.<br>CU-08 – Importar y normalizar agenda operativa.                                                                                                             |
| <b>Actores:</b>                                                                         | Operador del centro de control.                                                                                                                                                                                                                                            |
| <b>Precondiciones:</b>                                                                  | <ul style="list-style-type: none"> <li>– El operador se encuentra autenticado y autorizado.</li> <li>– El panel web mantiene una conexión activa con el canal de comunicación en tiempo real.</li> <li>– El panel web cuenta con conectividad hacia el sistema.</li> </ul> |
| <b>Postcondiciones:</b>                                                                 | <ul style="list-style-type: none"> <li>– El panel web recibió la nueva actualización de ubicación y esta es reflejada en tiempo real.</li> </ul>                                                                                                                           |
| <b>FLUJO NORMAL</b>                                                                     |                                                                                                                                                                                                                                                                            |
| <b>Acción del Usuario</b>                                                               | <b>Respuesta del sistema</b>                                                                                                                                                                                                                                               |
| 1. El operador mantiene el panel web abierto y conectado al sistema de monitoreo.       |                                                                                                                                                                                                                                                                            |
|                                                                                         | 2. El sistema detecta que una nueva actualización de ubicación ha sido procesada correctamente.                                                                                                                                                                            |
|                                                                                         | 3. El sistema envía automáticamente la actualización al panel web a través del canal de comunicación en tiempo real, incluyendo la agenda de la cuadrilla asociada.                                                                                                        |
|                                                                                         | 4. El panel web actualiza la visualización de la cuadrilla correspondiente.                                                                                                                                                                                                |
| <b>FLUJO ALTERNATIVO</b>                                                                |                                                                                                                                                                                                                                                                            |
| 1a. No hay instancias del panel web conectadas al canal de comunicación en tiempo real. | 2a. El sistema no envía la notificación en tiempo real, la información queda disponible para ser obtenida en una futura conexión (CU-02).                                                                                                                                  |
| 1b. El panel web pierde la conexión con el canal de comunicación en tiempo real.        | 2b. El sistema interrumpe el envío de notificaciones hasta que la conexión sea restablecida.                                                                                                                                                                               |

Fuente: Elaboración propia

Tabla 4. Caso de uso narrativo: CU04 – Listar archivos geoespaciales.

|                                                                                                                    |                                                                                                                                                                                                 |
|--------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre:                                                                                                            | CU04 – Listar archivos geoespaciales.                                                                                                                                                           |
| Descripción:                                                                                                       | Permitir a un usuario autorizado consultar el listado de archivos geoespaciales en el panel web.                                                                                                |
| Ref. Cruzadas:                                                                                                     | CU-05 – Gestionar archivos geoespaciales en formato KMZ.                                                                                                                                        |
| Actores:                                                                                                           | Operador del centro de control.                                                                                                                                                                 |
| Precondiciones:                                                                                                    | <ul style="list-style-type: none"><li>– El operador se encuentra autenticado y autorizado para acceder al panel web.</li><li>– El panel web cuenta con conectividad hacia el sistema.</li></ul> |
| Postcondiciones:                                                                                                   | <ul style="list-style-type: none"><li>– El panel web recibió el listado de archivos geoespaciales disponibles, o una lista vacía en el caso de no existir registros.</li></ul>                  |
| FLUJO NORMAL                                                                                                       |                                                                                                                                                                                                 |
| Acción del Usuario                                                                                                 | Respuesta del sistema                                                                                                                                                                           |
| 1. El operador accede a la sección de archivos geoespaciales                                                       |                                                                                                                                                                                                 |
|                                                                                                                    | 2. El sistema valida la autenticación y autorización del operador.                                                                                                                              |
|                                                                                                                    | 3. El sistema consulta los archivos geoespaciales y sus metadatos.                                                                                                                              |
|                                                                                                                    | 4. El sistema entrega el listado de archivos al panel web con sus metadatos respectivos.                                                                                                        |
| FLUJO ALTERNATIVO                                                                                                  |                                                                                                                                                                                                 |
| 1a. El operador accede a la sección de archivos geoespaciales, pero no existen archivos geoespaciales registrados. | 2a. El sistema entrega una lista vacía de archivos geoespaciales.                                                                                                                               |

Fuente: Elaboración propia

Tabla 5. Caso de uso narrativo: CU05 – Visualizar archivo geoespacial en el panel web.

|                                                                                                                        |                                                                                                                                                                                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                                                         | CU05 – Visualizar archivo geoespacial en el panel web.                                                                                                                                                                                                                                                      |
| <b>Descripción:</b>                                                                                                    | Permitir al operador del centro de control seleccionar uno o más archivos geoespaciales desde el panel web para que sean cargados y mostrados como capas sobre el mapa mediante la API de Google Maps.                                                                                                      |
| <b>Ref. Cruzadas:</b>                                                                                                  | CU-04 – Listar archivos geoespaciales.                                                                                                                                                                                                                                                                      |
| <b>Actores:</b>                                                                                                        | Operador del centro de control.                                                                                                                                                                                                                                                                             |
| <b>Precondiciones:</b>                                                                                                 | <ul style="list-style-type: none"> <li>– El operador se encuentra autenticado y autorizado para acceder al panel web.</li> <li>– El panel web cuenta con conectividad hacia el sistema.</li> <li>– El panel web ha obtenido previamente el listado de archivos geoespaciales disponibles (CU-04)</li> </ul> |
| <b>Postcondiciones:</b>                                                                                                | <ul style="list-style-type: none"> <li>– El archivo KMZ se visualizó como capa en el mapa del panel web.</li> </ul>                                                                                                                                                                                         |
| <b>FLUJO NORMAL</b>                                                                                                    |                                                                                                                                                                                                                                                                                                             |
| <b>Acción del Usuario</b>                                                                                              | <b>Respuesta del sistema</b>                                                                                                                                                                                                                                                                                |
| 1. El operador selecciona un archivo geoespacial desde el listado de archivos disponibles (CU-04)                      |                                                                                                                                                                                                                                                                                                             |
|                                                                                                                        | 2. El sistema recupera el contenido del archivo KMZ seleccionado.                                                                                                                                                                                                                                           |
|                                                                                                                        | 3. El sistema entrega el archivo KMZ al panel web.                                                                                                                                                                                                                                                          |
|                                                                                                                        | 4. El panel web carga el archivo y lo visualiza como capa en el mapa mediante la API de Google Maps.                                                                                                                                                                                                        |
| <b>FLUJO ALTERNATIVO</b>                                                                                               |                                                                                                                                                                                                                                                                                                             |
| 1a. El operador activa la visualización de un archivo KMZ pero ocurre un error al obtener el archivo desde el sistema. | 2a. El sistema retorna el código de error correspondiente al problema encontrado.                                                                                                                                                                                                                           |

Fuente: Elaboración propia

Tabla 6. Caso de uso narrativo: CU06 – Cargar/Actualizar archivos geoespaciales KMZ.

|                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                                                                                                 | CU06 – Cargar/Actualizar archivos geoespaciales KMZ.                                                                                                                                                                                                                                                            |
| <b>Descripción:</b>                                                                                                                                            | Permitir al administrador del sistema cargar un nuevo archivo geoespacial KMZ o actualizar uno existente (archivo o metadatos), asegurando unicidad de nombre y disponibilidad posterior en el panel web.                                                                                                       |
| <b>Ref. Cruzadas:</b>                                                                                                                                          | CU-04 – Listar archivos geoespaciales.                                                                                                                                                                                                                                                                          |
| <b>Actores:</b>                                                                                                                                                | Administrador del sistema.                                                                                                                                                                                                                                                                                      |
| <b>Precondiciones:</b>                                                                                                                                         | <ul style="list-style-type: none"><li>– El administrador se encuentra autenticado y autorizado para acceder al panel web.</li><li>– El panel web cuenta con conectividad hacia el sistema.</li><li>– Para actualizaciones, el panel web ha obtenido previamente el listado de archivos geoespaciales.</li></ul> |
| <b>Postcondiciones:</b>                                                                                                                                        | <ul style="list-style-type: none"><li>– El archivo KMZ quedó registrado o actualizado correctamente.</li></ul>                                                                                                                                                                                                  |
| <b>FLUJO NORMAL</b>                                                                                                                                            |                                                                                                                                                                                                                                                                                                                 |
| <b>Acción del Usuario</b>                                                                                                                                      | <b>Respuesta del sistema</b>                                                                                                                                                                                                                                                                                    |
| 1. El administrador accede a la sección de administración de archivos geoespaciales.                                                                           | 2. El sistema valida la autenticación y autorización del administrador.                                                                                                                                                                                                                                         |
| 3. El administrador selecciona la opción de cargar un nuevo archivo o actualizar uno existente e ingresa los datos correspondientes (archivo KMZ y metadatos). | 4. El sistema valida los datos ingresados y verifica que no exista otro archivo con el mismo nombre en caso de ser un nuevo registro.                                                                                                                                                                           |
| 5. El administrador confirma la operación.                                                                                                                     | 6. El sistema almacena el archivo o actualiza los metadatos según corresponda.                                                                                                                                                                                                                                  |
|                                                                                                                                                                | 7. El sistema registra la operación y confirma el resultado al administrador.                                                                                                                                                                                                                                   |
| <b>FLUJO ALTERNATIVO</b>                                                                                                                                       |                                                                                                                                                                                                                                                                                                                 |
| 3a. El administrador intenta cargar/actualizar un archivo con un formato invalido o no soportado                                                               | 4a. El sistema rechaza la operación e informa el error correspondiente.                                                                                                                                                                                                                                         |
| 3b. El administrador intenta cargar un archivo utilizando un nombre ya registrado.                                                                             | 4b. El sistema rechaza la operación por nombre duplicado y no realiza cambios.                                                                                                                                                                                                                                  |

Fuente: Elaboración propia

Tabla 7. Caso de uso narrativo: CU07 – Eliminar archivo geoespacial KMZ.

|                                                                                          |                                                                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre:                                                                                  | CU07 – Eliminar un archivo geoespacial KMZ.                                                                                                                                                                                                                                               |
| Descripción:                                                                             | Permite al administrador eliminar un archivo geoespacial KMZ previamente registrado del sistema.                                                                                                                                                                                          |
| Ref. Cruzadas:                                                                           | CU-04 – Listar archivos geoespaciales.                                                                                                                                                                                                                                                    |
| Actores:                                                                                 | Administrador del sistema.                                                                                                                                                                                                                                                                |
| Precondiciones:                                                                          | <ul style="list-style-type: none"><li>– El administrador se encuentra autenticado y autorizado para acceder al panel web.</li><li>– El panel web cuenta con conectividad hacia el sistema.</li><li>– El panel web ha obtenido previamente el listado de archivos geoespaciales.</li></ul> |
| Postcondiciones:                                                                         | <ul style="list-style-type: none"><li>– El archivo KMZ quedó eliminado del repositorio.</li></ul>                                                                                                                                                                                         |
| FLUJO NORMAL                                                                             |                                                                                                                                                                                                                                                                                           |
| Acción del Usuario                                                                       | Respuesta del sistema                                                                                                                                                                                                                                                                     |
| 1. El administrador selecciona un archivo KMZ desde la lista de archivos administrables. |                                                                                                                                                                                                                                                                                           |
|                                                                                          | 2. El sistema valida la autenticación y autorización del administrador                                                                                                                                                                                                                    |
| 3. El administrador solicita eliminar el archivo seleccionado                            |                                                                                                                                                                                                                                                                                           |
|                                                                                          | 4. El sistema elimina el archivo del repositorio y actualiza los registros asociados.                                                                                                                                                                                                     |
|                                                                                          | 5. El sistema registra la operación y confirma el resultado al administrador.                                                                                                                                                                                                             |
| FLUJO ALTERNATIVO                                                                        |                                                                                                                                                                                                                                                                                           |
| 1a. Un usuario sin permisos de administración intenta acceder a la funcionalidad.        | 2a. El sistema deniega el acceso.                                                                                                                                                                                                                                                         |

Fuente: Elaboración propia

Tabla 8. Caso de uso narrativo: CU08 – Importar y normalizar agenda operativa.

|                                                                                       |                                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                        | CU08 – Importar y normalizar agenda operativa.                                                                                                                                                                   |
| <b>Descripción:</b>                                                                   | Permitir al administrador del sistema cargar un archivo Excel desde el panel web para importar y normalizar la agenda operativa, actualizando la base de datos de trabajos y estados asociados a las cuadrillas. |
| <b>Ref. Cruzadas:</b>                                                                 | CU-02 – Obtener ubicaciones más recientes por cuadrilla.<br>CU-03 – Notificar actualización del panel web.                                                                                                       |
| <b>Actores:</b>                                                                       | Administrador del sistema.                                                                                                                                                                                       |
| <b>Precondiciones:</b>                                                                | <ul style="list-style-type: none"> <li>– El sistema se encuentra operativo.</li> <li>– El administrador se encuentra autenticado y autorizado.</li> </ul>                                                        |
| <b>Postcondiciones:</b>                                                               | <ul style="list-style-type: none"> <li>– La agenda se guardó en formato normalizado.</li> <li>– Los trabajos y estados quedaron disponibles para su asociación por cuadrilla en el panel web.</li> </ul>         |
| <b>FLUJO NORMAL</b>                                                                   |                                                                                                                                                                                                                  |
| <b>Acción del Usuario</b>                                                             | <b>Respuesta del sistema</b>                                                                                                                                                                                     |
| 1. El administrador accede a la sección de importación de agenda y confirma la carga. | 2. El sistema valida que el archivo tenga un formato y extensión compatibles.                                                                                                                                    |
|                                                                                       | 3. El sistema normaliza la información e incorpora la agenda a la base de datos.                                                                                                                                 |
|                                                                                       | 4. El sistema envía una actualización de estado de las cuadrillas asociadas (CU-03).                                                                                                                             |
|                                                                                       | 5. El sistema retorna un mensaje de éxito.                                                                                                                                                                       |
| <b>FLUJO ALTERNATIVO</b>                                                              |                                                                                                                                                                                                                  |
| 1a. La fuente de datos contiene datos incompletos o inconsistentes.                   | 2a. El sistema registra el error de importación y cancela la carga.                                                                                                                                              |
| 3a. Ocurre un error durante el procesamiento o normalización de datos.                | 4a. El sistema aborta la importación, no realiza cambios en la base de datos e informa el error ocurrido al administrador.                                                                                       |

Fuente: Elaboración propia

Tabla 9. Caso de uso narrativo: CU09 – Autenticación y autorización de usuarios.

|                                                                                    |                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                     | CU09 – Autenticación y autorización de usuarios.                                                                                                                                                                                 |
| <b>Descripción:</b>                                                                | Permitir que un usuario acceda al sistema mediante autenticación y una vez autenticado, se aplica la autorización según los roles asignados para habilitar o restringir funcionalidades del sistema.                             |
| <b>Ref. Cruzadas:</b>                                                              | CU-10 – Gestionar usuarios.<br>CU-11 – Asignar roles a usuario.                                                                                                                                                                  |
| <b>Actores:</b>                                                                    | Usuario del sistema (Operador del centro de control, Administrador del sistema y Técnico en terreno).                                                                                                                            |
| <b>Precondiciones:</b>                                                             | <ul style="list-style-type: none"> <li>– El usuario se encuentra registrado en el sistema.</li> <li>– El sistema se encuentra operativo.</li> </ul>                                                                              |
| <b>Postcondiciones:</b>                                                            | <ul style="list-style-type: none"> <li>– El usuario se autenticó con un token válido.</li> <li>– El sistema aplicó las reglas de autorización según los roles asignados, habilitando el acceso a las funcionalidades.</li> </ul> |
| <b>FLUJO NORMAL</b>                                                                |                                                                                                                                                                                                                                  |
| <b>Acción del Usuario</b>                                                          | <b>Respuesta del sistema</b>                                                                                                                                                                                                     |
| 1. El usuario accede a la pantalla de inicio de sesión e ingresa sus credenciales. |                                                                                                                                                                                                                                  |
|                                                                                    | 2. El sistema valida las credenciales y autentica al usuario.                                                                                                                                                                    |
|                                                                                    | 3. El sistema obtiene los roles asociados al usuario.                                                                                                                                                                            |
|                                                                                    | 4. El sistema crea un token y habilita el acceso a las funcionalidades permitidas para el rol.                                                                                                                                   |
|                                                                                    | 5. El sistema retorna el token de acceso y redirige al usuario a la pantalla correspondiente a su rol.                                                                                                                           |
| <b>FLUJO ALTERNATIVO</b>                                                           |                                                                                                                                                                                                                                  |
| 1a. El usuario ingresa credenciales inválidas.                                     | 2a. El sistema rechaza el inicio de sesión.                                                                                                                                                                                      |

Fuente: Elaboración propia

Tabla 10. Caso de uso narrativo: CU10 – Gestionar usuarios.

|                                                                                        |                                                                                                         |
|----------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                                         | CU10 – Gestionar usuarios.                                                                              |
| <b>Descripción:</b>                                                                    | Permitir al administrador del sistema crear, modificar y eliminar usuarios que acceden a la plataforma. |
| <b>Ref. Cruzadas:</b>                                                                  | CU-09 – Autenticación y autorización de usuarios.<br>CU-11 – Asignar roles a usuario.                   |
| <b>Actores:</b>                                                                        | Administrador del sistema.                                                                              |
| <b>Precondiciones:</b>                                                                 | – El administrador se encuentra autenticado y autorizado con permisos de administración de usuarios.    |
| <b>Postcondiciones:</b>                                                                | – El usuario quedó creado, actualizado o eliminado según la acción realizada.                           |
| <b>FLUJO NORMAL</b>                                                                    |                                                                                                         |
| <b>Acción del Usuario</b>                                                              | <b>Respuesta del sistema</b>                                                                            |
| 1. El administrador accede al módulo de administración de usuarios.                    |                                                                                                         |
|                                                                                        | 2. El sistema valida las credenciales y autorización del administrador.                                 |
| 3. El administrador envía la acción a realizar (creación, modificación o eliminación). |                                                                                                         |
|                                                                                        | 4. El sistema valida los datos incluidos en la solicitud.                                               |
|                                                                                        | 5. El sistema ejecuta la operación solicitada sobre el usuario seleccionado.                            |
|                                                                                        | 6. El sistema registra la operación y retorna el resultado de la solicitud.                             |
| <b>FLUJO ALTERNATIVO</b>                                                               |                                                                                                         |
| 3a. El administrador envía una solicitud con datos incompletos o inválidos.            | 4a. El sistema rechaza la operación y retorna el error.                                                 |
| 3b. El administrador intenta crear un usuario con identificador ya registrado.         | 4b. El sistema rechaza la operación por duplicidad y retorna el error.                                  |

Fuente: Elaboración propia

Tabla 11. Caso de uso narrativo: CU11 – Asignar roles a usuarios.

|                                                                             |                                                                                                                                                                                                                                                                                          |
|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre:                                                                     | CU11 – Asignar roles a usuarios.                                                                                                                                                                                                                                                         |
| Descripción:                                                                | Permitir al administrador del sistema asignar, modificar o remover roles asociados a un usuario con el fin de controlar el acceso a las distintas funcionalidades del sistema.                                                                                                           |
| Ref. Cruzadas:                                                              | CU-09 – Autenticación y autorización de usuarios.<br>CU-10 – Gestionar usuarios.                                                                                                                                                                                                         |
| Actores:                                                                    | Administrador del sistema.                                                                                                                                                                                                                                                               |
| Precondiciones:                                                             | <ul style="list-style-type: none"><li>– El administrador se encuentra autenticado y autorizado con permisos de administración de usuarios.</li><li>– El usuario objetivo ya se encuentra registrado.</li><li>– Los roles por asignar ya se encuentran definidos en el sistema.</li></ul> |
| Postcondiciones:                                                            | <ul style="list-style-type: none"><li>– Los roles asociados al usuario fueron actualizados.</li></ul>                                                                                                                                                                                    |
| FLUJO NORMAL                                                                |                                                                                                                                                                                                                                                                                          |
| Acción del Usuario                                                          | Respuesta del sistema                                                                                                                                                                                                                                                                    |
| 1. El administrador accede al módulo de administración de usuarios.         |                                                                                                                                                                                                                                                                                          |
|                                                                             | 2. El sistema valida la sesión y los permisos mediante el token.                                                                                                                                                                                                                         |
| 3. El administrador selecciona un usuario y define los nuevos roles.        |                                                                                                                                                                                                                                                                                          |
|                                                                             | 4. El sistema valida los datos incluidos en la solicitud.                                                                                                                                                                                                                                |
|                                                                             | 5. El sistema actualiza los roles del usuario seleccionado.                                                                                                                                                                                                                              |
|                                                                             | 6. El sistema registra la operación y retorna el resultado de la solicitud.                                                                                                                                                                                                              |
| FLUJO ALTERNATIVO                                                           |                                                                                                                                                                                                                                                                                          |
| 3a. El administrador envía una solicitud con roles incompletos o inválidos. | 4a. El sistema rechaza la operación y retorna el error.                                                                                                                                                                                                                                  |

Fuente: Elaboración propia

Tabla 12. Caso de uso narrativo: CU12 – Cerrar sesión.

|                                                                   |                                                                                                                                             |
|-------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre:</b>                                                    | CU12 – Cerrar sesión.                                                                                                                       |
| <b>Descripción:</b>                                               | Permitir a un usuario autenticado cerrar su sesión actual, invalidando el token de acceso.                                                  |
| <b>Ref. Cruzadas:</b>                                             | CU-09 – Autenticación y autorización de usuarios.                                                                                           |
| <b>Actores:</b>                                                   | Usuario del sistema (Operador del centro de control, Administrador del sistema y Técnico en terreno).                                       |
| <b>Precondiciones:</b>                                            | – El usuario se encuentra autenticado con un token de sesión válido.                                                                        |
| <b>Postcondiciones:</b>                                           | – El token de sesión del usuario quedó inválido.<br>– El usuario debe volver a autenticarse para acceder a las funcionalidades del sistema. |
| <b>FLUJO NORMAL</b>                                               |                                                                                                                                             |
| <b>Acción del Usuario</b>                                         | <b>Respuesta del sistema</b>                                                                                                                |
| 1. El usuario envía una solicitud de cierre de sesión.            |                                                                                                                                             |
|                                                                   | 2. El sistema valida el token de sesión.                                                                                                    |
|                                                                   | 3. El sistema invalida la sesión o token asociado al usuario.                                                                               |
|                                                                   | 4. El sistema retorna una respuesta de éxito.                                                                                               |
| <b>FLUJO ALTERNATIVO</b>                                          |                                                                                                                                             |
| 1a. El usuario intenta cerrar sesión sin tener una sesión activa. | 4a. El sistema rechaza la solicitud e informa que no hay sesión activa. .                                                                   |

Fuente: Elaboración propia

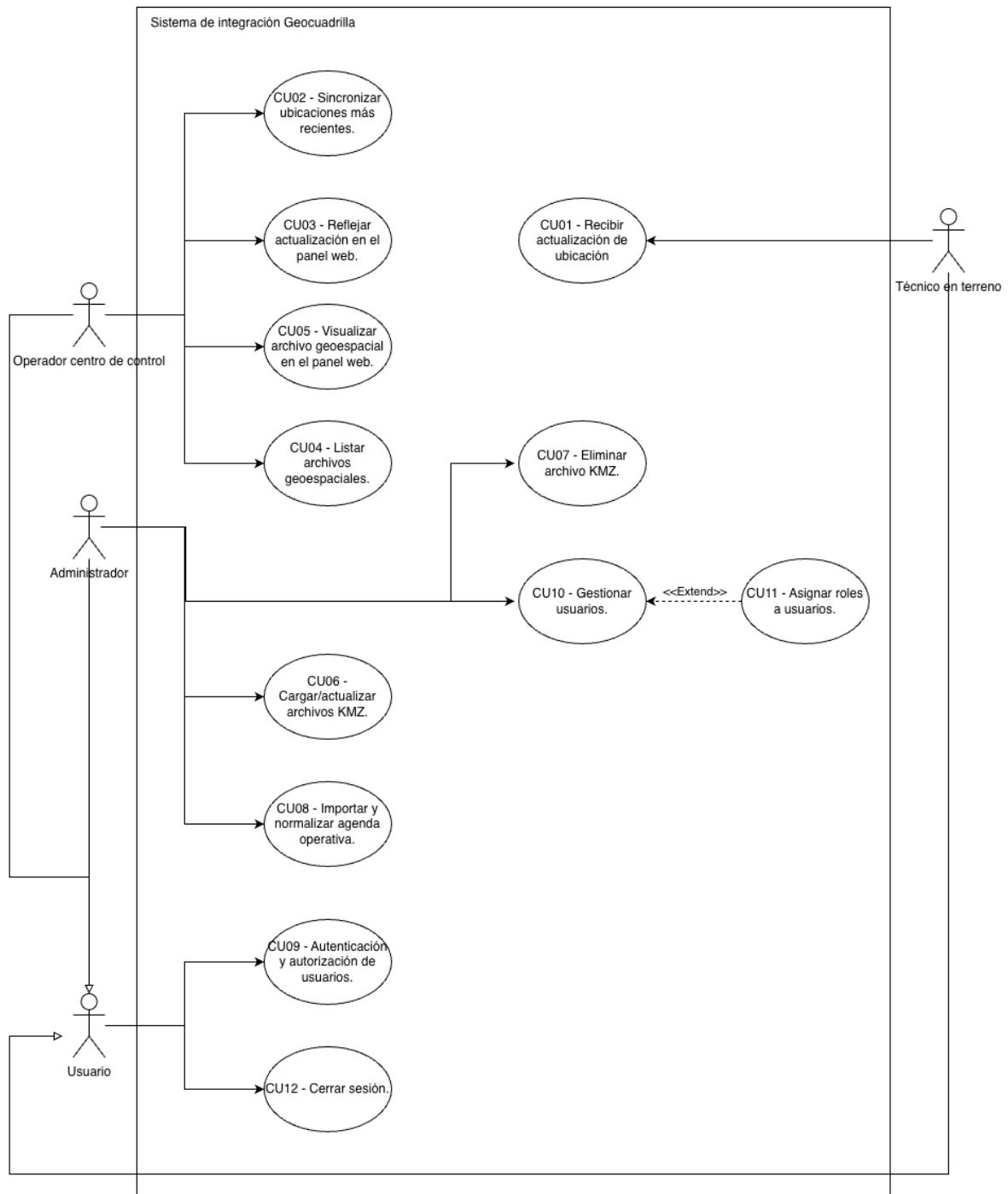


Figura 2. Casos de uso.  
Fuente: Elaboración propia.

## 3.5 Diseño del sistema

### 3.5.1 Arquitectura General

La solución se diseña como un sistema compuesto por una aplicación móvil, un *backend* de integración, un *bróker* de mensajería, servicios de almacenamiento y un panel web de monitoreo.

A diferencia de arquitecturas que separan estos roles en procesos independientes, el *backend* consolida en un único proceso todos los componentes responsables de la recepción, procesamiento y distribución de la información operativa.

La aplicación móvil actúa como emisora de eventos de ubicación, enviando periódicamente las coordenadas al *backend* mediante solicitudes HTTP/REST. El *backend* recibe estas actualizaciones de forma síncrona a través de su API REST y las persiste en una tabla de *outbox*, desacoplando la recepción del procesamiento posterior con el objetivo de minimizar la latencia en la confirmación enviada al dispositivo móvil. Un servicio publicador, ejecutándose dentro del mismo proceso, monitorea periódicamente la tabla de *outbox* y publica los mensajes pendientes hacia el *bróker* de mensajería RabbitMQ. Este mecanismo garantiza que ningún mensaje se pierda ante una falla transitoria, ya que los mensajes permanecen en la tabla de *outbox* hasta que su publicación es confirmada por el *bróker*.

Un servicio consumidor, también ejecutándose dentro del mismo proceso, recibe los mensajes desde la cola de RabbitMQ, ejecuta la lógica de procesamiento correspondiente, persiste la información en la base de datos relacional y notifica las actualizaciones al panel web del centro de control mediante SignalR, estableciendo así el canal de comunicación en tiempo real con los operadores.

Adicionalmente, el sistema dispone de un servicio de almacenamiento de objetos (MinIO) para la gestión de archivos geoespaciales en formato KMZ, los cuales son administrados y servidos mediante endpoints expuestos por la API REST del *backend*.

Si bien en el entorno de demostración actual todos los componentes — API REST, servicio publicador del *outbox* y servicio consumidor de RabbitMQ — se ejecutan dentro de un único proceso, la arquitectura fue diseñada considerando su eventual separación en procesos independientes para un entorno de producción. En particular, el servicio publicador del *outbox* puede extraerse como un proceso autónomo sin modificaciones al núcleo del sistema, dado que su única dependencia es la tabla de *outbox* en la base de datos relacional. Esta decisión de diseño permite escalar o redistribuir los componentes

según las necesidades operativas del patrocinador sin requerir cambios estructurales en el sistema.

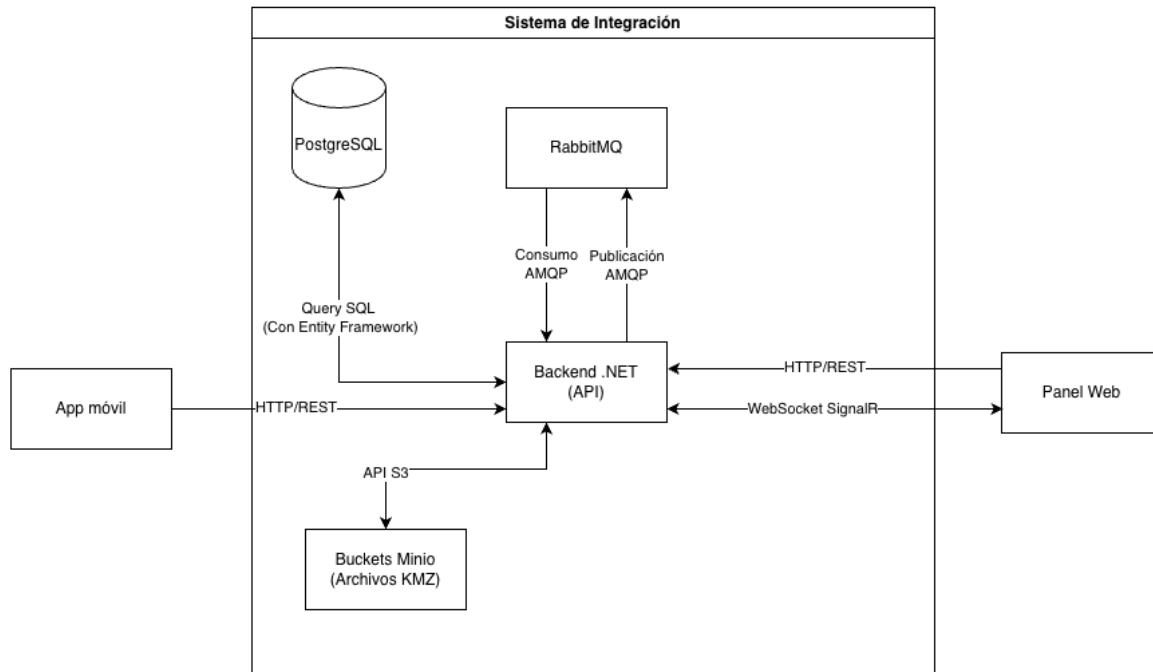


Figura 3. Arquitectura general.

Fuente: Elaboración propia.

### 3.5.2 Arquitectura lógica del *backend*

El *backend* se implementa siguiendo los principios de *Clean architecture* [6], adoptando un enfoque cercano a la Arquitectura Hexagonal (Puertos y Adaptadores). Bajo este paradigma, la lógica central del sistema se organiza en componentes con dependencias dirigidas hacia el núcleo (Dominio y Aplicación), mientras que los detalles tecnológicos (base de datos, mensajería, almacenamiento y *frameworks*) se implementan como adaptadores externos que dependen del núcleo.

La estructura lógica se compone de las siguientes componentes:

- **Dominio:** Contiene las entidades y reglas de negocio principales que modelan la información central del sistema. Estas entidades incorporan validaciones de consistencia para evitar estados inválidos y se mantienen independientes de los *frameworks* o tecnologías específicas.

- **Aplicación:** Implementa los casos de uso del sistema mediante servicios de aplicación. Define interfaces (puertos) que representan las operaciones requeridas desde el exterior, tales como persistencia, mensajería o almacenamiento, manteniendo la lógica de orquestación desacoplada de los detalles de la infraestructura.
- **Adaptadores de entrada:** Exponen la funcionalidad del sistema hacia los clientes externos. Incluyen los controladores *REST* y el *hub* de comunicación en tiempo real mediante la librería *SignalR (WebSockets)*, encargados de traducir las solicitudes externas a comandos u operaciones de la capa de aplicación.
- **Persistencia:** Implementación de repositorios y acceso a datos mediante el *framework ORM Entity Framework Core* sobre *PostgreSQL*.
- **Mensajería:** Integración con *RabbitMQ* para la publicación y consumo de eventos. Incluye un servicio *listener* encargado de consumir los mensajes desde la cola, invocar los casos de uso correspondientes a la capa de aplicación, persistir la información y notificar las actualizaciones al panel web mediante *SignalR*.
- **Almacenamiento de objetos:** Integración con el servicio *MinIO* mediante una API compatible con *AWS S3* para la gestión de archivos geoespaciales en formato *KMZ*.
- **Seguridad:** Mecanismos de infraestructura para autenticación, incluyendo la generación y validación de tokens *JWT* y la aplicación de políticas de acceso.

Este diseño favorece los atributos de calidad como la mantenibilidad y la testabilidad, mediante el aislamiento de la lógica del dominio y casos de uso de los detalles tecnológicos de la implementación, permitiendo reemplazar adaptadores o actualizar librerías con un impacto mínimo en el núcleo del negocio.

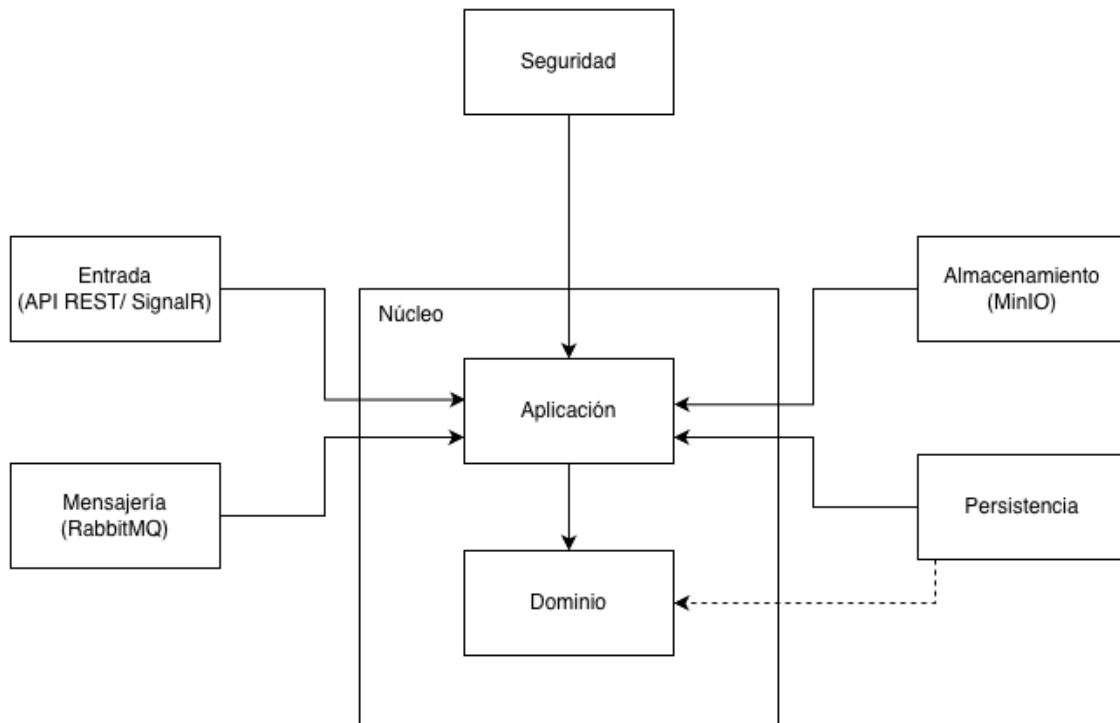


Figura 4. Arquitectura de lógica del backend.

Fuente: Elaboración propia.

### 3.5.3 Modelo lógico de datos

Para la persistencia de la información se diseñó un modelo de datos relacional normalizado, orientado a garantizar la integridad referencial y evitar la redundancia. El modelo se estructura en torno a tres ejes principales: la gestión de usuarios (entidades *User*), la operación en terreno (*Movil*, *LocationUpdate* y *AgendaEntry*) y la gestión de archivos KMZ (*FileMetadata*).

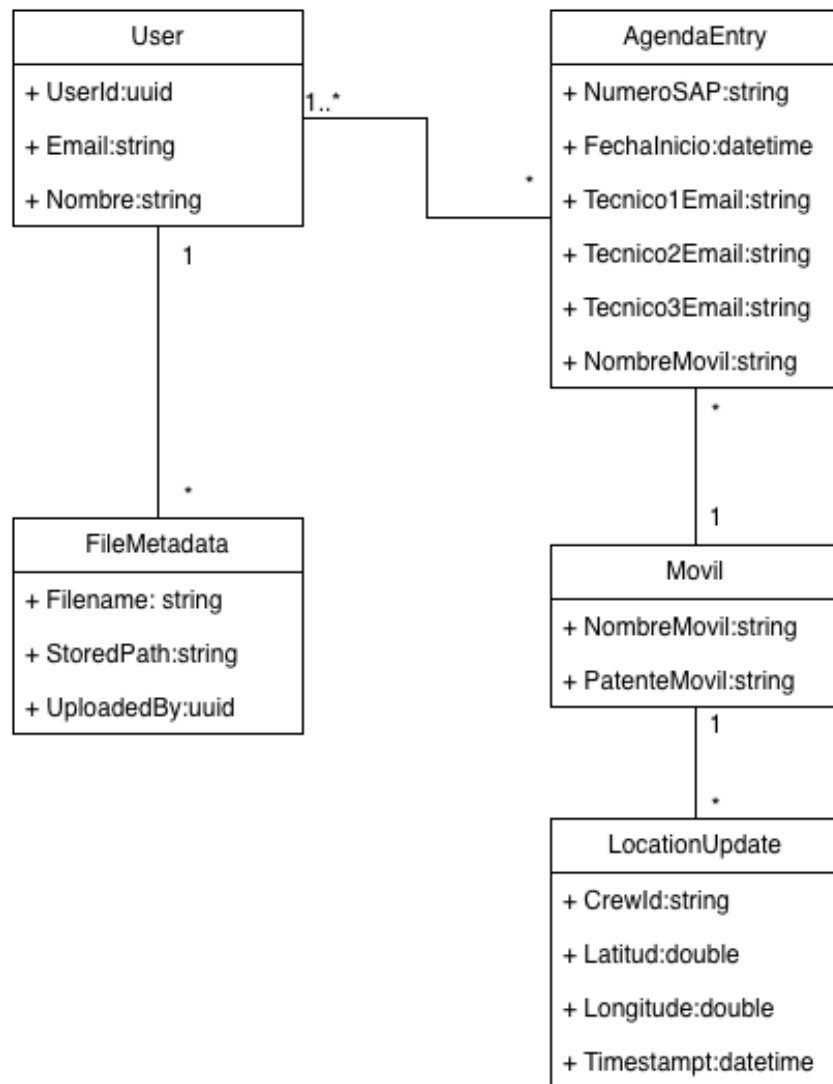


Figura 5. Modelo Lógico  
Fuente: Elaboración propia

### 3.5.4 Modelo de clases

El modelo de clases presentado en la Figura 3.5 representa las principales entidades del dominio del sistema y sus relaciones. El diagrama abstrae los detalles de infraestructura y persistencia, enfocándose en los conceptos centrales del negocio, tales como usuario, roles, móviles, agenda operativa, archivos geoespaciales y actualizaciones de ubicación.

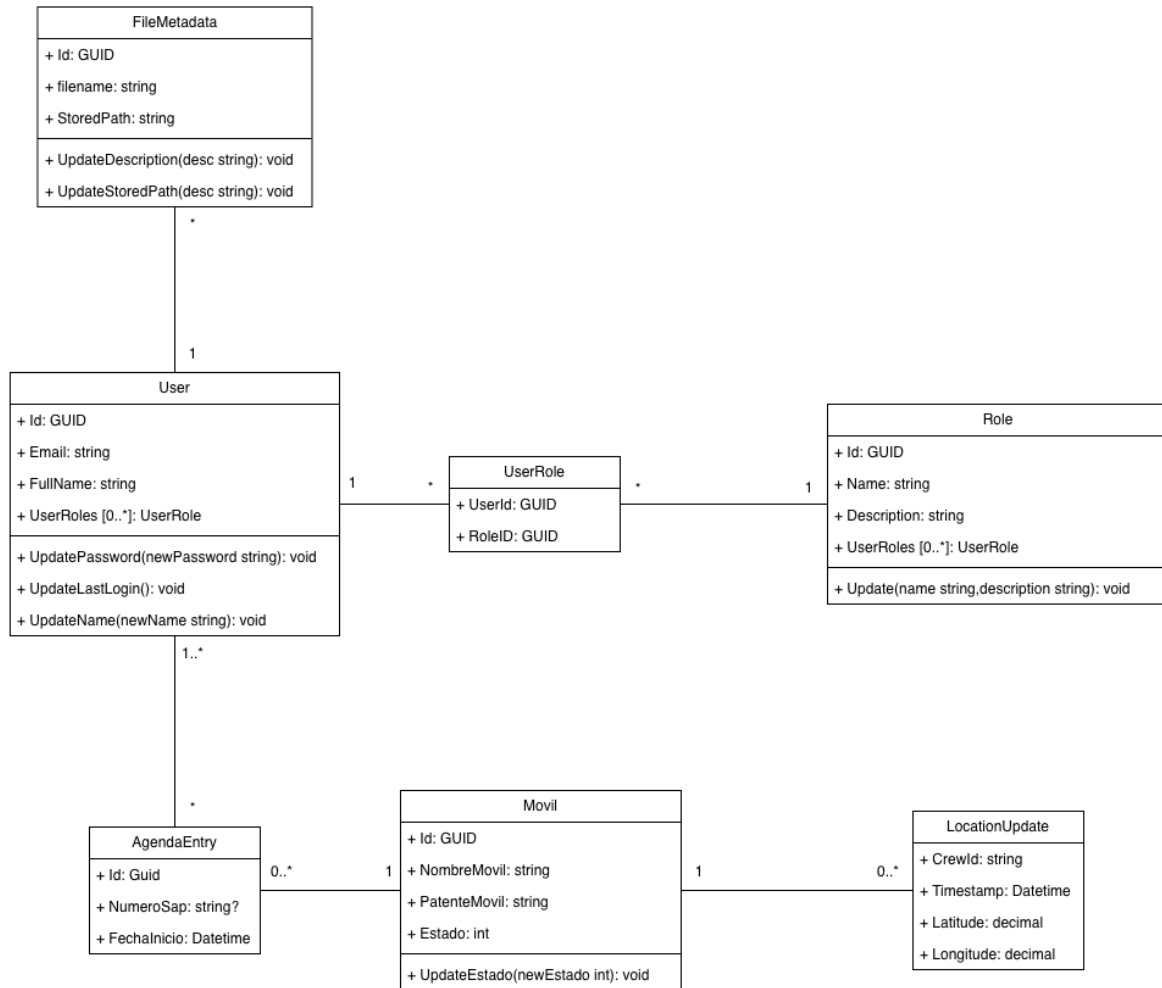


Figura 6. Diagrama de clases.

Fuente: Elaboración propia.

### 3.4.5 Flujo de datos de actualización de ubicación

El sistema contempla dos flujos de datos principales: la inicialización del panel web y el procesamiento de actualizaciones de ubicación. Ambos flujos se describen a continuación y se ilustran en los diagramas de secuencia de las Figuras **Error! Reference source not found.** y Figura 3.7 respectivamente.

#### **Flujo 1: Inicialización del Panel Web:**

Ya habiendo obtenido el token JWT, el cliente establece una conexión WebSocket con *LocationHub* en la ruta `/hubs/location`, momento en que el *hub* registra la conexión mediante *ConnectionManagementService*.

Una vez conectado, el Panel Web invoca `GetLatestLocations()` para obtener el estado actual de las cuadrillas. El *hub* delega esta operación a *LocationUpdateService*, que consulta las últimas ubicaciones registradas a través de *LocationRepository* y las enriquece con datos operativos consultando *AgendaRepository* por cada cuadrilla, obteniendo información como patente del móvil, nombre, tipo de actividad y técnicos asignados.

El *hub* filtra los resultados para retener únicamente las actualizaciones del día en curso y retorna el snapshot inicial al cliente.

Finalmente, el Panel Web invoca `SubscribeToCrews([crewIds])` para suscribirse a las actualizaciones en tiempo real de las cuadrillas de su interés. El *hub* agrega la conexión a los grupos correspondientes mediante *ConnectionManagementService* y confirma la suscripción al cliente. A partir de este momento, el Panel Web recibirá eventos `LocationUpdated` de forma automática cada vez que se procese una nueva actualización.

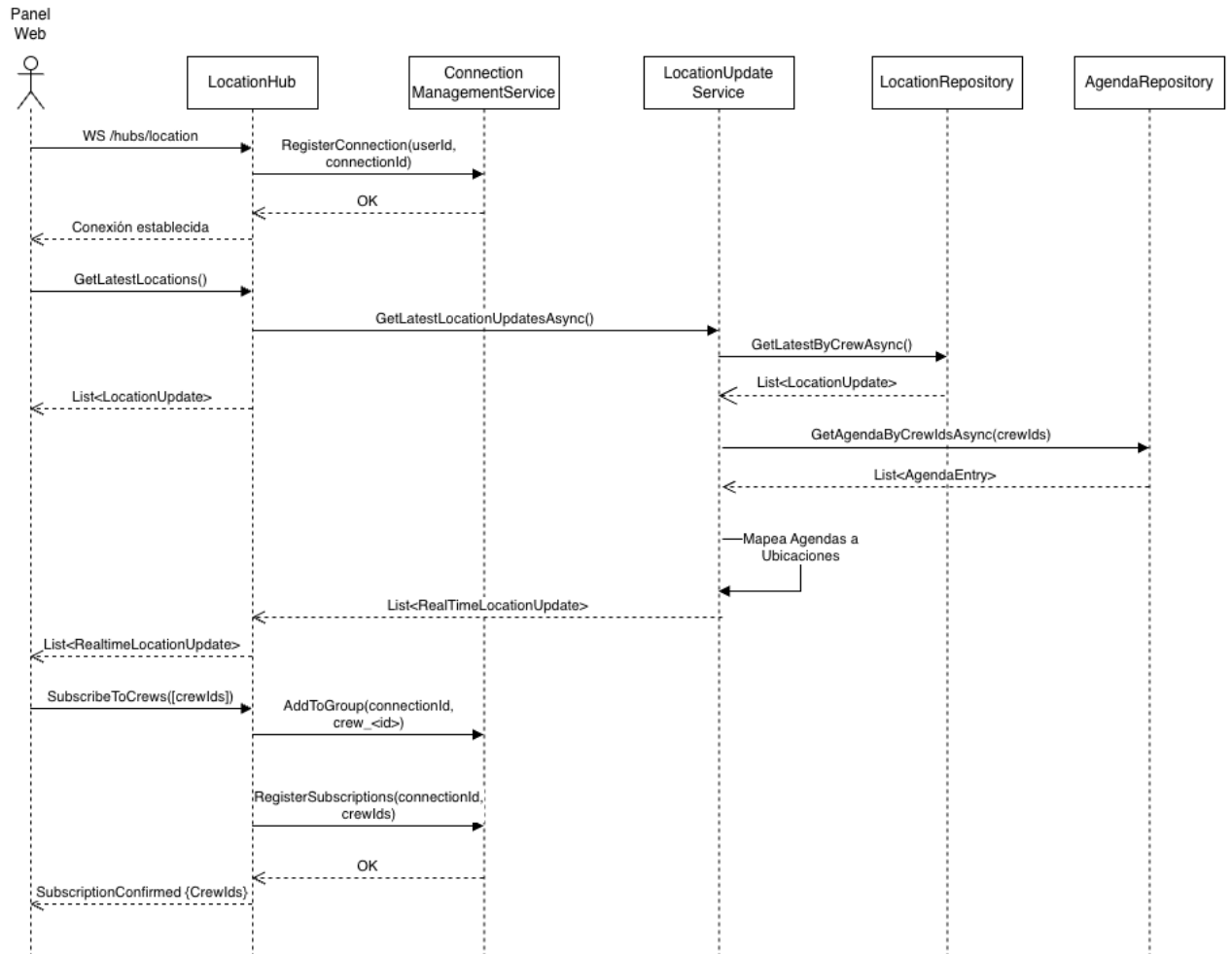


Figura 7. Diagrama de secuencia para subscripción a actualizaciones.

Fuente: Elaboración propia

**Flujo 2: Recepción y procesamiento de actualización de ubicación.**

El flujo se inicia cuando la aplicación móvil envía una actualización de ubicación mediante una solicitud HTTP POST al `LocationUpdatesController`. El controlador persiste el mensaje en la tabla de *outbox* a través de `LocationRepository` y retorna una confirmación HTTP 200 al dispositivo de forma inmediata, sin esperar a que el procesamiento se complete. De forma independiente, `OutboxPublisherService` ejecuta un ciclo de *polling* cada 250 milisegundos consultando los mensajes pendientes en la tabla de *outbox*. Por cada mensaje pendiente, publica el evento hacia `RabbitMqListenerService` mediante `BasicPublish`, marca el mensaje como procesado y persiste el cambio de estado en la base de datos.

El intervalo de 250 ms fue seleccionado considerando que la tabla *outbox* se vacía de forma progresiva a medida que los mensajes son publicados, por lo que las consultas operan sobre un conjunto acotado de registros pendientes, manteniendo el costo de cada ciclo bajo y compatible con el requisito de latencia definido en RNF-01.

Al recibir el mensaje, `RabbitMqListenerService` consulta la agenda operativa en `AgendaRepository` utilizando el email del técnico para resolver los datos de la cuadrilla asociada. Con la información enriquecida, persiste la ubicación en `LocationRepository` mediante `CreateAsync` y notifica la actualización al Panel Web a través de `LocationHub`, que emite el evento `LocationUpdated` a todos los clientes suscritos al grupo correspondiente. Finalmente, el listener envía el `BasicAck` a `RabbitMQ` confirmando el procesamiento exitoso del mensaje.

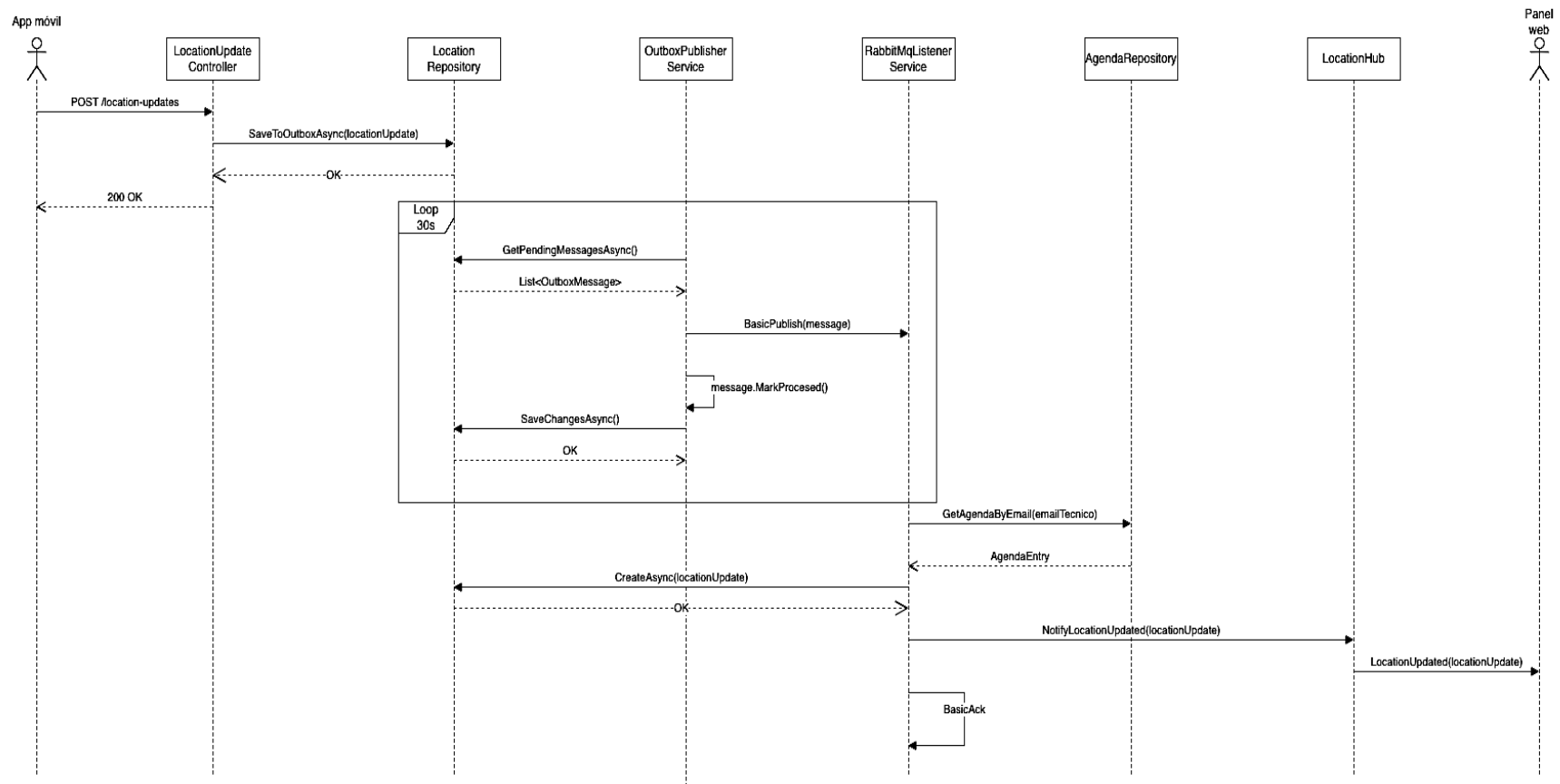


Figura 8. Diagrama de secuencia para la publicación de una actualización.  
Fuente: Elaboración propia

### 3.5.5 Diseño de despliegue

El sistema se despliega en un servidor *on-premise* administrado mediante Proxmox VE, plataforma de virtualización que permite alojar múltiples contenedores Linux (LXC) de forma aislada sobre el mismo hardware físico. Esta estrategia permite separar responsabilidades de red e infraestructura sin requerir múltiples servidores físicos.

El servidor contempla dos contenedores LXC. El primero, denominado LXC 1: *Ingress*, actúa como punto de entrada de red y aloja el daemon Cloudflared, responsable de mantener un túnel persistente hacia el servicio externo Cloudflare. Este túnel permite exponer el sistema a internet sin necesidad de abrir puertos en el firewall ni configurar IP pública, delegando la terminación TLS y el enrutamiento del tráfico externo a la infraestructura de Cloudflare.

El segundo contenedor, LXC 2: *Backend*, aloja toda la lógica del sistema mediante Docker Engine. Dentro de este entorno se despliegan cuatro contenedores Docker orquestados mediante Docker Compose: la API REST del *backend*, PostgreSQL como sistema de persistencia relacional, RabbitMQ como *bróker* de mensajería y MinIO como servicio de almacenamiento de objetos.

El uso de Docker Compose permite definir la configuración de todos los servicios en un único archivo declarativo, simplificando el despliegue y asegurando la reproducibilidad del entorno.

Los clientes externos acceden al sistema a través de Cloudflare mediante HTTPS y *WebSocket*. El tráfico es recibido por Cloudflare, reenviado por el túnel hacia Cloudflared en LXC 1, y desde allí dirigido a la API REST en LXC 2. La aplicación móvil se comunica exclusivamente mediante HTTP/REST, mientras que el navegador web del operador establece adicionalmente una conexión *WebSocket* persistente con el *hub* de *SignalR* para recibir actualizaciones en tiempo real.

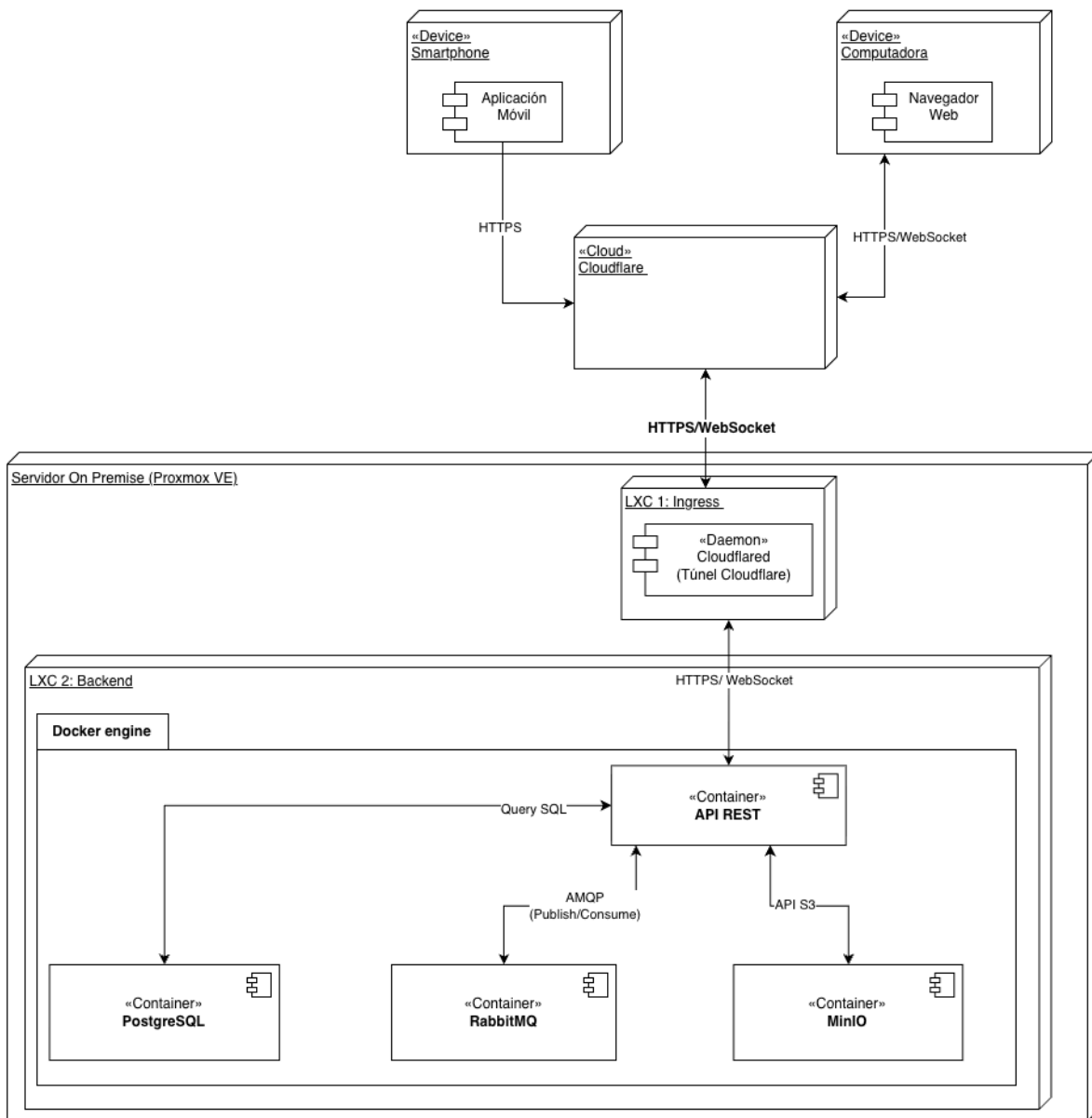


Figura 9. Diagrama de despliegue del sistema para demostración.

Fuente: Elaboración propia

### 3.6 Diseño de interfaces del sistema

El sistema de integraciones expone sus funcionalidades mediante *endpoints* REST implementados mediante HTTP, utilizando el formato JSON para el intercambio de datos.

Algunos endpoints requieren que el usuario se haya autenticado mediante el *endpoint* de *login* y posteriormente incluir el *token JWT* obtenido mediante el encabezado `Authorization: Bearer <token>`.

Esta **sección** abarca la **definición** de los *endpoints* del sistema, los detalles de implementación serán abordados en el siguiente capítulo.

#### 3.6.1 Catálogo de Endpoints

La siguiente tabla presenta los principales *endpoints* del *backend* y sus funciones.

Tabla 13. Catálogo de endpoints.

| Método | Ruta                  | Descripción                                          | Autenticación |
|--------|-----------------------|------------------------------------------------------|---------------|
| POST   | /api/location-updates | Registrar actualización de ubicación                 | Sí            |
| POST   | /api/Auth/login       | Autenticación de usuario                             | No            |
| POST   | /api/Auth/logout      | Cierra la sesión actual del usuario revocando token. | Sí            |
| POST   | /api/Auth/refresh     | Renovación de token                                  | Sí            |
| GET    | /api/Auth/me          | Obtener información del usuario autenticado          | Sí            |
| POST   | /api/Auth/register    | Registra un usuario                                  | Sí            |
| POST   | /api/Files/upload     | Cargar archivo KMZ                                   | Sí            |
| GET    | /api/Files            | Listar archivos KMZ                                  | Sí            |
| PUT    | /api/Files/{id}       | Actualizar metadatos del archivo                     | Sí            |
| DELETE | /api/Files/{id}       | Borra el archivo indicado                            | Sí            |
| GET    | /api/Files/{id}       | Descargar archivo KMZ                                | Sí            |
| GET    | /api/Users            | Listar usuarios                                      | Sí            |
| GET    | /api/Users/{id}       | Detalle del usuario                                  | Sí            |
| PUT    | /api/Users/{id}/roles | Asigna roles al usuario                              | Sí            |

Fuente: Elaboración propia.

### 3.6.2 Especificación detallada de endpoints críticos

Tabla 14. Especificación Autenticación de usuario.

|                           |                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre                    | Autenticación de usuario                                                                                                                                                                                                                                                                                                                                          |
| Método HTTP               | POST                                                                                                                                                                                                                                                                                                                                                              |
| URL                       | /api/Auth/login                                                                                                                                                                                                                                                                                                                                                   |
| Autenticación requerida   | No                                                                                                                                                                                                                                                                                                                                                                |
| Descripción               | Permite a un usuario autenticarse en el sistema mediante credenciales de acceso, obteniendo un token JWT para su uso en solicitudes posteriores. El endpoint aplica un límite de 5 intentos por minuto por IP para prevenir ataques de fuerza bruta.                                                                                                              |
| Ejemplo solicitud         | <pre>{   "userName": "operador@empresa.cl",   "password": "Password123!",   "rememberMe": false }</pre>                                                                                                                                                                                                                                                           |
| Campos solicitud          | <ul style="list-style-type: none"> <li>- userName: Correo electrónico del usuario.</li> <li>- password: Contraseña del usuario.</li> <li>- rememberMe: Indica si se debe extender la duración del token.</li> </ul>                                                                                                                                               |
| Ejemplo respuesta         | <p>Código HTTP: 200 OK</p> <pre>{   "isSuccess": true,   "message": "Login successful",   "data": {     "accessToken": "eyJhbGciOi...",     "refreshToken": "f4d8f1...",     "expiresAt": "2026-02-16T15:40:00Z",     "user": {       "userName": "operador",       "email": "operador@empresa.cl",       "roles": ["Operador"]     }   },   "errors": [] }</pre> |
| Posibles códigos de error | <ul style="list-style-type: none"> <li>- 400 Datos inválidos.</li> <li>- 401 Credenciales incorrectas.</li> <li>- 423 Cuenta bloqueada.</li> <li>- 429 Demasiados intentos de autenticación.</li> </ul>                                                                                                                                                           |

Fuente: Elaboración propia.

Tabla 15. Especificación Carga archivo KMZ.

|                           |                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre                    | Carga de archivo geoespacial                                                                                                                                                                                                                                                                                                                                                             |
| Método HTTP               | POST                                                                                                                                                                                                                                                                                                                                                                                     |
| URL                       | /api/Files/upload                                                                                                                                                                                                                                                                                                                                                                        |
| Autenticación requerida   | Si                                                                                                                                                                                                                                                                                                                                                                                       |
| Descripción               | Permite a un usuario autorizado cargar un archivo geoespacial en formato KMZ al sistema. El archivo es almacenado en el repositorio de objetos MinIO y sus metadatos quedan registrados en la base de datos relacional.                                                                                                                                                                  |
| Ejemplo solicitud         | Content-Type: multipart/form-data<br>- file: archivo binario (.kmz o .kml)<br>- description: "Líneas de alta tensión zona norte" (opcional, máx. 1000 caracteres)                                                                                                                                                                                                                        |
| Ejemplo respuesta         | Código HTTP: 201 Created<br><pre>{   "isSuccess": true,   "message": "File uploaded successfully",   "data": {     "id": "9d71ad37-...",     "fileName": "lineas_220kv.kmz",     "contentType": "application/vnd.google-earth.kmz",     "fileSize": 348221,     "uploadedAt": "2026-02-16T13:00:00Z",     "description": "Líneas de alta tensión zona norte"   },   "errors": [] }</pre> |
| Posibles códigos de error | <ul style="list-style-type: none"> <li>- 400 Formato de archivo no soportado o archivo vacío.</li> <li>- 401 Usuario no autenticado.</li> <li>- 403 Permisos insuficientes.</li> <li>- 413 Archivo demasiado grande.</li> </ul>                                                                                                                                                          |

Fuente: Elaboración propia.

Tabla 16. Especificación Listado de archivos KMZ.

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre                    | Listar archivos KMZ                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Método HTTP               | GET                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| URL                       | /api/Files                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Autenticación requerida   | Si                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Descripción               | Retorna el listado paginado de archivos geoespaciales disponibles en el sistema, incluyendo sus metadatos. El panel web consume este endpoint para construir el listado de capas disponibles para su visualización                                                                                                                                                                                                                              |
| Parámetros de consulta    | -pageNumber: Número de página (por defecto 1).<br>-pageSize: Cantidad de resultados por página (por defecto 20, máximo 100).<br>- contentTypeFilter: Filtro opcional por tipo de contenido.                                                                                                                                                                                                                                                     |
| Ejemplo de respuesta      | Código HTTP: 200 OK<br><pre>{ "isSuccess": true,   "data":     { "items":       [ { "id": "9d71ad37-...",         "fileName": "lineas_220kv.kmz",         "fileSize": 348221,         "uploadedAt": "2026-02-16T13:00:00Z",         "uploadedByUsername": "admin",         "description": "Líneas de alta tensión zona norte" }       ],       "currentPage": 1,       "totalPages": 1,       "totalCount": 1 },       "errors": []     }</pre> |
| Posibles códigos de error | - 400 Parámetros de consulta inválidos.<br>- 401 Usuario no autenticado.<br>- 403 Permisos insuficientes.                                                                                                                                                                                                                                                                                                                                       |

Fuente: Elaboración propia.

Tabla 17. Especificación Actualización de ubicación.

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre                    | Actualización de ubicación                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Método HTTP               | POST                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| URL                       | /api/location-updates                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Autenticación requerida   | Si                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Descripción               | Permite a la aplicación móvil enviar actualizaciones periódicas de ubicación asociadas a un técnico en terreno.                                                                                                                                                                                                                                                                                                                                                      |
| Ejemplo solicitud         | <pre>{   "emailTecnico": "tecnico@empresa.cl",   "timestamp": "2026-02-16T14:30:00Z",   "latitude": -33.4569,   "longitude": -70.6483,   "metadata": {     "accuracyMeters": 9,     "speedKmh": 38   },   "source": "mobile-app" }</pre>                                                                                                                                                                                                                             |
| Campos solicitud          | <ul style="list-style-type: none"> <li>- emailTecnico: Identificador del técnico asociado.</li> <li>- timestamp: Fecha y hora en formato ISO 8601.</li> <li>- latitude: Latitud en grados decimales.</li> <li>- longitude: Longitud en grados decimales.</li> <li>- metadata: información adicional opcional incluye precisión de la ubicación y la velocidad del vehículo al tomar la ubicación.</li> <li>- source: Identificador del origen del mensaje</li> </ul> |
| Ejemplo respuesta         | <p>Código HTTP: 200 OK</p> <pre>{   "isSuccess": true,   "message": "Location update accepted.",   "data": null,   "errors": [] }</pre>                                                                                                                                                                                                                                                                                                                              |
| Posibles códigos de error | <ul style="list-style-type: none"> <li>- 400 Datos invalidos.</li> <li>- 401 Usuario no autenticado.</li> <li>- 503 Sistema no disponible</li> </ul>                                                                                                                                                                                                                                                                                                                 |

Fuente: Elaboración propia.

### 3.6.3 Canal de comunicación en tiempo real (SignalR)

La comunicación en tiempo real entre el sistema de integración y el panel web se realiza mediante un *hub* de SignalR expuesto en la ruta */hubs/location*. A diferencia de los endpoints REST, este canal establece una conexión persistente basada en *WebSockets*, permitiendo que el servidor notifique activamente a los clientes conectados ante nuevos eventos sin necesidad de sondeo.

La conexión al *hub* requiere autenticación mediante token JWT, el cual puede ser enviado como parámetro de consulta *access\_token* en la URL de conexión, dado que los WebSockets no permiten el uso de encabezados HTTP estándar.

Tabla 18. Métodos del cliente hacia el servidor.

| Método               | Parámetros        | Descripción                                                             |
|----------------------|-------------------|-------------------------------------------------------------------------|
| SubscribeToCrews     | crewIds: string[] | Suscribe al cliente a las actualizaciones de las cuadrillas indicadas.  |
| UnsubscribeFromCrews | crewIds: string[] | Cancela la suscripción a las cuadrillas indicadas.                      |
| GetLatestLocations   |                   | Solicita las últimas ubicaciones conocidas de las cuadrillas suscritas. |

Fuente: Elaboración propia.

Tabla 19. Eventos del servidor hacia el cliente.

| Evento                  | Descripción                                                                                                        |
|-------------------------|--------------------------------------------------------------------------------------------------------------------|
| LocationUpdated         | Notifica una nueva actualización de ubicación procesada, incluyendo coordenadas y agenda asociada de la cuadrilla. |
| SubscriptionsConfirmed  | Confirma que la suscripción a las cuadrillas solicitadas fue registrada correctamente.                             |
| UnsubscriptionConfirmed | Confirma la cancelación de la suscripción.                                                                         |

Fuente: Elaboración propia.

## 4 Implementación del Sistema

### 4.1 Tecnologías utilizadas

El sistema de integración fue implementado utilizando *.NET 8* como plataforma principal de desarrollo del *backend*, aprovechando su soporte para aplicaciones de alto rendimiento, concurrencia y comunicación en tiempo real.

Para la persistencia de datos estructurados se utilizó *PostgreSQL*, accedido mediante *Entity Framework Core*, mientras que el almacenamiento de archivos geoespaciales en formato KMZ se realizó mediante *MinIO*, un sistema de almacenamiento de objetos compatible con la API S3.

La mensajería asíncrona se implementó utilizando *RabbitMQ*, el cual actúa como intermediario entre la aplicación móvil y el sistema de integración. Para la comunicación en tiempo real con el panel web se utilizó *SignalR*, permitiendo la difusión de actualizaciones sin necesidad de sondeo por parte del cliente.

El sistema fue desplegado utilizando *Docker Compose*, lo que permitió orquestar todos los servicios necesarios en un entorno controlado y reproducible.

### 4.2 Implementación del *backend* de integración

El *backend* del sistema corresponde a una aplicación ASP.NET Core estructurada en capas, de acuerdo con la arquitectura descrita en el Capítulo 3. La implementación se centra en coordinar la recepción de eventos de ubicación, su persistencia y su difusión hacia los clientes conectados.

La capa de Presentación expone endpoints REST para las funcionalidades administrativas y un *hub* de *SignalR* para la comunicación en tiempo real. Esta capa no contiene lógica de negocio, limitándose a delegar la ejecución de las operaciones a la capa de Aplicación.

### 4.3 Procesamiento de actualizaciones de ubicación

La recepción de actualizaciones de ubicación se implementa en el controlador *LocationUpdatesController*, el cual expone el *endpoint* `POST /api/location-updates`. Este controlador aplica dos mecanismos antes de procesar la solicitud: la validación de autorización mediante token JWT y el control de concurrencia mediante un semáforo (`_writeLimiter`), cuyo propósito es limitar la cantidad de solicitudes simultáneas que

acceden al controlador. Cuando se alcanza el límite configurado, las solicitudes entrantes quedan en espera hasta que una conexión quede disponible, evitando así que picos de tráfico saturen la base de datos y se produzca pérdida de información.

Una vez superados estos controles, el modelo de entrada ejecuta su propia validación mediante el método `Validate()`, que verifica las siguientes condiciones:

- *emailTecnico* es requerido, no puede superar los 320 caracteres y debe tener formato de correo electrónico válido.
- *timestamp* es requerido y no puede corresponder a un momento superior a una hora en el futuro respecto a la hora UTC actual.
- *latitude* debe estar en el rango  $[-90, 90]$ .
- *longitude* debe estar en el rango  $[-180, 180]$ .

Si alguna validación falla, el sistema retorna una respuesta 400 Bad Request con el detalle de los errores. En caso contrario, el controlador delega la persistencia al servicio `_locationUpdateOutboxService`, el cual registra el mensaje en una tabla dedicada de la base de datos que actúa de manera similar al patrón *outbox*. Este patrón desacopla la recepción del mensaje de su procesamiento posterior, permitiendo que el controlador confirme la recepción al cliente de forma inmediata sin esperar a que el mensaje sea procesado y enviado al panel web.

```
/// <summary>
/// Receives location updates from mobile clients and persists them in outbox for async RabbitMQ publishing.
/// </summary>
[HttpPost( template: "/api/location-updates")]
[Authorize]
[ProducesResponseType(typeof(BaseResponse<object>)), statusCode: StatusCodes.Status200OK]
[ProducesResponseType(typeof(BaseResponse<object>)), statusCode: StatusCodes.Status400BadRequest]
[ProducesResponseType(typeof(BaseResponse<object>)), statusCode: StatusCodes.Status401Unauthorized]
[ProducesResponseType(typeof(BaseResponse<object>)), statusCode: StatusCodes.Status500InternalServerError]]
@ roberto *
public async Task<ActionResult<BaseResponse<object>>> PublishAsync([FromBody] LocationUpdateMessage message)
{
    if (!ModelState.IsValid)
    {
        return ValidationErrors( errors: ModelState.ToDictionary(
            entry :KeyValuePair<string,ModelStateEntry?> => entry.Key,
            entry :KeyValuePair<string,ModelStateEntry?> => entry.Value?.Errors.Select(error => error.ErrorMessage).ToArray() ?? Array.Empty<string>());
    }

    var validationErrors :List<string> = message.Validate();
    if (validationErrors.Count > 0)
    {
        return BadRequest( error: BaseResponse<object>.Failure(validationErrors));
    }

    var acquired :bool = await _writeLimiter.TryAcquireAsync(HttpContext.RequestAborted);
    if (!acquired)
    {
        return BadRequest( StatusCodes.Status503ServiceUnavailable,
            BaseResponse<object>.Failure("System busy. Please retry shortly."));
    }

    try
    {
        var payload :string = JsonConvert.SerializeObject(message);
        var enqueueResult = await _locationUpdateOutboxService.EnqueueLocationUpdateAsync(payload, HttpContext.RequestAborted);
        if (!enqueueResult.IsSuccess)
        {
            var errorMessage = enqueueResult.ErrorMessage ?? "Failed to store location update.";
            _logger.LogError("Outbox enqueue failed for {EmailTecnico}: {ErrorMessage}", message.EmailTecnico, errorMessage);
            return BadRequest( StatusCodes.Status500InternalServerError, BaseResponse<object>.Failure(errorMessage));
        }
    }
    finally
    {
        _writeLimiter.Release();
    }

    _logger.LogInformation("Stored location update in outbox for {EmailTecnico} at {Timestamp}",
        message.EmailTecnico, message.Timestamp);

    return Success( message: "Location update accepted.");
}
```

Figura 10. LocationUpdatesController.cs PublishAsync()

Fuente: Elaboración propia

En *LocationUpdateOutboxService.cs* se implementa la escritura del mensaje en el buzón de salida en la base de datos. Mediante este proceso podemos minimizar el tiempo de interacción entre la aplicación móvil y el servidor.

```
0+1 usages
public async Task<ServiceResult> EnqueueLocationUpdateAsync(string payload, CancellationToken cancellationToken = default)
{
    try
    {
        if (string.IsNullOrEmpty(payload))
        {
            return ServiceResult.Failure("Location update payload cannot be empty.");
        }

        var outboxMessage = new OutboxMessage(LocationUpdateMessageType, payload);
        _outboxRepository.Add(outboxMessage);
        await _unitOfWork.SaveChangesAsync(cancellationToken);

        return ServiceResult.Success();
    }
    catch (OperationCanceledException) when (cancellationToken.IsCancellationRequested)
    {
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Failed to enqueue location update payload into outbox.");
        return ServiceResult.Failure(errorMessage: "Failed to store location update in outbox.", ex);
    }
}
```

Figura 11. LocationUpdateOutboxService.cs EnqueueLocationUpdateAsync()

Fuente: Elaboración propia

El siguiente paso del flujo de recepción es tomar estos mensajes salientes de la base de datos y publicarlos en la cola de mensajes en RabbitMQ, este paso está a cargo de un servicio llamado *OutboxLocationUpdatePublisher*.

```
protected override async Task ExecuteAsync(CancellationToken stoppingToken)
{
    while (!stoppingToken.IsCancellationRequested)
    {
        try
        {
            using var scope = _serviceProvider.CreateScope();
            var dbContext = scope.ServiceProvider.GetRequiredService<GeocudrillaDbContext>();
            var publisher = scope.ServiceProvider.GetRequiredService<ILocationUpdateQueuePublisher>();

            var pending :List<OutboxMessage> = await dbContext.OutboxMessages // DbSet<OutboxMessage>
                .Where(x :OutboxMessage => x.ProcessedAt == null && x.Attempts < _maxAttempts) // IQueryable<OutboxMessage>
                .OrderBy(x :OutboxMessage => x.CreatedAt) // IOrderedQueryable<OutboxMessage>
                .Take(_batchSize) // IQueryable<OutboxMessage>
                .ToListAsync(stoppingToken); // Task<List<...>>

            if (pending.Count == 0)
            {
                await Task.Delay(_pollInterval, stoppingToken);
                continue;
            }

            foreach (var message in pending)
            {
                try
                {
                    await publisher.PublishRawAsync(message.Payload, stoppingToken);
                    message.MarkProcessed();
                }
                catch (Exception ex)
                {
                    message.MarkFailed(ex.Message);
                    _logger.LogWarning(ex, message: "Failed to publish outbox message {OutboxId}", message.Id);
                }
            }

            await dbContext.SaveChangesAsync(stoppingToken);
        }
        catch (OperationCanceledException) when (stoppingToken.IsCancellationRequested)
        {
            // Service shutdown requested, exit gracefully
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, message: "Unexpected error while processing outbox messages.");
            await Task.Delay(_pollInterval, stoppingToken);
        }
    }
}
```

Figura 12. OutboxLocationUpdatePublisher.cs

Fuente: Elaboración propia

El proceso descrito hasta este punto está pensado para ser ejecutado en un nodo separado del *backend* principal, de esta manera la recepción de las coordenadas y el procesamiento de estas está desacoplada y nos permite soportar un alto *throughput* de mensajes y minimizar el tiempo de respuesta para la aplicación móvil sin perder información.

A continuación, el servicio *RabbitMqListenerService* se encarga de consumir los mensajes de la cola y procesarlos para continuar el flujo.

Con el método *SetUpConsumer()* definimos un proceso que consumirá mensajes para una cola definida.

```
/// <summary>
/// Sets up a consumer for a specific queue with error handling.
/// </summary>
[1 usage] [2 Roberto Romero]
private void SetUpConsumer(string queueName, Func<string, ulong, Task> handleAction)
{
    if (_channel == null)
    {
        _logger.LogError("Cannot set up consumer: channel is null");
        return;
    }

    var consumer = new EventingBasicConsumer(_channel);

    consumer.Received += async (model :object?, eventArgs) =>
    {
        var content :string = Encoding.UTF8.GetString( bytes: eventArgs.Body.ToArray());
        var deliveryTag :ulong = eventArgs.DeliveryTag;

        _logger.LogDebug("Received message from {Queue}: {Content}", queueName, content);

        try
        {
            await handleAction(content, deliveryTag);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, message: "Error processing message from {Queue}: {Message}", queueName, ex.Message);

            // Send to dead letter queue and NACK the message
            await PublishToDeadLetterQueue(queueName, eventArgs.Body.ToArray(), error: ex.Message);
            _channel.BasicNack(deliveryTag, multiple: false, requeue: false);
        }
    };

    // Start consuming messages
    _channel.BasicConsume(queue: queueName, autoAck: false, consumer: consumer);

    _logger.LogInformation("Started consuming messages from queue: {Queue}", queueName);
}
```

Figura 13. RabbitMqListenerService.cs SetupConsumer()

Fuente: Elaboración propia

*HandleLocationUpdateMessage()* es el encargado de procesar los mensajes consumidos de la cola de actualizaciones. Es en este proceso en donde realizamos un cruce entre la agenda de trabajo y la información del técnico que publicó la actualización, de esta manera podemos informar el estado actual de trabajo de la cuadrilla.

```
/// <summary>
/// Handles incoming location update messages.
/// </summary>
[1] usage  [2] Roberto Romero +1
private async Task HandleLocationUpdateMessage(string content, ulong deliveryTag)
{
    try
    {
        _logger.LogInformation("Processing location update message: {Content}", content);

        // Deserialize the message
        var locationMessage = JsonConvert.DeserializeObject<LocationUpdateMessage>(content);

        if (locationMessage == null){...}

        // Validate the message
        var validationErrors :List<string> = locationMessage.Validate();
        if (validationErrors.Count > 0){...}

        // Use a service scope to resolve dependencies
        using var scope = _serviceProvider.CreateScope();
        var serviceProvider = scope.ServiceProvider;
        var locationUpdateService = serviceProvider.GetRequiredService<ILocationUpdateService>();
        var agendaEntryRepository = serviceProvider.GetRequiredService<IAgendaEntryRepository>();
        var agendaEntry = await agendaEntryRepository.GetLatestByTechnicianEmailAsync(locationMessage.EmailTecnico, CancellationToken.None);

        if (agendaEntry == null || string.IsNullOrEmpty(agendaEntry.NombreMovil)){...}

        var crewId :string = agendaEntry.NombreMovil.Trim();
        var realtimeLocationUpdate = CreateRealtimeLocationUpdate(crewId, locationMessage, agendaEntry);
        _logger.LogInformation("Resolved technician {EmailTecnico} to crew {CrewId}",
            locationMessage.EmailTecnico, crewId);

        // Create the DTO for the service
        var createLocationUpdateDto = new CreateLocationUpdateDto
        {...};

        // Process the location update
        var result = await locationUpdateService.CreateAsync(createLocationUpdateDto);

        if (result.IsSuccess)
        {
            _logger.LogInformation("Location update processed successfully for crew {CrewId} at {Timestamp}",
                crewId, locationMessage.Timestamp);

            // Broadcast real-time location update to connected clients
            await BroadcastLocationUpdate(realtimeLocationUpdate);

            // Acknowledge the message after successful processing
            _channel?.BasicAck(deliveryTag, multiple: false);
        }
        else{...}
    }
    catch (JsonException ex){...}
    catch (Exception ex){...}
}
```

Figura 14. RabbitMqListenerService.cs HandleLocationUpdateMessage()

Fuente: Elaboración propia

Con *locationUpdateService.CreateAsync()* realizamos la escritura de esta recepción en la base de datos principal del *backend*. Para finalizar este proceso, hacemos la publicación de la actualización de ubicación en el *Hub SignalR* con *BroadcastLocationUpdate()* y realizamos la confirmación de recepción del mensaje de la cola con *\_channel?.BasicAck()* para que el mensaje consumido sea eliminado.

Por último, la llamada a *BroadcastLocationUpdate()* envía la actualización al panel web mediante *WebSockets* con la librería *SignalR*.

```
/// <summary>
/// Broadcasts a location update to SignalR clients subscribed to the crew's group.
/// Uses thread-safe broadcasting to prevent race conditions during concurrent message processing.
/// </summary>
[2 usages] [Roberto Romero +1]
private async Task BroadcastLocationUpdate(RealtimeLocationUpdate realtimeLocationUpdate)
{
    try
    {
        // Broadcast to clients subscribed to this crew's group
        // The SignalR hub context is thread-safe by design, but we ensure proper error handling
        await _hubContext.Clients.Group($"crew_{realtimeLocationUpdate.CrewId}") // IClientProxy
            .SendAsync(method: "LocationUpdated", realtimeLocationUpdate); // Task

        _logger.LogDebug("Broadcasted real-time location update for crew {CrewId} at {Timestamp}",
            realtimeLocationUpdate.CrewId, realtimeLocationUpdate.Timestamp);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Error broadcasting location update for crew {CrewId}: {Message}",
            realtimeLocationUpdate.CrewId, ex.Message);

        // Don't let SignalR broadcasting errors break the database processing
        // The location data is still saved even if the real-time broadcast fails
    }
}
```

Figura 4.6 RabbitMqListenerService.cs BroadcastLocationUpdate()

#### 4.4 Persistencia de datos

La persistencia de datos estructurados se implementó mediante *Entity Framework Core*, siguiendo el patrón de repositorios.

Las actualizaciones de ubicación se almacenan como registros históricos, manteniendo un historial completo que permite su consulta posterior. No se implementaron operaciones de actualización ni eliminación sobre estos registros, garantizando la integridad e inmutabilidad de la información histórica.

No se definieron índices adicionales sobre la tabla de actualizaciones de ubicación, dado que el enfoque principal del sistema es la ingesta y difusión en tiempo real, y no la explotación analítica intensiva de los datos.

## 4.5 Gestión de archivos geoespaciales

Los archivos geoespaciales en formato KMZ se gestionan mediante un sistema CRUD expuesto por el *backend*. Los archivos se almacenan físicamente en un *bucket* único de MinIO, mientras que sus metadatos (nombre, tipo, tamaño, ruta de almacenamiento y usuario asociado) se almacenan en la base de datos relacional.

El *backend* actúa como intermediario exclusivo para el acceso a los archivos, evitando que los clientes interactúen directamente con el sistema de almacenamiento de objetos. Esta estrategia permite centralizar el control de acceso y la validación de las operaciones.

Después de pasar por una validación básica en el controlador del endpoint *Files*, las funciones para administrar los archivos KMZ son manejadas en *FileService.cs*. Comenzando por el proceso de subir un archivo, luego de recibir la solicitud se intenta subir este archivo al servicio *MinIO* mediante *\_fileStorageService.UploadFileAsync()*, si este proceso es exitoso se procede a guardar los metadatos de este nuevo archivo en la base de datos.

```
public class FileService : IFileService
{
    public async Task<ServiceResult<FileMetadataDto>> UploadFileAsync(
        try
        {
            return await _unitOfWork.ExecuteInTransactionAsync( operation: async () =>
            {
                // Upload file to storage
                var uploadResult = await _fileStorageService.UploadFileAsync(
                    fileStream, fileMetadata.FileName, fileMetadata.ContentType, cancellationToken); // Task<ServiceResult<...>>

                if (!uploadResult.IsSuccess)
                    return ServiceResult<FileMetadataDto>.Failure($"File storage failed: {uploadResult.ErrorMessage}");

                // Create file metadata entity
                var fileEntity = new FileMetadata(
                    fileMetadata.FileName,
                    storedPath: uploadResult.Data!,
                    fileMetadata.ContentType,
                    fileMetadata.FileSize,
                    uploadedByUserId,
                    fileMetadata.Description);

                _fileMetadataRepository.Add(fileEntity);
                await _unitOfWork.SaveChangesAsync(cancellationToken);

                _logger.LogInformation("File uploaded successfully. FileId: {FileId}, StorageKey: {StorageKey}",
                    fileEntity.Id, uploadResult.Data);

                var dto = MapToFileMetadataDto(fileEntity);
                return ServiceResult<FileMetadataDto>.Success(dto);
            }, cancellationToken); // Task<ServiceResult<...>>
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, message: "Error uploading file: {FileName}", fileMetadata?.FileName);
            return ServiceResult<FileMetadataDto>.Failure(errorMessage: "An error occurred while uploading the file", ex);
        }
    }
}
```

Figura 16. FileService.cs UploadFileAsync()

Fuente: Elaboración propia

La interacción con el servicio *MinIO* es realizada en *FileStorageService.cs* mediante el SDK de *MinIO* para .NET. Luego de que *UploadFileAsync()* realiza algunas validaciones la función *UploadWithRetryAsync()* es llamada. Esta última función realiza varios intentos de subir el archivo hasta que este es exitoso o el límite de intentos es superado, aumentando el tiempo de espera entre intentos de manera exponencial entre cada intento con el fin de evitar sobrecargar el servicio.

```
private async Task<ServiceResult> UploadWithRetryAsync(Stream fileStream, string storageKey, string contentType, CancellationToken cancellationToken)
{
    var attempt = 0;
    Exception? lastException = null;

    while (attempt < _settings.MaxRetryAttempts)
    {
        try
        {
            // Reset stream position if possible
            if (fileStream.CanSeek)
            {
                fileStream.Position = 0;
            }

            var putObjectArgs = new PutObjectArgs()
                .WithBucket(_settings.BucketName)
                .WithObject(storageKey)
                .WithStreamData(fileStream)
                .WithObjectSize(fileStream.Length)
                .WithContentType(contentType);

            await _minioClient.PutObjectAsync(putObjectArgs, cancellationToken);
            return ServiceResult.Success();
        }
        catch (Exception ex)
        {
            lastException = ex;
            attempt++;

            if (attempt < _settings.MaxRetryAttempts)
            {
                _logger.LogWarning(ex, message: "File upload attempt {Attempt} failed, retrying. Storage key: {StorageKey}",
                    attempt, storageKey);

                // Wait before retry with exponential backoff
                var delay :TimeSpan = TimeSpan.FromMilliseconds(Math.Pow(2, attempt - 1) * 1000);
                await Task.Delay(delay, cancellationToken);
            }
        }
    }

    _logger.LogError(lastException, message: "All file upload attempts failed. Storage key: {StorageKey}", storageKey);
    return ServiceResult.Failure(errorMessage: $"Upload failed after {_settings.MaxRetryAttempts} attempts: {lastException?.Message}", lastException);
}
```

Figura 17. FileStorageService.cs UploadWithRetryAsync()

Fuente: Elaboración propia

El listado de archivos es manejado por la función *GetAllAsync()*, este permite obtener un listado de los metadatos de los archivos almacenados en la base de datos. La función permite filtrar según tipo de archivo lo que permite adaptar el sistema en caso de que *MinIO* sea utilizado para almacenar otros tipos de archivos además de KMZ.

```
public async Task<ServiceResult<PaginatedResult<FileMetadataDto>>> GetAllAsync(
    int pageNumber = 1,
    int pageSize = 20,
    string? contentTypeFilter = null,
    CancellationToken cancellationToken = default)
{
    try
    {
        if (pageNumber <= 0)
            return ServiceResult<PaginatedResult<FileMetadataDto>>.Failure("Page number must be greater than zero");

        if (pageSize <= 0 || pageSize > 100)
            return ServiceResult<PaginatedResult<FileMetadataDto>>.Failure("Page size must be between 1 and 100");

        _logger.LogDebug("Getting files with pagination. Page: {PageNumber}, Size: {PageSize}, ContentType: {ContentType}",
            pageNumber, pageSize, contentTypeFilter);

        IEnumerable<FileMetadata> files;
        int totalCount;

        // Apply filters based on provided parameters
        if (!string.IsNullOrEmpty(contentTypeFilter))
        {
            var contentTypeFiles :IEnumerable<FileMetadata> = await _fileMetadataRepository.GetFilesByContentTypeAsync(contentTypeFilter, cancellationToken);
            totalCount = contentTypeFiles.Count();
            files = contentTypeFiles // IEnumerable<FileMetadata>
                .OrderByDescending(f :FileMetadata => f.UploadedAt) // IOrderedEnumerable<FileMetadata>
                .Skip((pageNumber - 1) * pageSize)
                .Take(pageSize); // IEnumerable<FileMetadata>
        }
        else
        {
            files = await _fileMetadataRepository.GetFilesPagedAsync(pageNumber, pageSize, cancellationToken);
            totalCount = await _fileMetadataRepository.CountAsync(cancellationToken);
        }

        var dtos :List<FileMetadataDto> = files.Select(MapToFileMetadataDto).ToList();
        var paginatedResult = new PaginatedResult<FileMetadataDto>(dtos, pageNumber, pageSize, totalCount);

        return ServiceResult<PaginatedResult<FileMetadataDto>>.Success(paginatedResult);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Error getting files with pagination");
        return ServiceResult<PaginatedResult<FileMetadataDto>>.Failure(errorMessage: "An error occurred while retrieving files", ex);
    }
}
```

Figura 18. 19 FileService.cs GetAllAsync()

Fuente: Elaboración propia

Con la información obtenida en el listado, el usuario puede descargar un archivo en particular mediante el *endpoint* `/api/Files/{id}`. La interacción con *MinIO* es manejada por la función *DownloadWithRetryAsync()*. Al igual que la función de subida, la descarga reintenta la operación hasta que esta es exitosa o el límite de intentos es alcanzado, aumentando el tiempo entre intentos de manera exponencial al igual que la función de subida.

```
1 usage 2 Roberto Romero
private async Task<ServiceResult<Stream>> DownloadWithRetryAsync(string storageKey, CancellationToken cancellationToken)
{
    var attempt = 0;
    Exception? lastException = null;

    while (attempt < _settings.MaxRetryAttempts)
    {
        try
        {
            var memoryStream = new MemoryStream();

            var getObjectArgs = new GetObjectArgs()
                .WithBucket(_settings.BucketName)
                .WithObject(storageKey)
                .WithCallbackStream(stream => stream.CopyTo(memoryStream));

            await _minioClient.GetObjectAsync(getObjectArgs, cancellationToken);

            memoryStream.Position = 0;
            return ServiceResult<Stream>.Success(memoryStream);
        }
        catch (Exception ex)
        {
            lastException = ex;
            attempt++;

            if (attempt < _settings.MaxRetryAttempts)
            {
                _logger.LogWarning(ex, message: "File download attempt {Attempt} failed, retrying. Storage key: {StorageKey}",
                    attempt, storageKey);

                // Wait before retry with exponential backoff
                var delay :TimeSpan = TimeSpan.FromMilliseconds(Math.Pow(2, attempt - 1) * 1000);
                await Task.Delay(delay, cancellationToken);
            }
        }
    }

    _logger.LogError(lastException, message: "All file download attempts failed. Storage key: {StorageKey}", storageKey);
    return ServiceResult<Stream>.Failure(errorMessage: $"Download failed after {_settings.MaxRetryAttempts} attempts: {lastException?.Message}", lastException);
}
```

Figura 20. FileService.cs DownloadWithRetryAsync()

Fuente: Elaboración propia

## 4.6 Integración de agenda operativa

La agenda operativa de las cuadrillas se integra mediante la importación de archivos Excel provistos por la organización. El sistema procesa estos archivos, extrae la información relevante y la normaliza para su uso interno, permitiendo asociar vehículos, personal y tareas operativas.

Esta integración corresponde a una importación manual en la versión actual del sistema. No se implementó sincronización automática con sistemas externos, la cual se considera como una posible extensión futura del proyecto.

La agenda es importada por la función *ImportAsync()* la cual es invocada al partir el programa. Esta lee cada fila del archivo Excel y los agrega a la base de datos si es que estos aún no existen.

```
public sealed class AgendaWorkbookImportService : IAgendaWorkbookImportService
{
    public async Task ImportAsync(CancellationToken cancellationToken = default)
    {
        try
        {
            var filePath = ResolveExcelPath(_options.AgendaExcelPath);
            filePath = "/Users/robertoromero/RiderProjects/Geocuadrilla/import_files/Trabajo en Terreno.xlsx";

            if (!File.Exists(filePath)){...}

            using var workbook = new XLWorkbook(filePath);
            if (!workbook.TryGetWorksheet(WorksheetName, out var worksheet)){...}

            var rows :List<IXLRow> = worksheet.RowsUsed().Skip(1).ToList();
            if (rows.Count == 0){...}

            var existingEntries :Dictionary<string,AgendaEntry> = await _dbContext.AgendaEntries // DbSet<AgendaEntry>
                .AsTracking() // IQueryable<AgendaEntry>
                .ToDictionaryAsync(entry => entry.Id, StringComparer.OrdinalIgnoreCase, cancellationToken); // Task<Dictionary<...>>

            var usersByEmail :Dictionary<string,User> = await _dbContext.Users // DbSet<User>
                .AsNoTracking() // IQueryable<User>
                .ToDictionaryAsync(user => user.Email.Value, StringComparer.OrdinalIgnoreCase, cancellationToken); // Task<Dictionary<...>>

            var movilesByPatente :Dictionary<string,Movil> = await _dbContext.Moviles // DbSet<Movil>
                .AsNoTracking() // IQueryable<Movil>
                .ToDictionaryAsync(m :Movil => m.Id, StringComparer.OrdinalIgnoreCase, cancellationToken); // Task<Dictionary<...>>

            var missingUsers = new HashSet<string>(StringComparer.OrdinalIgnoreCase);
            var missingMoviles = new HashSet<string>(StringComparer.OrdinalIgnoreCase);

            var added = 0;
            var updated = 0;

            foreach (var row in rows)
            {
                cancellationToken.ThrowIfCancellationRequested();

                if (!TryReadRow(row, out var record :AgendaRow , out var warning :string?)){...}

                var tecnico1UserId :Guid? = ResolveUserId(record.Tecnico1Email, usersByEmail, missingUsers);
                var tecnico2UserId :Guid? = ResolveUserId(record.Tecnico2Email, usersByEmail, missingUsers);
                var tecnico3UserId :Guid? = ResolveUserId(record.Tecnico3Email, usersByEmail, missingUsers);

                Movil? movil = null;
                if (!string.IsNullOrEmpty(record.PatenteMovil)){...}

                if (existingEntries.TryGetValue(record.Id, out var entry)){...}
                else{...}
            }
        }
    }
}
```

Figura 21. AgendaWorkbookImportService.cs ImportAsync()  
Fuente: Elaboración propia

#### 4.7 Comunicación en tiempo real con el panel web

La difusión de actualizaciones hacia el panel web se realiza mediante *SignalR*, utilizando comunicación bidireccional basada en *WebSockets*. El *backend* organiza a los clientes en grupos, permitiendo enviar actualizaciones únicamente a aquellos clientes suscritos a un contexto específico.

Una vez que una actualización de ubicación es procesada y almacenada, el sistema envía automáticamente la información al grupo correspondiente, logrando una visualización en tiempo real sin requerir solicitudes explícitas por parte del cliente.

El primer paso en el proceso de comunicación es obtener un listado de las cuadrillas activas en terreno y su estado actual, esto se logra mediante la función *GetLatestLocations()* de *LocationHub.cs*.

Con los *Ids* de las cuadrillas obtenidas del paso anterior, el *frontend* puede suscribirse a los eventos de actualización para las cuadrillas que se seleccionen. Esta funcionalidad de suscripción por *Id* permite, por ejemplo, que un centro de control regional solo se suscriba a las cuadrillas de su región.

La suscripción por *Id* es manejada por la función *SubscribeToCrews()* en *LocationHub.cs*. Cuando recibimos el listado de identificadores primero agregamos a quien llama a la función al grupo de suscriptores para la cuadrilla, de manera que cada cuadrilla puede enviar a diferentes suscriptores sus actualizaciones. Esta suscripción a los grupos es realizada de manera paralela para acelerar la acción y esperamos a que todas terminen con *await Task.WhenAll()*

Luego, *\_connectionManagementService* se encarga de mantener un control de las suscripciones para poder manejar las desconexiones y aplicar límites de suscripciones a las cuadrillas.

```
public async Task<IEnumerable<RealtimeLocationUpdate>> GetLatestLocations()
{
    if (!_connectionManagementService.CanSendMessage(Context.ConnectionId))
    {
        await Clients.Caller.SendAsync( method: "Error", new
        {
            ...
        }); // Task
        return Enumerable.Empty<RealtimeLocationUpdate>();
    }

    _connectionManagementService.RecordMessageSent(Context.ConnectionId);

    try
    {
        var result = await _locationUpdateService.GetLatestLocationUpdatesAsync();

        if (!result.IsSuccess){...}

        if (result.Data == null){...}

        var todayUtcDate = DateTime.UtcNow.Date;
        var todaysUpdates :CrewLatestLocationDto[] = result.Data
            .Where(dto => dto.LastUpdateTimestamp.Date == todayUtcDate) // IEnumerable<CrewLatestLocationDto>
            .ToArray();

        if (todaysUpdates.Length == 0)
        {
            _logger.LogInformation(
                "Filtered latest locations for connection {ConnectionId}: no crews reported updates today",
                Context.ConnectionId);

            return Enumerable.Empty<RealtimeLocationUpdate>();
        }

        var locations :RealtimeLocationUpdate[] = todaysUpdates // CrewLatestLocationDto[]
            .Select(dto => new RealtimeLocationUpdate
            {
                CrewId = dto.CrewId,
                Timestamp = dto.LastUpdateTimestamp,
                Latitude = dto.Latitude,
                Longitude = dto.Longitude,
                IsOnline = !dto.IsStale,
                PatenteMovil = dto.PatenteMovil,
                NombreMovil = dto.NombreMovil,
                TipoActividadId = dto.TipoActividadId,
                TipoActividadNombre = dto.TipoActividadNombre,
                TechnicianEmails = dto.TechnicianEmails
            }) // IEnumerable<RealtimeLocationUpdate>
            .ToArray();

        _logger.LogDebug("Returned {Count} latest locations for connection {ConnectionId}",
            locations.Length, Context.ConnectionId);

        return locations;
    }
}
```

Figura 22. LocationHub.cs GetLatestLocations()

Fuente: Elaboración propia

```
public async Task SubscribeToCrews(string[] crewIds)
{
    if (!_connectionManagementService.CanSendMessage(Context.ConnectionId))
    {
        await Clients.Caller.SendAsync( method: "Error", new
        {
            ...}); // Task
        return;
    }

    _connectionManagementService.RecordMessageSent(Context.ConnectionId);

    if (crewIds == null || crewIds.Length == 0){...}

    try
    {
        var normalizedCrewIds :string[] = crewIds //string[]
            .Where(id :string => !string.IsNullOrEmpty(id))
            .Select(id :string => id.Trim())
            .Where(id :string => id.Length > 0)
            .Distinct(StringComparer.OrdinalIgnoreCase) // IEnumerable<string>
            .ToArray();

        if (normalizedCrewIds.Length == 0)
        {
            await Clients.Caller.SendAsync( method: "Error", new
            {
                Code = "INVALID_CREW_IDS",
                Message = "No valid crew IDs provided."
            }); // Task
            return;
        }

        // Add to SignalR groups in parallel for performance
        var groupTasks :IEnumerable<Task> = normalizedCrewIds.Select(crewId :string =>
            Groups.AddToGroupAsync(Context.ConnectionId, groupName: $"crew_{crewId}"));

        await Task.WhenAll(groupTasks);

        // Track subscriptions for cleanup
        _connectionManagementService.AddCrewSubscriptions(Context.ConnectionId, normalizedCrewIds);

        _logger.LogInformation("Client {ConnectionId} subscribed to {CrewCount}/{RequestedCount} crews: {CrewIds}",
            Context.ConnectionId, normalizedCrewIds.Length, crewIds.Length,
            string.Join(" ", normalizedCrewIds));

        // Notify client of successful subscription
        await Clients.Caller.SendAsync( method: "SubscriptionConfirmed", new
        {
            CrewIds = normalizedCrewIds,
            Message = $"Successfully subscribed to {normalizedCrewIds.Length} crews"
        }); // Task
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Error subscribing to crews for connection {ConnectionId}", Context.ConnectionId);

        await Clients.Caller.SendAsync( method: "Error", new
        {
            Code = "SUBSCRIPTION_ERROR",
            Message = "An error occurred while subscribing to crews. Please try again."
        }); // Task
    }
}
```

Figura 23. LocationHub.cs SubscribeToCrews()  
Fuente: Elaboración propia

## 4.8 Autenticación y seguridad

Para autenticar el usuario debe llamar al *endpoint* `POST /api/Auth/login` con su correo y contraseña. El controlador de este *endpoint* aplica un limitador de velocidad que, con la configuración actual, evita que una IP tenga más de 5 intentos en 1 minuto para evitar ataques por fuerza bruta.

```
[HttpPost( template: "Login")]
[AllowAnonymous]
[EnableRateLimiting( policyName: "Login")]
[ProducesResponseType(typeof(BaseResponse<AuthenticationResult>), statusCode: StatusCodes.Status200OK)]
[ProducesResponseType(typeof(BaseResponse<object>), statusCode: StatusCodes.Status400BadRequest)]
[ProducesResponseType(typeof(BaseResponse<object>), statusCode: StatusCodes.Status401Unauthorized)]
[ProducesResponseType(typeof(BaseResponse<object>), statusCode: StatusCodes.Status423Locked)]
[ProducesResponseType(typeof(BaseResponse<object>), statusCode: StatusCodes.Status429TooManyRequests)]
// Roberto Romero +1
public async Task<ActionResult<BaseResponse<AuthenticationResult>>> LoginAsync(
    [FromBody] LoginRequest request,
    CancellationToken cancellationToken = default)
{
    // Validate model state
    if (!ModelState.IsValid){...}

    try
    {
        // Add request context information
        request.IpAddress = HttpContext.GetClientIpAddress();
        request.UserAgent = HttpContext.Request.Headers.UserAgent.FirstOrDefault();

        _logger.LogInformation("Login attempt for user: {UserName} from IP: {IpAddress}",
            request.UserName, request.IpAddress);

        var result = await _authenticationService.AuthenticateAsync(request, cancellationToken);

        if (result.IsSuccess)
        {
            _logger.LogInformation("Successful login for user: {UserName}", request.UserName);
            return Success(result, message: "Login successful");
        }

        // Determine appropriate HTTP status code based on error message
        var statusCode:int = result.ErrorMessage switch
        {
            var msg:string? when msg.Contains("locked", StringComparison.OrdinalIgnoreCase) => StatusCodes.Status423Locked,
            var msg:string? when msg.Contains("invalid", StringComparison.OrdinalIgnoreCase) => StatusCodes.Status401Unauthorized,
            var msg:string? when msg.Contains("not found", StringComparison.OrdinalIgnoreCase) => StatusCodes.Status401Unauthorized,
            _ => StatusCodes.Status401Unauthorized
        };

        return StatusCode(statusCode, BaseResponse<AuthenticationResult>.Failure(result.ErrorMessage ?? "Authentication failed"));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Error during login attempt for user: {UserName}", request.UserName);
        return StatusCode(StatusCodes.Status500InternalServerError, BaseResponse<AuthenticationResult>.Failure("An unexpected error occurred during login"));
    }
}
```

Figura 24. AuthController.cs LoginAsync()

Fuente: Elaboración propia

La contraseña del usuario es verificada mediante bibliotecas nativas de .NET con `user.VerifyPassword()`, de ser correctas las credenciales proporcionadas se emite un token JWT que verifica la validez de la sesión.

Por motivos de seguridad, este *endpoint* tiene un tiempo mínimo de respuesta, actualmente 100 milisegundos, para contrarrestar ataques de *timing* que puedan filtrar información de los usuarios.

```
public async Task<AuthenticationResult> AuthenticateAsync(LoginRequest loginRequest, CancellationToken cancellationToken = default)
{
    if (loginRequest == null)
        throw new ArgumentNullException(nameof(loginRequest));

    var startTime = DateTime.UtcNow;
    const int minResponseTimeMs = 100; // Minimum response time to prevent timing attacks
    const int maxFailedAttempts = 5;
    const int lockoutMinutes = 30;

    try
    {
        _logger.LogInformation("Authentication attempt for user: {UserName}", loginRequest.UserName);

        // Get user with roles by username or email
        var user = await _userRepository.GetUserWithRolesByUsernameAsync(loginRequest.UserName, cancellationToken)
            ?? await _userRepository.GetUserWithRolesByEmailAsync(loginRequest.UserName, cancellationToken);

        // Always perform password verification to maintain consistent timing
        var isPasswordValid = false;
        if (user != null)
        {
            isPasswordValid = user.VerifyPassword(loginRequest.Password);
        }
        else{...}

        // Handle user not found case
        if (user == null){...}

        // Check if account is locked BEFORE validating credentials
        if (user.IsAccountLocked()){...}

        // Check if account is active
        if (!user.IsActive){...}

        // Validate credentials
        if (!isPasswordValid){...}

        var authenticationResult = await GenerateAuthenticationResultAsync(
            user,
            loginRequest.IpAddress,
            loginRequest.UserAgent,
            cancellationToken);

        _logger.LogInformation("Successful authentication for user: {UserName}", loginRequest.UserName);

        await EnsureMinimumResponseTime(startTime, minResponseTimeMs, cancellationToken);
        return authenticationResult;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Error during authentication for user: {UserName}", loginRequest.UserName);
        await EnsureMinimumResponseTime(startTime, minResponseTimeMs, cancellationToken);
        return AuthenticationResult.Failure(errorMessage: "Authentication failed due to an internal error");
    }
}
```

Figura 25. AuthenticationService.cs AuthenticateAsync()

Fuente: Elaboración propia

Cuando un usuario excede la cantidad máxima de intentos, su cuenta es bloqueada por cierta cantidad de minutos.

La cantidad de intentos permitidos y los minutos que la cuenta es bloqueada son definidos en *AuthenticateAsync()*, actualmente se permiten 5 intentos y el *timeout* es de 30 minutos.

```
public void RecordFailedLogin(int maxAttempts, int lockoutMinutes, bool useProgressiveLockout = true)
{
    if (maxAttempts <= 0)
        throw new ArgumentException("Max attempts must be greater than zero", nameof(maxAttempts));
    if (lockoutMinutes <= 0)
        throw new ArgumentException("Lockout minutes must be greater than zero", nameof(lockoutMinutes));

    LastFailedLoginAt = DateTime.UtcNow;
    FailedLoginAttempts++;

    if (FailedLoginAttempts >= maxAttempts)
    {
        LockAccount(lockoutMinutes, useProgressiveLockout);
    }

    MarkAsUpdated();
}
```

Figura 26. User.cs RecordFailedLogin()

Fuente: Elaboración propia

#### 4.9 Gestión de usuarios.

El sistema expone un módulo de gestión de usuarios mediante endpoints REST que permiten al administrador crear, consultar, modificar y eliminar usuarios de la plataforma, implementando eliminación lógica (*soft delete*) para preservar la integridad referencial de los registros históricos. La asignación de roles se gestiona mediante un *endpoint* dedicado que reemplaza el conjunto de roles asociados al usuario, complementando el flujo de autenticación descrito en la sección anterior.

Las funciones del controlador */Users* son ejecutadas por *UserService.cs*

```
public async Task<ServiceResult<PaginatedResult<UserDto>>> GetAllAsync(int pageNumber = 1, int pageSize = 10, bool activeOnly = true, CancellationToken cancellationToken = default)
{
    try
    {
        if (pageNumber <= 0)
            return ServiceResult<PaginatedResult<UserDto>>.Failure("Page number must be greater than zero");

        if (pageSize <= 0 || pageSize > 100)
            return ServiceResult<PaginatedResult<UserDto>>.Failure("Page size must be between 1 and 100");

        IEnumerable<User> users;
        int totalCount;

        if (activeOnly)
        {
            users = await _userRepository.GetActiveUsersPagedAsync(pageNumber, pageSize, cancellationToken);
            totalCount = await _userRepository.GetActiveUsersCountAsync(cancellationToken);
        }
        else
        {
            users = await _userRepository.GetPagedAsync(pageNumber, pageSize, cancellationToken);
            totalCount = await _userRepository.CountAsync(cancellationToken);
        }

        var userDtos :List<UserDto> = users.Select(u :User => MapToUserDto(u, includeRoles: false)).ToList();

        var paginatedResult = new PaginatedResult<UserDto>
        {
            Items = userDtos,
            CurrentPage = pageNumber,
            PageSize = pageSize,
            TotalCount = totalCount,
            TotalPages = (int)Math.Ceiling((double)totalCount / pageSize)
        };

        return ServiceResult<PaginatedResult<UserDto>>.Success(paginatedResult);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Error getting all users with pagination");
        return ServiceResult<PaginatedResult<UserDto>>.Failure(errorMessage: "An error occurred while retrieving users", ex);
    }
}
```

Figura 27. GetAllAsync(), función q  
Fuente: Elaboración propia

#### 4.10 Despliegue del sistema.

El prototipo fue desplegado en una máquina virtual alojada en un servidor propio, la ejecución fue asistida por el uso de contenedores Docker.

Para facilitar el funcionamiento del prototipo todos los servicios del sistema se despliegan mediante un solo *Docker compose*, sin embargo, en un sistema productivo es posible ejecutar los diferentes servicios en servidores o nodos diferentes para asegurar una alta disponibilidad.

El archivo `Docker-compose.yaml` presentado a continuación corresponde a la definición del sistema para la demostración final.

```
services:
  postgres:
    image: postgres:16-alpine
    container_name: geocudrilla-postgres
    environment:
      - POSTGRES_DB=${POSTGRES_DB:-geocudrilla2}
      - POSTGRES_USER=${POSTGRES_USER:-postgres}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./scripts/init-db.sql:/docker-entrypoint-initdb.d/init-db.sql
    networks:
      - geocudrilla-network
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER:-postgres} -d ${POSTGRES_DB:-geocudrilla2}"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 10s

  rabbitmq:
    image: rabbitmq:3.13-management-alpine
    container_name: geocudrilla-rabbitmq
    hostname: geocudrilla-rabbitmq
    environment:
      - RABBITMQ_DEFAULT_USER=${RABBITMQ_USER:-admin}
      - RABBITMQ_DEFAULT_PASS=${RABBITMQ_PASSWORD}
      - RABBITMQ_DEFAULT_VHOST=${RABBITMQ_VHOST:-/}
    ports:
      - "5672:5672"      # AMQP port
      - "15672:15672"    # Management UI port
    volumes:
      - rabbitmq_data:/var/lib/rabbitmq
    networks:
      - geocudrilla-network
    healthcheck:
      test: ["CMD", "rabbitmq-diagnostics", "ping"]
      interval: 15s
      timeout: 10s
      retries: 5
      start_period: 20s

  minio:
    image: minio/minio:latest
    container_name: geocudrilla-minio
    environment:
      - MINIO_ROOT_USER=${MINIO_ROOT_USER:-admin}
      - MINIO_ROOT_PASSWORD=${MINIO_ROOT_PASSWORD}
    ports:
      - "9000:9000"      # MinIO API
      - "9001:9001"      # MinIO Console UI
    volumes:
      - minio_data:/data
    command: server /data --console-address ":9001"
    networks:
      - geocudrilla-network
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]
      interval: 15s
      timeout: 10s
      retries: 3
      start_period: 20s

  api:
    build:
      context: .
      dockerfile: Geocudrilla-2/Dockerfile
```

```
container_name: geocuadrilla-api
environment:
  - ASPNETCORE_ENVIRONMENT=${ASPNETCORE_ENVIRONMENT:-Docker}
  - ASPNETCORE_URLS=http://+:8080
  - ConnectionStrings__DefaultConnection=Host=postgres;Port=5432;Database=${POSTGRES_DB:-geocuadrilla2};Username=${POSTGRES_USER:-postgres};Password=${POSTGRES_PASSWORD};Include
Error Detail=true
  - JwtSettings__SecretKey=${JWT_SECRET}
  - RabbitMQ__Host=rabbitmq
  - RabbitMQ__Username=${RABBITMQ_USER:-admin}
  - RabbitMQ__Password=${RABBITMQ_PASSWORD}
  - RabbitMQ__VirtualHost=${RABBITMQ_VHOST:-/}
  - MinIO__Endpoint=minio:9000
  - MinIO__AccessKey=${MINIO_ROOT_USER:-admin}
  - MinIO__SecretKey=${MINIO_ROOT_PASSWORD}
  - MinIO__BucketName=${MINIO_BUCKET_NAME:-geocuadrilla-files}
ports:
  - "8081:8080"
depends on:
  postgres:
    condition: service_healthy
  rabbitmq:
    condition: service_healthy
  minio:
    condition: service_healthy
networks:
  - geocuadrilla-network
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
  interval: 30s
  timeout: 10s
  retries: 3
  start_period: 40s

volumes:
  postgres_data:
    driver: local
  rabbitmq_data:
    driver: local
  minio_data:
    driver: local

networks:
  geocuadrilla-network:
    driver: bridge
```

Figura 28. Definición de servicios en Docker Compose.

Fuente: Elaboración propia

La ejecución del sistema puede ser observada mediante el comando `docker ps`.

```
+ ~ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS
c86efeb1fbdd   geocuadrilla-api                   "dotnet Geocuadrilla..." 6 minutes ago  Up 5 minutes (unhealthy)  8081/tcp, 0.0.0.0:5133->8080/tcp, 0.0.0.0:8081->8080/tcp, [::]:5133->8080/tcp, [::]:8081->8080/tcp
a20d5ffc70a    postgres:16-alpine                "docker-entrypoint.s..." 6 minutes ago  Up 6 minutes (healthy)   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp, 0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp
fcfe064ad79    rabbitmq:3.13-management-alpine    "docker-entrypoint.s..." 10 days ago   Up 11 minutes (healthy)  4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, [::]:5672->5672/tcp, 0.0.0.0:15672->15672/tcp, [::]:15672->15672/tcp
f019c130a3a5   minio/minio:latest                "/usr/bin/docker-ent..." 10 days ago   Up 11 minutes (healthy)  0.0.0.0:9000-9001->9000-9001/tcp, [::]:9000-9001->9000-9001/tcp
```

Figura 29. Servicios siendo ejecutados con Docker Compose.

Fuente: Elaboración propia

## 5 Resultados y validación del sistema

### 5.1 Contexto de validación

La validación del sistema de integración se realizó en un entorno de prueba funcional, correspondiente a un servidor propio en el cual se desplegaron de manera simultánea todos los servicios del sistema mediante *Docker Compose*. Este entorno incluyó el *backend* de integración, el *bróker* de mensajería *RabbitMQ*, la base de datos *PostgreSQL*, el sistema de almacenamiento de objetos *MinIO* y los servicios de comunicación en tiempo real.

Para evaluar la funcionalidad de los *endpoints* se utilizó el programa *Bruno*, el cual, de manera similar a *Postman*, nos permite realizar solicitudes HTTP y WebSocket.

### 5.2 Validación de requisitos funcionales

#### - RF-01 Recepción de actualizaciones de ubicación:

Se verificó el correcto funcionamiento del endpoint `POST /api/location-updates` mediante la ejecución de solicitudes de prueba. El sistema recibió satisfactoriamente una actualización de ubicación conteniendo los campos requeridos: identificador del técnico, marca de tiempo en formato ISO 8601, latitud y longitud en grados decimales, y origen de la solicitud. El sistema respondió con código HTTP 200 y el mensaje "Location update accepted." en un tiempo de 69 ms, confirmando la recepción y el encolamiento del mensaje para su procesamiento asíncrono mediante el patrón *outbox*. La persistencia y difusión de esta actualización es evidenciada en los puntos RF-02 y RF-03 respectivamente.

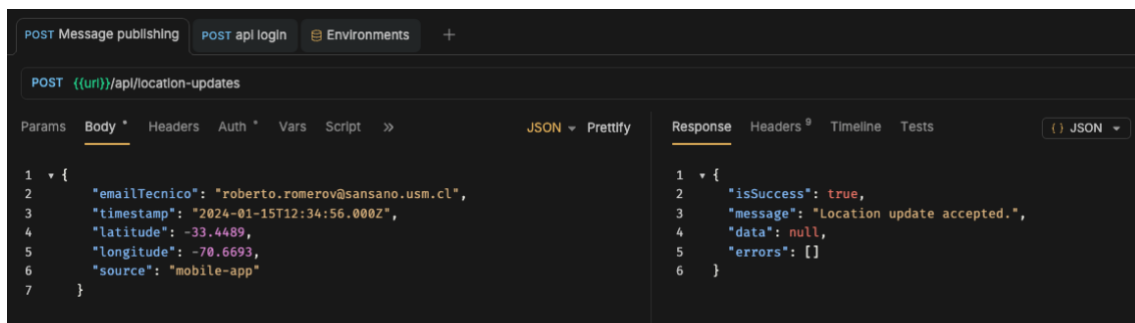


Figura 30. Request y respuesta de actualización de ubicación

Fuente: Elaboración propia.

## - RF-02 Registro histórico de ubicaciones

Se verificó la persistencia de las actualizaciones de ubicación en la base de datos PostgreSQL mediante la consulta directa a la tabla de registros históricos. Como se observa en la figura, cada actualización procesada quedó almacenada como un registro independiente, incluyendo el identificador único, el nombre de la cuadrilla asociada, las coordenadas geográficas en grados decimales, y la marca de tiempo en formato UTC. La tabla evidencia un historial continuo de posiciones correspondiente a múltiples cuadrillas operando de forma simultánea.

|                                       |              |                                   |                     |                     |                                   |
|---------------------------------------|--------------|-----------------------------------|---------------------|---------------------|-----------------------------------|
| 4e410992-dbf9-458d-89ce-2ebbb13f8f23  | Tecnored-002 | 2026-01-30 09:33:25.037000 +00:00 | -33.103020050000000 | -71.677477420000000 | 2026-01-30 20:40:49.206891 +00:00 |
| 2638df35-724c-4c45-99a2-a480df10df0   | Tecnored-002 | 2026-01-30 08:17:28.228000 +00:00 | -33.102947330000000 | -71.677330570000000 | 2026-01-30 20:40:49.206891 +00:00 |
| 9ff5cfd5-7ab6-44b0-856e-d32dea939b53  | Tecnored-003 | 2026-01-30 06:46:04.449000 +00:00 | -32.987274140000000 | -71.524674580000000 | 2026-01-30 20:40:49.206890 +00:00 |
| 1c74e232-09ee-414f-a2f1-b3aa9f6f8e5b2 | Tecnored-002 | 2026-01-30 06:31:18.104000 +00:00 | -33.045845070000000 | -71.476005640000000 | 2026-01-30 20:40:49.206890 +00:00 |
| 05f7d74-e31-4ecb-a120-1357085d6be8    | Tecnored-004 | 2026-01-30 06:18:11.844000 +00:00 | -32.885615240000000 | -71.256060130000000 | 2026-01-30 20:40:49.206596 +00:00 |
| 1ecd179d-2f0a-4e0c-b8d4-740ae04ce7b0  | Tecnored-002 | 2026-01-23 08:28:38.000000 +00:00 | -33.353225910247986 | -70.720766265179620 | 2026-01-23 00:29:00.381284 +00:00 |
| 9fe097df-d2c6-42dc-8166-b1af56b736ab  | Tecnored-002 | 2026-01-23 08:28:33.000000 +00:00 | -33.557244587006700 | -70.557546172554480 | 2026-01-23 00:28:50.176335 +00:00 |
| 1beba53d-ec0b-495e-9d54-39817f0d93eb  | Tecnored-002 | 2026-01-23 08:28:28.000000 +00:00 | -33.595290205102810 | -70.514027426738450 | 2026-01-23 00:28:39.877131 +00:00 |
| e454edc6-c0bf-4efd-9aa8-5ac6b77da8b3  | Tecnored-002 | 2026-01-23 08:28:23.000000 +00:00 | -33.382735708426104 | -70.501468214790040 | 2026-01-23 00:28:29.362228 +00:00 |
| 1ecfd0f0-999a-48bd-9e62-559a74a24ef8  | Tecnored-002 | 2026-01-23 08:28:18.000000 +00:00 | -33.561887915507510 | -70.665127311968870 | 2026-01-23 00:28:19.077989 +00:00 |
| b62f0dbf-e511-420e-b9ac-a7a6b2b88f89  | Tecnored-002 | 2026-01-22 23:58:12.000000 +00:00 | -33.508936915389920 | -70.734267311930170 | 2026-01-22 23:58:39.384303 +00:00 |
| 93d5da43-5bad-47dc-bccb-adccc704810e  | Tecnored-002 | 2026-01-22 23:58:07.000000 +00:00 | -33.340629331759686 | -70.851982575608370 | 2026-01-22 23:58:28.470869 +00:00 |
| 22e5e1e6-8341-487a-8ec7-c62acc8dace5f | Tecnored-002 | 2026-01-22 23:58:02.000000 +00:00 | -33.330647841252830 | -70.655786970462910 | 2026-01-22 23:58:17.735120 +00:00 |
| 01642861-6313-4207-bcd5-0197080b7e10f | Tecnored-002 | 2026-01-22 23:57:57.000000 +00:00 | -33.384535698997560 | -70.573428728173510 | 2026-01-22 23:58:06.084580 +00:00 |
| 05339ae3-3323-4641-a947-4356a600f505  | Tecnored-002 | 2026-01-22 23:57:53.000000 +00:00 | -33.463258220162544 | -70.553199541271770 | 2026-01-22 23:58:02.828559 +00:00 |
| 4ffff4c90-8f18-4aac-871c-bbc3803e3254 | Tecnored-002 | 2026-01-22 23:57:52.000000 +00:00 | -33.413673103724356 | -70.782470672376330 | 2026-01-22 23:57:53.249965 +00:00 |
| f2cf846f-831e-49b8-9a87-82dedcac3b42  | Tecnored-002 | 2026-01-22 23:54:33.000000 +00:00 | -33.444203764732420 | -70.856405168266890 | 2026-01-22 23:54:57.876723 +00:00 |
| fd95f422-febb-454c-958e-7a73a34edc08  | Tecnored-002 | 2026-01-22 23:54:32.000000 +00:00 | -33.507459325550130 | -70.602012986993830 | 2026-01-22 23:54:51.397152 +00:00 |
| fdcl3ec-1bc7-447a-aa82-af78227f9fad   | Tecnored-002 | 2026-01-22 23:54:31.000000 +00:00 | -33.311992090806280 | -70.526507375720780 | 2026-01-22 23:54:50.286242 +00:00 |
| 87435d7e-f129-44e0-803b-72bffa522880  | Tecnored-002 | 2026-01-22 23:54:12.000000 +00:00 | -33.529303677202590 | -70.831412771686590 | 2026-01-22 23:54:48.788390 +00:00 |
| 2fc8ad9a-6c14-4b1f-8009-6cf6e6964ed4  | Tecnored-002 | 2026-01-22 23:54:11.000000 +00:00 | -33.379315801606260 | -70.798180241433930 | 2026-01-22 23:54:39.673933 +00:00 |

Figura 31. Tabla 'LocationUpdates' en la base de datos.

Fuente: Elaboración propia.

## - RF-03 Notificación automática al panel web

Se validó el mecanismo de notificación automática al panel web mediante la ejecución de un flujo de prueba completo sobre el canal de comunicación en tiempo real, utilizando la herramienta Bruno para establecer conexiones *WebSocket* contra el *hub* expuesto en */hub/location*.

En primer lugar, se estableció una conexión con el *hub* y se invocó el método *GetLatestLocations*, obteniendo el listado de cuadrillas operativas junto con sus últimas ubicaciones registradas, confirmando que el sistema es capaz de entregar el estado inicial al momento de la conexión.

A continuación, se invocó el método *SubscribeToCrews* para suscribir al cliente a las notificaciones de una cuadrilla específica, recibiendo del servidor una confirmación de suscripción mediante el evento *SubscriptionsConfirmed*.

Una vez establecida la suscripción, se envió una actualización de ubicación para la cuadrilla suscrita mediante el *endpoint* *POST /api/location-updates*. El servidor notificó automáticamente la actualización al cliente a través del canal *WebSocket*,

sin requerir solicitud explícita por parte de este. Como se observa en la Figura 35, la diferencia de tiempo entre el envío de la actualización y su recepción en el canal fue inferior a 1 segundo, evidenciando la baja latencia del proceso de integración y difusión incluso con una conexión de datos móviles.

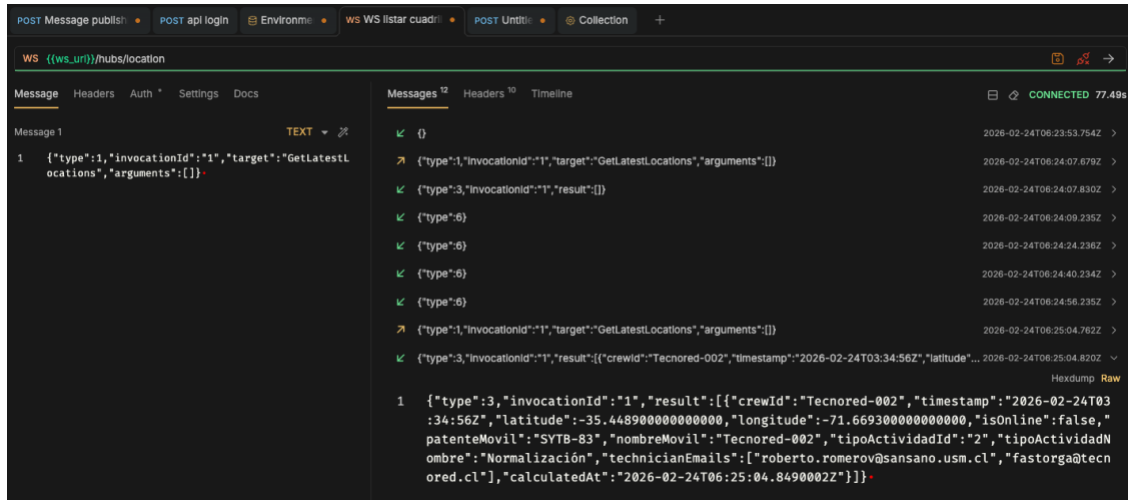


Figura 32. Listado de cuadrillas por WebSocket.

Fuente: Elaboración propia.

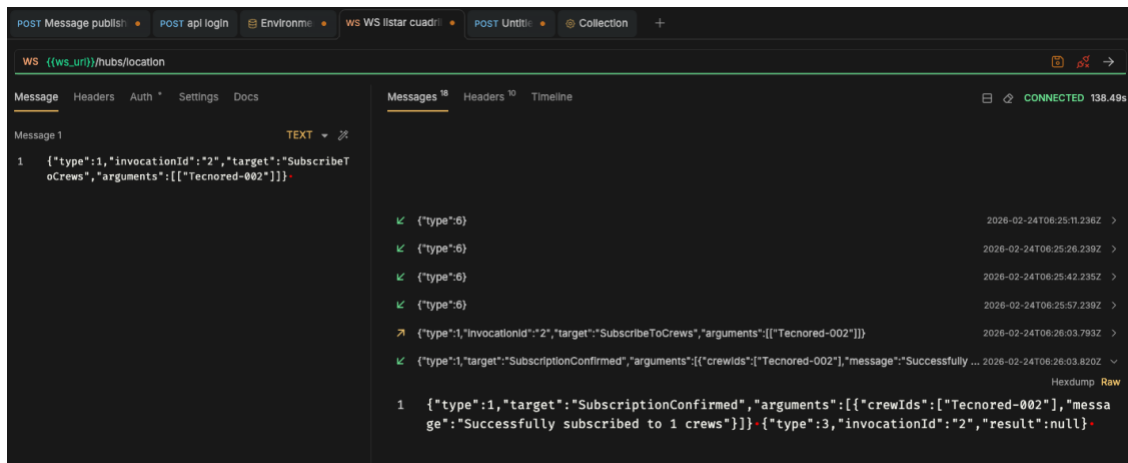


Figura 33. Suscripción a eventos de la cuadrilla y confirmación de suscripción.

Fuente: Elaboración propia.

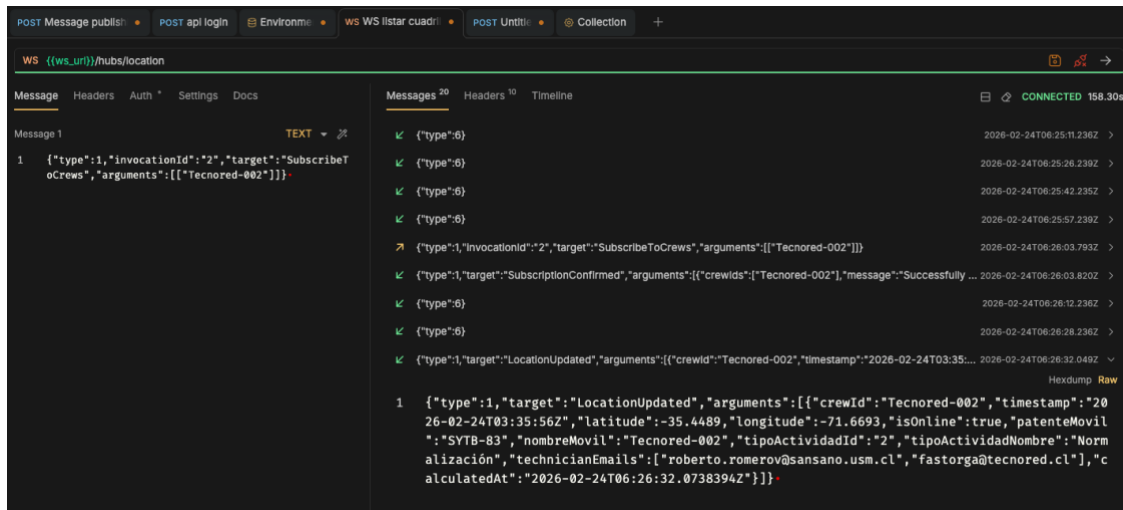


Figura 34. Recepción de la actualización de ubicación  
Fuente: Elaboración propia.

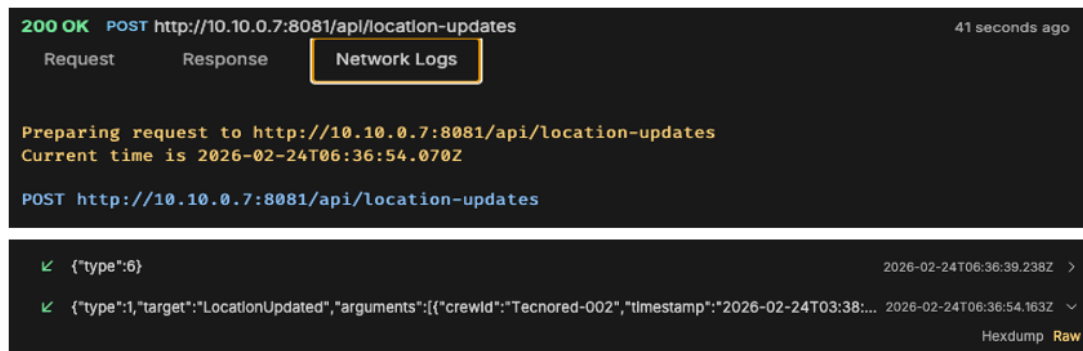


Figura 35. Comparación de marcas de tiempo entre la solicitud HTTP POST (arriba)  
y la notificación WebSocket (abajo).

#### - RF-04 Gestión de archivos geoespaciales

Se validó el ciclo completo de operaciones CRUD sobre archivos geoespaciales en formato KMZ, ejecutando cada operación de forma secuencial mediante solicitudes HTTP al *backend* y verificando las respuestas obtenidas.

La validación inició con la consulta del listado de archivos existentes en el sistema mediante el método GET, obteniendo correctamente los metadatos de los archivos registrados. A continuación, se descargó uno de los archivos listados, verificando que el contenido retornado por el sistema correspondía al archivo original almacenado en MinIO. Posteriormente, se ejecutó la eliminación del archivo descargado mediante el

método DELETE, confirmando su remoción tanto del repositorio de objetos como de los registros de la base de datos. El archivo fue cargado nuevamente al sistema mediante el método POST, validando el proceso de subida y el registro de sus metadatos. Finalmente, se modificó la descripción del archivo mediante el método PUT, verificando que la actualización de metadatos opera de forma independiente al archivo físico almacenado.

Todas las operaciones retornaron los códigos HTTP esperados (200 OK y 201 Created, confirmando el correcto funcionamiento del sistema CRUD para la gestión de archivos geoespaciales.

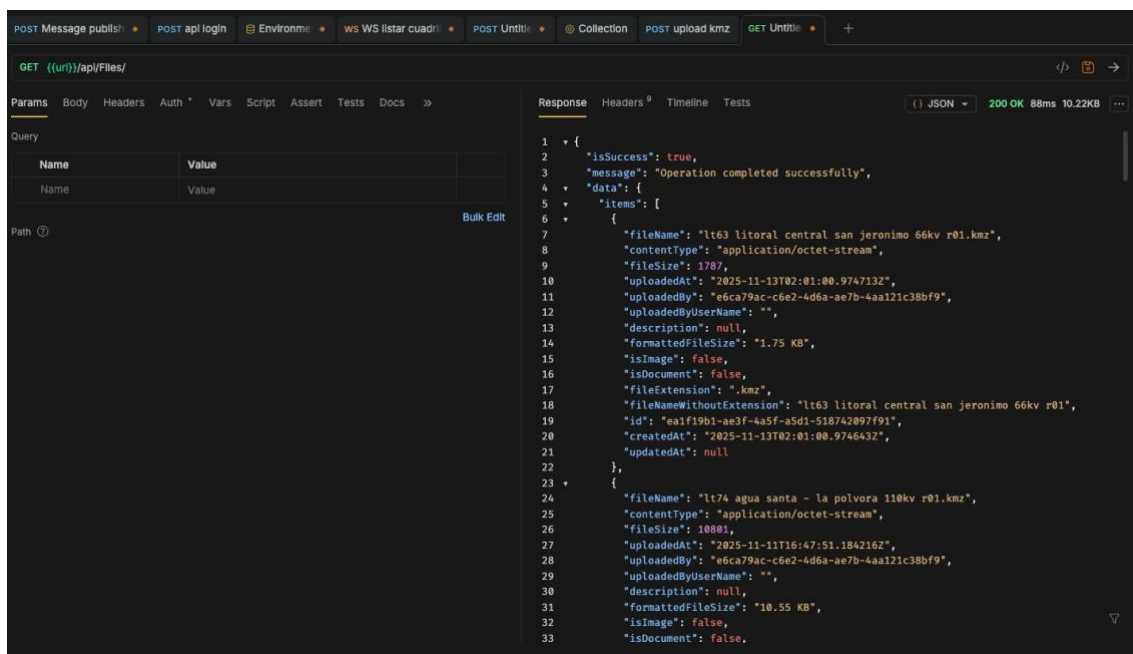


Figura 36. Listado de archivos KMZ en el sistema

Fuente: Elaboración propia.

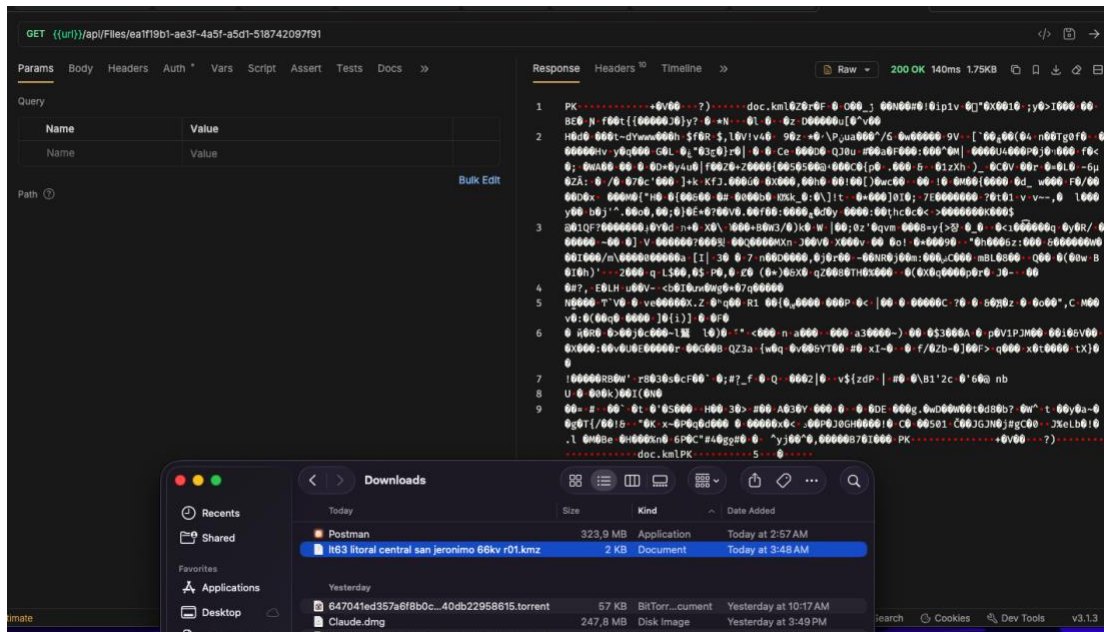


Figura 37. Descarga de un archivo KMZ

Fuente: Elaboración propia.

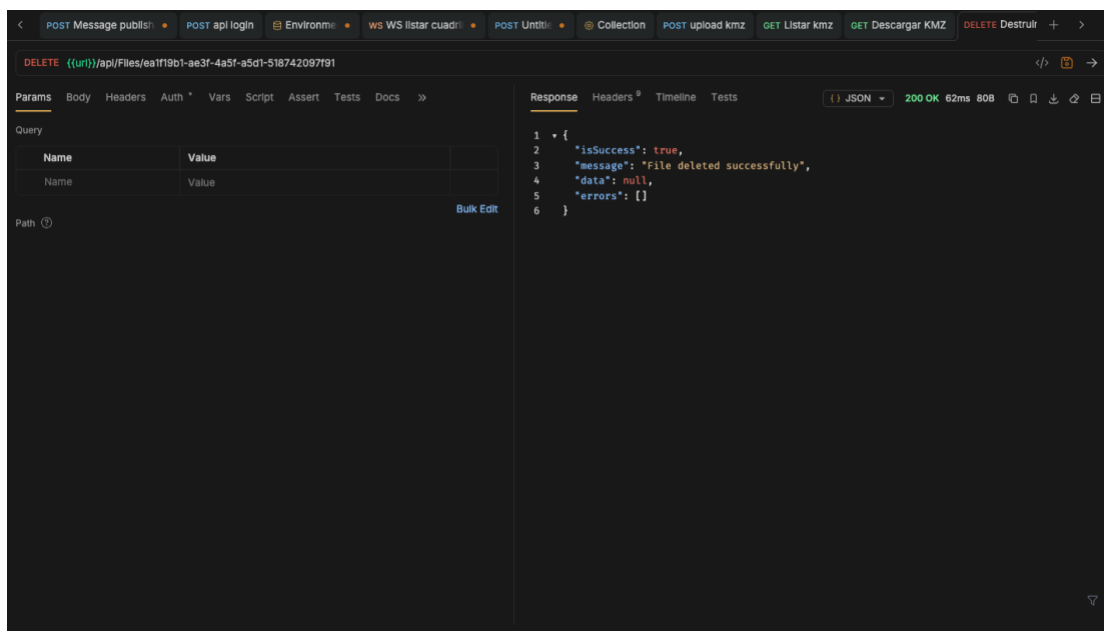


Figura 38. Eliminación de un archivo KMZ

Fuente: Elaboración propia.

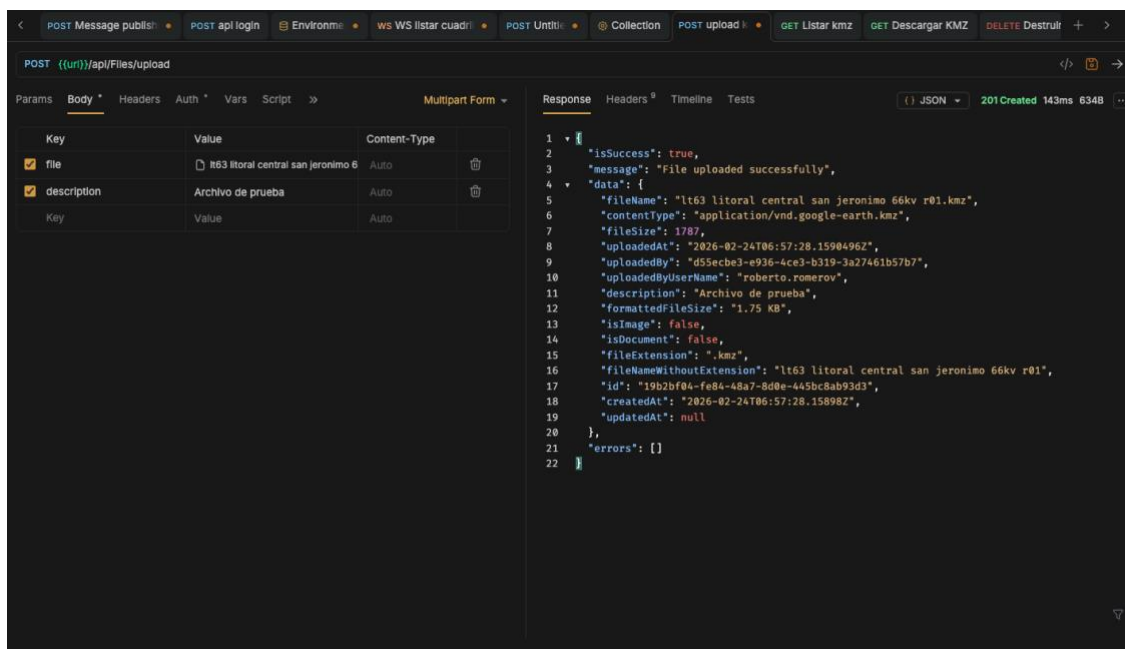


Figura 39. Archivo KMZ cargado con éxito  
Fuente: Elaboración propia.

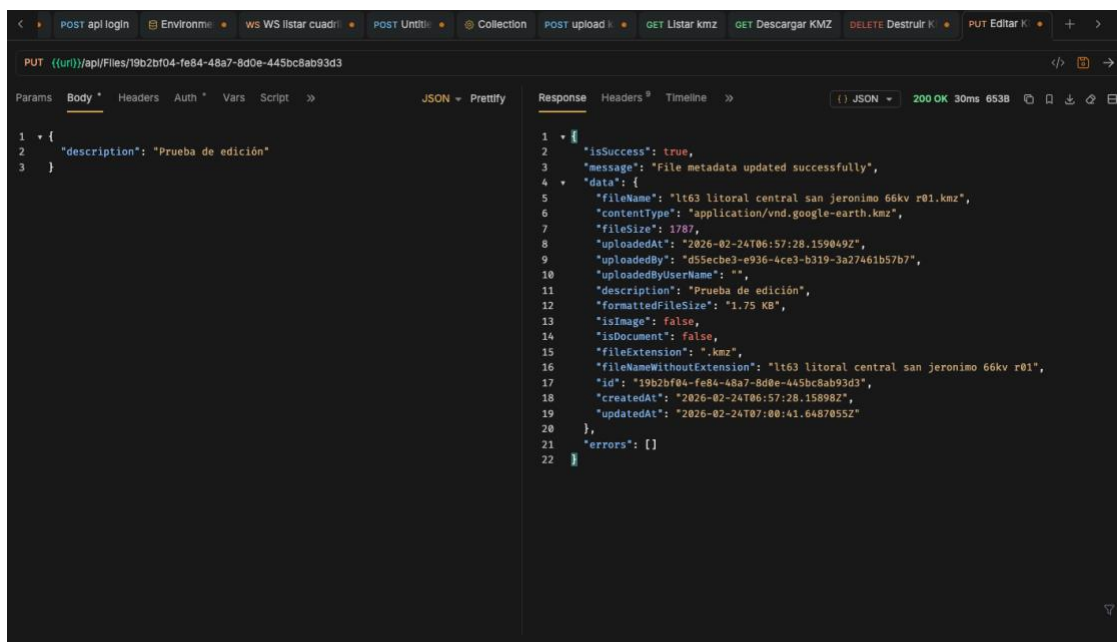


Figura 40. Edición de un archivo KMZ existente  
Fuente: Elaboración propia.

- **RF-05 Integración de agenda operativa**

Se validó la importación y normalización de la agenda operativa desde archivos Excel provistos por Chilquinta Transmisión. El sistema procesó correctamente la información contenida en estos archivos, asociando para cada cuadrilla monitoreada el nombre del técnico asignado, la patente del vehículo y el tipo de actividad en ejecución.

La efectividad de esta integración queda evidenciada en la **Error! Reference source not found.**, correspondiente a la validación de RF-03, donde se observa que el *payload* de la actualización de ubicación notificada al panel web incluye, además de las coordenadas geográficas, los campos operativos provenientes de la agenda normalizada. Dado que esta información existe exclusivamente en el archivo Excel importado, su presencia en el mensaje publicado confirma que el proceso de importación, normalización y asociación de datos operativos funciona correctamente de extremo a extremo.

### 5.3 Validación de requisitos no funcionales

- **RNF-01 – Rendimiento (latencia de actualización)**

La latencia del sistema fue evaluada en el contexto de la validación de RF-03, donde se midió el tiempo transcurrido entre el envío de una actualización de ubicación por parte de la aplicación móvil y su recepción en el panel web a través del canal *WebSocket*. Como se evidencia en las figuras correspondientes a dicha validación, la diferencia de tiempo entre ambos eventos fue inferior a 1 segundo, cumpliendo con el requisito establecido de una latencia máxima de 2 segundos bajo condiciones normales de operación. Este resultado fue obtenido en un entorno de prueba funcional con una frecuencia de actualización de 5 segundos y utilizando datos móviles.

- **RNF-02 – Capacidad de procesamiento**

Se evaluó la capacidad de procesamiento del sistema mediante una prueba de carga desarrollada en Python, utilizando `ThreadPoolExecutor` para simular solicitudes concurrentes al endpoint `POST /api/location-updates`. La prueba consistió en el envío de mensajes con diferentes parámetros de cantidad y de simultaneidad, llegando hasta 30000 mensajes enviados por 150 diferentes hilos de manera simultánea.

El sistema procesó la totalidad de las solicitudes sin registrar ningún fallo, obteniendo hasta 30.000 respuestas HTTP 200 de un total de 30.000 enviadas. El tiempo total de ejecución para la prueba con mayor cantidad de mensajes fue de 45,8 segundos,

con un tiempo promedio por solicitud de 1,5 ms. Estos resultados confirman que la arquitectura basada en el patrón *outbox* y mensajería asíncrona mediante RabbitMQ permite absorber volúmenes de carga elevados sin degradar el tiempo de respuesta ni producir pérdida de mensajes, cumpliendo ampliamente con el requisito de capacidad de procesamiento definido.

```
def main() -> int:
    if len(sys.argv) < 3:
        usage()
        return 1

    base_url = sys.argv[1]
    token = sys.argv[2]
    email = sys.argv[3] if len(sys.argv) > 3 else DEFAULT_EMAIL
    count = int(sys.argv[4]) if len(sys.argv) > 4 else 1
    parallel = int(sys.argv[5]) if len(sys.argv) > 5 else 1

    if count < 1 or parallel < 1:
        print("count and parallel must be >= 1")
        return 1

    start = time.perf_counter()

    successes = 0
    with ThreadPoolExecutor(max_workers=min(parallel, count)) as executor:
        futures = [
            executor.submit(post_location, base_url, token, email, i, count)
            for i in range(1, count + 1)
        ]

        for future in as_completed(futures):
            if future.result():
                successes += 1

    elapsed = time.perf_counter() - start
    failures = count - successes
    avg = elapsed / count

    print(f"Done. total={count} ok={successes} failed={failures}")
    print(f"Time: total={elapsed:.3f}s avg_per_request={avg:.3f}s parallel={min(parallel, count)}")

    return 0 if failures == 0 else 1

if __name__ == "__main__":
    raise SystemExit(main())
```

Figura 41. Script Python para prueba de carga en paralelo.

Fuente: Elaboración propia.

```
[4977/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4995/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4983/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4993/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4977/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4986/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4976/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4982/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4990/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4994/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4996/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4992/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4997/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4999/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4991/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[4998/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
[5000/5000] status=200 body={"isSuccess":true,"message":"Location update accepted.","data":null,"errors":[]}
Done. total=5000 ok=5000 failed=0
Time: total=11.009s avg_per_request=0.002s parallel=25

→ Geocuadrilla git:(master) ×
```

Figura 42. Resultado de la ejecución del script con 5000 mensajes y 25 hilos.  
Fuente: Elaboración propia.

La siguiente tabla presenta los resultados de diferentes parámetros de ejecución del *script* de prueba de carga.

Tabla 20. Resultados de pruebas de carga.

| Cantidad de mensajes | Cantidad de hilos | Tiempo total [s] | Throughput promedio [mensajes/tiempo] | % Errores |
|----------------------|-------------------|------------------|---------------------------------------|-----------|
| 500                  | 10                | 1.534            | 332.96                                | 0%        |
| 5000                 | 15                | 10.808           | 462.6                                 | 0%        |
| 15000                | 60                | 24.666           | 608.12                                | 0%        |
| 20000                | 120               | 31.515           | 634.62                                | 0%        |
| 30000                | 150               | 45.766           | 655.5                                 | 0%        |

Fuente: Elaboración propia.

#### - RNF-03 – Disponibilidad operativa

Durante el período de validación, el sistema se mantuvo operativo de forma continua, procesando las actualizaciones de ubicación y atendiendo las solicitudes administrativas sin interrupciones perceptibles. La ejecución exitosa de la totalidad de los casos de prueba asociados a los requisitos funcionales RF-01 a RF-05 evidencia que todos los servicios del sistema — *backend*, RabbitMQ, PostgreSQL y MinIO — permanecieron disponibles y coordinados durante el desarrollo de las pruebas, cumpliendo con el requisito de disponibilidad operativa definido.

#### - RNF-04 – Seguridad de acceso

Se verificó que el sistema restringe el acceso a sus funcionalidades mediante autenticación basada en tokens JWT. Para validar este mecanismo, se ejecutó una

solicitud al endpoint `POST /api/location-updates` sin incluir credenciales de autenticación, obteniendo una respuesta `HTTP 401 Unauthorized` en 14 ms, confirmando que el sistema deniega correctamente el acceso a usuarios no autenticados. Adicionalmente, la validación del proceso de autenticación demostró que el sistema emite tokens con roles y permisos correctamente asociados al usuario, habilitando el acceso únicamente a las funcionalidades correspondientes a su perfil. El endpoint de autenticación incorpora además mecanismos de protección contra ataques de fuerza bruta mediante limitación de intentos por IP y bloqueo temporal de cuentas ante intentos fallidos reiterados.

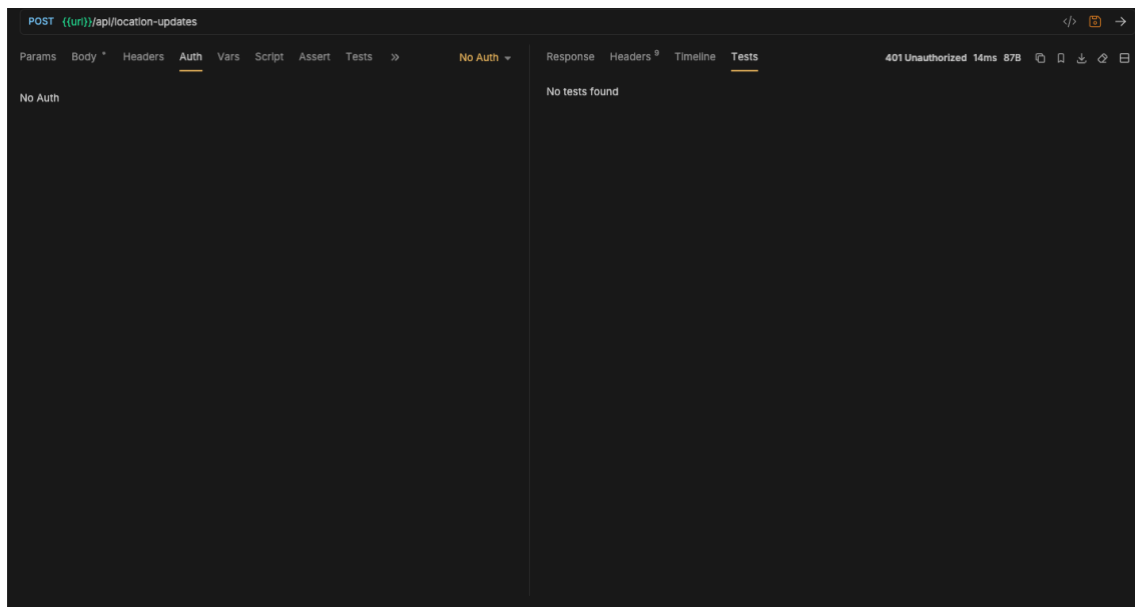


Figura 43. Error 401 Unauthorized al no enviar token.

Fuente: Elaboración propia.

## 5.4 Validación con el patrocinador

El sistema fue presentado y validado en instancias de revisión con Chilquinta Transmisión, donde se mostró su funcionamiento en tiempo real y se recibió retroalimentación sobre las funcionalidades implementadas. Asimismo, la solución fue exhibida en la Feria de Software de la Universidad Técnica Federico Santa María, permitiendo demostrar el sistema en un entorno operativo completo y en tiempo real.

La retroalimentación obtenida confirmó que el sistema cumple con el objetivo de proporcionar una herramienta centralizada para la supervisión de cuadrillas en terreno, mejorando la visibilidad operativa y apoyando la toma de decisiones del centro de control.

## 5.5 Síntesis de resultados

Los resultados obtenidos confirman el cumplimiento de los cinco requisitos funcionales y los cuatro requisitos no funcionales definidos para el sistema.

La recepción, persistencia y difusión de actualizaciones de ubicación opera correctamente de extremo a extremo, con una latencia inferior a 1 segundo entre el envío desde el dispositivo móvil y la recepción en el panel web, cumpliendo con el umbral de 2 segundos establecido en el objetivo general y en RNF-01.

La prueba de carga demostró que la arquitectura basada en el patrón *Outbox* y mensajería asíncrona absorbe ráfagas de hasta 30.000 mensajes concurrentes sin registrar errores, alcanzando un *throughput* máximo de 655 mensajes por segundo con un tiempo promedio de respuesta de 1,5 ms por solicitud.

La gestión de archivos geospaciales KMZ y la importación de la agenda operativa fueron validadas funcionalmente, verificando el ciclo completo de operaciones y la correcta asociación de datos operativos a las actualizaciones de ubicación.

Finalmente, los mecanismos de seguridad demostraron restringir el acceso exclusivamente a usuarios autenticados y autorizados, incluyendo protección contra ataques de fuerza bruta.

## 6 Conclusiones y trabajo futuro

### 6.1 Conclusiones

El presente trabajo de título tuvo como objetivo el diseño e implementación de un sistema de integración que permitiera habilitar el monitoreo en tiempo real de cuadrillas en terreno para Chilquinta Transmisión. A partir del desarrollo realizado y de los resultados obtenidos, se concluye que el objetivo general y los objetivos específicos planteados fueron cumplidos satisfactoriamente.

El sistema implementado permite recibir actualizaciones de ubicación desde una aplicación móvil mediante un esquema de mensajería asíncrona, procesar y almacenar dicha información de forma persistente y difundirla en tiempo real hacia un panel web de supervisión. Asimismo, se logró integrar información operativa relevante, como la agenda de las cuadrillas y archivos geoespaciales en formato KMZ, centralizando estos datos en un único componente de integración.

La arquitectura adoptada, basada en el desacoplamiento entre ingesta y procesamiento mediante un bróker de mensajería y el patrón *Outbox*, demostró ser adecuada para el propósito del proyecto. Esta decisión de diseño permitió alcanzar un *throughput* de 655 mensajes por segundo y absorber ráfagas de hasta 30.000 mensajes concurrentes sin registrar errores ni degradar los tiempos de respuesta. La comunicación en tiempo real mediante *WebSocket* y *SignalR* permitió una latencia de difusión inferior a 1 segundo en las pruebas realizadas, cumpliendo con el requisito de latencia máxima de 2 segundos establecido en el objetivo general. El uso de una arquitectura en capas permitió una clara separación de responsabilidades, favoreciendo la mantenibilidad, extensibilidad y comprensión del sistema dentro del alcance definido.

La principal contribución de este trabajo radica en la construcción de un componente de integración funcional que resuelve un problema operativo concreto: la falta de visibilidad en tiempo real sobre la ubicación y estado de las cuadrillas en terreno. Al centralizar la recepción, procesamiento y distribución de datos operativos en un único sistema desacoplado, la solución propuesta sienta las bases para que Chilquinta Transmisión pueda reducir tiempos de respuesta ante incidentes, mejorar la asignación de cuadrillas y mantener un historial auditable de las operaciones de campo. Si bien el sistema fue validado en un entorno de prototipo, su arquitectura modular y el uso de tecnologías estándar de la industria facilitan su transición hacia un entorno productivo.

La validación funcional del sistema se realizó tanto en un entorno de prueba controlado como en instancias de revisión con el patrocinador. En particular, el sistema fue

presentado y demostrado en funcionamiento durante la Feria de Software de la Universidad Técnica Federico Santa María, instancia en la cual se expuso la solución completa, integrada y operativa. Esta actividad permitió validar el sistema en un contexto académico-práctico, recibiendo retroalimentación directa de docentes, estudiantes y profesionales, lo que reafirmó la pertinencia y viabilidad de la solución desarrollada.

## **6.2 Limitaciones del trabajo**

El alcance del presente trabajo se centró exclusivamente en el sistema de integración, por lo que no se abordó el desarrollo de la aplicación móvil utilizada por las cuadrillas ni la implementación del panel web de monitoreo.

En cuanto a la validación, si bien se realizaron pruebas de carga que demostraron la capacidad de procesamiento del sistema bajo escenarios de alta concurrencia, la medición de latencia de extremo a extremo se basó en observaciones puntuales durante las pruebas funcionales y no en una instrumentación sistemática con muestreo estadístico. La incorporación de métricas automatizadas de latencia se identifica como una mejora necesaria para un entorno productivo. Asimismo, no se implementaron pruebas automatizadas, limitándose la validación funcional a pruebas manuales.

La integración de la agenda operativa se realizó mediante la importación manual de archivos Excel, sin implementar mecanismos de sincronización automática con sistemas externos. Estas limitaciones no afectan el cumplimiento de los objetivos del proyecto, pero deben ser consideradas al evaluar el sistema en un contexto productivo.

## **6.3 Trabajo futuro**

A partir del trabajo realizado, se identifican diversas líneas de evolución y mejora del sistema de integración.

En primer lugar, se contempla la implementación del sistema en las instalaciones de Chilquinta Transmisión, lo que permitiría validar su funcionamiento en un entorno corporativo real, considerando las políticas internas de infraestructura, seguridad y operación del patrocinador. Esta etapa incluiría el despliegue de los servicios en la infraestructura institucional y su integración con los entornos existentes.

En segundo lugar, se plantea la incorporación de datos provenientes de sistemas de GPS vehicular, en aquellos casos en que dicha información esté disponible. Esta integración

permitiría complementar los datos obtenidos desde la aplicación móvil, enriqueciendo la visión operativa del sistema y reduciendo la dependencia exclusiva del dispositivo móvil del técnico en terreno.

Una mejora relevante para la transición a producción es la integración con el ecosistema Microsoft de la organización. Actualmente el sistema gestiona la autenticación mediante cuentas locales con JWT; sin embargo, Chilquinta Transmisión utiliza cuentas corporativas Microsoft, por lo que la incorporación de un inicio de sesión mediante *Microsoft Entra ID* (anteriormente *Azure Active Directory*) permitiría a los usuarios autenticarse con sus credenciales institucionales existentes, eliminando la necesidad de gestionar cuentas independientes y simplificando la administración de accesos. Adicionalmente, esta integración habilitaría el acceso programático mediante Microsoft *Graph API* al archivo Excel compartido en el que las cuadrillas mantienen su agenda operativa a través de PowerApps. Esto permitiría reemplazar la importación manual actual por una lectura automática e incremental de la agenda directamente desde su fuente, resolviendo de forma conjunta las limitaciones identificadas tanto en la autenticación como en la sincronización de la agenda operativa.

Otra línea de trabajo futuro corresponde a la instrumentación de métricas y monitoreo del sistema, incorporando mecanismos que permitan medir de forma continua y automatizada la latencia de actualización, la tasa de procesamiento de mensajes y el uso de recursos, con el fin de evaluar y optimizar el rendimiento del sistema bajo distintos escenarios de carga en producción.

Finalmente, se consideran posibles extensiones funcionales, tales como mejoras en la visualización de información histórica, el enriquecimiento de los datos geoespaciales y la integración con otros sistemas corporativos, manteniendo siempre al sistema de integración como un componente desacoplado y reutilizable.

## **6.4 Agradecimientos**

A Jazmín, mi prometida, por ser mi principal apoyo y pilar fundamental durante todo este proceso.

A Aiko, por el ánimo constante en los momentos más exigentes.

A mi familia, por su apoyo permanente y confianza incondicional.

A Catalina y Waldo, por su amistad y compañía, incluso a la distancia.

Al profesor guía Óscar Carrasco, por su orientación y apoyo durante el desarrollo de este trabajo.

## 7 Bibliografía

- [1] G. H. y. B. Woolf, Enterprise Integration Patterns: Designing, Building and Messaging Solutions, Addison-Wesley, 2003.
- [2] Amazon Web Services, «Event-Driven Architecture,» [En línea]. Available: <https://aws.amazon.com/event-driven-architecture/>. [Último acceso: Febrero 2026]
- [3] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» 2000. [En línea]. Available: [https://roy.gbiv.com/pubs/dissertation/fielding\\_dissertation.pdf](https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf). [Último acceso: Febrero 2026].
- [4] R. E. y. S. B. Navathe, Fundamentals of Database Systems, Pearson, 2015.
- [5] C. Richardson, «Pattern: Transactional Outbox,» [En línea]. Available: <https://microservices.io/patterns/data/transactional-outbox.html>. [Último acceso: Febrero 2026].
- [6] R. C. Martin, «The Clean Code Blog,» 13 08 2012. [En línea]. Available: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>. [Último acceso: Febrero 2026].
- [7] M. Fowler, «Repository,» 2003. [En línea]. Available: <https://martinfowler.com/eaaCatalog/repository.html>. [Último acceso: Febrero 2026].
- [8] RabbitMQ, «RabbitMQ 4.2 Documentation,» [En línea]. Available: <https://www.rabbitmq.com/docs>. [Último acceso: Febrero 2026]
- [9] Amazon Web Services, «What is a RESTful API?,» [En línea]. Available: <https://aws.amazon.com/what-is/restful-api/>. [Último acceso: Febrero 2026]