

Diseño e Implementación del Backend Móvil para la Gestión de Información Crítica en Emergencias de Bomberos

Natán Abelardo Jasón Apablaza Valencia

natan.apablaza@usm.cl

Lais San Martín Navarro
Profesora Guía

Francis Fuentes
Profesora Correferente

Resumen: Este proyecto presenta el desarrollo de una aplicación móvil orientada a la gestión y difusión de planes de emergencia en entornos comunitarios e institucionales. La iniciativa surge de la necesidad de optimizar el acceso a información crítica durante situaciones de riesgo, mejorando tanto la preparación de los usuarios como la capacidad de respuesta de los equipos de emergencia, incluidos bomberos. La aplicación se construyó utilizando Flutter, debido a su capacidad multiplataforma y a su integración con el ecosistema de servicios en la nube de Firebase, lo que permite gestionar datos en tiempo real y garantizar disponibilidad incluso en contextos de conectividad limitada. El resultado es una herramienta digital que centraliza información de emergencia, facilita rutas de evacuación y fortalece los procesos de actuación frente a desastres.

Palabras Clave: Planes de emergencia, Información crítica, Flutter, Aplicaciones móviles, Firebase.



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: Diseño e Implementación del Backend Móvil para la Gestión de Información Crítica en Emergencias de Bomberos

Nombre del candidato(a): Natán Abelardo Jasón Apablaza Valencia

Carrera / Grado: Ingeniería en Informática

Campus: Sede Viña del Mar

Departamento: Electrónica e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, _Lais San Martín Navarro, 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: 13-01-2026 **Firma:**

Estudiante o Candidato(a):

Fecha: 13-01-2026 **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 Presentación del Desafío

En Chile, la normativa vigente exige que edificios, condominios, empresas e instituciones educacionales cuenten con un Plan de Emergencia que establezca procedimientos claros frente a una situación crítica. Estos planes incluyen la definición de responsabilidades, rutas de evacuación, puntos de encuentro seguros, identificación de materiales peligrosos y mecanismos de comunicación interna y externa.

A pesar de su relevancia, la realidad es que esta información rara vez está disponible en el momento de una emergencia. Generalmente los planes se encuentran en formato físico o en archivos dispersos, lo que dificulta su acceso inmediato por parte de equipos de primera respuesta, como Bomberos, Carabineros o personal de salud. La falta de disponibilidad de estos documentos compromete la efectividad en la toma de decisiones y, en casos extremos, puede poner en riesgo vidas humanas.

Específicamente, los Bomberos de Viña del Mar enfrentan la dificultad de acudir a emergencias sin contar con información previa sobre el lugar afectado, lo que retrasa las acciones iniciales y limita la eficiencia del plan de acción. Resolver este desafío es esencial para fortalecer la coordinación entre la Central de Comunicaciones, los bomberos en terreno y la comunidad en general.

1.1.1 Organización Operativa de Bomberos

Para entender cuál es el problema, es necesario conocer la estructura bajo la cual funcionan los Cuerpos de Bomberos en Chile. La operación se divide en dos actores: las Unidades en Terreno y la Central de Comunicaciones. La Central actúa como el cerebro operativo, es quien procesa la información inicial y toma las decisiones de despacho; también, gestiona recursos, monitorea estado de carros y provee apoyo logístico a los voluntarios. Por su parte, las Unidades en Terreno son las que acuden al lugar para atender la emergencia, dependiendo de los antecedentes y directrices asignadas por la Central de Comunicaciones.

1.2 Definición del Problema

Actualmente, los sistemas de gestión de planes de emergencia presentan diversas limitaciones que afectan su efectividad en situaciones críticas. Una de ellas es la fragmentación de la información, ya que los planes suelen encontrarse en formato físico o dispersos en múltiples documentos digitales, lo que dificulta su consolidación. A esto se suma el acceso no confiable, pues durante una emergencia los equipos de primera respuesta no disponen de un canal único y seguro para acceder a esta información esencial. Asimismo, existe una falta de interoperabilidad, dado que no se cuenta con mecanismos efectivos que conecten los datos de los planes con sistemas externos utilizados por Bomberos, como plataformas de geolocalización o aplicaciones de gestión de incidentes. Finalmente, se observa una baja usabilidad en terreno, debido a que la información no está sintetizada ni optimizada para su uso rápido en contextos de presión y riesgo.

Estas limitaciones provocan retrasos en la toma de decisiones, incrementan la exposición a riesgos y generan una respuesta menos coordinada frente a emergencias.

1.3 Propuesta de Solución General

La solución planteada consiste en FastPlan, una plataforma tecnológica compuesta por una aplicación web y una aplicación móvil que permite digitalizar, centralizar y distribuir los planes de emergencia de edificios. Desde la interfaz web, el Bombero de la Central puede definir un radio de riesgo en torno al siniestro, lo que permite identificar automáticamente qué planes cercanos deben ser enviados a la unidad en terreno. El sistema está diseñado para abordar las principales limitaciones actuales, proporcionando herramientas adaptadas a las necesidades de distintos actores involucrados en la gestión de emergencias.

Por un lado, entrega a la Central de Comunicaciones una herramienta para cargar, gestionar y enviar planes de emergencia en formato digital. Al mismo tiempo, ofrece a los bomberos en terreno acceso a información crítica (como grifos, salidas, materiales peligrosos y accesos especiales) en un formato visual e intuitivo, complementado con mapas interactivos y notificaciones en tiempo real. Con ello, FastPlan se orienta a mejorar la coordinación entre distintos actores, optimizar la toma de decisiones en terreno y fomentar la preparación ciudadana mediante el uso de tecnología y datos centralizados.

1.4 Aporte del Trabajo de Título

El presente trabajo de título se centra en el diseño e implementación del backend móvil de FastPlan, es decir, la capa técnica que soporta la aplicación utilizada por los bomberos en terreno. Este desarrollo constituye un componente esencial del sistema, ya que asegura que la información crítica llegue de manera oportuna y confiable a quienes la necesitan durante una emergencia.

El aporte específico de este trabajo considera la implementación de un backend robusto que garantice la recepción, procesamiento y entrega de información en dispositivos móviles tipo tablet. Además, incorpora la sincronización de datos y la mensajería en tiempo real entre la Central de Comunicaciones y los carros de bomberos, junto con el desarrollo de mecanismos de gestión de sesión persistente, lo que permite un acceso rápido sin necesidad de credenciales en cada inicio. También contempla la representación de información de manera visual e intuitiva, mediante iconografía y mapas interactivos optimizados para contextos de presión, y la exploración de integraciones con sistemas externos que asistan en la navegación hacia el lugar del siniestro.

En síntesis, el backend móvil constituye la columna vertebral técnica que posibilita a los bomberos recibir información clave en terreno de forma inmediata y confiable, aportando directamente a la solución global del sistema.

1.5 Objetivo General y Objetivos Específicos

Objetivo General

Diseñar e implementar el backend móvil de FastPlan, de manera que los bomberos en terreno puedan acceder en menos de 5 segundos a la información crítica de planes de emergencia desde dispositivos móviles, asegurando una disponibilidad del 99% y una interfaz optimizada para su uso en contextos de emergencia.

Objetivos Específicos

1. Diseñar la arquitectura del backend móvil integrando un servicio externo de sincronización en tiempo real, asegurando compatibilidad con la Central de Comunicaciones.
2. Implementar un módulo de mensajería en tiempo real que permita recibir y gestionar datos desde la Central de Comunicaciones con una latencia máxima de 5 segundos.
3. Desarrollar un sistema de autenticación y almacenamiento local seguro que permita guardar credenciales y datos relevantes de forma persistente, logrando que al menos el 95% de las pruebas de reconexión no requieran reingreso de credenciales.
4. Implementar mapas interactivos y elementos gráficos que representen información crítica (grifos, salidas, materiales peligrosos), de manera que al menos el 80% de estos elementos estén disponibles y funcionales en la aplicación móvil.
5. Realizar pruebas de robustez y confiabilidad mediante al menos 5 escenarios simulados de emergencia, alcanzando un 95% de éxito en la entrega de información sin errores críticos, antes del último mes del proyecto.

1.6 Metodología y Plan de Trabajo

El desarrollo del backend móvil se llevará a cabo bajo un enfoque ágil incremental, utilizando prácticas de Scrum. Esto permite entregar valor de forma temprana y ajustarse a los requerimientos que puedan surgir durante la implementación.

Las fases de trabajo durante el año 2025 son:

Levantamiento de requerimientos: análisis de necesidades específicas de los bomberos en terreno.

Diseño de arquitectura: definición de la estructura del backend y sus integraciones externas.

Implementación inicial: desarrollo de módulos de conexión con la plataforma seleccionada para comunicación en tiempo real y gestión de sesión.

Iteraciones incrementales: agregar funcionalidades de visualización, notificaciones y geolocalización.

Pruebas y validación: verificación de la robustez utilizando pruebas unitarias, herramientas de análisis estático de código y pruebas de carga simuladas.

Documentación y cierre: generación de documentación académica.



Figura 1: Diagrama básico que representa los tiempos de la metodología

Fuente: elaboración propia en Canva

1.7 Organización del Informe

El documento se estructura de la siguiente manera:

Capítulo 1: Introducción (contexto, problema, propuesta, objetivos y metodología).

Capítulo 2: Marco Conceptual (revisión de literatura sobre sistemas de emergencia, backend móvil y uso de tecnologías en primera respuesta).

Capítulo 3: Diseño de la Solución (arquitectura propuesta del backend móvil).

Capítulo 4: Implementación (detalles técnicos de desarrollo).

Capítulo 5: Pruebas y Validación.

Capítulo 6: Conclusiones y Trabajo Futuro.

Capítulo 7: Agradecimientos

Capítulo 8: Referencias

2 Marco Conceptual

2.1 Emergencia

La Real Academia Española (RAE) define una emergencia como una “situación de grave riesgo, catástrofe o calamidad pública que requiere la intervención coordinada de los poderes públicos y de los ciudadanos para la protección y socorro de personas y bienes”. [1] En términos generales, una emergencia se caracteriza por ser un evento súbito, inesperado y potencialmente peligroso, que interrumpe el desarrollo normal de las actividades y pone en riesgo la vida, la integridad de las personas, el medioambiente o la infraestructura.

En el ámbito de la gestión del riesgo, el concepto de emergencia se diferencia de otros como incidente o desastre: un incidente puede resolverse con recursos inmediatos disponibles, mientras que una emergencia exige una respuesta organizada y planificada, y un desastre implica un impacto mayor que supera la capacidad de respuesta local.

De esta manera, comprender qué es una emergencia resulta fundamental para el desarrollo de planes y sistemas que permitan minimizar sus efectos y optimizar los tiempos de respuesta, especialmente en contextos críticos como la labor de los cuerpos

de bomberos, donde la rapidez y la precisión en la entrega de información son determinantes para la seguridad de la comunidad.

2.2 Planes de Emergencia

Un plan de emergencia corresponde a un conjunto de procedimientos organizados y previamente establecidos que buscan enfrentar de manera eficiente situaciones imprevistas que ponen en riesgo la vida, la integridad de las personas o la infraestructura. Según el Ministerio de Vivienda y Urbanismo (MINVU), un plan de emergencia es "la planificación y organización humana para la utilización óptima de los medios técnicos previstos, con la finalidad de reducir al máximo las consecuencias económicas y humanas de una situación de emergencia". [2]

Los objetivos principales de un plan de emergencia son:

- Proteger la vida y la seguridad de las personas expuestas a un evento de riesgo.
- Reducir al mínimo las pérdidas materiales y los daños a la infraestructura.
- Establecer protocolos claros de acción que permitan una respuesta rápida y coordinada.
- Garantizar la continuidad de las operaciones esenciales una vez controlada la emergencia.

En la gestión del riesgo, los planes de emergencia son una herramienta clave de prevención y mitigación. Su utilidad radica en anticipar escenarios críticos, definir responsabilidades, optimizar la asignación de recursos y coordinar la comunicación entre los distintos actores involucrados. Por ello, los planes de emergencia son un componente esencial dentro de la seguridad ocupacional, la protección civil y la resiliencia comunitaria.

En el contexto nacional, la normativa chilena establece diversas obligaciones relacionadas con la elaboración e implementación de planes de emergencia, particularmente en edificios, empresas y espacios públicos.

La Ley N.º 21.442, sobre Copropiedad Inmobiliaria, en su Artículo 40, establece que los condominios y edificios deben contar con un Plan de Emergencia y Evacuación, aprobado por la autoridad competente, el cual debe considerar procedimientos de actuación frente a incendios, terremotos u otras situaciones que comprometan la seguridad de los residentes. [3] Específicamente, esta normativa exige que el plan incluya planos detallados que identifiquen la ubicación de las instalaciones de seguridad y combate de incendios, tales como la red seca, red húmeda, grifos y sistemas de corte de energía.

Asimismo, el Decreto Supremo N.º 594/1999 del Ministerio de Salud, que aprueba el Reglamento sobre Condiciones Sanitarias y Ambientales Básicas en los Lugares de Trabajo, exige a las empresas contar con medidas preventivas, planes de evacuación y sistemas de alarma en caso de emergencias, especialmente en recintos con alta concentración de trabajadores. [4] Esta reglamentación hace obligatorio mantener, identificar y señalizar elementos críticos como extintores, vías de evacuación despejadas y zonas de seguridad.

A nivel de protección civil, la Oficina Nacional de Emergencia (ONEMI, actualmente SENAPRED) ha elaborado manuales y guías técnicas que orientan a instituciones y organizaciones en la elaboración de planes de emergencia y continuidad operacional,

con el fin de estandarizar la respuesta ante catástrofes naturales y accidentes de gran magnitud.

En Chile, la existencia de planes de emergencia no es solo una buena práctica, sino también una obligación legal y reglamentaria, cuyo cumplimiento busca resguardar la vida de las personas y asegurar una respuesta eficiente ante eventualidades críticas. Asimismo, la normativa establece qué elementos físicos se registran y georreferencian, lo que define la información mínima que cualquier sistema de bases de datos para gestión de emergencias debe manejar.

2.3 Sistemas Similares Existentes

En el ámbito de la gestión de emergencias y seguridad, existen soluciones tecnológicas que buscan mejorar la comunicación y la disponibilidad de información crítica en situaciones de riesgo. Si bien cada plataforma tiene un enfoque distinto, su análisis resulta relevante para identificar buenas prácticas y áreas de mejora en la propuesta de FastPlan.

- **ViperGo:** aplicación utilizada actualmente por cuerpos de bomberos en Chile, cuyo objetivo principal es la gestión de alertas y el seguimiento de carros en tiempo real. Su fortaleza radica en la comunicación instantánea entre la Central y los equipos en terreno, aunque no contempla la integración de planes de emergencia detallados ni herramientas visuales específicas para cada recinto afectado.
- **LobbyControl:** desarrollada por Nicolás Uribe y Jorge Contreras, es un sistema orientado a la gestión de accesos en edificios, el cual incluye funcionalidades básicas de seguridad y monitoreo. Si bien no está enfocado en la respuesta a emergencias, representa un antecedente de cómo centralizar información sobre edificios y usuarios en tiempo real.
- **Smert:** es una evolución tecnológica creada por los mismos fundadores de LobbyControl, orientada a fortalecer la gestión integral de emergencias en instituciones públicas y privadas. Actualmente utilizada en lugares como el Hospital de Temuco, la plataforma incorpora herramientas que permiten identificar puntos críticos, ejecutar simulacros y elaborar reportes preventivos. Si bien ha logrado posicionarse como un referente nacional en sistematización de información preventiva, su alcance presenta limitaciones en cuanto a la respuesta operativa en tiempo real dentro del terreno de la emergencia.
- **Lumiform:** aplicación internacional centrada en listas de chequeo y protocolos de inspección en seguridad laboral. Ofrece formularios digitales y reportes automáticos, siendo útil en auditorías preventivas más que en emergencias.
- **FireMaint:** software especializado en el mantenimiento de sistemas contra incendios, que permite gestionar inspecciones y asegurar la operatividad de equipos de protección. Su alcance es más técnico-preventivo que operativo.

La mayoría de las herramientas revisadas priorizan la prevención, inspección o comunicación básica, sin integrar de forma completa los planes de emergencia en un flujo de trabajo entre Central de Comunicaciones y bomberos en terreno. La diferencia principal de FastPlan es que se centra en el momento inicial y crítico de la emergencia,

entregando información sintetizada, visual y en tiempo real a quienes la necesitan para salvar vidas.

2.4 Aplicaciones de Geolocalización

Las herramientas de geolocalización constituyen un pilar en la gestión moderna de emergencias, ya que permiten optimizar el desplazamiento de las unidades de primera respuesta y proporcionar visibilidad en tiempo real del lugar afectado.

- **Google Maps:** ofrece rutas optimizadas e información detallada de lugares. Sin embargo, su uso en emergencias puede verse limitado por su falta de redirección de tráfico cuando hay rutas más rápidas o tráfico congestionado.
- **Waze:** aplicación colaborativa que destaca por la rapidez en la actualización del estado vial, alertas comunitarias y rutas alternativas. En contextos de emergencia, puede reducir tiempos de llegada gracias a su enfoque en incidentes en tiempo real.
- **OpenStreetMap (OSM):** plataforma de código abierto que permite integrar mapas personalizables y añadir capas de información (ejemplo: ubicación de grifos o salidas de emergencia). Su flexibilidad lo convierte en una herramienta ideal para aplicaciones especializadas como FastPlan.

La integración de estas herramientas en sistemas de respuesta a emergencias debe considerar factores como la latencia en la actualización de datos, facilidad de integración vía API y el acceso offline en caso de pérdida de conectividad. En el caso de FastPlan, la incorporación inicial de OSM responde a su carácter abierto y personalizable, mientras que la integración con Waze busca mejorar la navegación en tiempo real hacia el sitio de la emergencia.

2.5 Interconexión Aplicación Web y Móvil

La comunicación entre la Central de Comunicaciones (aplicación web) y los bomberos en terreno (aplicación móvil) es un requisito crítico para garantizar la disponibilidad de datos en tiempo real. Existen diversos enfoques técnicos que permiten establecer esta interconexión:

- **APIs REST:** modelo ampliamente utilizado para la comunicación cliente-servidor. Su principal ventaja es la simplicidad y estandarización, aunque en contextos de tiempo real puede generar retrasos debido al modelo basado en peticiones.
- **WebSockets:** protocolo que habilita comunicación bidireccional en tiempo real entre cliente y servidor. Es más eficiente que las consultas periódicas llamadas polling, y reduce la latencia, lo que lo hace idóneo para aplicaciones críticas.
- **Firebase Realtime Database / Firestore:** servicios en la nube que permiten la sincronización automática de datos con baja latencia y alta disponibilidad. Firestore utiliza una estructura NoSQL basada en colecciones y documentos, bastante parecida al formato JSON, donde cada documento almacena información en pares clave-valor y puede contener objetos anidados. Cuenta con un plan gratuito limitado a 50.000 lecturas diarias; en entornos de producción se aplican costos adicionales de \$0,06 USD por cada 100.000 lecturas extra, por lo que es

necesario dimensionar correctamente el uso esperado antes de su implementación. [5]

Para FastPlan, la estrategia es usar Firestore, lo que asegura tanto la persistencia de datos como la inmediatez en la comunicación Central-Bomberos.

2.6 Tecnologías de Backend Móvil

En el desarrollo móvil, el concepto de backend no siempre se refiere únicamente a servidores externos, sino también a la lógica y servicios internos que permiten procesar datos dentro del propio dispositivo o en conexión con servicios en la nube. Entre las principales tecnologías destacan:

- **Flutter (con Dart):** Aunque Flutter es principalmente un framework de frontend multiplataforma, gracias a su arquitectura permite implementar lógica de negocio, manejo de estados, persistencia local (con SQLite o Hive) y comunicación directa con APIs. Esto le permite actuar como un “backend ligero” dentro de la aplicación móvil, delegando solo tareas críticas a servidores externos.
- **Swift (iOS):** En el ecosistema Apple, Swift no solo permite construir interfaces, sino también la lógica de negocio, validaciones, cifrado, almacenamiento seguro con Keychain, integración con CoreData y servicios locales. Muchas apps iOS implementan gran parte de su backend en el dispositivo, reduciendo la dependencia de servidores externos.
- **Kotlin (Android):** Kotlin, junto con Android Jetpack, permite implementar servicios internos, control de flujo, persistencia con Room Database, y acceso directo a APIs REST o GraphQL. Actúa como backend móvil al gestionar seguridad, procesamiento de datos y sincronización con la nube.
- **React Native (con JavaScript/TypeScript):** es un framework híbrido, es decir, se puede crear código tanto para web como móvil, el cual tiene la capacidad de backend que permite incluir lógica de negocio y manejar almacenamiento interno.
- **Xamarin (con C#):** al igual que React Native este también es híbrido, además, ofrece integración con librerías .NET para la gestión de procesos y bases de datos locales.

En este proyecto se eligió Flutter como framework principal debido a su capacidad de generar aplicaciones multiplataforma (Android e iOS) a partir de una única base de código, lo que reduce significativamente el tiempo y costo de desarrollo. Además, Flutter ofrece una amplia integración con servicios de backend como Firebase y Firestore, permitiendo gestionar datos en tiempo real, autenticación y almacenamiento de manera nativa y eficiente dentro de la aplicación móvil.

2.7 Tabla de Versiones y Compatibilidad

El desarrollo y despliegue de una aplicación móvil funcional requiere garantizar la compatibilidad entre los distintos componentes tecnológicos que la conforman. Esta sección presenta una tabla resumen con las versiones recomendadas de cada herramienta o servicio utilizado, asegurando su correcta integración, estabilidad y

soporte a largo plazo. Además, se consideran aspectos clave como la disponibilidad en entornos gratuitos (free tier) y la compatibilidad entre bibliotecas, frameworks y sistemas operativos. Esta información es crucial para evitar conflictos durante la implementación y para facilitar futuras actualizaciones o mantenimientos del sistema.

Tabla 1: Versiones recomendadas y compatibilidad entre tecnologías utilizadas en el desarrollo del sistema

Fuente: elaboración propia

Paquete/Framework	Versión	Breve Descripción
Flutter SDK	3.9.0	Framework principal para desarrollo multiplataforma.
firebase_core	4.0.0	Conecta la app con Firebase.
cloud_firestore	6.0.0	Base de datos NoSQL en tiempo real.
path_provider	2.1.5	Acceso a directorios del sistema (documentos, caché).
flutter_launcher_icons	0.14.4	Generación automática de íconos de app.
flutter_map	5.0.0	Visualización de mapas interactivos.
latlong2	0.9.0	Manejo de coordenadas geográficas.
connectivity_plus	6.0.3	Detección del estado de la red (online/offline).
google_fonts	6.3.1	Uso de tipografías de Google Fonts en la UI.
intl	0.18.0	Formateo de fechas, números e internacionalización.
flutter_test	SDK	Herramientas de testing de Flutter.
flutter_lints	5.0.0	Buenas prácticas y reglas de estilo de código.
url_launcher	6.1.10	Permite abrir enlaces externos y lanzar aplicaciones como el navegador, correo o GPS.
shared_preferences	2.5.3	Almacenamiento local persistente de datos simple en formato clave-valor.
http	1.5.0	Cliente para realizar peticiones HTTP hacia APIs o servicios remotos.
cached_network_image	3.0.0	Carga y caché de imágenes web con soporte para placeholders y manejo de errores.
mockito	5.4.0	Crea objetos simulados (mocks) durante las pruebas unitarias, permitiendo evaluar la lógica de negocio sin depender de servicios reales.
build_runner	2.3.3	Utilidad que ejecuta generadores de código de manera automática, necesaria para procesos como la creación de mocks.

2.8 Modelado UML y Conceptos Asociados

El Lenguaje Unificado de Modelado (UML) constituye una herramienta esencial para representar gráficamente los componentes y relaciones de un sistema complejo. En el contexto de FastPlan, los diagramas UML permiten planificar, documentar y comunicar la solución propuesta.



Los principales diagramas aplicados son:

- **Diagrama de casos de uso:** describe las interacciones entre los actores (Central de Comunicaciones, bomberos en terreno y ciudadanos) y el sistema.
- **Diagrama de clases:** modela las entidades clave y su comunicación dentro del código.
- **Diagrama de secuencia:** representa el flujo de mensajes entre la aplicación web, el backend móvil y Firebase durante el envío y recepción de un plan de emergencia.
- **Diagrama de despliegue:** muestra la infraestructura física y lógica, incluyendo servidores, base de datos y dispositivos móviles.

El uso de UML permite anticipar problemas de diseño, estandarizar la comunicación entre desarrolladores y asegurar que la arquitectura final se alinee con los objetivos planteados para FastPlan.

3 Diseño de la solución

Después de contextualizar el desafío operativo de los Bomberos de Viña del Mar y revisar las tecnologías relevantes en el marco conceptual, este capítulo presenta el diseño de la solución FastPlan, estableciendo los fundamentos técnicos que guían el desarrollo del backend móvil. Esta sección detalla la arquitectura general del sistema, la arquitectura específica de la aplicación y backend móvil, el modelo de datos en Firestore y los flujos esenciales del sistema.

3.1 Análisis de Riesgos y Mitigaciones

Algunos de los riesgos que se consideraron importantes de evaluar fueron posibles riesgos durante y posteriores al desarrollo del proyecto. Basándose en la matriz de riesgo (ver tabla 2), se utilizará como métrica la probabilidad de ocurrencia, medida en puntuación de uno (1) a cinco (5), y la gravedad del riesgo, medido con la misma puntuación. Estas métricas se multiplican entre sí para dar como total el impacto de riesgo. Se establecen tres niveles de prioridad que clasifican el resultado de este producto: Alta (mayor o igual a 15), que requiere acciones inmediatas; media (entre 6 y 12), riesgos operativos que demandan prevención y monitoreo; y baja (de 1 a 5), considerada de impacto menor y aceptable para la continuidad del servicio.

Riesgo 1: Pérdida de conectividad en terreno. La tablet del bombero en terreno puede perder la conexión a internet durante el desplazamiento o en el lugar de la emergencia. Esto podría impedir la recepción de datos vitales.

Gravedad: Crítica (5)

Probabilidad: Probable (4)

Impacto de riesgo: 20

Prioridad: Alta.

Estrategia: Mitigar el impacto mediante el diseño de una capa de persistencia de datos que utilice la caché local del dispositivo. Esto asegura que si el plan fue recibido inicialmente con conexión, permanecerá accesible contribuyendo al objetivo de alta disponibilidad.

Riesgo 2: Latencia en la recepción de la emergencia. Un retraso notorio entre el despacho desde la Central y la recepción en la tablet atenta contra el actuar de bomberos y la capacidad de prepararse antes de llegar al lugar del siniestro.

Gravedad: Mayor (4)

Probabilidad: Poco probable (2)

Impacto de riesgo: 8

Prioridad: Media.

Estrategia: Evitar la latencia mediante el diseño de una arquitectura reactiva tipo push. En lugar de un modelo pull (consultas periódicas HTTP/API), se debe

optar por servicios que utilicen listeners en tiempo real, asegurando que los datos sean "empujados" al dispositivo instantáneamente.

Riesgo 3: Pérdida de sesión del carro. Si la app se cierra o el sistema operativo la finaliza en segundo plano, forzar al bombero a reingresar los datos de inicio de sesión en medio de una emergencia es un fallo crítico.

Gravedad: Crítica (5)

Probabilidad: Poco probable (2)

Impacto de riesgo: 10

Prioridad: Media.

Estrategia: Evitar la interrupción del servicio implementando un sistema de sesión persistente. Tras el primer registro exitoso de la patente del carro, esta debe guardarse en el almacenamiento local seguro del dispositivo, permitiendo que la aplicación omita la autenticación en todos los inicios futuros.

Riesgo 4: Costos operativos Firebase. El diseño de consultas ineficiente, un error de lógica que genere bucles de lectura, o un evento de catástrofe que dispare miles de accesos simultáneos, podría generar costos operativos imprevistos y prohibitivos.

Gravedad: Mayor (4)

Probabilidad: Ocasional (3)

Impacto de riesgo: 12

Prioridad: Media.

Estrategia: Diseñar la estructura de la base de datos NoSQL desnormalizada para que la interfaz móvil pueda obtener toda la información de una emergencia, incluyendo el plan principal y los secundarios, con una sola operación de lectura.

Riesgo 5: Dependencia de la calidad de los datos de origen. La utilidad de la aplicación es nula si los datos del plan de emergencia son incorrectos u obsoletos. Decisiones en terreno basadas en esta información errónea podrían ser contraproducentes o peligrosas.

Gravedad: Crítica (5)

Probabilidad: Ocasional (3)

Impacto de riesgo: 15

Prioridad: Alta.

Estrategia: Incluir obligatoriamente en el diseño de la interfaz de la aplicación móvil la fecha de "última actualización" del plan, permitiendo al bombero evaluar la confiabilidad de la información que está visualizando.

Riesgo 6: Agotamiento de recursos del dispositivo (Batería). Al ser una aplicación de respuesta a emergencias, debe estar siempre disponible. Un diseño que mantenga listeners de datos activos o procesos en segundo plano de forma ineficiente podría agotar la batería de la tablet, dejándola inoperativa en el momento de un despacho.

Gravedad: Mayor (4)

Probabilidad: Ocasional (3)

Impacto de riesgo: 12

Prioridad: Media.

Estrategia: Diseñar la gestión de los listeners de datos para que se activen y desactiven siguiendo el ciclo de vida de la aplicación, para asegurar que no se ejecuten procesos de escucha innecesarios cuando la aplicación no está en primer plano.

Riesgo 7: Dependencia de servicios de mapas externos. La aplicación depende de proveedores de mapas de terceros para la visualización de elementos críticos georreferenciados. Si estos servicios cambian sus APIs, introducen costos, o sufren una caída, la funcionalidad de geolocalización de la app se vería críticamente comprometida.

Gravedad: Significativa (3)

Probabilidad: Poco probable (2)

Impacto de riesgo: 6

Prioridad: Media.

Estrategia: Diseñar la capa de visualización de mapas de forma modular y abstracta, de modo que el proveedor de mapas pueda ser intercambiado por otro sin requerir una reingeniería completa de la aplicación.

Estos riesgos, se representan gráficamente utilizando sus identificadores numéricos mediante la siguiente matriz.

Tabla 2: Matriz de riesgo basada en Probabilidad y Gravedad

Fuente: elaboración propia

Probabilidad/ Gravedad	Mínimo (1)	Menor (2)	Significativa (3)	Mayor (4)	Crítica (5)
Raro (1)					
Poco probable (2)			7	2	3
Ocasional (3)				6-4	5
Probable (4)					1
Casi seguro (5)					

3.2 Requisitos Funcionales

Para el diseño de la solución móvil, se establecieron los siguientes requisitos funcionales que el backend móvil debe satisfacer. Algunos de estos requisitos derivan tanto de los riesgos detectados, como de los objetivos específicos del proyecto (definidos en la tabla como OG, para el objetivo general, y OE para los específicos).

Tabla 3: Tabla de Requisitos Funcionales

Fuente: elaboración propia

ID	Nombre	Descripción	Objetivo Relacionado
RF-01	Autenticación Persistente	El bombero debe registrar su dispositivo mediante una patente y una clave única, manteniendo la sesión iniciada de forma persistente para evitar que el usuario tenga que autenticarse cada vez que abra la app.	OE-3
RF-02	Recepción de Emergencias	El bombero debe recibir en su dispositivo los despachos de emergencia asignados a la patente del usuario, con el fin de mostrar el plan principal y los planes secundarios con una latencia mínima.	OE-2
RF-03	Visualización Geoespacial	El bombero debe poder visualizar un mapa interactivo centrado en la ubicación de la emergencia, incluyendo las posiciones del plan principal y los planes secundarios.	OE-4
RF-04	Despliegue de información crítica	El bombero debe ser capaz de visualizar en el mapa los marcadores correspondientes a la información crítica del plan activo, representando al menos el 80% de los elementos disponibles.	OE-4
RF-05	Gestión de Estado	El bombero debe poder marcar una emergencia como "Atendida". Esta acción debe notificar al backend web y marcar el	OE-1

		carro como disponible a nivel de base de datos para futuras emergencias.	
RF-06	Operación Offline	La aplicación debe detectar automáticamente la pérdida de conectividad y mantener visible la información actual, mostrando además un indicador claro de estado offline para garantizar su disponibilidad continua.	OG
RF-07	Navegación Asistida	La aplicación debe permitir al bombero iniciar la navegación GPS hacia la ubicación de la emergencia mediante aplicaciones externas	OG.

3.3 Casos de Uso

Estos Casos de Uso, sirven para modelar el comportamiento del sistema móvil y mostrar como los requisitos funcionales se convierten en interacciones reales. El actor principal es llamado Bombero en Terreno (Bombero ET en el diagrama).

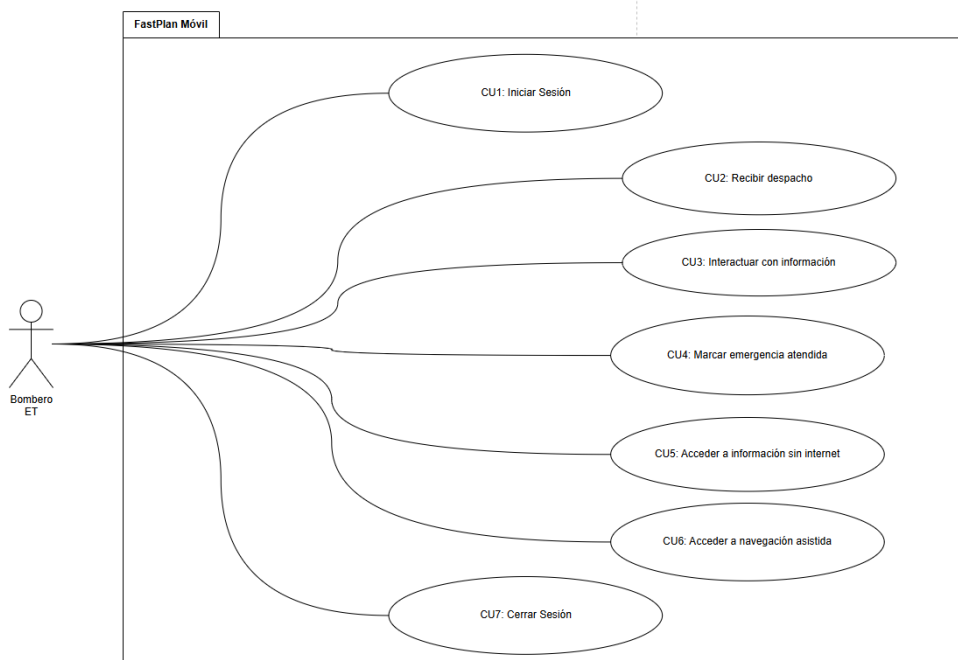


Figura 2: Diagrama que representa la interacción del bombero con los Casos de Uso
Fuente: elaboración propia en draw.io

La figura anterior muestra una visión general del sistema móvil, pero para comprender su lógica interna, es necesario profundizar en cada interacción. A continuación, se presentan las tablas detalladas de los Caso de Uso con precondiciones, flujos principales, flujos alternativos, postcondiciones y como se relacionan con los requisitos funcionales:

Tabla 4: Descripción del caso de uso: Iniciar Sesión
Fuente: elaboración propia

CU-01 Iniciar Sesión	
Actor	Bombero en terreno.
Precondiciones	La tablet debe tener conexión a internet.
Requerimiento Funcional Asociado	- RF-01.
Flujo principal	1. El bombero entra a la aplicación. 2. El sistema consulta al almacenamiento local si existen credenciales ingresadas. 3. El almacenamiento local no tiene registros asociados. 4. El sistema muestra la pantalla de bienvenida y luego da inicio. 5. El bombero ingresa la patente del carro con la clave registrada dentro de Firebase y presiona "Validar y Entrar". 6. El sistema guarda la patente en el almacenamiento local persistente. 7. El sistema redirige al home e inicia el listener de Firestore para esa patente.
Flujo alternativo	3.1 El almacenamiento local tiene registros asociados. 3.2 El sistema redirige al home e inicia el listener de Firestore para esa patente.
Postcondiciones	El bombero accede a la aplicación y puede recibir planes de emergencia según la patente registrada.

Tabla 5: Descripción del caso de uso: Recibir despacho
Fuente: elaboración propia

CU-02 Recibir despacho	
Actor	Bombero en terreno.
Precondiciones	- La sesión debe estar iniciada. - No hay emergencia activa. - Debe tener conexión a internet.
Requerimiento Funcional Asociado	- RF-02.
Flujo principal	1. La Central de Comunicaciones asigna una emergencia a la patente del carro desde la interfaz web y la información llega a Cloud Firestore. 2. El listener de la app recibe el nuevo objeto que contiene la información del siniestro. 3. El sistema actualiza su estado interno, estableciendo el plan principal del despacho como plan activo. 4. Bombero recibe el plan de emergencia en el dispositivo con toda la información relacionada.
Flujo alternativo	2.1 El dispositivo pierde la conexión a internet en el momento del despacho. 2.2 El listener no recibe el nuevo documento desde Firestore. 2.3 El sistema mantiene visible la última información guardada en memoria local.

Postcondiciones	El bombero accede a la información enviada desde la central sin necesidad de refrescar manualmente.
------------------------	---

Tabla 6: Descripción del caso de uso: Interactuar con información
Fuente: elaboración propia

CU-03 Interactuar con información	
Actor	Bombero en terreno.
Precondiciones	- La sesión debe estar iniciada. - Hay una emergencia activa.
Requerimiento Funcional Asociado	- RF-02. - RF-03. - RF-04.
Flujo principal	1. El Bombero observa el mapa, la iconografía de los elementos georreferenciados en el mapa y el detalle de todos los elementos críticos a la derecha. 2. El Bombero presiona un marcador de elemento crítico en el mapa. 3. El sistema muestra un modal con la información del elemento.
Flujo alternativo	No existe flujo alternativo.
Postcondiciones	La información de los elementos críticos es georreferenciada, interactiva y de fácil acceso.

Tabla 7: Descripción del caso de uso: Marcar emergencia atendida
Fuente: elaboración propia

CU-04 Marcar emergencia atendida	
Actor	Bombero en terreno.
Precondiciones	- La sesión debe estar iniciada. - Hay una emergencia activa.
Requerimiento Funcional Asociado	- RF-05.
Flujo principal	1. El Bombero ha finalizado con sus labores en terreno y considera que la emergencia ha concluido, por lo que presiona el botón "Atendida" en el banner superior de la pantalla principal. 2. El sistema muestra un diálogo de confirmación. 3. El bombero presiona "Confirmar". 4. El sistema realiza una llamada al repositorio del backend web para marcar el carro como disponible 5. El sistema llama al repositorio para eliminar el documento de despacho en Firestore. 6. El sistema muestra el Home con el mensaje "Sin emergencias activas".
Flujo alternativo	3.2 El bombero presiona Cancelar en el diálogo de confirmación. 3.3 El sistema descarta la acción y mantiene la emergencia activa. 4.1 La llamada al backend web falla.

	4.2 El sistema muestra un mensaje de error indicando que no se pudo actualizar el estado del carro. 4.3 La emergencia permanece activa hasta que la operación pueda ser reintentada.
Postcondiciones	- El documento en Firestore se elimina. - El listener del móvil recibe un null. - La interfaz de la app vuelve automáticamente al estado "Sin emergencias activas".

Tabla 8: Descripción del caso de uso: Acceder a información sin internet

Fuente: elaboración propia

CU-05 Acceder a información sin internet	
Actor	Bombero en terreno.
Precondiciones	- La sesión debe estar iniciada. - Hay una emergencia activa. - En un contexto inicial, se debe tener conexión a internet
Requerimiento Funcional Asociado	- RF-02. - RF-03. - RF-06.
Flujo principal	1. El bombero recibe el plan de emergencia en su tablet. 2. El sistema almacena localmente la información del plan y del mapa visualizado. 3. El Sistema Operativo pierde conexión a internet por cualquier motivo. 4. El Sistema muestra la información del plan de emergencia y el mapa que están almacenados localmente.
Flujo alternativo	2.1 El plan de emergencia no estaba previamente almacenados localmente. 2.2 El sistema no cuenta con datos locales para mostrar. 2.3 Se despliega un mensaje indicando que no hay información disponible sin conexión. 2.4 El sistema continúa intentando reconectar en segundo plano.
Postcondiciones	El bombero tiene acceso a la información en entornos de difícil acceso a internet.

Tabla 9: Descripción del caso de uso: Acceder a navegación asistida

Fuente: elaboración propia

CU-06 Acceder a navegación asistida	
Actor	Bombero en terreno.
Precondiciones	- La sesión debe estar iniciada. - Acceso a internet. - La opción de Ubicación en el Sistema Operativo debe estar habilitada. - La información del plan de emergencia está desplegada
Requerimiento Funcional Asociado	- RF-07.
Flujo principal	1. El bombero necesita saber cuál es la ruta para llegar al siniestro.

	2. El bombero selecciona la opción "Llegar en Waze". 3. El sistema almacena las coordenadas de la ubicación actual y del destino. 4. El sistema abre la aplicación Waze con las coordenadas almacenadas anteriormente.
Flujo alternativo	2.1 La tablet no cuenta con Waze instalado. 2.2 El sistema muestra un mensaje de error y no abre nada.
Postcondiciones	El bombero cuenta con la ruta óptima para llegar al destino según Waze.

Tabla 10: Descripción del caso de uso: Cerrar Sesión
Fuente: elaboración propia

CU-07 Cerrar Sesión	
Actor	Bombero en terreno.
Precondiciones	- La sesión debe estar iniciada. - Acceso a internet.
Requerimiento Funcional Asociado	- Sin requerimiento funcional asociado.
Flujo principal	1. El bombero en terreno desea cambiar la tablet de un carro al otro. 2. El bombero selecciona la opción cerrar sesión para posteriormente usar las credenciales de otro carro. 3. El sistema borra el archivo creado en el Inicio de Sesión. 4. El sistema destruye las vistas y redirige al Login.
Flujo alternativo	Sin flujo alternativo.
Postcondiciones	La tablet puede ser usada para diferentes unidades de bomberos sin necesidad de ser exclusiva de una.

3.4 Visión General

Para comprender el funcionamiento integral de FastPlan y cómo se gestiona la comunicación entre la Central de Comunicaciones y los bomberos en terreno, es necesario presentar la arquitectura general del sistema. FastPlan no es únicamente una aplicación móvil; su operación depende de un ecosistema de componentes que trabajan de forma coordinada para permitir el registro, gestión, despacho y sincronización de los planes de emergencia en tiempo real.

El sistema contempla una plataforma web utilizada por la Central de Comunicaciones, y una aplicación móvil destinada a los bomberos en terreno. A continuación, se describen los principales componentes de ambas plataformas:

Frontend web: Es la interfaz visual que utiliza el Bombero de la Central de Comunicaciones. Desde esta plataforma se accede a las funcionalidades operativas de FastPlan, como la búsqueda de planes, la administración de carros y la carga de nuevos planes con sus elementos críticos georreferenciados.

Backend web: El intermediario principal entre el frontend, la base de datos y Firebase. Gestiona la lógica del sistema, procesa las solicitudes de la Central y publica los despachos hacia Firebase. Está desplegado mediante contenedores Docker para asegurar su portabilidad y escalabilidad.

Base de datos MySQL: Guarda la información estructurada del sistema, como los usuarios, las unidades registradas, el historial de planes y otros metadatos importantes. Se optó por un motor SQL debido a su solidez transaccional y su capacidad para mantener la integridad referencial entre las entidades.

Firebase: Es la plataforma encargada de la sincronización inmediata con la aplicación móvil y gestiona las credenciales de acceso. Almacena los planes de emergencia listos para despacho. Su arquitectura permite una latencia muy baja y listeners hacia los dispositivos en terreno.

Aplicación móvil: Es la herramienta operativa del bombero en terreno. Recibe en tiempo real los planes de emergencia desde Firestore, los visualiza de forma geográfica y ofrece las funciones clave, como la navegación asistida, la lectura de elementos críticos y la gestión del estado de la emergencia.

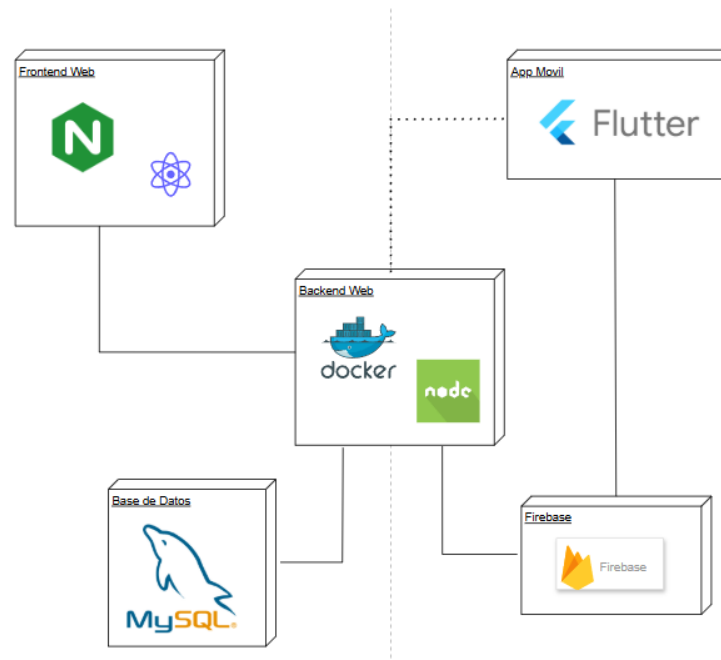


Figura 3: Diagrama de despliegue que muestra el sistema FastPlan
Fuente: elaboración propia en draw.io

3.5 Diseño de Modelo de Datos

Es importante aclarar que, aunque Firebase Firestore cumple un rol central en el almacenamiento y sincronización de datos, este no actúa como el backend del sistema, sino que proporciona servicios administrados en la nube; como base de datos, autenticación y sincronización en tiempo real que complementan la arquitectura propia

del sistema. La lógica de negocio, el procesamiento del despacho y la gestión operativa son responsabilidad de las capas internas de la aplicación móvil.

Asimismo, es fundamental describir la estructura concreta del modelo de datos utilizado que debe tratar el backend móvil. La información almacenada en Firestore se organiza en colecciones de documentos, donde cada documento contiene pares clave-valor y puede incluir subcolecciones, lo que permite tener estructuras jerárquicas flexibles. [6] En este caso, cada documento de la colección “planes” se identifica por la patente del carro y contiene toda la información necesaria para desplegar el plan de emergencia en tiempo real. Su estructura está organizada principalmente en torno a tres componentes:

- Metadatos del despacho, como la fecha y hora de emisión.
- Plan de emergencia principal, el cual contiene la información crítica del siniestro.
- Planes secundarios, con la información colindante al siniestro según el radio de acción.

La naturaleza flexible de Firestore permite que estos documentos evolucionen sin romper compatibilidad, lo cual es especialmente útil considerando que los planes de emergencia pueden variar según el tipo de edificio e información disponible. A continuación, se presenta la descomposición del documento almacenado en la colección planes, explicando los detalles de sus objetos internos y los campos relevantes mediante tablas explicativas.

Tabla 11: Documento guardado en Firebase

Fuente: elaboración propia

Clave	Tipo de dato	Descripción
fecha_envio	String	Fecha y hora en que la Central de Comunicaciones envía la información hacia la aplicación móvil.
plan_principal	Objeto	Contiene la información central del lugar del siniestro. Incluye todos los datos críticos necesarios para la respuesta operativa.
planes_secundarios	Array	Arreglo que almacena otros planes de emergencia ubicados dentro del radio de acción cercano al siniestro. Cada elemento del arreglo posee la misma estructura del objeto plan_principal.

Dado que planes_secundarios replica la estructura de plan_principal, se describe únicamente este último en la tabla siguiente.

Tabla 12: Objeto plan_principal del documento

Fuente: elaboración propia

Clave	Tipo de dato	Descripción
id_plan	Number	Identificador único del plan de emergencia dentro de la base de datos.
nombre_lugar	String	Nombre descriptivo del lugar asociado a la dirección completa (ej. “Mall Marina”).

direccion	String	Dirección completa del lugar del siniestro.
elementos_criticos	Array	Lista de objetos que representan los elementos críticos del lugar, junto con su información operativa relevante.
fecha_actualizacion	Date	Fecha y hora de la última actualización del plan de emergencia.
fecha_creacion	Date	Fecha en que se creó el plan de emergencia en el sistema.
tipo_edificio	String	Tipo de infraestructura, en formato texto (ej. escuela, edificio, condominio, etc.).
latitud	Number	Coordenada geográfica (latitud) correspondiente a la ubicación del siniestro.
longitud	Number	Coordenada geográfica (longitud) correspondiente a la ubicación del siniestro

Tabla 13: Array elementos_criticos del objeto plan_principal

Fuente: elaboración propia

Clave	Tipo de dato	Descripción
ubicacion	String	Ubicación referencial dentro del edificio o sector (ej. "Segundo piso, sector sur"). No corresponde a una dirección formal.
observaciones	String	Son comentarios o notas adicionales del elemento crítico
tipo_elemento	String	Tipo de elemento crítico en formato texto (ej. grifo, zona de evacuación, tablero eléctrico).
latitud	Number	Coordenada geográfica de la ubicación del elemento crítico.
longitud	Number	Coordenada geográfica de la ubicación del elemento crítico.

Ahora, de forma general, el documento almacenado en la colección planes, se vería tal que así:

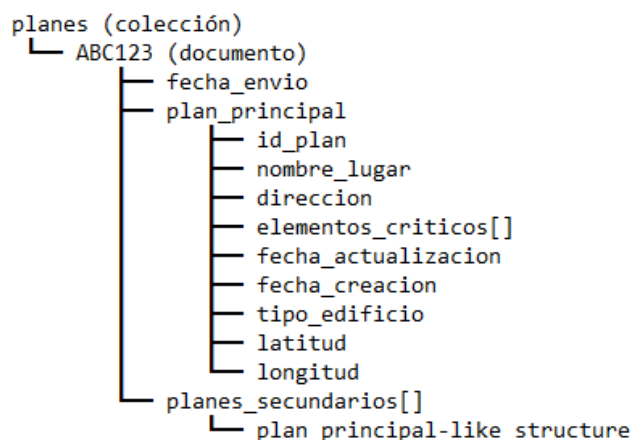


Figura 4: Estructura del documento almacenado en la colección planes de Firestore
Fuente: elaboración propia en bloc de notas

3.6 Flujo de Datos hacia el Dispositivo Móvil

Dado que se ha presentado la visión general del sistema y qué información se almacena en Firestore, es fundamental comprender la ruta que sigue un plan de emergencia desde su selección en la plataforma web hasta su visualización final en la aplicación móvil. Esta interacción entre componentes constituye el núcleo operativo y da soporte al objetivo principal del sistema: entregar la información crítica del siniestro al bombero en terreno con la menor latencia posible. Es este flujo de datos que se explica a continuación:

1. **Búsqueda del Siniestro:** El bombero de la Central de Comunicaciones ingresa a la sección “Planes de Emergencia” y realiza la búsqueda de la dirección asociada al siniestro. El frontend web procesa esta solicitud y retorna la información del plan disponible para esa ubicación.
2. **Seleccionar enviar:** Luego de identificar el plan adecuado, el bombero selecciona la opción de envío según el carro bomba que se desee. Esta acción indica al sistema que debe despachar la información del siniestro hacia los equipos en terreno.
3. **Envío de identificadores:** El frontend web envía al backend web los identificadores necesarios para completar la operación, tales como la patente del carro que atenderá la emergencia, el identificador del plan principal y los identificadores de los planes secundarios asociados.
4. **Solicita datos:** Tras recibir estos identificadores, el backend web consulta a la base de datos para obtener toda la información relevante del plan de emergencia y los elementos críticos vinculados. Esta consulta retorna los datos necesarios para preparar el despacho hacia los dispositivos móviles.
5. **Publicar Plan en Firebase:** Una vez recibida la información desde la base de datos, el backend web construye un objeto JSON con la estructura definida para la aplicación móvil. Este JSON es enviado a Firebase Firestore, donde se almacena dentro de la colección planes, utilizando como identificador del documento la patente del carro.
6. **Suscripción al documento:** La aplicación móvil, correspondiente al carro despachado, mantiene una suscripción activa al documento de Firestore asociado a su patente. Cuando Firebase detecta la actualización generada en el paso anterior, envía automáticamente un snapshot con la nueva información al cliente móvil. Al recibir este snapshot actualizado, la aplicación procesa el contenido,

genera el modelo de datos correspondiente y finalmente actualiza la vista, mostrando el plan de emergencia vigente al bombero en terreno.

Todo lo descrito se puede observar de manera más gráfica mediante la siguiente figura:

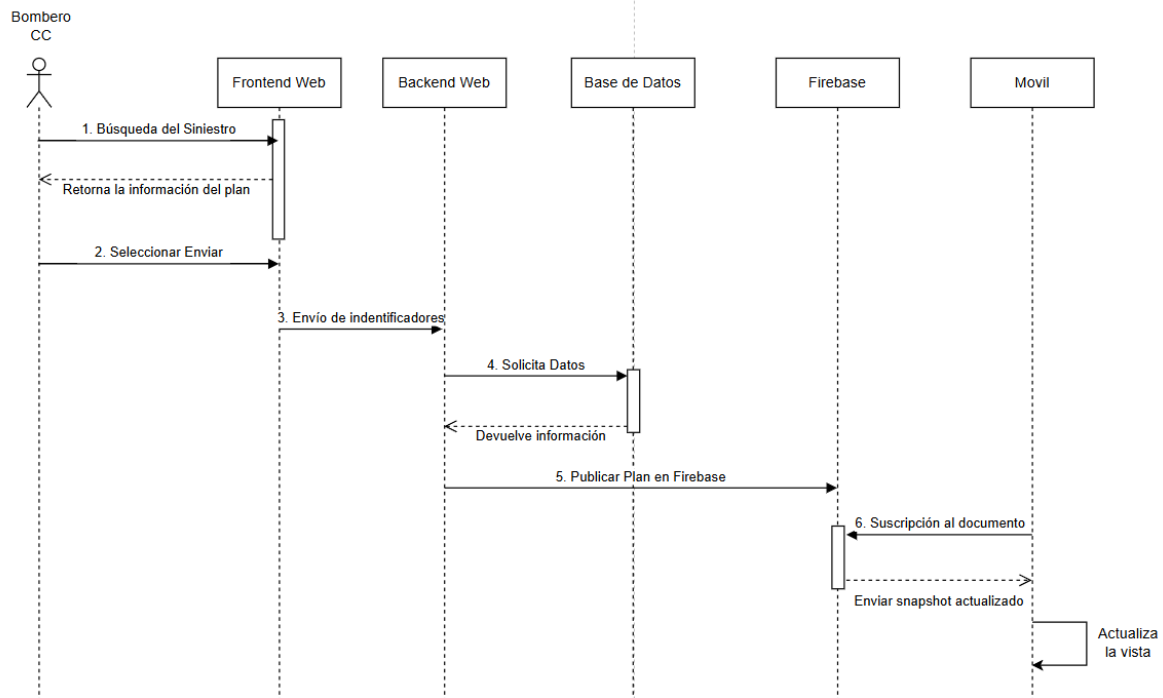


Figura 5: Diagrama de secuencia que retrata el flujo de información desde la Central de Comunicaciones hasta la aplicación móvil

Fuente: elaboración propia en draw.io

3.7 Arquitectura Backend Móvil

El fundamento de este punto está dado por un enfoque por capas inspirado en los principios de Clean Architecture, con el objetivo de garantizar separación de responsabilidades, alta mantenibilidad y testabilidad. [7] Dado que la aplicación opera en un contexto crítico de respuesta a emergencias, esta estructura permite que el sistema evolucione sin comprometer su estabilidad y facilite la incorporación de nuevas tecnologías o cambios futuros.

La división entre Presentación, Dominio y Datos permite que la interfaz de usuario, la lógica de negocio y el manejo de información evolucionen de manera independiente. [7] La capa de Dominio define los modelos y contratos centrales, la de Datos implementa estas reglas interactuando tanto con Firestore como con el almacenamiento local, y la capa de Presentación gestiona las vistas y widgets que reaccionan a los flujos de información en tiempo real. La organización de esta arquitectura mejora la claridad interna, permite realizar pruebas unitarias sin depender directamente de Firebase o Flutter y aumenta la confiabilidad del sistema.

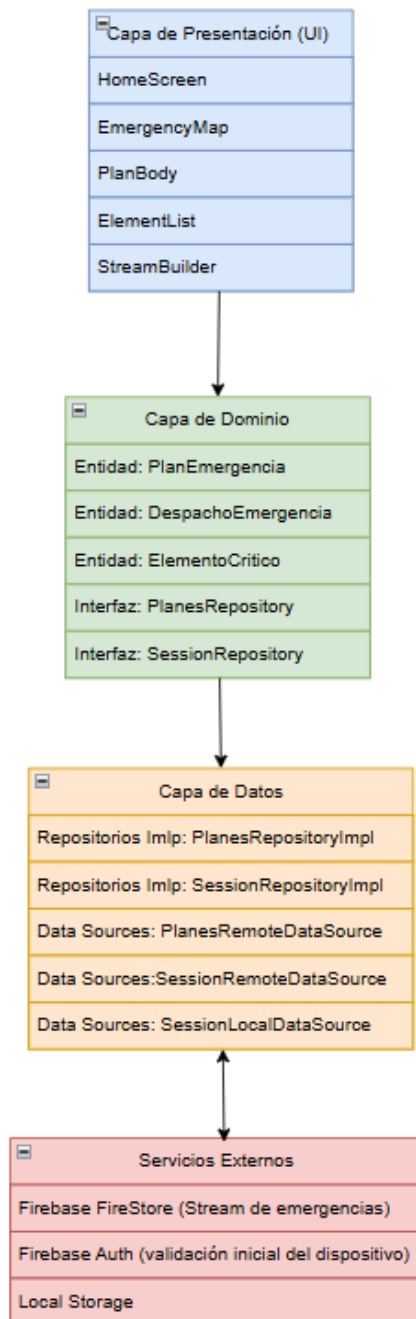


Figura 6: Representación de los principios Clean Architecture del backend móvil

Fuente: elaboración propia en draw.io

3.8 Diseño de Autenticación y Sesión Persistente

El mecanismo de autenticación debe equilibrar seguridad y usabilidad en un contexto operativo donde el acceso debe ser inmediato, pero limitado estrictamente a dispositivos autorizados. Para cumplir con este requisito, se evita el ingreso frecuente de credenciales, pero también se descarta cualquier solución que dependa únicamente del

almacenamiento local, dado que no permite verificar la legitimidad del dispositivo ni garantiza control centralizado.

Para resolver este desafío, se diseñó un sistema de autenticación híbrido que combina un servicio de validación centralizado con una sesión persistente en el dispositivo. Firebase Authentication actúa como proveedor de identidad, creando un usuario por cada carro bomba mediante un identificador basado en la patente y una contraseña robusta de uso interno. El almacenamiento local solo conserva un indicador de que el dispositivo ya fue validado, sin guardar contraseñas ni información sensible, lo que permite mantener rapidez sin comprometer la seguridad. [8]

El flujo de autenticación inicial ocurre solo la primera vez que la aplicación se ejecuta en la tablet: al no encontrar una sesión previa, la app solicita patente y contraseña y envía estas credenciales directamente a Firebase Auth mediante `signInWithEmailAndPassword`. Si la validación es exitosa, se almacena localmente el indicador de sesión y Firebase gestiona su token persistente; si no, se rechaza el acceso. Una vez autenticado el dispositivo, el usuario es redirigido a la pantalla principal, iniciando la suscripción al plan de emergencia correspondiente.

Registro de Carro

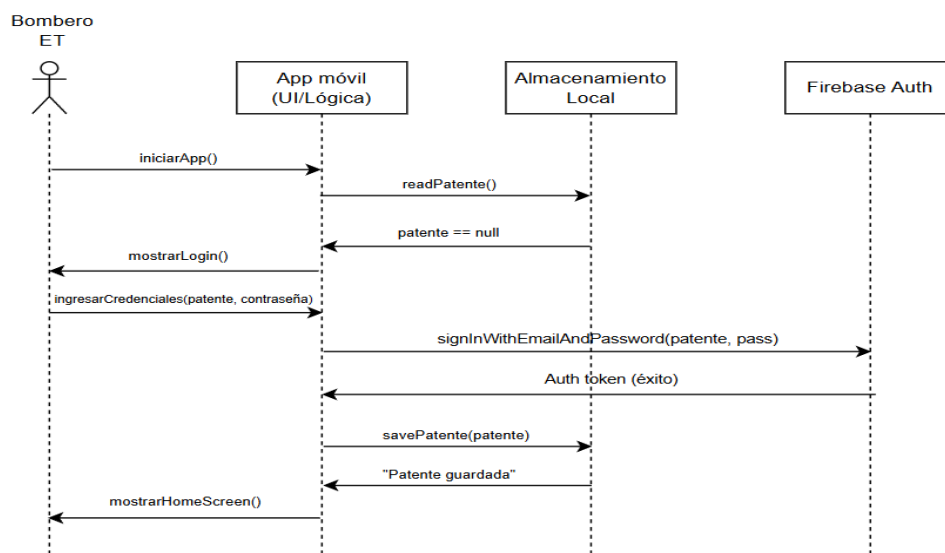


Figura 7: Diagrama de Secuencia que retrata la validación de credenciales del dispositivo

Fuente: elaboración propia en draw.io

El segundo flujo corresponde al inicio subsecuente de la aplicación, que será el comportamiento predominante durante la operación real. En esta circunstancia, la aplicación verifica si existe una patente registrada en el almacenamiento local. Si la encuentra, deduce que el dispositivo ya fue validado previamente, omite la pantalla de autenticación y dirige al usuario directamente al HomeScreen, lo que cumple con el requisito de acceso inmediato en situaciones de emergencia. De forma paralela, la aplicación utiliza la patente almacenada para suscribirse a los documentos de Firestore correspondientes al carro, lo que permite recibir despachos en tiempo real. En segundo

plano, el SDK de Firebase Auth renueva y valida su token interno sin intervención del usuario, asegurando una sesión segura y persistente.

Carro Registrado

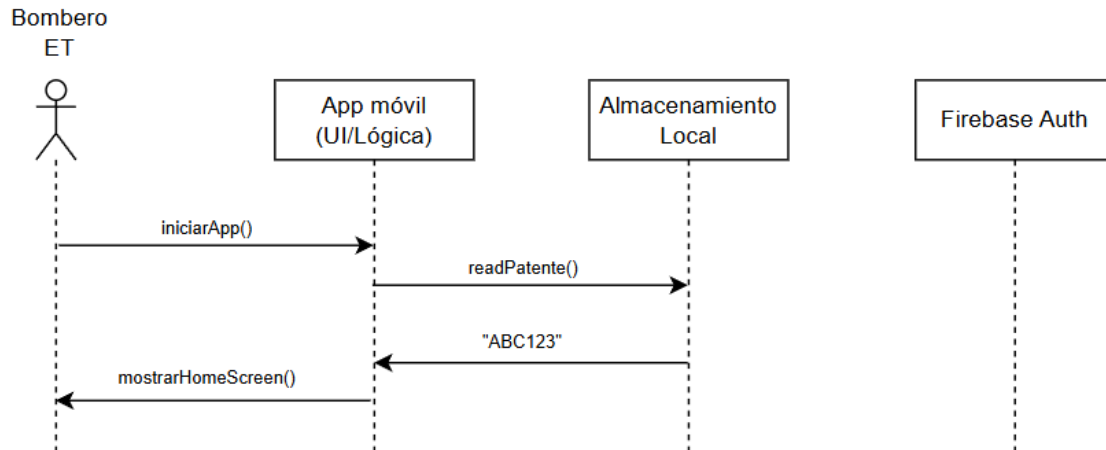


Figura 8: Diagrama de Secuencia que retrata el acceso a Home cuando el dispositivo ya ingresó sus credenciales de acceso

Fuente: elaboración propia en draw.io

Este diseño permite garantizar que cada tablet sea un dispositivo identificado y autorizado usando una validación inicial robusta, mientras que la persistencia local de la sesión asegura la rapidez de acceso necesaria en el contexto operativo de los bomberos. De esta manera, se cumplen con los requisitos de seguridad, disponibilidad e inmediatez que demanda su entorno de uso.

3.9 Diagrama de Componentes

El siguiente diagrama visualiza la conexión entre la aplicación móvil con Firebase Authentication, Firestore y el backend del área web.

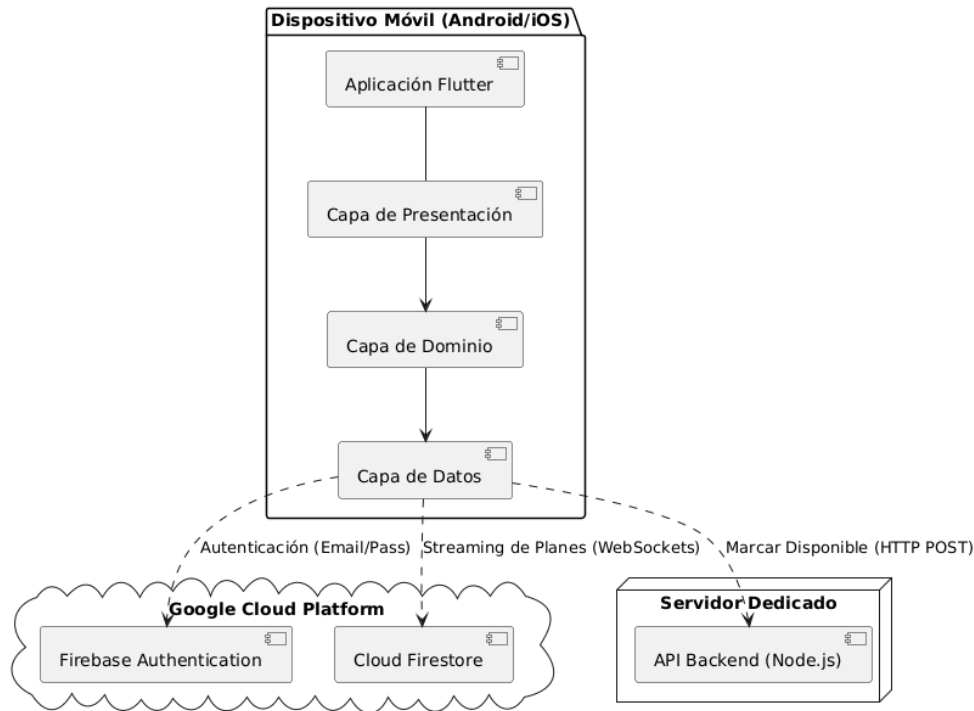


Figura 9: Diagrama de Componentes que destaca la modularidad de la aplicación móvil y sus interacciones con servicios externos

Fuente: elaboración propia en PlantUML

3.10 Clases

Tomando en cuenta el diseño de arquitectura propuesto para la aplicación, se detallarán las clases más importantes involucradas por cada capa.

3.10.1 Capa de Presentación

Esta capa agrupa los componentes visuales (llamadas widgets) y la lógica de interfaz encargada de la interacción con el usuario.

Onboarding: Pantalla inicial de carga.

- initState(): Ejecuta la lógica de arranque, verificando si existe una sesión activa para redirigir automáticamente al Home o al Login

Login: Gestiona el formulario de autenticación.

- patenteController/passwordController: Controladores que capturan el texto ingresado al repositorio de sesión.

- `_iniciarSesión()`: Método asíncrono que orquesta la validación de credenciales invocando al repositorio de sesión.

HomeScreen: Estructura principal de la aplicación post-login.

- `patente`: Almacena el identificador de la unidad operativa actual para gestionar las suscripciones de datos.

PlanBody: Widget contenedor que gestiona el estado de la emergencia activa.

- `_registrarMetricas(despacho)`: Función técnica encargada de calcular y registrar la latencia de llegada de los datos y el tamaño del payload para las pruebas que se harán posteriormente.

EmergencyLayout: Layout que organiza la información visual de una emergencia activa.

- `onAtendida`: Callback que se ejecuta cuando el bombero confirma que la emergencia ha finalizado.

EmergencyMap: Componente de visualización geoespacial.

- `Despacho`: Objeto de datos que contiene las coordenadas para renderizar el plan principal, secundarios y elementos críticos en el mapa

ElementList y **EmergencyBanner**: Widgets especializados en mostrar el listado textual de elementos críticos y la cabecera informativa del siniestro respectivamente.

3.10.2 Capa de Dominio

Define los contratos y modelos puros, sin dependencias de librerías externas. [7]

SessionRepository: Interfaz que establece el contrato para la gestión de usuarios.

- `login()`: Define la operación de autenticación remota.
- `savePatente()` / `readPatente()`: Definen la persistencia local de la sesión.
- `clearSession()`: Define el cierre de sesión y limpieza de datos.

PlanesRepository: Interfaz que establece el contrato para la gestión de datos operativos.

- `listenToPlan()`: Establece el flujo de datos en tiempo real para recibir despachos.
- `deletePlan()`: Operación para eliminar/cerrar un despacho activo.
- `marcarCarroDisponible()`: Operación transaccional para actualizar el estado de la unidad en el backend.

PlanEmergencia y **DespachoEmergencia**: Modelos de datos que encapsulan la información del negocio.

- `fromFirestore(data)`: Factory Method que actúa como adaptador, transformando y validando los datos crudos provenientes de la base de datos en objetos tipados y seguros.

industria. A continuación, se detalla su aplicación específica en el contexto del backend móvil.

3.11.1 Patrón Repository

Para mantener una separación clara entre la lógica de negocio y las operaciones de acceso a datos, se implementó el patrón Repository. Esto permite que la aplicación interactúe con sus fuentes de datos mediante contratos bien definidos, evitando que las capas superiores dependan de detalles específicos de almacenamiento o comunicación externa. [9]

En la capa de dominio se establecieron interfaces como `PlanesRepository` y `SessionRepository`, las cuales describen las operaciones requeridas por el sistema (como escuchar planes o gestionar sesiones). Su implementación concreta se encuentra en la capa de datos, donde clases como `PlanesRepositoryImpl` y `SessionRepositoryImpl` coordinan el acceso a los `DataSources` remotos y locales.

Este patrón responde al Principio de Inversión de Dependencias, permitiendo reemplazar o modificar la tecnología utilizada (por ejemplo, migrar desde Cloud Firestore a otro servicio en la nube) sin alterar la lógica de la aplicación ni su interfaz. [7]

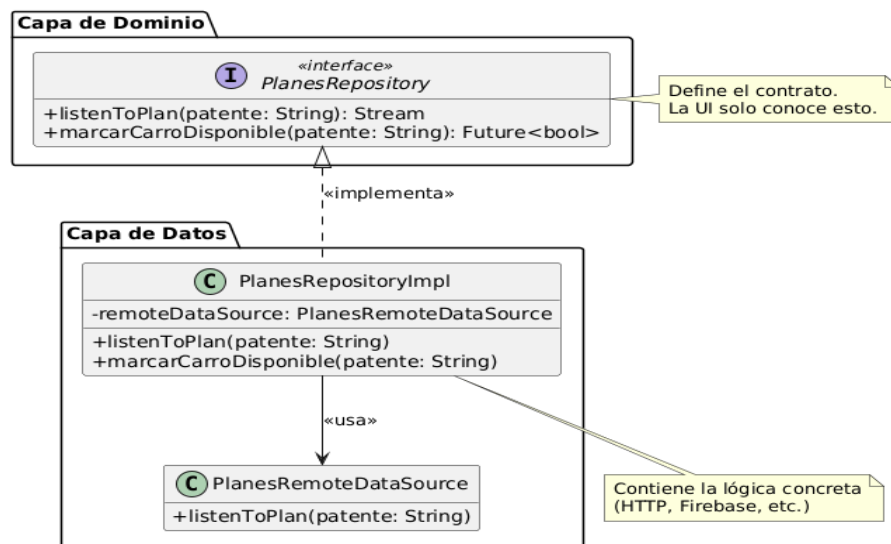


Figura 11: Diagrama de Clases que muestra la encapsulación de la creación del objeto
Fuente: elaboración propia en PlantUML

3.11.2 Patrón Observer

Como se necesita reaccionar inmediatamente a la información enviada desde la Central de Comunicaciones, se utilizó el patrón Observer mediante el uso de Streams. Este mecanismo permite mantener una suscripción activa a los cambios publicados en Firestore, de modo que, cuando un documento es actualizado, la aplicación recibe una notificación automática y actualiza la interfaz sin intervención del usuario. [10]

Este comportamiento se concreta en métodos como `listenToPlan`, que se vinculan a los cambios del documento asociado a la patente del carro. La utilización de Streams elimina

la necesidad de consultas periódicas, optimiza el consumo de red y reduce la latencia en la entrega de información crítica. [11]

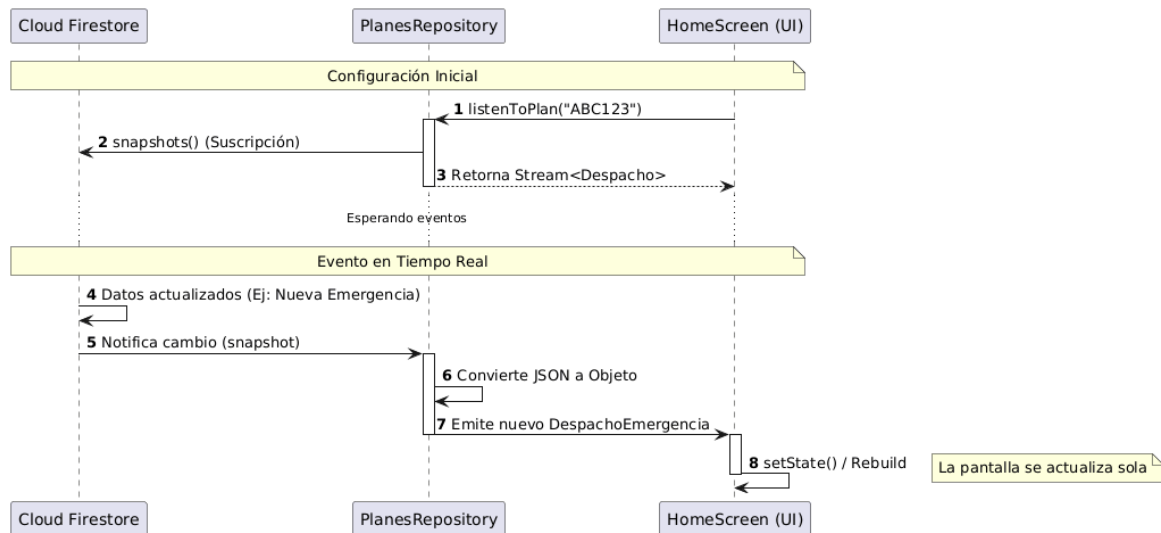


Figura 12: Diagrama de Secuencia que retrata la comunicación entre el home, PlanesRepository y Cloud Firestore

Fuente: elaboración propia en PlantUML

3.11.3 Patrón Factory

Para manejar de forma consistente la creación de objetos de dominio a partir de datos no estructurados, se utilizaron métodos de fábrica estáticos. Este patrón permite encapsular la lógica de transformación y validación de los documentos obtenidos desde Firestore. [10]

Ejemplos de esto son los métodos `PlanEmergencia.fromFirestore` y `DespachoEmergencia.fromFirestore`, los cuales se encargan de traducir estructuras JSON, asignar valores por defecto en caso de ausencia de datos y convertir tipos específicos de Firebase a tipos nativos de Dart. Con esto se garantiza que los objetos consumidos por la capa de presentación sean siempre válidos, homogéneos y seguros.

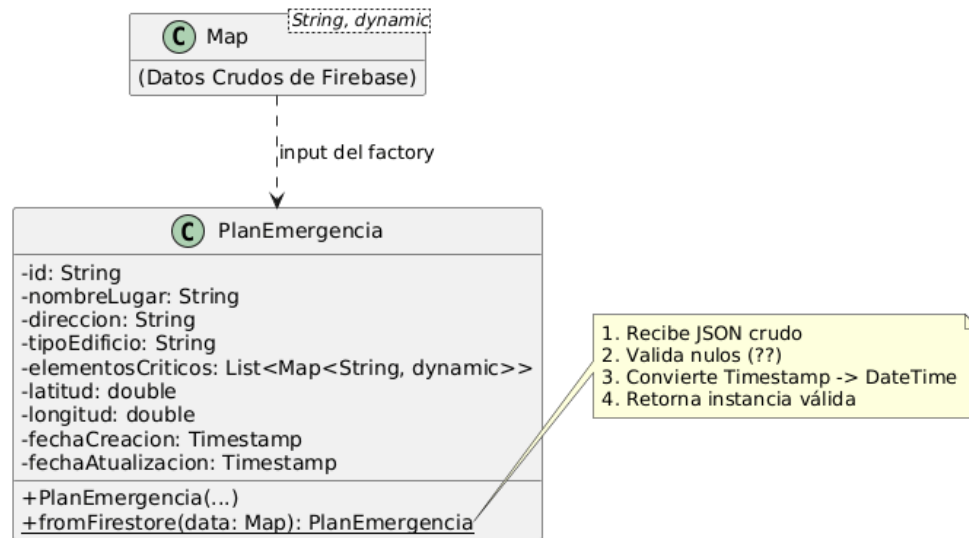


Figura 13: Diagrama de Clases que muestra el uso del Patrón Factory

Fuente: elaboración propia en PlantUML

3.11.4 Inyección de Dependencias

Para reducir el acoplamiento entre clases y mejorar la capacidad de prueba del sistema, se aplicó el principio de Inyección de Dependencias. Las clases reciben sus colaboraciones a través del constructor, lo que facilita sustituirlas por implementaciones alternativas durante las pruebas unitarias. [12]

Este patrón se observa tanto en la construcción de los repositorios como en los widgets principales. Por ejemplo, `SessionRepositoryImpl` recibe instancias de `SessionLocalDataSource` y `AuthRemoteDataSource`; del mismo modo, las pantallas reciben repositorios ya configurados, permitiendo controlar su comportamiento de forma aislada en entornos de prueba.

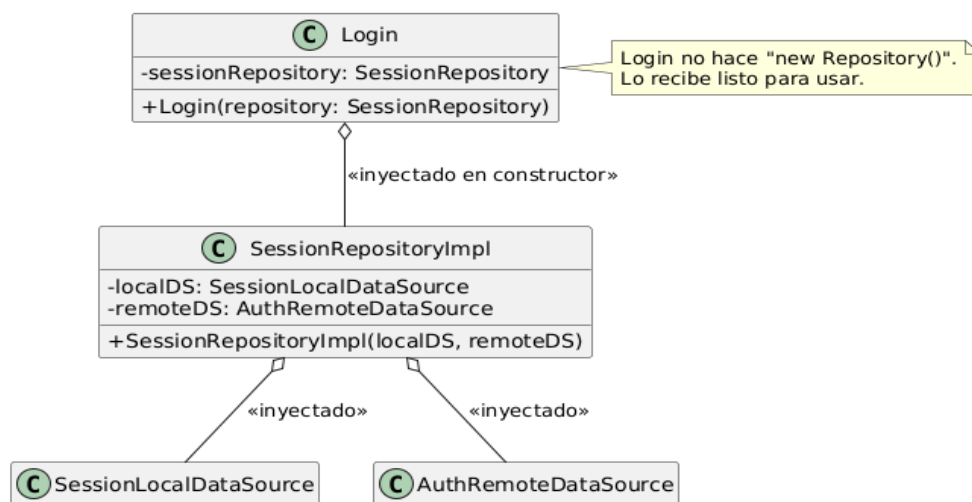


Figura 14: Diagrama de Clases que muestra el uso del patrón de Inyección de dependencias

Fuente: elaboración propia en PlantUML

3.12 Boceto de la Interfaz de Mapas e Información Crítica

El diseño de la interfaz de mapas debe permitir que la información crítica almacenada en Firestore pudiera interpretarse de forma inmediata, integrándose de manera coherente con la arquitectura y requisitos funcionales. El diseño organiza la información en dos vistas complementarias:

1. Ubicado en la mitad inferior de la pantalla, está el Mapa principal, donde se despliega la ubicación del plan principal, los planes secundarios y todos los elementos críticos contenidos en el documento recibido desde Firestore. Cada marcador se construye a partir de los datos de latitud y longitud definidos en el objeto plan_principal y en su arreglo de elementos críticos. Al lado de este mapa, hay un panel lateral en formato de lista, que presenta la misma información de manera textual, agrupada por tipo de elemento. Esta agrupación permite obtener una lectura rápida del entorno operativo sin depender únicamente de la interpretación visual del mapa.
2. Ubicado en la parte superior de la pantalla, un panel para interacciones rápidas, acordes a la dinámica operativa del bombero en terreno. El usuario puede ver la información relevante del plan de emergencia, iniciar la navegación asistida hacia el siniestro con Waze, y dar por atendida la emergencia.

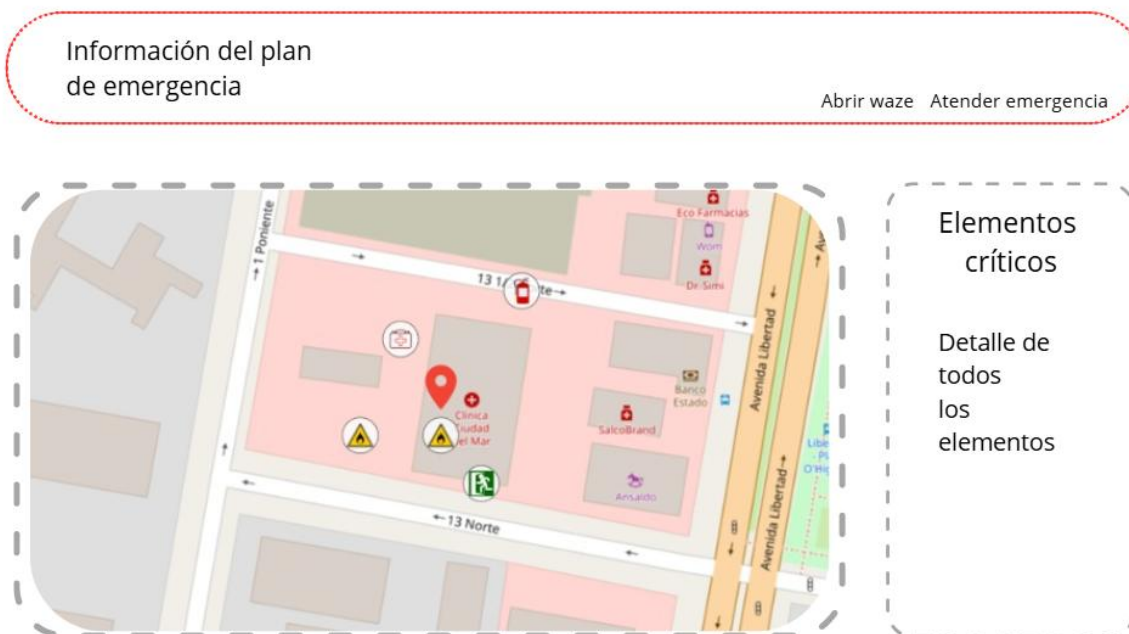


Figura 15: Esqueleto del home cuando existe una emergencia

Fuente: elaboración propia en Canva

4 Implementación

En este capítulo se detalla la materialización de la arquitectura propuesta, enfocándose en demostrar como lo desarrollado está acorde a lo definido en el capítulo de Diseño, como los desafíos de comunicación en tiempo real, seguridad en la autenticación y consistencia de datos.

4.1 Módulo de Autenticación y Gestión de Sesión

A continuación, se muestra el módulo de autenticación y sesión persistente acorde a lo definido en el capítulo 3.8.

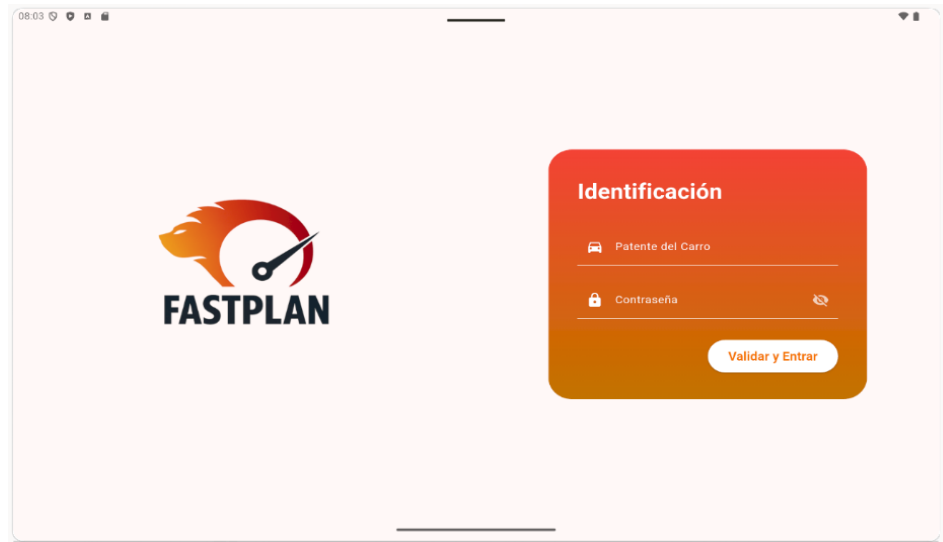


Figura 16: Pantalla del Login cuando no se ha ingresado ninguna credencial

Fuente: screenshot de la aplicación real

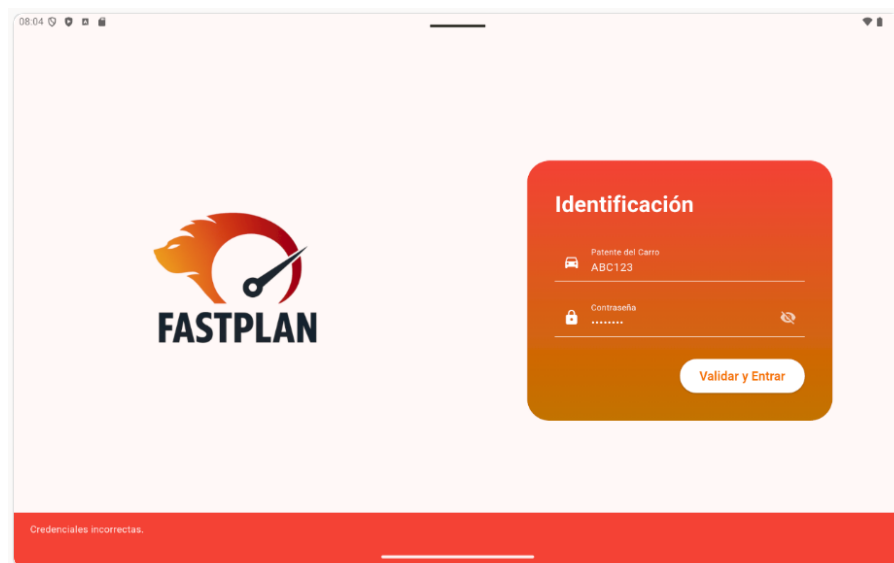


Figura 17: Pantalla del Login cuando se ingresan credenciales inválidas

Fuente: screenshot de la aplicación real

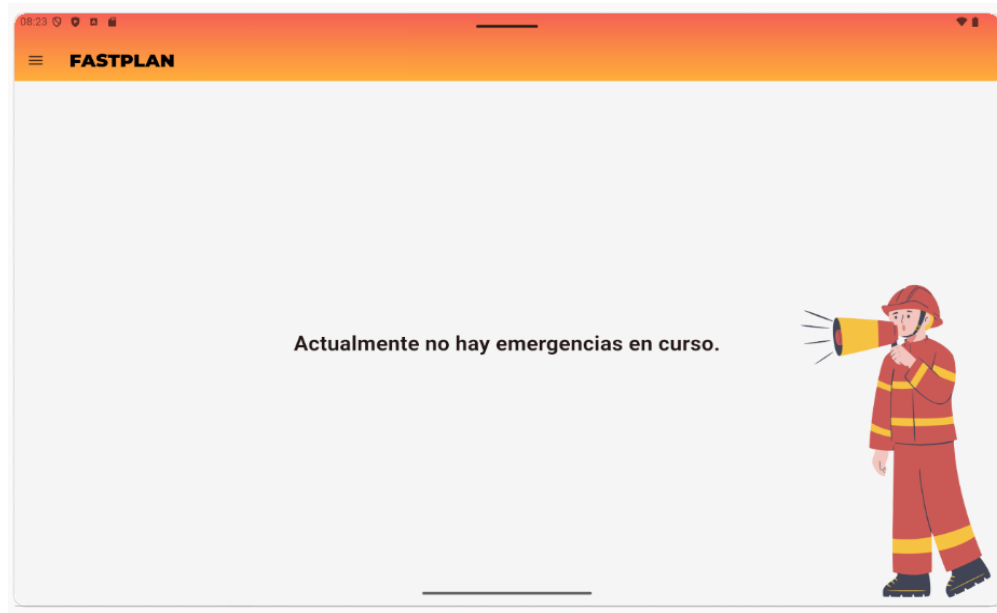


Figura 18: Redirección al Home producto del login exitoso
Fuente: screenshot de la aplicación real

Como comentario adicional, esta última figura también es producto de haber entrado a la aplicación cuando ya se inició sesión anteriormente.

4.2 Sincronización de Datos en Tiempo Real

El flujo reactivo de Firestore explicado en el capítulo de Diseño, permite que el despliegue de información ocurra prácticamente al instante, cumpliendo con los tiempos de respuesta requeridos por la operación en terreno.

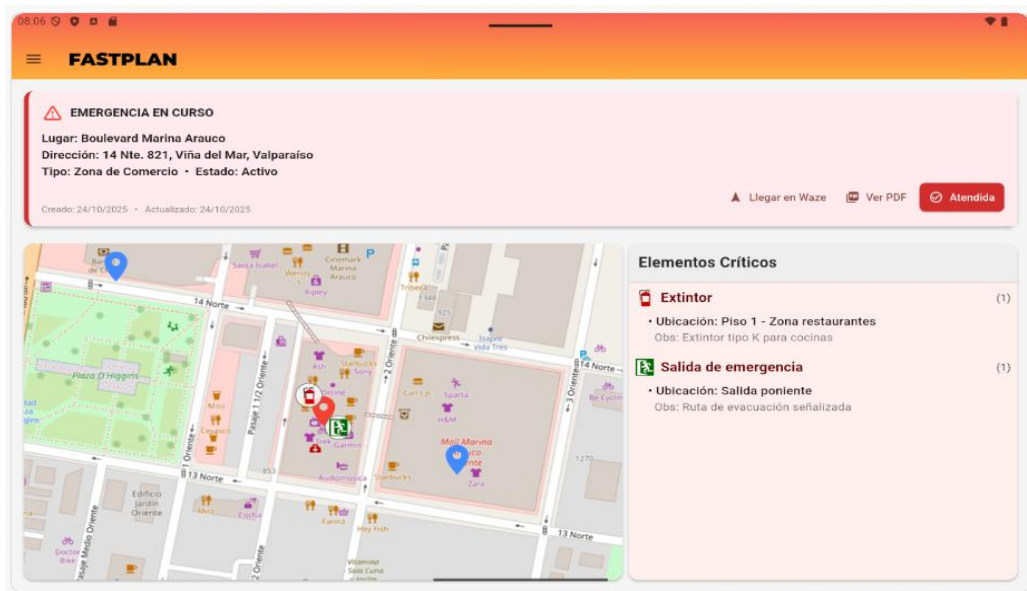


Figura 19: Home de la aplicación cuando existe una emergencia en curso
Fuente: screenshot de la aplicación real

Para verificar que los tiempos cumplen con los estándares de una emergencia y del objetivo general, lea el subcapítulo 5.2.1.

4.3 Gestión de Estados Operativos

Además de recibir información desde la Central, la unidad debe reportar su propio estado operativo. Cuando la tripulación finaliza una emergencia, la aplicación permite marcar la unidad como disponible mediante una acción explícita en la interfaz.

A diferencia de la escucha en tiempo real, esta operación requiere comunicación directa con el backend web. Por ello, se implementó un flujo de escritura a través de una solicitud HTTP POST realizada desde PlanesRepositoryImpl. El sistema interpreta el resultado de la operación según el código de estado HTTP. Si la solicitud se completó exitosamente (código 200), se confirma al usuario que el cambio fue aceptado por la Central. En cambio, si hay errores en el servidor o hay un problema con la request enviada desde el backend móvil, entonces se informa al usuario que la operación no fue registrada, evitando inconsistencias entre el estado visible en el dispositivo y el estado registrado en la aplicación web. Este mecanismo mantiene la coherencia entre la aplicación móvil y la plataforma web de gestión.

A continuación, se demuestra la gestión de estados operativos de los carros bomba en el Sistema FastPlan



The screenshot displays the FastPlan web application interface. On the left is a sidebar with navigation links: 'Central de Comunicaciones', 'Planes de Emergencia', 'Carros de la central', 'Subir Plan en PDF', and 'Cerrar Sesión'. The main content area is titled 'Despacho para: Universidad Tecnica Federico Santa Maria Sede Viña del Mar' with the address 'Dirección: Av. Federico Sta. Maria 6090' and a 'Volver a la lista' link. It features a 'Mapa de Zona de Peligro' with a radius of 0.25 km, showing a map of the area. Below the map is the 'Información del Plan' section, which includes 'Fecha de creación: 10-11-2025', 'Fecha de actualización: 10-11-2025', and 'Tipo de edificio: Escuela'. On the right, the 'Unidades Disponibles' section lists 13 units (U-1 to U-13) with their respective patent numbers and operational status (e.g., 'Disponible', 'No Disponible'). Each unit has an 'Enviar Información' button.

Unidad	Patente	Estado	Acción
U-1	ABCD12	Disponible	Enviar Información
U-2	KJTR34	No Disponible	Enviar Información
U-3	ZXCV56	Disponible	Enviar Información
U-4	A1B2C3	Disponible	Enviar Información
U-11	QWERT12	Disponible	Enviar Información
U-12	PLMNB34	Disponible	Enviar Información
U-13	ZXCVB09	Disponible	Enviar Información

Figura 20: Listado de unidades con su estado de disponibilidad operativa

Fuente: screenshot de la aplicación web

En este caso la tablet ingresó con las credenciales de la unidad con patente "ABCD12" (demás está destacar que esta patente registrada de bomberos no es real), y al seleccionar enviar información el estado de la unidad queda en "No Disponible".

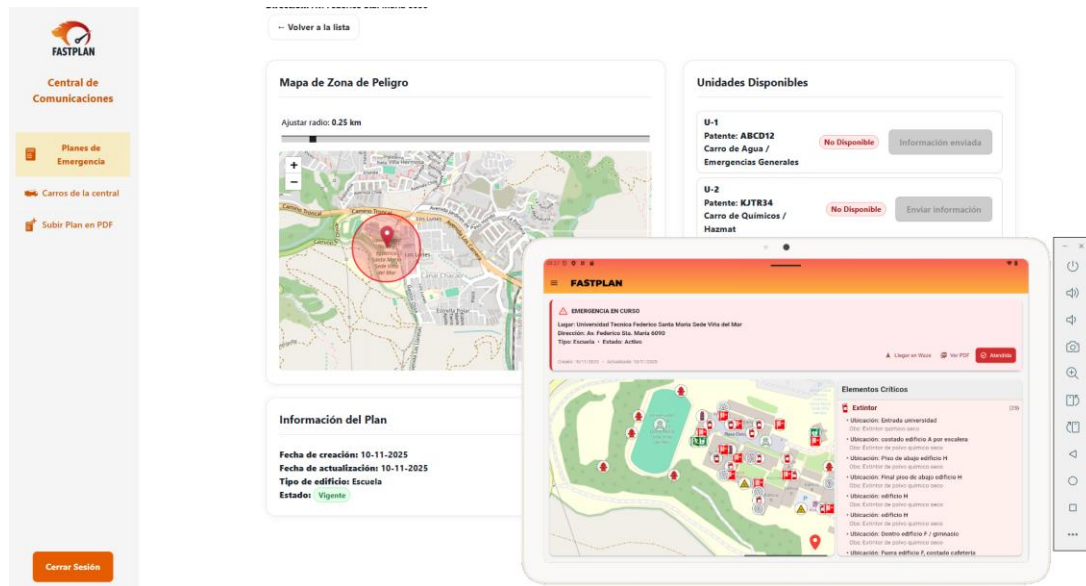


Figura 21: Visualización de la información en la tablet y el cambio de disponibilidad al recibirla

Fuente: screenshot del sistema FastPlan

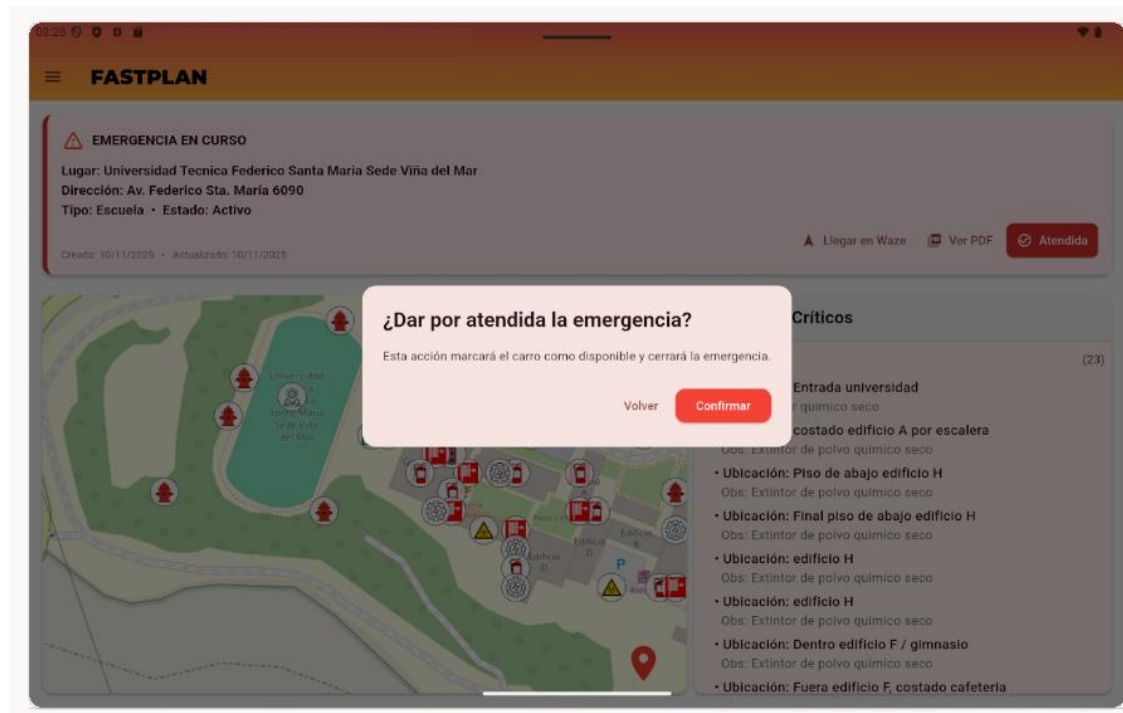


Figura 22: Confirmación para dar atendida la emergencia

Fuente: screenshot de la aplicación real

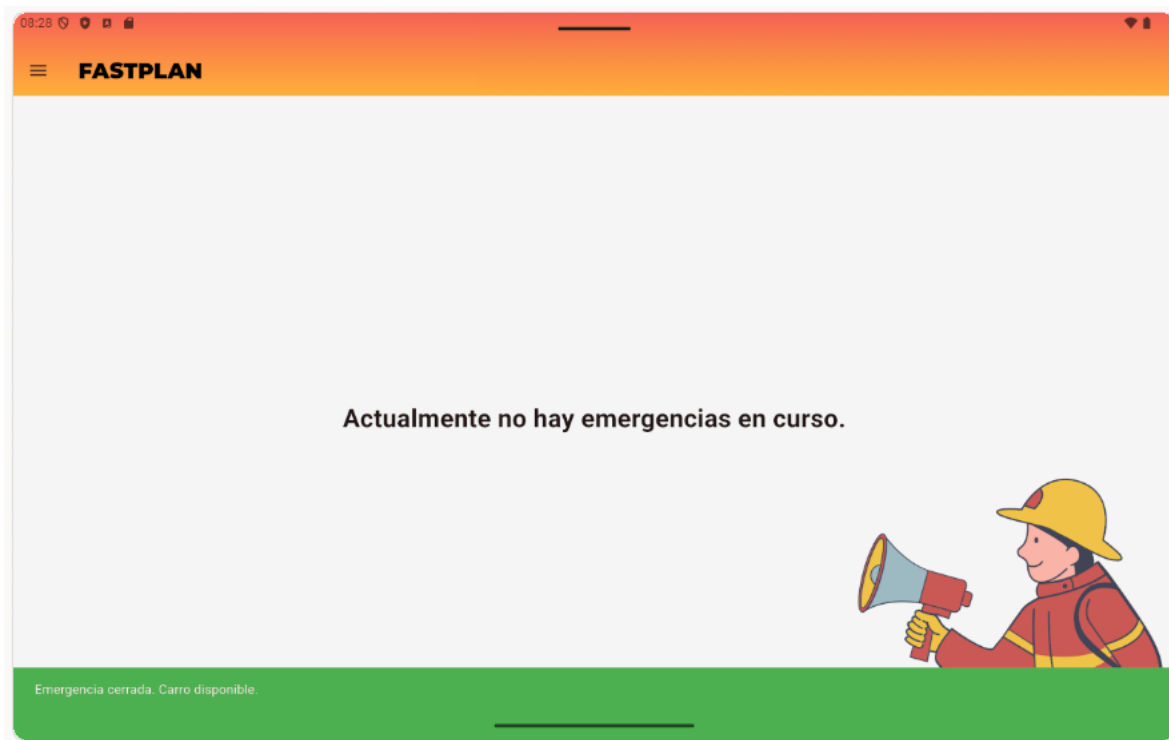


Figura 23: Emergencia atendida y cambio de disponibilidad del carro

Fuente: screenshot de la aplicación real

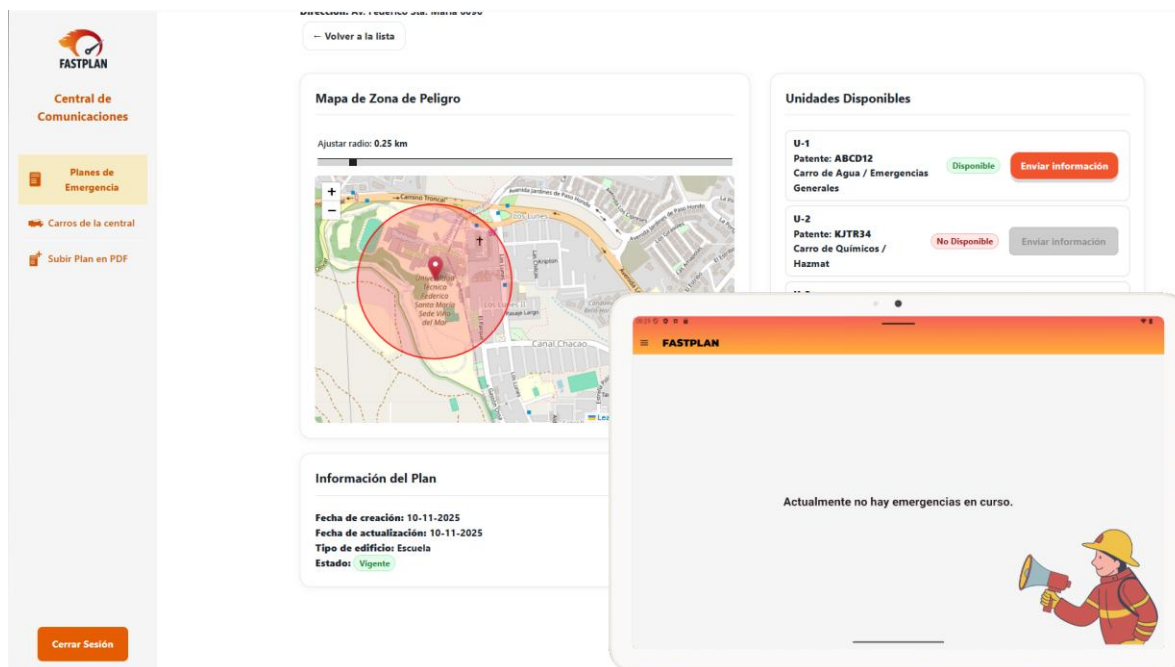


Figura 24: Cambio de disponibilidad del carro en el sistema FastPlan

Fuente: screenshot del Sistema FastPlan

5 Pruebas y Validación

Con el desarrollo completo de la solución, este capítulo presenta el proceso de verificación del funcionamiento a través de diferentes estrategias de prueba orientadas a evaluar la calidad, estabilidad y el desempeño de la aplicación. Este capítulo integra evaluaciones técnicas que permiten comprobar la consistencia interna del sistema y su comportamiento bajo condiciones reales de uso.

5.1 Ejecución y Análisis de Pruebas Unitarias

El proceso de aseguramiento de calidad se centró en validar la lógica de negocio y el comportamiento de los componentes críticos de la aplicación de manera aislada. Para ello se desarrolló una suite de pruebas automatizadas utilizando el entorno nativo de flutter test, abarcando tanto los repositorios como los modelos del dominio.

5.1.1 Validación de Lógica de Negocio y Repositorios

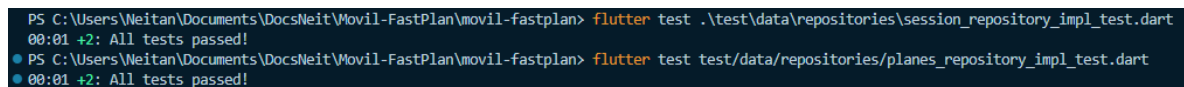
Las pruebas asociadas a la gestión de sesión confirmaron el comportamiento esperado frente a escenarios de autenticación exitosa y fallida. Empleando implementaciones simuladas, llamadas mocks, se verificó que el repositorio responsable no persiste datos locales cuando la autenticación remota retorna un error, reforzando la seguridad del dispositivo y garantizando un flujo de acceso confiable.

Resultado: All tests passed (2/2 pruebas exitosas).

En el caso de la comunicación de planes, las pruebas unitarias validaron que los métodos encargados de escuchar planes en tiempo real y de marcar disponibilidad delegan correctamente las operaciones a sus fuentes de datos. Esto permitió comprobar que la lógica de sincronización y comunicación se comporta de manera coherente ante cambios en la red o en la API backend.

Resultado: All tests passed (2/2 pruebas exitosas).

Todo lo anterior se deja en evidencia en el siguiente log.



```
PS C:\Users\Weitan\Documents\Docs\Neit\Movil-FastPlan\movil-fastplan> flutter test .\test\data\repositories\session_repository_impl_test.dart
00:01 +2: All tests passed!
PS C:\Users\Weitan\Documents\Docs\Neit\Movil-FastPlan\movil-fastplan> flutter test test\data\repositories\planes_repository_impl_test.dart
00:01 +2: All tests passed!
```

Figura 25: Resultado obtenido al ejecutar los test de los repositorios

Fuente: screenshot de los logs en Visual Studio Code

5.1.2 Detección de errores en modelos de datos

La ejecución de pruebas en los modelos del dominio permitió verificar la consistencia en la conversión de datos provenientes de Firestore. En el caso de DespachoEmergencia, se confirmó que la lógica aplicada en fromFirestore procesa correctamente estructuras anidadas y listas relacionadas, demostrando que el sistema es capaz de interpretar documentos complejos sin pérdida de información.

Resultado: All tests passed (2/2 pruebas exitosas).

Un hallazgo relevante surgió durante la validación del modelo PlanEmergencia, donde el sistema reportó la excepción "type 'String' is not a subtype of type 'Timestamp'". Este error reveló una inconsistencia en la asignación de tipos dentro del método de fábrica, la cual podía pasar inadvertida en compilación pero que provocaría fallos en tiempo de ejecución.

```
PS C:\Users\Neitan\Documents\DocsNeit\Movil-FastPlan\movil-fastplan> flutter test test/domain/models/despacho_emergencia_test.dart
00:01 +2: All tests passed!
PS C:\Users\Neitan\Documents\DocsNeit\Movil-FastPlan\movil-fastplan> flutter test test/domain/models/plan_emergencia_test.dart
00:01 +0 -1: PlanEmergencia Model Test fromFirestore debería retornar un PlanEmergencia válido con datos completos [E]
type 'String' is not a subtype of type 'Timestamp'
package:fastplan_movil/domain/models/plan_emergencia.dart 39:39  new PlanEmergencia.fromFirestore
test/domain/models/plan_emergencia_test.dart 29:37                main.<fn>.<fn>

To run this test again: D:\flutter\bin\cache\dart-sdk\bin\dart.exe test C:\Users\Neitan\Documents\DocsNeit\Movil-FastPlan\movil-fastplan\test\domain\models\plan_emergencia_test.dart -p vm --plain-name "PlanEmergencia Model Test fromFirestore debería retornar un PlanEmergencia válido con datos completos"
00:01 +0 -2: PlanEmergencia Model Test fromFirestore debería asignar valores por defecto si faltan campos [E]
type 'String' is not a subtype of type 'Timestamp'
package:fastplan_movil/domain/models/plan_emergencia.dart 39:39  new PlanEmergencia.fromFirestore
test/domain/models/plan_emergencia_test.dart 47:37                main.<fn>.<fn>

To run this test again: D:\flutter\bin\cache\dart-sdk\bin\dart.exe test C:\Users\Neitan\Documents\DocsNeit\Movil-FastPlan\movil-fastplan\test\domain\models\plan_emergencia_test.dart -p vm --plain-name "PlanEmergencia Model Test fromFirestore debería asignar valores por defecto si faltan campos"
00:01 +0 -2: Some tests failed.
```

Figura 26: Resultado obtenido al ejecutar los test de los modelos

Fuente: screenshot de los logs en Visual Studio Code

Dado el error, se prosigue a corregir el modelo estructurando correctamente como Timestamp y se ejecuta nuevamente el último test para verificar que los problemas ya están solucionados.

```
PS C:\Users\Neitan\Documents\DocsNeit\Movil-FastPlan\movil-fastplan> flutter test .\test\domain\models\plan_emergencia_test.dart
00:01 +2: All tests passed!
PS C:\Users\Neitan\Documents\DocsNeit\Movil-FastPlan\movil-fastplan>
```

Figura 27: Resultado obtenido al corregir el modelo plan_emergencia

Fuente: screenshot de los logs en Visual Studio Code

5.2 Validación de Objetivos

En este subcapítulo se presentan los resultados correspondientes a la validación de los objetivos, comenzando con el objetivo general y posteriormente los objetivos específicos.

5.2.1 Objetivo General

En este caso, el objetivo general se desglosará en sus tres aspectos medibles:

1. "(...) acceder en menos de 5 segundos a la información crítica de planes de emergencia desde dispositivos móviles (...)"
2. "(...) disponibilidad del 99% (...)"
3. "(...) interfaz optimizada (...)"

Para lograr el primer aspecto medible, se definió un procedimiento que consiste en buscar el plan desde el frontend web, ajustar el radio de búsqueda y luego iniciar una medición: desde el backend se envía un Timestamp con la hora del envío que se almacena en el documento, luego este se recibe en el móvil y se realiza la diferencia, en el momento que se crea la vista, entre ambos tiempos para obtener el tiempo aproximado de envío. Este método permitió obtener mediciones de un escenario real de uso, al operar sobre un entorno híbrido con backend web y base de datos en la nube, y el frontend ejecutándose de manera local.

Tabla 14: Mediciones sobre el rápido acceso a la información crítica

Fuente: elaboración propia

Contenido del Plan	Planes secundarios dentro del radio	Tamaño del Plan (KB)	Tiempo (s)	Conexión a Internet	¿Tiempo menor a 5 segundos?
Plan real de la UTFSM Sede JMC	0	12,98	1,63	Wifi	Sí
Plan real de la UTFSM Sede JMC	0	12,98	0,8	Datos Móviles 5G	Sí
Plan real de la UTFSM Sede JMC	0	12,98	0,1	Wifi	Sí
Plan Demo Casino de Viña del Mar (Radio 250 m)	2	0,36	1,39	Datos Móviles 5G	Sí
Plan Demo Boulevard Marina Arauco (Radio 300m)	3	1,39	0,5	Datos Móviles 5G	Sí
Plan Demo Clínica Ciudad del Mar (Radio 600m)	3	2,41	1,46	Wifi	Sí
Plan Demo Espacio Urbano (Radio 1,95km)	6	3,36	1,1	Wifi	Sí
Plan Demo Mall Marina (Radio 500m)	5	4,86	0,9	Wifi	Sí
Plan Demo Torre Coraceros (Radio 900m)	5	3,79	1,84	Wifi	Sí
Plan Demo Mall Marina Oriente (Radio 2km)	6	4,7	0,8	Datos Móviles 5G	Sí

```
D8$$SyntheticClass:0)]  
I/flutter (11487): --- REGISTRO DE PRUEBA DE LATENCIA ---  
I/flutter (11487): Tamaño Total Despacho: 12.98 KB  
I/flutter (11487): Tiempo Total: 0.085 s  
I/flutter (11487): Cumple (<5s): SÍ  
I/flutter (11487): -----  
I/flutter (11487): Firestore snapshot for ABCD12: {estado: enviado,
```

Figura 28: Logs de la prueba de latencia

Fuente: screenshot de la terminal en Visual Studio Code

Los resultados muestran la aplicación logró cargar y desplegar cada plan en tiempos que oscilaron entre 0,1 y 1,84 segundos, siempre por debajo del umbral de cinco segundos definido por el objetivo del proyecto. En todos los casos evaluados, incluyendo planes reales y planes demo distribuidos en diferentes radios geográficos, la aplicación mantuvo un desempeño consistentemente adecuado, validando que el mecanismo de sincronización y recuperación de datos cumple con los requisitos de rapidez establecidos para un entorno operativo de emergencia.

Continuando con el segundo aspecto medible, la validación de la disponibilidad del 99% combina la fiabilidad de la infraestructura en la nube con la solidez de la arquitectura móvil desconectada.

1. Como respaldo teórico, el acuerdo de nivel de servicio (SLA) de Google Cloud, establece que Firebase garantiza un uptime mensual del 99,99% para Firestore. Este estándar supera el requisito del 99% definido en los objetivos del proyecto, delegando la responsabilidad del uptime del servidor a un proveedor de nivel mundial. [13]
2. Dado que la disponibilidad operativa de Bomberos no puede depender exclusivamente de la conexión a internet, se validó el desacoplamiento de la aplicación. La arquitectura implementada permite usar el caché para almacenar ahí los documentos accedidos.

Se dispone de un registro audiovisual completo de esta prueba [14].

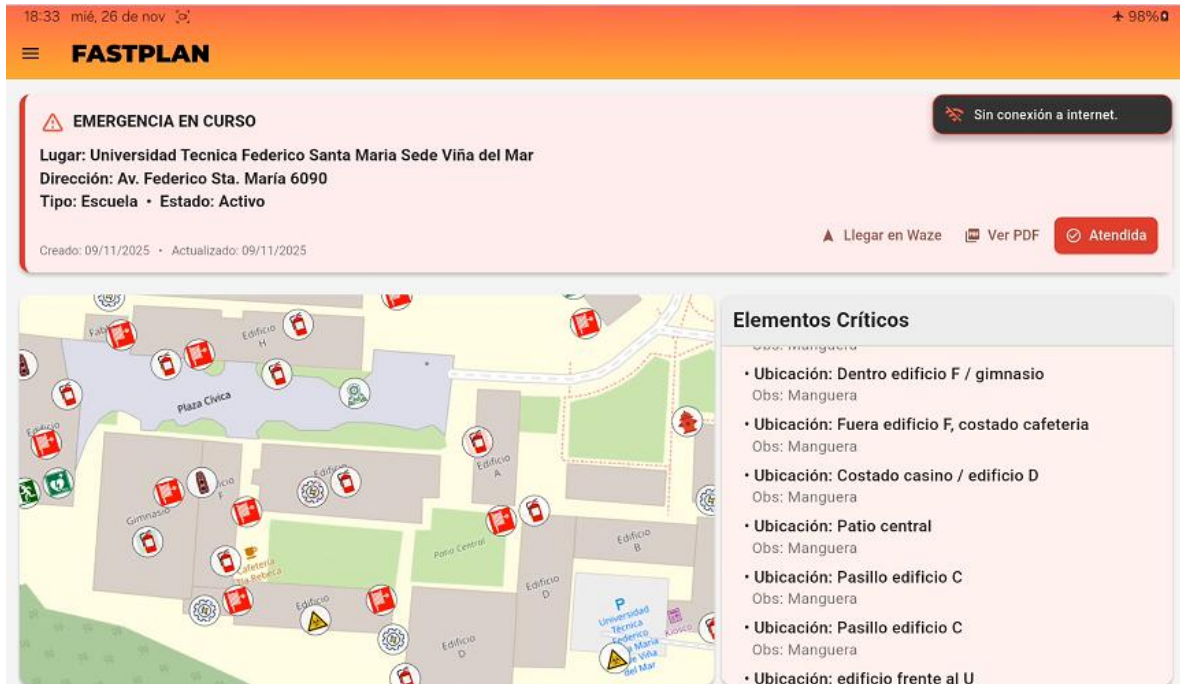


Figura 29: Prueba de funcionamiento offline y persistencia de datos
Fuente: screenshot del registro audiovisual

Continuando, durante la última validación del objetivo general, se ejecutaron tres escenarios de interacción intensiva típicos de una emergencia: manipulación del mapa (zoom/pan), interacción táctil con elementos críticos y scroll por listas de información. El objetivo fue verificar el cumplimiento del estándar de 60 FPS (16.6 ms por cuadro). A continuación, los resultados de la prueba de estrés.

Interacción con el mapa: El hilo de UI presentó un tiempo de renderizado promedio cercano a 3,8 ms, muy por debajo del umbral de 16 ms necesario para garantizar una experiencia fluida a 60 FPS. Por su parte, el hilo de Raster registró picos puntuales de hasta 19,5 ms asociados a la carga inicial de teselas, pero mantuvo un promedio de 9,0 ms, lo que permitió una navegación estable y sin interrupciones perceptibles. Los resultados observados se alinean con un promedio efectivo de 59 FPS y con una ausencia total de *jank*, evidenciando una ejecución estable del motor gráfico.



Figura 30: Registro del *performance overlay* mientras se interactúa con el mapa
Fuente: screenshot Flutter DevTools

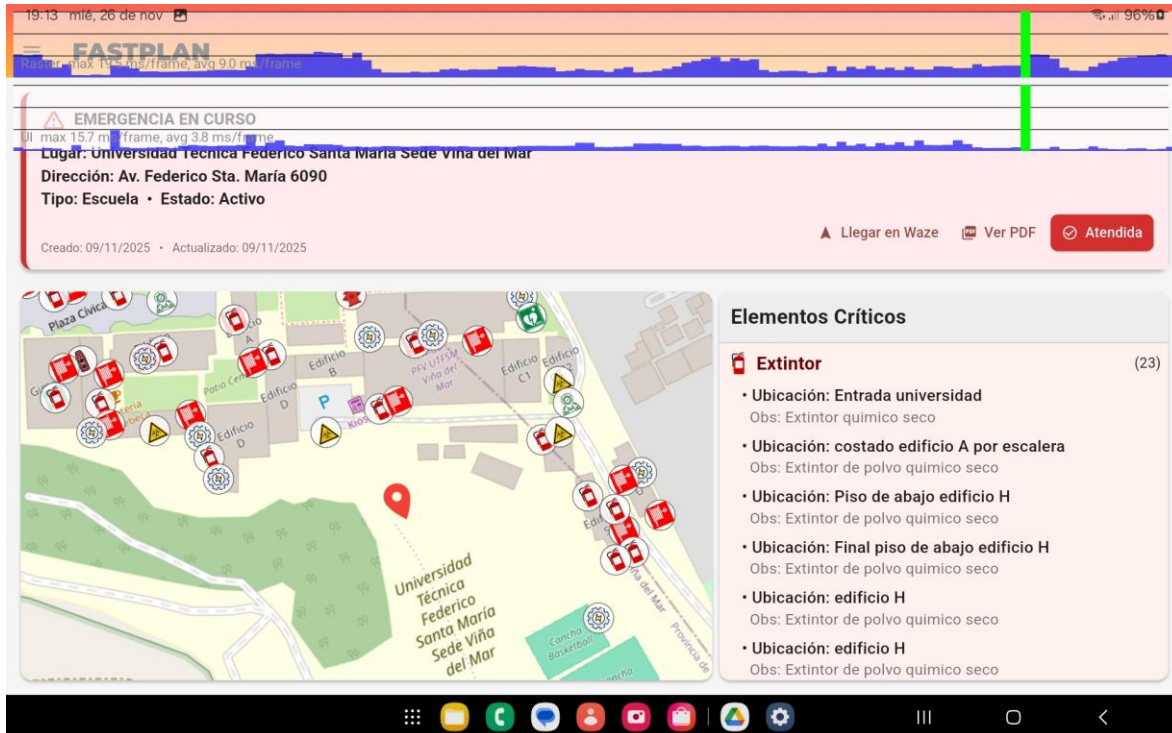


Figura 31: Vista móvil del *performance overlay*
Fuente: screenshot del registro audiovisual

Interacción con elementos críticos: El tiempo de construcción de widgets en el hilo de UI promedió 3,2 ms, mientras que el hilo de Raster registró un promedio de 9,9 ms, valores que permiten que la respuesta táctil sea percibida de forma inmediata (menor a 100 ms).

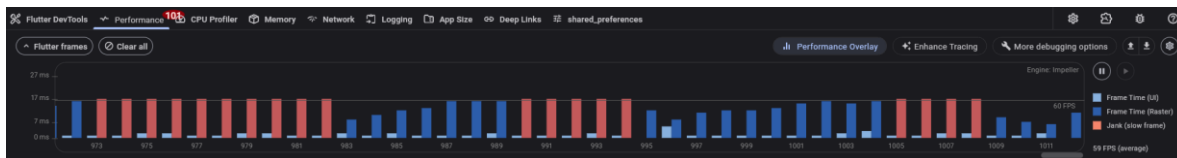


Figura 32: Registro del *performance overlay* mientras se interactúa con los elementos críticos del mapa
Fuente: screenshot Flutter DevTools

La captura del dispositivo confirma un renderizado estable, con barras verdes predominantes en el *Performance Overlay*, demostrando que la superposición de elementos sobre el mapa no afecta la fluidez de la aplicación.

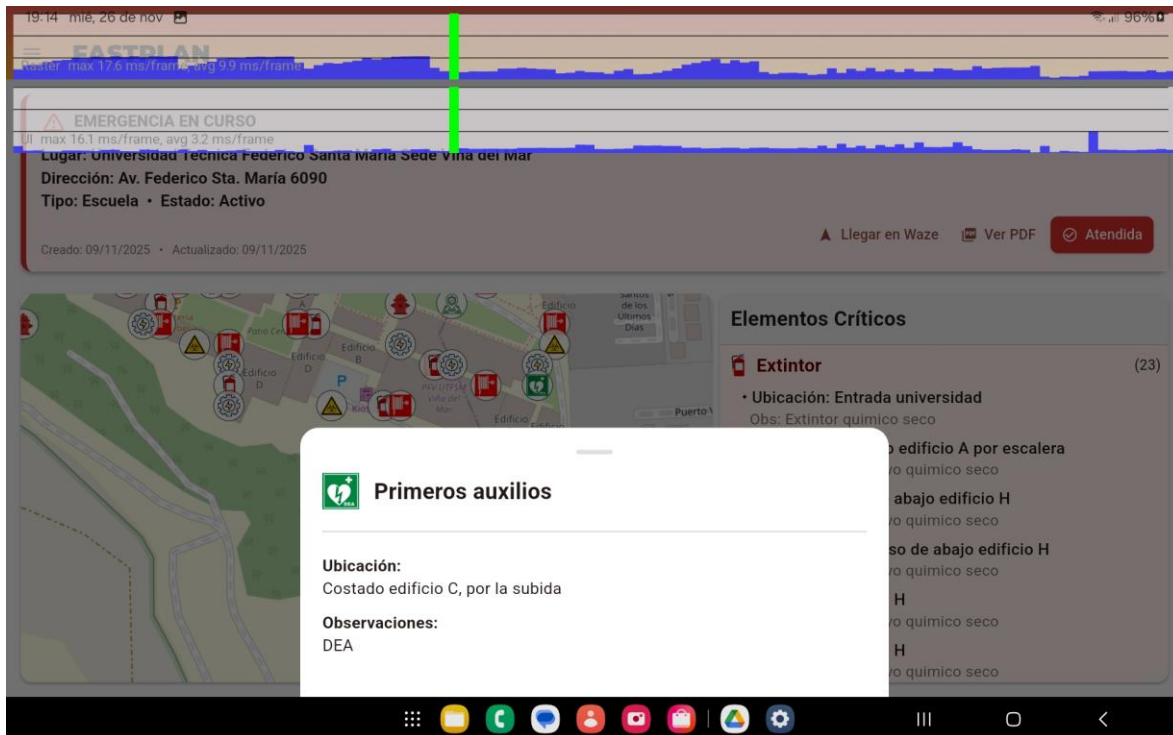


Figura 33: Vista móvil del *performance overlay* al interactuar con elementos críticos dentro del mapa

Fuente: screenshot del registro audiovisual

Scroll por elementos críticos: el motor de renderizado Impeller mostró un desempeño altamente eficiente: el hilo de UI descendió a un promedio de 1,6 ms y el de Raster a 5,9 ms. Se identificó un único pico aislado de 31,6 ms en el hilo de Raster, asociado a la carga inicial de imágenes, el cual no generó impacto perceptible gracias a la gestión interna de caché. En conjunto, estos resultados reflejan un promedio sostenido de 59 FPS y una experiencia de desplazamiento fluida y reactiva, consistente con los criterios de una interfaz optimizada para situaciones de emergencia.



Figura 34: Registro del *performance overlay* mientras se hace scroll en la vista lateral de los elementos críticos

Fuente: screenshot Flutter DevTools

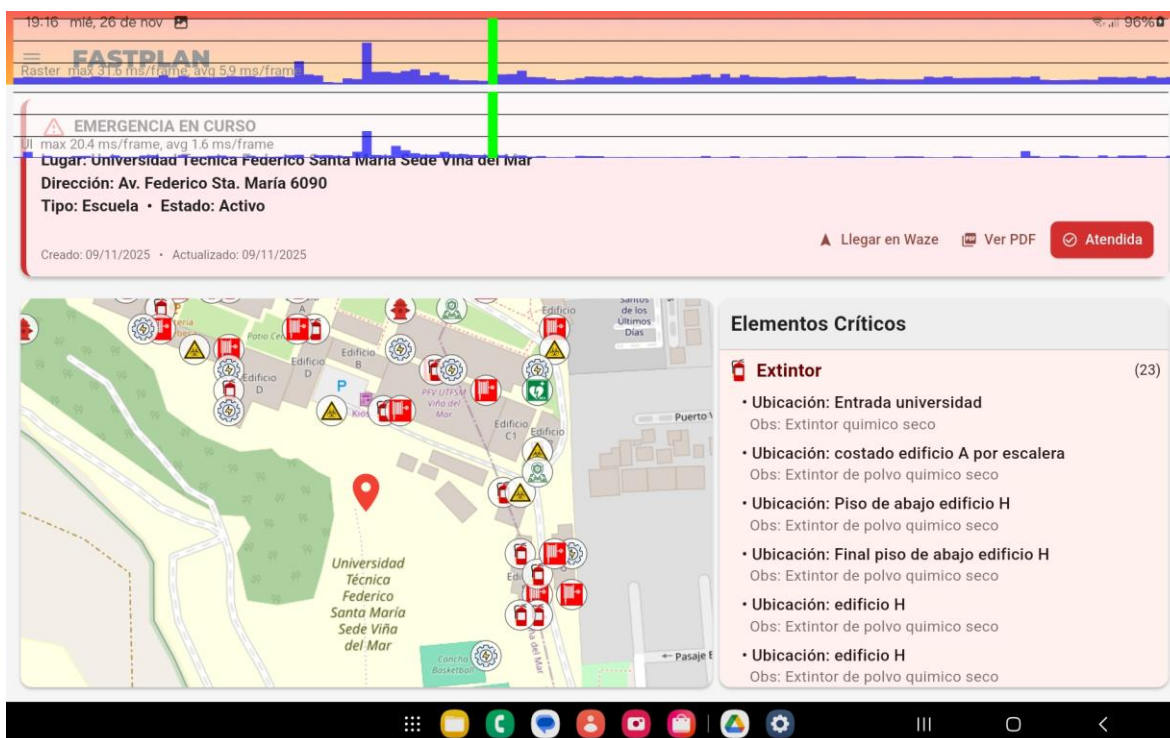


Figura 35: Vista móvil del *performance overlay* al desplazarse por el contenedor de elementos críticos al lado del mapa

Fuente: screenshot del registro audiovisual

El análisis cuantitativo mediante Flutter DevTools confirma que la aplicación cumple con el requisito de una interfaz optimizada. En el 95% del tiempo de ejecución, tanto el hilo lógico como el gráfico operaron con márgenes de holgura superiores al 40% respecto al límite de 16 ms, garantizando una respuesta inmediata a las acciones del bombero incluso bajo carga de trabajo gráfica.

5.2.2 Validación de Objetivos

A lo largo del desarrollo de todo el proyecto los objetivos específicos han sido abordados de una u otra forma. Para dejar en evidencia y formalizar el cumplimiento de estos, se presenta la siguiente matriz de trazabilidad que vincula cada objetivo específico con la evidencia técnica obtenida durante la fase de pruebas.

Tabla 15: Evidencia del cumplimiento técnico de los objetivos específicos

Fuente: elaboración propia

Objetivo específico	Estrategia de validación	Resultado obtenido	Estado
OE1: Diseñar arquitectura backend con servicio de sincronización real-time.	Implementación de Clean Architecture y patrón Observer con Firestore.	Comunicación bidireccional funcional validada en diagramas de secuencia y pruebas de integración.	Cumplido.
OE2: Implementar mensajería con latencia máxima de 5 segundos.	Medición instrumentada de latencia punto a punto en 10 pruebas de carga.	Promedio de latencia de 1.05 segundos, con 100% de pruebas bajo el umbral crítico de 5s.	Cumplido.
OE3: Autenticación y sistema de almacenamiento local seguro.	Prueba de reinicio de aplicación y verificación de token local.	Acceso directo al Home sin reingreso de credenciales en pruebas de ciclo de vida.	Cumplido.
OE4: Implementar mapas interactivos y elementos gráficos.	Inspección visual y pruebas de renderizado de marcadores georreferenciados.	Visualización correcta de grifos, salidas y zonas de seguridad sobre el mapa vectorial.	Cumplido.
OE5: Pruebas de robustez y confiabilidad.	Ejecución de pruebas unitarias y simulación de escenarios offline.	100% de aprobación en tests unitarios críticos y validación exitosa de persistencia sin conexión.	Cumplido.

6 Conclusiones y Trabajo Futuro

La investigación logró resolver la problemática de acceso fragmentado a la información en contextos de emergencia mediante el diseño e implementación del backend móvil de FastPlan, cumpliendo así con el Objetivo General (OG) de transformar un procedimiento tradicionalmente estático en un sistema digital reactivo y sincronizado en tiempo real. El uso de Flutter, Cloud Firestore y la aplicación de Clean Architecture en el backend móvil, permitió construir un sistema modular, escalable y desacoplado, validando el diseño de una arquitectura capaz de integrar servicios externos de sincronización en tiempo real (OE1).

Las pruebas instrumentadas confirmaron que la solución supera los requisitos operativos definidos (OE5). Específicamente, la latencia promedio de acceso a información crítica fue de 1.05 segundos, muy por debajo del límite establecido inicialmente de 5 segundos para la mensajería en tiempo real (OE2). También, el mecanismo de persistencia local garantizó continuidad operativa incluso en escenarios sin conectividad, satisfaciendo los criterios definidos (OE3). A nivel de interfaz, las pruebas de estrés gráfico demostraron un rendimiento estable de 60 FPS en la visualización de mapas y elementos críticos (OE4). En general, estos resultados no solo validan que se cumple con el propósito técnico, sino que convierten los planes de emergencia en una herramienta táctica efectiva para la toma de decisiones en terreno.

Si bien el sistema cumple plenamente con los objetivos iniciales, su arquitectura habilita múltiples líneas de evolución futura. Entre ellas destacan la integración con sistemas CAD para automatizar la entrega de información al momento del despacho, la incorporación de tecnologías de posicionamiento en interiores, el desarrollo de un portal web colaborativo para comunidades y administradores. También, está la necesidad de fortalecer el modelo de seguridad, mediante una validación más robusta de dispositivos, mecanismos avanzados de protección de sesiones y una estrategia integral para mitigar riesgos asociados al uso de credenciales estáticas. Estas mejoras proyectan a FastPlan como un ecosistema integral de gestión de emergencias con impacto creciente en la seguridad pública.



7 Agradecimientos

A Dios por la vida, la salud tanto mía como de todos mis seres queridos, y por darme la capacidad para esforzarme en todo este periodo.

A la Universidad Técnica Federico Santa María, por haber tenido la oportunidad de ser parte de ella y adoptar sus valores como estudiante y persona.

A mi madre y padre, que sus esfuerzos diarios me inspiran a tener las mismas fuerzas que ellos. Por inculcarme hábitos y buenos valores que me han ayudado a seguir adelante en momentos difíciles.

A mi novia y futura esposa Sidney, quien ha sido mi apoyo incondicional y mi mayor fuente de amor.

A mi abuelita Tita, por el amor que ella ha tenido conmigo.

A mi familia, que siempre me han apoyado y amado en toda mi vida.

A mis amigos y compañeros de Universidad, que con su compañía han hecho que este periodo no solo sea enriquecedor académicamente, sino que también muy divertido de afrontar junto a ellos. Especialmente a Miguel Alfaro, Matia Zambelli, Sofía Jara, Fernando Bastías, Felipe Bravo y Matías Pino.

8 Referencias

- [1] Real Academia Española. (s. f.). Emergencia. Diccionario panhispánico del español jurídico. Recuperado el 29 de septiembre de 2025, de <https://dpej.rae.es/lema/emergencia>
- [2] Ministerio de Vivienda y Urbanismo (MINVU). (2020). Guía para elaborar un buen plan de emergencia. Gobierno de Chile. <https://www.minvu.gob.cl/wp-content/uploads/2020/09/Guia-para-elaborar-un-buen-Plan-de-Emergencia-Minvu.pdf>
- [3] Biblioteca del Congreso Nacional de Chile (BCN). (2022). Ley N.º 21.442 sobre Copropiedad Inmobiliaria. Recuperado el 29 de septiembre de 2025, de <https://www.bcn.cl/leychile/navegar?idNorma=1174663>
- [4] Ministerio de Salud (MINSAL). (1999). Decreto Supremo N.º 594: Aprueba reglamento sobre condiciones sanitarias y ambientales básicas en los lugares de trabajo. Biblioteca del Congreso Nacional de Chile. <https://www.bcn.cl/leychile/navegar?idNorma=16839>
- [5] Google. (2025). Ejemplo de facturación de Cloud Firestore. Firebase. Recuperado el 3 de octubre de 2025, de <https://firebase.google.com/docs/firestore/billing-example?hl=es-419>
- [6] Google. (s. f.). Modelo de datos de Cloud Firestore. Firebase Documentation. Recuperado el 5 de octubre de 2025, de <https://firebase.google.com/docs/firestore/data-model>
- [7] Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.
- [8] Google. (s. f.). Firebase Authentication. Firebase Documentation. Recuperado el 5 de octubre de 2025, de <https://firebase.google.com/docs/auth>
- [9] Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional.
- [10] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
- [11] Google. (s. f.). Get real-time updates with Cloud Firestore. Firebase Documentation. Recuperado el 5 de octubre de 2025, de <https://firebase.google.com/docs/firestore/query-data/listen>
- [12] Fowler, M. (2004). Inversion of Control Containers and the Dependency Injection pattern. MartinFowler.com. <https://martinfowler.com/articles/injection.html>
- [13] Google Cloud. (s. f.). Acuerdo de Nivel de Servicio (SLA) de Cloud Firestore. Recuperado el 5 de octubre de 2025, de <https://cloud.google.com/firestore/sla>
- [14] Apablaza, N. (2025). Prueba de funcionamiento offline y persistencia de datos [Archivo de Video]. Google Drive. https://drive.google.com/file/d/1a_uFAy5FLg4qRWCvNBEDuc6AdMYi6EDV/view?usp=drive_link