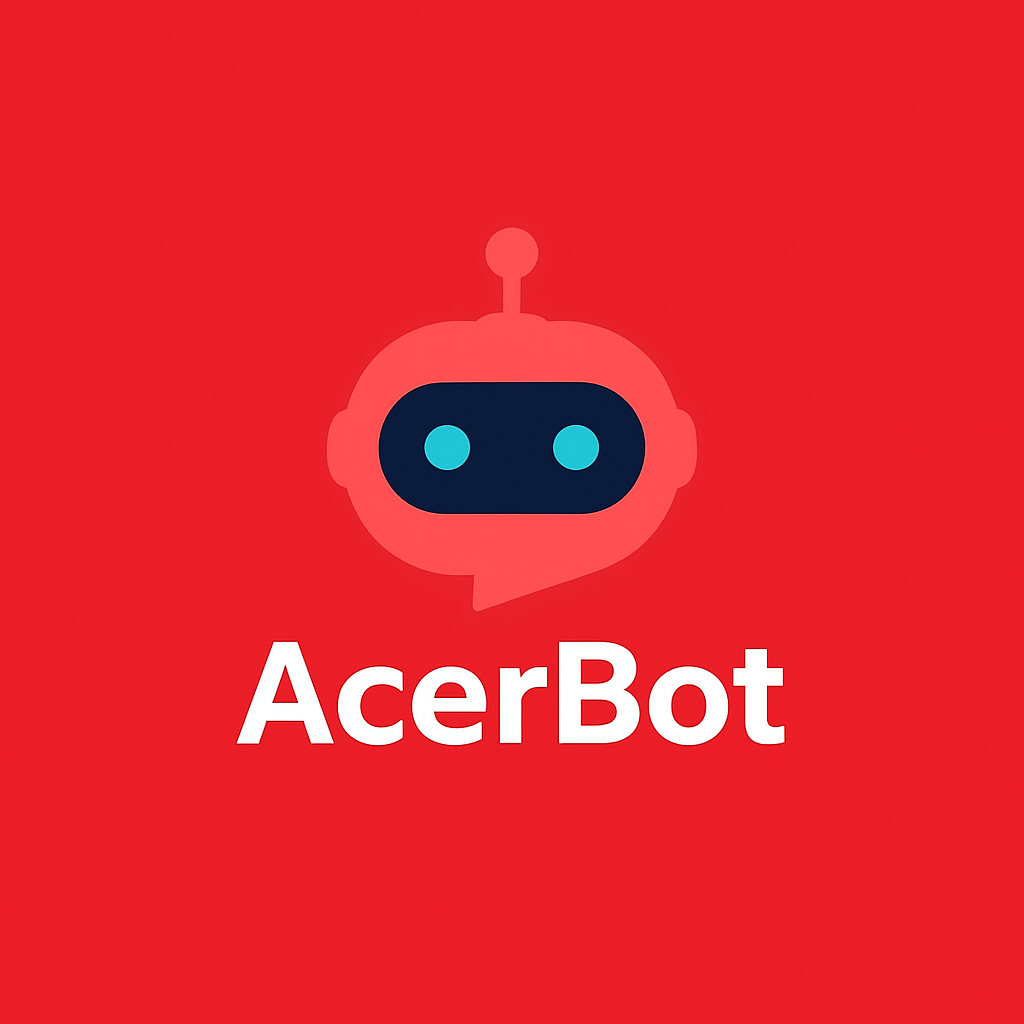
**Informe Final**: Acer Bot Automatización de Gestión de Buses



| Integrante | Vicente José López González |
| --- | --- |
| Asignatura | Capstone |
| Sección | PTY 4614-002D |
| Docente | Luis Ricardo Valdivia Pinto |
| Fecha | 20/11/2025 |

[**Abstract 3**](#_bm1a0j4ft9wu)

[**Resumen 3**](#_tf61yvs2dfj2)

[**Introducción 3**](#_uodywl46tigk)

[**Relevancia del Proyecto 3**](#_sfv6vigsln3f)

[**Necesidades del Negocio 4**](#_ti98kgu7lmq6)

[**Necesidad del Bot 5**](#_xb3k527r9w4b)

[**Objetivos Generales y Específicos 6**](#_xfoa2yk8s0fs)

[**Control de Ingreso del sistema 6**](#_p66g54agtwhr)

[**Metodología Scrum 6**](#_4l15f4fapnl8)

[**Carta Gantt 7**](#_upmsfzr6pto1)

[**Diseño y Arquitectura del Sistema 8**](#_298xpc9q78l9)

[**Creación del Bot mediante Bot Father 11**](#_mp3wrguiu1w9)

[**Desarrollo Técnico del Sistema 12**](#_rjkd17dlj3ix)

[**Base de Datos - Modelo y Evidencias 13**](#_rzyj4379cxnl)

[**Evidencias de Documentación 15**](#_4f5azg7bea3o)

[**Evidencias de Sistema 15**](#_mg0lto5rfwse)

[**Evidencias de Base de Datos 16**](#_9310cyjumz4z)

[**Pruebas y Validación (QA) 16**](#_hh5h0l8om7p2)

[**Fortalezas y Limitaciones 16**](#_npmcoc26c7p5)

[**Competencias del Perfil de Egreso 16**](#_p92f8mh0oxph)

[**Intereses y Proyección Profesional 16**](#_fukbs79pjxqc)

[**Conclusión 17**](#_ufteco54n0ai)

[**Reflexión (Inglés) 17**](#_m1oldqnn41az)

[**Bibliografía 18**](#_58vcne901zv8)

[**Anexos 19**](#_d1dtwzkfsrsl)

# Abstract

Acer Bot is an automated fleet-management system designed to detect TAG toll passages, validate authorized bus routes, and provide real-time alerts to administrators. The system integrates Cloud Services, GPS APIs, TAG databases, and a conversational bot built in Telegram through BotFather. Using an agile Scrum methodology, the project implements a scalable backend, a real-time database in Firestore, automated route-validation logic, and a communication layer for operational queries. Acer Bot improves traceability, operational safety, and automation for transport companies, delivering a modern and reliable technological solution aligned with industry standards.

# Resumen

Acer Bot es un sistema automatizado de gestión operativa para buses que permite detectar pasos por TAG, validar rutas autorizadas y generar alertas en tiempo real. El sistema integra servicios en la nube, APIs GPS, bases TAG y un bot conversacional desarrollado en Telegram mediante Bot Father. Utilizando Scrum como metodología ágil, el proyecto incorpora una arquitectura escalable, una base de datos en Firestore, validación automática de rutas y comunicación operativa directa. Acer Bot mejora la trazabilidad, reduce pérdidas operativas y aporta una solución tecnológica moderna y alineada con las necesidades reales de la industria del transporte.

# Introducción

El proyecto Acer Bot responde a la creciente necesidad de las empresas de transporte de automatizar el control de rutas, reducir costos asociados a desvíos no autorizados y mejorar la trazabilidad operativa. Debido a que la revisión manual de TAG, GPS y rutas es ineficiente y propensa a errores, se plantea una solución tecnológica basada en automatización, servicios cloud y datos en tiempo real.

La industria del transporte terrestre presenta un alto grado de complejidad operativa: múltiples rutas, choferes, buses, cambios de turnos, variaciones de tráfico y costos asociados al uso de autopistas concesionadas. Acer Bot se posiciona como un sistema capaz de monitorear todo este proceso de forma automatizada mediante detección GPS y validación dinámica.

# Relevancia del Proyecto

El transporte urbano y rural en Chile depende de una correcta supervisión. No obstante, la mayoría de los sistemas actuales siguen siendo manuales, lentos y poco transparentes.

Acer Bot es relevante porque:

* Automatiza procesos que consumen tiempo humano.
* Reduce costos asociados al mal uso de TAG.
* Mejora el control operativo en tiempo real.
* Evita pérdidas económicas por desvíos.
* Aumenta la seguridad del servicio.
* Apoya la transformación digital del mercado de transporte.

Se trata de un problema real, con impacto directo en empresas, administración de flotas, conductores y usuarios.

# Necesidades del Negocio

Las necesidades que justifican este proyecto son:

1. **Control Automático de TAG**

Evitar gastos por pórticos no autorizados

1. **Seguimiento de buses sin intervención humana**

La automatización reduce errores humanos y mejora los tiempos de respuesta.

1. **Transparencia operativa**

Contar con registros precisos, no alterables y en tiempo real.

1. **Trazabilidad completa**

Saber dónde está cada bus, qué ruta sigue y que chofer lo conduce.

1. **Centralización de datos**

Un solo sistema reemplaza planillas, mensajes y reportes manuales.

1. **Alertas inmediatas**

Las empresas requieren saber al instante si un bus se desvió.

1. **Reducción de costos operacionales**

Las empresas gastan millones anuales por TAG indebidos.

Estas necesidades fueron levantadas durante el proceso de análisis del proyecto.

# Necesidad del Bot

El sistema es ideal para:

**Empresas de buses interurbanos y rurales**

* ChileBus
* Buses Pullman
* Turbus
* Buses Nilahue

**Ventaja:**

Tienen numerosas rutas que cruzan autopistas con TAG, por lo que el control es crítico.

**Empresas de Transporte escolar y privado**

* Transportes escolares regulados
* Empresas de traslado de personal (mineras, bancos, fábricas)

**Ventaja:**

Necesitan reportes para auditorías, seguridad y cumplimiento normativo.

**Empresas municipales y de apoyo público**

* Programas de traslado comunal
* Buses municipales gratuitos

**Ventaja:**

Garantizan transparencia del gasto público

**¿Por qué estas empresas necesitan un Bot?**

Porque el sistema:

* Evita pérdidas económicas
* Automatiza procesos que antes eran manuales
* Permite auditar rutas históricas
* Centraliza todo en un bot
* Aumenta la seguridad y confiabilidad
* Es escalable y puede crecer con la empresa
* Reduce costos de personal administrativo

# Objetivos Generales y Específicos

**General:**

Desarrollar un sistema automatizado denominado Acer Bot, que permita gestionar en tiempo real la operación de buses, mediante integración de APIs, servicios cloud, almacenamiento de datos y un bot conversacional que centraliza la información operativa.

**Específicos:**

* Diseñar una arquitectura escalable en la nube.
* Crear una base de datos operativa en tiempo real.
* Integrar API GPS y bases TAG.
* Implementar un sistema automático de alertas.
* Validar el sistema con pruebas reales de funcionamiento.

# Control de Ingreso del sistema

Acer Bot incluye un **control de ingreso seguro**:

* Solo usuarios autorizados pueden acceder al bot.
* El sistema reconoce el **ID único de Telegram** de cada administrador.
* Los accesos quedan registrados en la base de datos.

Este mecanismo evita fugas de información y garantiza que las funciones internas del bot sean únicamente para personal autorizado.

# Metodología Scrum

El proyecto se desarrolló bajo un enfoque ágil basado en Scrum, adaptado a un desarrollador único.

**Roles asumidos**

* Product Owner
* Scrum Master
* Equipo de desarrollo

**Duración de los Sprints**

Cada sprint tuvo una duración de 7 días, permitiendo entregas incrementales estables.

**Sprint 1 - Análisis**

* Requerimientos
* Estudio de APIs
* Alcance del proyecto

**Sprint 2 - Diseño**

* Arquitectura
* Base de datos
* Diagrama de flujo del bot

**Sprint 3 - Implementación I**

* Creación del bot Bot Father
* Conexión básica a Firebase
* Primeros comandos

**Sprint 4 - Implementación II**

* Detección de TAG
* Validación de rutas
* Alertas automáticas

**Sprint 5 - Implementación y pruebas**

* Pruebas funcionales
* Documentación
* Ajustes finales.

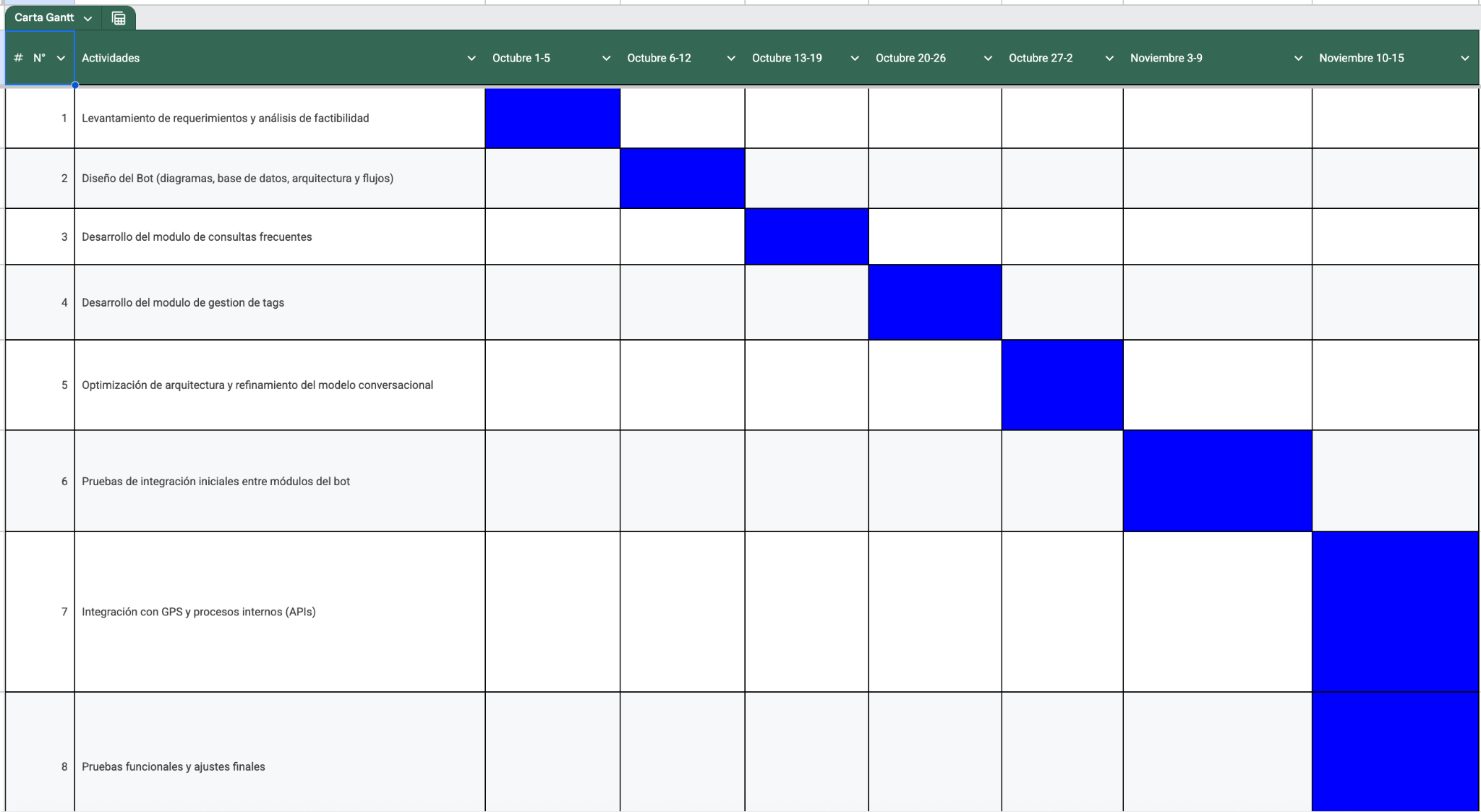
Scrum permitió enfrentar un proyecto complejo de manera flexible y ordenada.

# Carta Gantt

La Carta Gantt muestra la planificación completa del proyecto Acer Bot, organizada semana a semana. En ella se visualizan todas las etapas del desarrollo: levantamiento de requerimientos, diseño del sistema, creación del bot, implementación de los módulos (GPS, TAG, validación de rutas), integración con Firestore, pruebas y documentación final.

Cada actividad está distribuida en bloques que representan los sprints de 7 días, permitiendo ver claramente qué tareas corresponden a cada iteración. La carta también destaca el orden lógico del trabajo: primero análisis y diseño, luego implementación, después pruebas y finalmente documentación y presentación.

Esta planificación permite mantener control del avance, definir dependencias entre tareas y asegurar que el proyecto se completará dentro de los plazos establecidos.



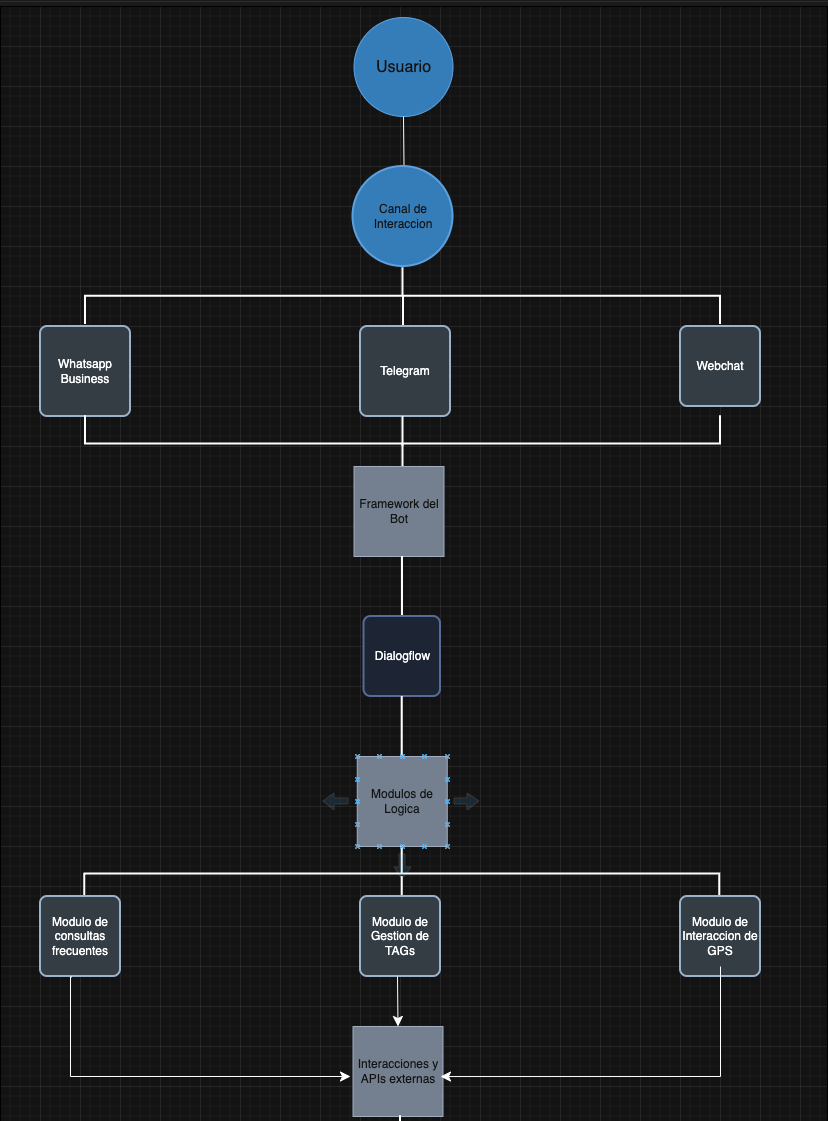
# Diseño y Arquitectura del Sistema

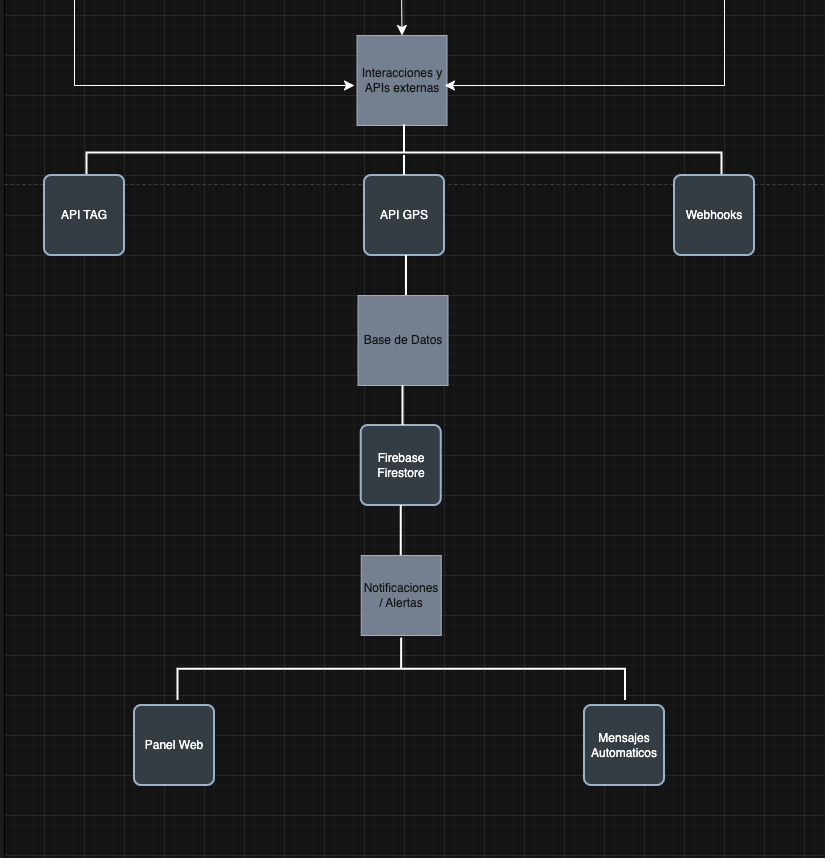
La arquitectura es modular:

1. **Bot Telegram**
2. **Cloud Functions en** [**Node.js**](http://node.js)
3. **Firestore (base de datos)**
4. **Módulo GPS**
5. **Módulo TAG**
6. **Sistema de alertas**

Todo funciona en tiempo real gracias a Firebase.

Diagrama del Sistema





# 

# 

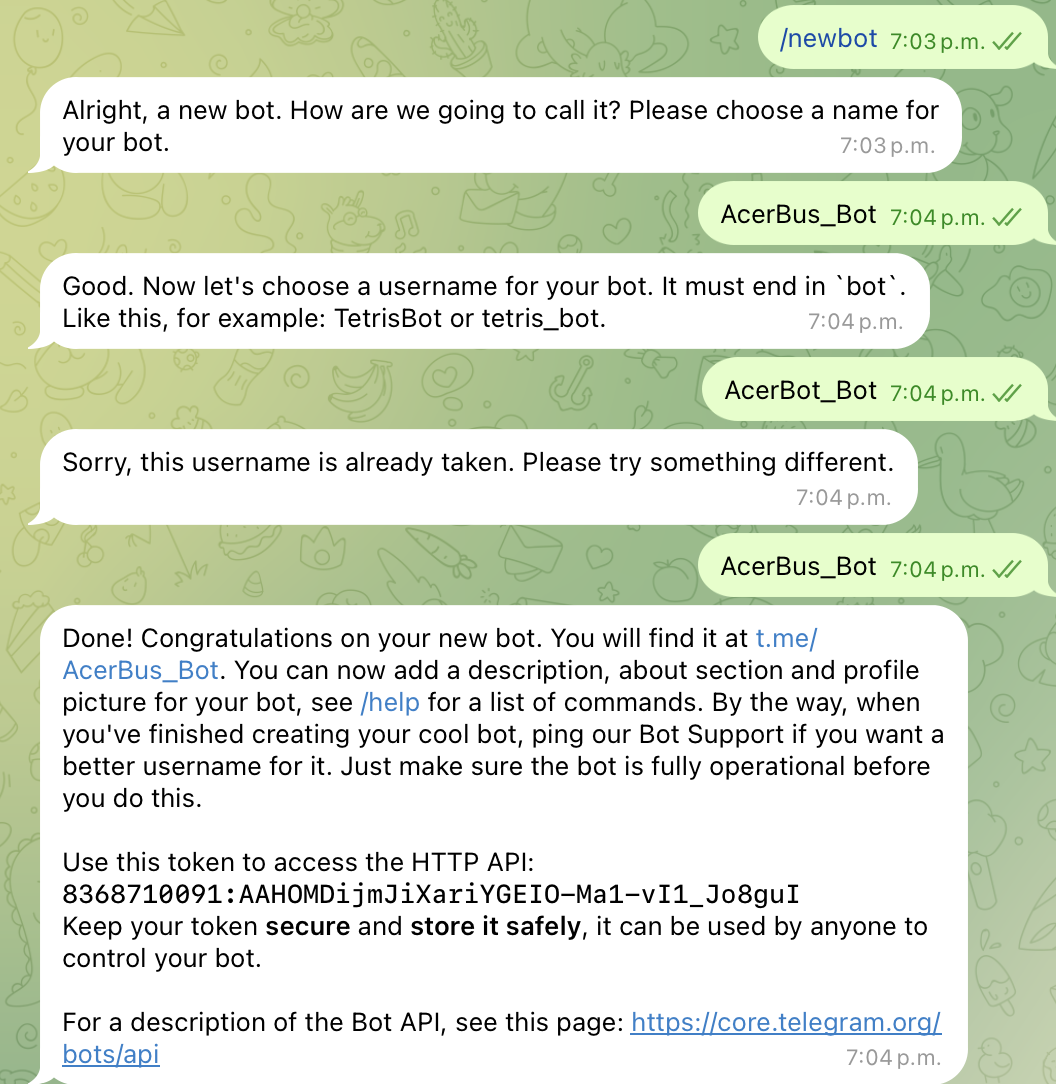
# 

# Creación del Bot mediante Bot Father

El bot de Acer Bot fue creado mediante @BotFather, la herramienta oficial de Telegram.

Proceso:

* Se ejecuto el comando /newbot
* Se definió el nombre: AcerBus\_Bot
* Se generó un token API único
* Se configuraron comandos básicos
* Se cargó foto y descripción
* Se protegió el token dentro de variables de entorno



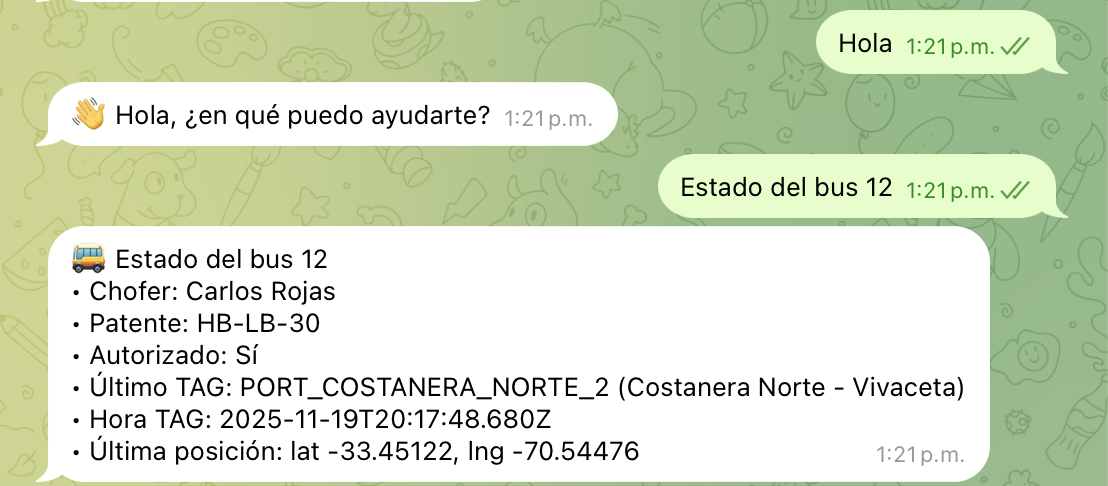
Esta evidencia demuestra que el bot es real y funciona.

# Desarrollo Técnico del Sistema

Incluye:

* Procesamiento de coordenadas GPS
* Comparación con pórticos TAG
* Validación de rutas autorizadas
* Creación de registros automáticos
* Generación de alertas
* Respuestas inteligentes mediante el bot.

Todo desarrollado en [Node.js](http://node.js) y Firebase



# 

# 

# Base de Datos - Modelo y Evidencias

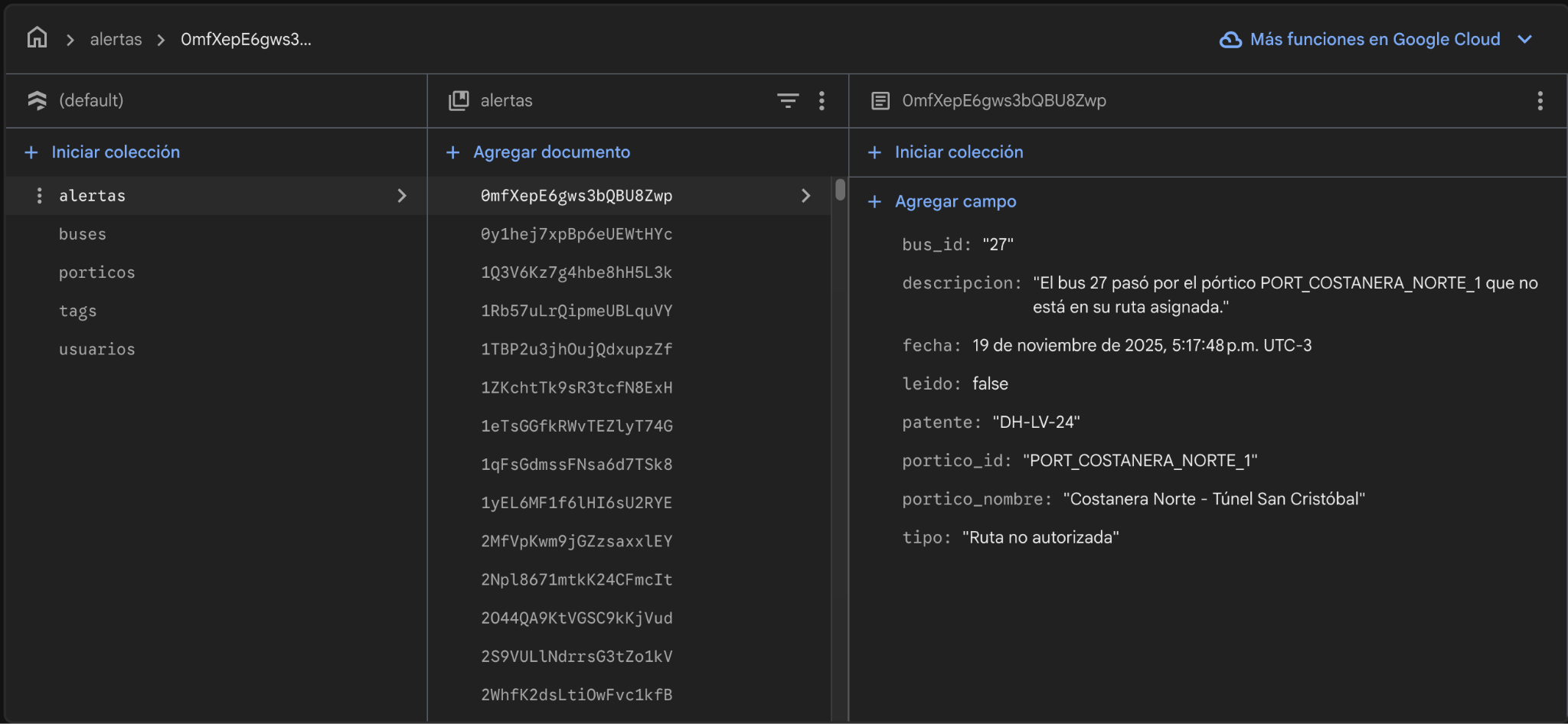
Firestore almacena:

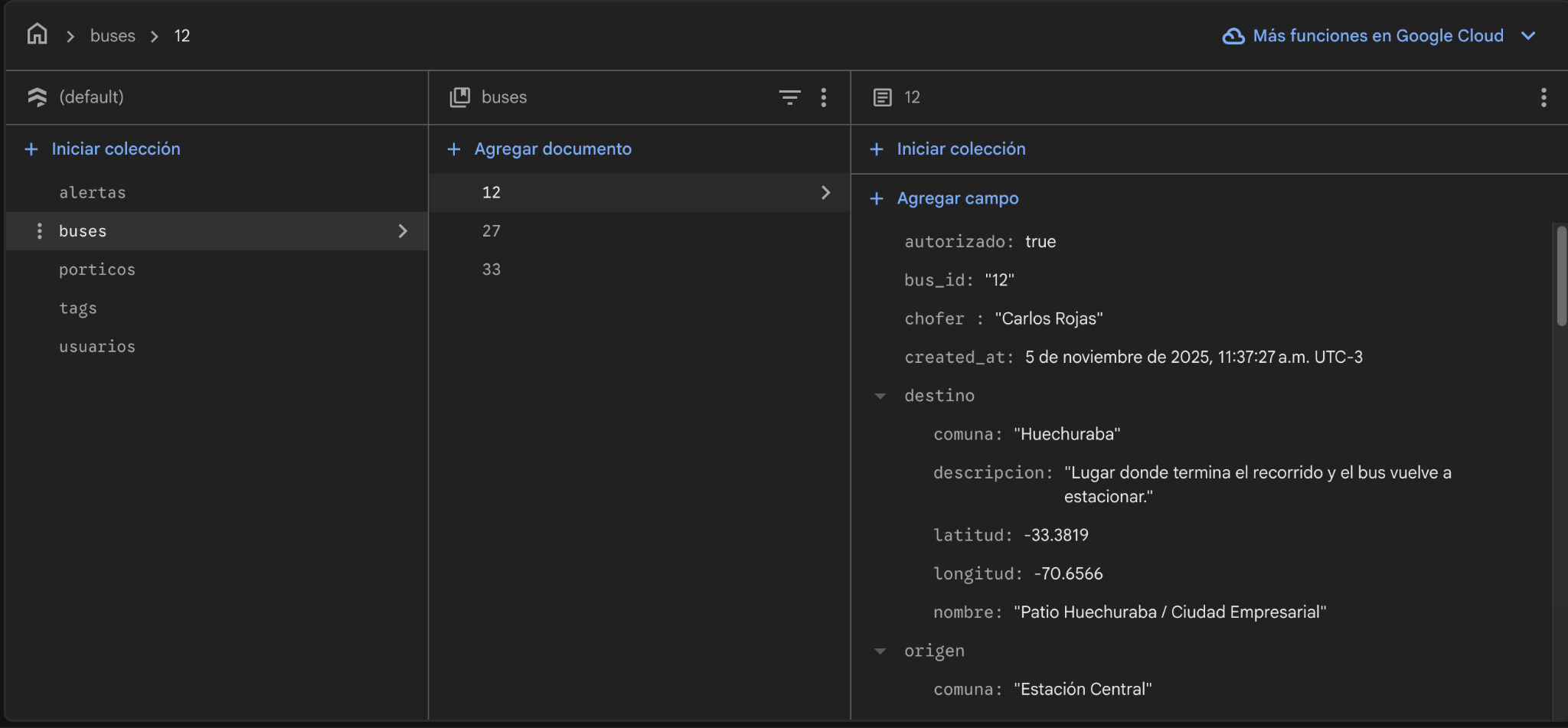
* Buses
* Rutas
* TAG
* Alertas

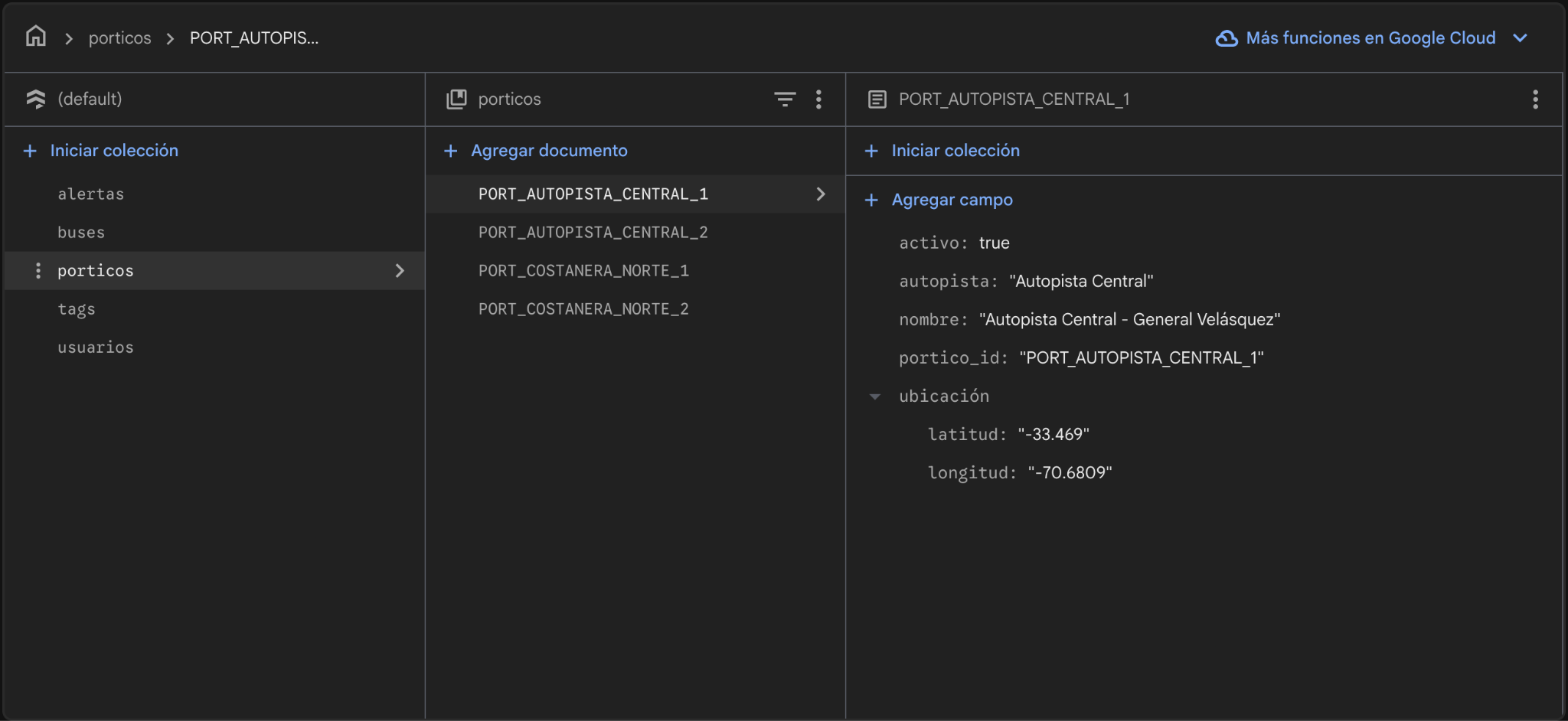
Cada evento contiene:

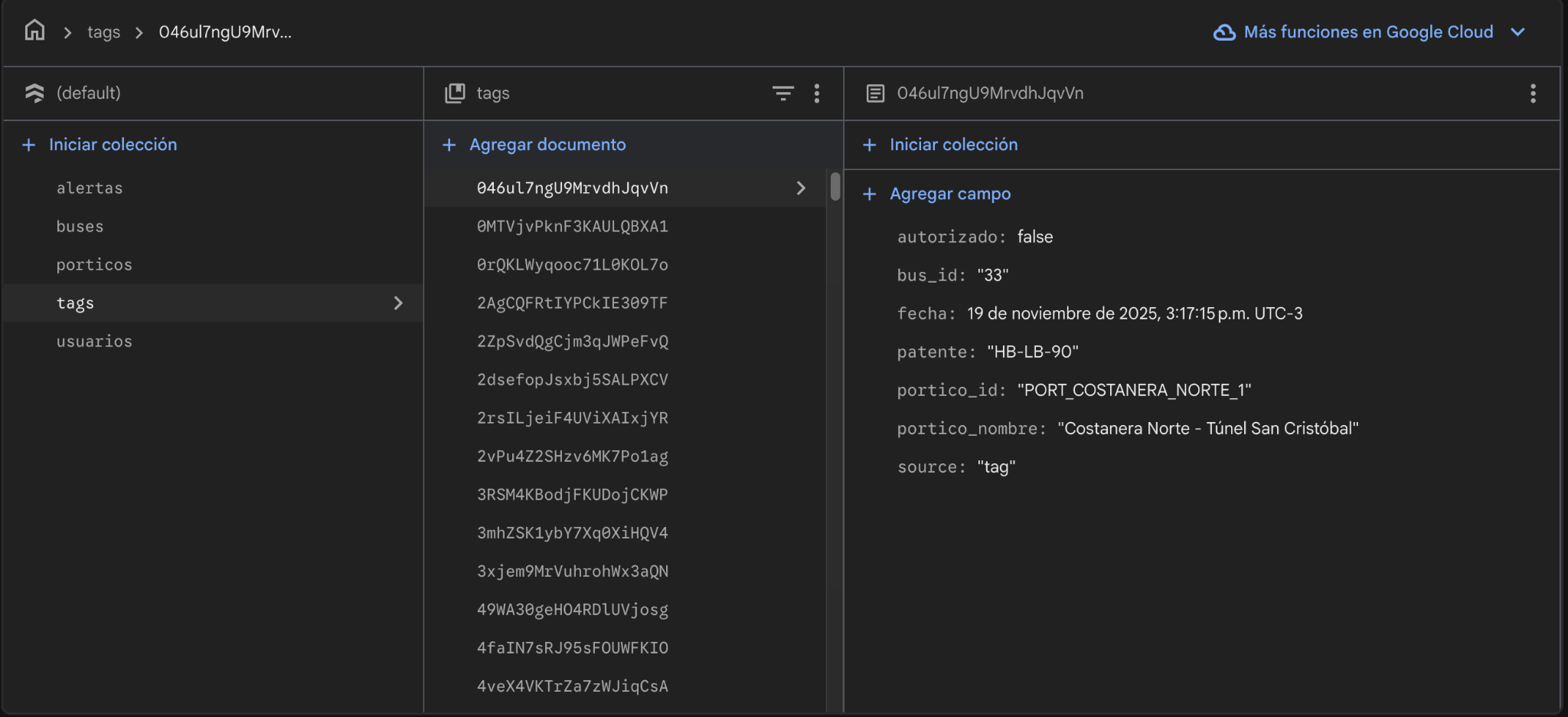
* Fecha
* Hora
* Bus
* Chofer
* Pórtico
* Estado
* Autorización

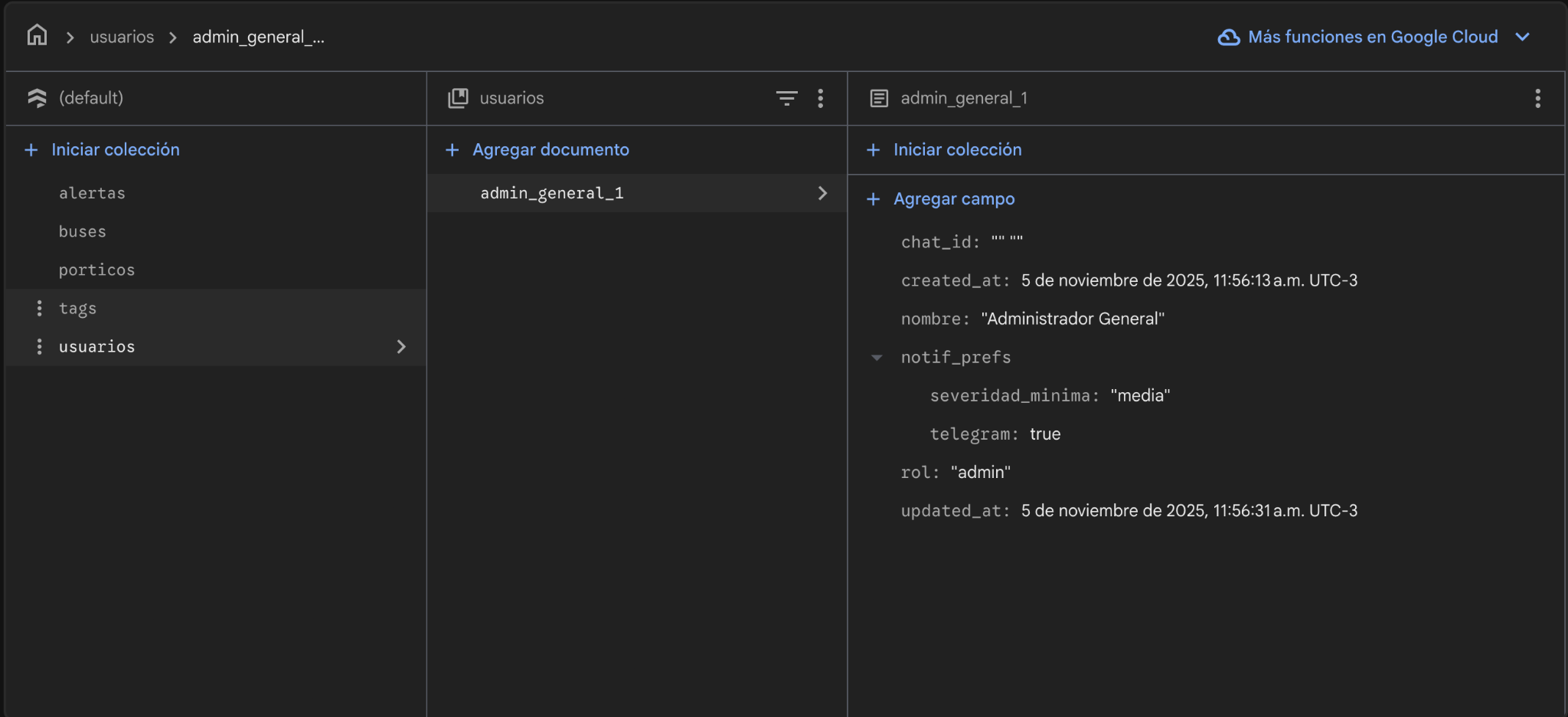
Es una base de datos escalable, rápida y fiable.











# Evidencias de Documentación

Las evidencias correspondientes al análisis, diseño, requerimientos, diagramas, metodología ágil, carta Gantt y uso del sistema de control de versiones Git se encuentran descritas en el documento “Evidencias de Documentación”.

Este documento contiene:

* Levantamiento de requerimientos
* Diagramas de flujos
* Arquitectura del sistema
* Backlog y sprints Scrum
* Carta Gantt

*Evidencias del Sistema.docx*

# Evidencias de Sistema

El funcionamiento del bot, la interacción con el usuario, la validación de rutas, la creación del bot mediante Bot Father, el registro de eventos y las alertas automáticas se encuentran documentados en “Evidencias del Sistema”.

Este documento incluye:

* Creación del bot en Bot Father
* Capturas del bot funcionando.
* Validación de rutas
* Sistema de alertas automáticas

*Evidencias del Sistema.docx*

# Evidencias de Base de Datos

La estructura completa de Firestore, los documentos reales y la integridad de los datos utilizados por el sistema se encuentran documentados en “Evidencias de Base de Datos”.

Este documento contiene:

* Colecciones Firestore
* Documentos de buses, alertas, usuarios y pórticos
* Estructura de documentos
* Ventajas de Firestore

*Evidencias de Base de Datos.docx*

# Pruebas y Validación (QA)

Pruebas realizadas:

* Pruebas unitarias a funciones clave.
* Pruebas de detección de TAG con coordenadas reales
* Simulación de rutas permitidas y no permitidas.
* Pruebas de alertas automáticas.
* Verificación de escritura y lectura en Firestore

# Fortalezas y Limitaciones

**Fortalezas:**

* Automatizado
* Tiempo real
* Escalable
* Bot Fácil de usar

**Limitaciones**

* Requiere conexión GPS estable

# Competencias del Perfil de Egreso

* Diseño de modelos de datos
* Integración de sistemas con API
* Desarrollo de software backend
* Gestión de proyectos informáticos
* Pruebas y validación

# Intereses y Proyección Profesional

El proyecto confirmó mi interés por:

* Automatización
* Backend y Cloud Computing
* Integración de APIs
* Arquitectura de software

Mi proyección va orientada a especializarme en empresas tecnológicas.

# Conclusión

El desarrollo de Acer Bot – Bot Automatizado de Gestión de Buses permitió abordar una problemática real del sector transporte mediante una solución tecnológica integral basada en automatización, servicios cloud y validación en tiempo real. A lo largo del proyecto se logró construir un sistema capaz de detectar pasos por TAG, validar rutas autorizadas, generar alertas inmediatas y centralizar información operativa mediante un bot conversacional, cumpliendo así con los objetivos generales y específicos planteados al inicio de la Fase 2.

La integración entre Node.js, Firebase Firestore, APIs GPS/TAG y Telegram proporcionó una arquitectura escalable, flexible y alineada con los estándares actuales de la industria. El uso de la metodología Scrum facilitó el avance incremental del sistema, permitiendo priorizar tareas, organizar fases de implementación y mejorar continuamente la solución mediante ciclos iterativos. Asimismo, la Carta Gantt brindó una visualización clara de todas las actividades y permitió gestionar adecuadamente los tiempos y dependencias del proyecto.

Acer Bot demuestra ser una solución viable para empresas de transporte que buscan optimizar su operación, reducir costos por desvíos, mejorar la trazabilidad y fortalecer el control interno. Además, el proyecto contribuye directamente al perfil profesional del ingeniero informático, aplicando competencias esenciales como desarrollo de software, automatización, diseño de modelos de datos, integración de servicios y gestión de proyectos tecnológicos. En síntesis, el trabajo realizado no solo cumple con los requisitos académicos, sino que también representa un aporte concreto a la transformación digital del sector transporte en Chile.

# Reflexión (Inglés)

Working on AcerBot was an experience that significantly strengthened both my technical and professional competencies. Throughout the development of this project, I was able to apply cloud services, API integrations, real-time data processing and backend automation in a practical and meaningful way. These are skills that directly align with the current demands of the industry and with my long-term career goals.

One of the most valuable aspects of the project was learning how to plan, structure and execute a complete technological solution from scratch. Using Scrum allowed me to work iteratively, manage the workload more effectively and make continuous improvements as new challenges appeared. It also helped me better understand how professional development teams organize themselves, even though this was an individual project.

This project also reinforced the importance of building systems that solve real problems for real industries. Understanding the operational needs of transportation companies made the development process more purposeful and motivated me to create a system that could truly help organizations reduce costs, improve traceability and enhance decision-making.

On a personal level, AcerBot helped me confirm my interest in automation, cloud architecture and intelligent operational systems. It encouraged me to continue learning about scalable infrastructure, API orchestration, real-time monitoring and the integration of bots as communication tools. This experience gives me confidence to face more complex technological challenges in the future and motivates me to grow professionally in areas related to backend engineering, cloud computing and digital transformation.

# Bibliografía

Firebase. (2024). *Cloud Firestore Documentation*. Google. <https://firebase.google.com/docs/firestore?hl=es-419>

Telegram. (2024). *Telegram Bot API – Bot Father Guide*. https://core.telegram.org/bots

Node.js Foundation. (2024). *Node.js Official Documentation*. https://nodejs.org/docs/latest/api/

Ministerio de Obras Públicas de Chile. (2024). *Sistema de Telepeaje TAG.* [*https://www.mop.gob.cl/*](https://www.mop.gob.cl/)

Costanera Norte. (2024). *Información técnica de pórticos TAG. https://web.costaneranorte.cl/*

# 

# 

# 

# 

# 

# 

# Anexos

**ANEXO 1 – Evidencias de Documentación**

Este anexo contiene toda la documentación generada durante la etapa de análisis y diseño del proyecto AcerBot. Incluye:

* Levantamiento de requerimientos funcionales y no funcionales
* Análisis de necesidades del negocio
* Diagramas de flujo del sistema
* Arquitectura general de la solución
* Diseño de la base de datos en Firestore
* Metodología Ágil Scrum
* Backlog, sprint planning y definición de roles
* Carta Gantt del proyecto

**ANEXO 2 – Evidencias del Sistema**

Este anexo agrupa las evidencias de funcionamiento del bot y de los distintos módulos desarrollados. Incluye:

* Creación del bot en BotFather
* Configuración de comandos y token (oculto en el informe)
* Ejecución real de comandos en Telegram
* Demostración de interacción del bot con lenguaje natural
* Validación de rutas autorizadas y no autorizadas
* Ejecución del sistema de alertas automáticas
* Capturas del bot detectando eventos TAG en tiempo real

**ANEXO 3 – Evidencias de Base de Datos**

Este anexo presenta la estructura de Firestore utilizada por el sistema y los registros generados durante las pruebas. Contiene:

Capturas de las colecciones Firestore del proyecto

* /buses
* /alertas
* /pórticos
* /tags
* /usuarios

Ejemplo de documentos reales generados por el sistema.

Validación de integridad de datos.

Ventajas del uso de Firestore en una arquitectura en tiempo real.

**ANEXO 4 – Código Fuente Relevante**

Incluye fragmentos clave del desarrollo:

* Cloud Functions en Node.js
* Validación de pórticos TAG
* Conversión y lectura de coordenadas GPS
* Alertas automáticas
* Conexión a Firebase
* Estructura modular del bot