Diseño de Arquitectura del Sistema - WayquiSafe

**Fecha: 28/08/2025**

**Autor:**

**Joan Nicol Cordova Bustamante**

**Alexander Villa Andia**

**Deyvid Brayan Vargaya Coaquira**

**Mijhael Amilkar Mejia Ballona**

**Versión:** 1.1

# 1. Introducción

Este documento describe de manera detallada la arquitectura del sistema para la aplicación móvil WaykiSafe, cuyo propósito es brindar seguridad a los turistas en la ciudad de Cusco mediante la detección de zonas de riesgo, el reporte de incidentes y la comunicación directa con autoridades y contactos de emergencia. Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas.

# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

El sistema adopta una arquitectura en capas, basada en el patrón cliente-servidor, y se organiza en tres capas principales:

* Capa de Presentación (Frontend): Interfaz con la que interactúa el usuario.
* Capa de Lógica de Negocio (Backend): Contiene la lógica que gobierna las operaciones del sistema.
* Capa de Persistencia de Datos (Base de Datos): Encargada del almacenamiento estructurado y recuperación de información.

## 2.2. Diagrama General de Arquitectura:

Diagrama

El contenido generado por IA puede ser incorrecto.

# 3. Descripción de Componentes Principales

## 3.1. Frontend (Presentación)

* **Framework:** Android Studio (Java)
* **Estilo:** Material Design + LottieFiles (animaciones interactivas)
* **Responsabilidades:**
  + Renderizar la interfaz gráfica de la aplicación.
  + Mostrar el mapa interactivo con zonas de riesgo.
  + Capturar entradas del usuario (reportes, calificaciones, botón de pánico).
  + Comunicarse con Firebase y APIs externas (Google Maps, WhatsApp).
  + Mostrar mensajes, alertas y notificaciones de riesgo.

## 3.2. Backend (Lógica de negocio)

* **Lenguaje y Servicios:** Firebase (Authentication, Realtime Database, Firestore, Cloud Functions, Cloud Messaging).
* **Responsabilidades:**
  + Gestionar autenticación segura (correo y Google).
  + Procesar reportes ciudadanos y almacenarlos en la BD.
  + Integrar datos de la API de Twitter/X y analizarlos con IA (NLP básico).
  + Emitir notificaciones push en tiempo real (zonas de riesgo, emergencias).
  + Validar ubicación del usuario y derivar llamadas según seguro médico.

### 3.2.1. Módulos del Backend

* authController.js: Manejo de login y registro
* taskController.js: Operaciones CRUD de tareas
* listController.js: Operaciones sobre listas de tareas
* middleware/auth.js: Validación de tokens JWT
* services/emailService.js: Envío de notificaciones

## 3.3. Base de Datos (Persistencia)

* **Sistema Gestor:** Firebase Realtime Database / Firestore.
* **Esquema de Datos:**
  + **Usuarios:** id, nombre, correo, teléfono, contactos de emergencia, tipo de seguro.
  + **Reportes:** id, tipo de incidente, nivel de riesgo (verde/naranja/rojo), ubicación, fecha/hora, usuario.
  + **Zonas:** id, coordenadas, nivel de riesgo, calificación de usuarios.
  + **Alertas:** id, usuario, ubicación, tipo de alerta, estado.

# 4. Integraciones Externas (Opcionales)

* Google Maps API: Para visualización de mapas y geolocalización.
* **API de Twitter/X:** Para recolección de datos públicos y detección de incidentes mediante IA.
* **WhatsApp API:** Para comunicación rápida con contactos de emergencia

# 5. Seguridad

* Autenticación de usuarios con Firebase Authentication.
* Cifrado y almacenamiento seguro de credenciales.
* Validación de datos de entrada para evitar inyecciones.
* Uso obligatorio de HTTPS/TLS para todas las comunicaciones.
* Tokens de sesión seguros con control de expiración.
* Activación de alertas mediante agitación del dispositivo, incluso en segundo plano.

# 6. Escalabilidad y Despliegue

Posibles problemas que pueden surgir durante el desarrollo.

* **Frontend (App móvil):** Distribuida a través de **Google Play Store.**
* **Backend:** Arquitectura **serverless** con Firebase, escalable automáticamente según la demanda.
* **Base de Datos:** Firestore y Realtime Database en la nube.
* **Monitoreo:** Firebase Crashlytics para detección de errores.
* La arquitectura permite escalar horizontalmente el backend y soportar más usuarios sin reconfiguración compleja.

# 7. Conclusiones

La arquitectura propuesta para **WaykiSafe** combina simplicidad, escalabilidad y seguridad. Aprovecha los servicios de Firebase y APIs externas para garantizar disponibilidad y rendimiento en tiempo real. Este diseño está alineado con las **buenas prácticas de desarrollo de aplicaciones móviles** y permite la evolución futura del sistema hacia funcionalidades más avanzadas como predicción de riesgos, integración con sistemas policiales y soporte multilenguaje.