Documentación Técnica – Proyecto Imbitis

# 1. Introducción

Imbitis es una aplicación móvil orientada a asistir a usuarios no capacitados durante una emergencia médica.   
Su propósito es guiar paso a paso en primeros auxilios, ofreciendo una interfaz simple, soporte por voz y acceso rápido al número de emergencia (131).  
La arquitectura del sistema se diseñó bajo el enfoque Modelo 4+1 de Kruchten, complementado con un estilo arquitectónico cliente–servidor con microservicios ligeros,  
para garantizar escalabilidad, mantenibilidad y respuesta en tiempo real.

# 2. Estilo Arquitectónico Utilizado

Tipo: Cliente–Servidor con microservicios y API REST.  
  
Descripción:  
- La aplicación móvil (cliente) realiza peticiones HTTP a los servicios backend.  
- El servidor backend (API) gestiona la lógica de negocio, almacenamiento de datos (guías, pasos, feedback) y autenticación mínima.  
- Una base de datos relacional (MySQL o Firebase Realtime Database) almacena usuarios, feedback y contenido multimedia.  
- Se utiliza un servicio de text-to-speech (TTS) para la generación de narraciones.  
  
Justificación:  
- Permite la actualización del contenido sin necesidad de reinstalar la app.  
- Se adapta a distintos dispositivos Android.  
- Facilita la integración futura con sistemas de emergencia nacionales (131).

# 3. Modelo 4+1

## 3.1 Vista Lógica (Modelo de Clases / Componentes)

Objetivo: Mostrar los módulos principales y cómo se relacionan.  
  
Componentes principales:  
1. UI Module: pantallas de inicio, selección de emergencia, guías paso a paso, feedback.  
2. Logic Module: controla flujo de pasos, validación de voz, navegación.  
3. EmergencyService Module: gestiona llamadas al 131 y accesos rápidos.  
4. Data Module: gestiona el almacenamiento local (SQLite o Firebase).  
5. Audio Module: controla narraciones y repetición automática.  
  
Relaciones:  
- UI interactúa con Logic para mostrar contenido.  
- Logic consulta Data y activa Audio o EmergencyService.

## 3.2 Vista de Desarrollo (Estructura de Código / Módulos)

Estructura del proyecto:  
/imbitis-app/  
│  
├── /ui/  
│ ├── main\_menu.dart  
│ ├── emergency\_list.dart  
│ ├── step\_guide.dart  
│ ├── feedback\_form.dart  
│  
├── /services/  
│ ├── audio\_service.dart  
│ ├── emergency\_call\_service.dart  
│ ├── feedback\_service.dart  
│  
├── /models/  
│ ├── Emergency.dart  
│ ├── Step.dart  
│ ├── Feedback.dart  
│  
├── /data/  
│ ├── local\_db.dart  
│ ├── api\_connection.dart  
│  
└── /assets/  
 ├── images/  
 ├── audio/  
  
Lenguaje sugerido: Flutter (Dart) o Kotlin (Android Nativo)  
Control de versiones: Git + GitHub

## 3.3 Vista de Procesos (Vista Dinámica / Concurrencia)

Objetivo: Representar cómo fluyen las interacciones en tiempo real.  
  
Flujo típico:  
1. El usuario abre Imbitis.  
2. Selecciona un tipo de emergencia.  
3. La app solicita al backend las instrucciones.  
4. El backend devuelve pasos y recursos multimedia.  
5. El módulo de voz reproduce la instrucción.  
6. El usuario confirma “Listo” (voz o toque).  
7. La app pasa al siguiente paso o permite llamar al 131.  
8. Al final, se ofrece un formulario de feedback.  
  
Procesos paralelos:  
- Reproducción de audio y control táctil/voz se ejecutan simultáneamente.  
- Llamada al 131 interrumpe todos los procesos en curso.

## 3.4 Vista Física (Despliegue / Infraestructura)

Infraestructura propuesta:  
[Usuario]  
 │  
 ▼  
[Móvil Android con Imbitis App]  
 │ REST API Calls  
 ▼  
[Servidor Backend]  
 │  
 ▼  
[Base de Datos]  
 │  
 ▼  
[Servicio TTS - Text-to-Speech]

## 3.5 Vista de Escenarios (Casos de Uso Principales)

Escenario 1 – Emergencia médica:  
El usuario abre la app, selecciona “Atragantamiento”, sigue los pasos y logra estabilizar a la víctima.  
  
Escenario 2 – Llamada directa:  
Usuario en pánico presiona el botón de emergencia y llama automáticamente al 131.  
  
Escenario 3 – Feedback:  
Después de usar la app, el usuario envía una calificación y comentario para mejorar las guías.

# 4. Modelo de Base de Datos (Simplificado)

Entidades principales:  
- Usuario (id\_usuario, nombre, correo)  
- Emergencia (id\_emergencia, nombre, descripcion)  
- GuiaPaso (id\_paso, id\_emergencia, orden, descripcion, imagen, audio)  
- Feedback (id\_feedback, id\_usuario, id\_emergencia, comentario, calificacion)  
  
Relaciones:  
- Un Usuario puede registrar varios Feedback.  
- Una Emergencia tiene muchos GuiaPaso.  
- GuiaPaso pertenece a una sola Emergencia.

# 5. Conclusión

El diseño de Imbitis bajo el modelo 4+1 ofrece una visión integral que asegura:  
- Modularidad y mantenibilidad del código.  
- Soporte multiplataforma.  
- Escalabilidad en Google Cloud.  
- Facilidad para futuras integraciones con sistemas de salud.