Documentacion fase 2.2

Proyecto Imbitis

Imbitis

Protege tu salud, cuida a los otros

16. Lista de verificación de aceptación

Esta tabla muestra y verifica el cumplimiento de los criterios de aceptación definidos para el Producto Mínimo Viable (MVP) del proyecto Imbitis.

IDENTIFICACIÓN	Criterio de Aceptación	Cumple (Sí/No)	Observaciones
CA01	La aplicación inicia correctamente y muestra el menú principal sin solicitar credenciales (acceso anónimo).	Si	Cumple con HU1.1. Acceso inmediato
CA02	El botón 131 inicia la intención de llamada de emergencia correctamente en el sistema operativo.	Si	Verificado. Lanza el marcador nativo con el número precargado.
CA03	Las guías se visualizan paso a paso, mostrando texto, imágenes y reproduciendo audio sincronizado (TTS).	Si	Sincronización correcta entre vista y narración.
CA04	El usuario puede navegar entre los pasos (Atrás/Siguiente) sin errores de flujo.	Si	Navegación fluida validada en pruebas.
CA05	El formulario de retroalimentación registra la calificación y comentarios anónimamente en la base de datos.	Si	Conectado exitosamente a tabla feedbacken Supabase.
CA06	El diseño de la interfaz se adapta a diferentes tamaños de pantalla (Responsive).	Si	Probado en simuladores y dispositivos físicos.
CA07	La aplicación no presenta cierres inesperados (crashes) durante el flujo crítico de una emergencia.	Si	Estabilidad verificada en Fase 3.
CA08	El Widget de pantalla de inicio permite acceder rápido a la aplicación.	Si	Verificado. Funcionalidad operativa tras corrección de permisos.

17. Registro de Bugs

El siguiente registro detalla los principales errores detectados durante el ciclo de desarrollo y su estado actual.

Error de identificación	Descripción	Estado	Prioridad	Responsable	Fase	Fecha Resolución	Observaciones
BUG01	El botón “Siguiente” se oculta detrás de la barra inferior de navegación.	**Cerrado**	Alta	Eugenio	Fase 2	15/10/25	Solucionado ajustando el *padding* inferior en el diseño XML.
BUG02	No se logra ejecutar el formato Widget en la pantalla de inicio.	**Cerrado**	Media	Eugenio	Fase 2	07/11/25	Se corrigieron los permisos en el manifiesto y la configuración del AppWidgetProvider.
BUG03	Problema con el formato y tamaño de letra en dispositivos pequeños.	**Cerrado**	Media	Cristóbal	Fase 1	30/08/25	Se estandarizaron las fuentes usando unidades sp(píxeles independientes de escala).

18. Informe de Cierre de Sprint/Release

El **Sprint 7** (Release Final) tuvo como objetivo completar y validar todas las funcionalidades principales de la aplicación Imbitis, incluyendo la corrección de errores críticos.

Resultados del Sprint:

- Finalización del menú principal y flujo de guías de emergencia.
- Integración completa del botón de llamada 131.
- Implementación de narración por voz sincronizada (TTS).
- **Solución del BUG02:** Implementación correcta del Widget de acceso rápido.
- Validación funcional con pruebas internas y externas (Checklist completo).

Conclusión:

El Sprint concluyó con el cumplimiento de todos los objetivos planificados, obteniendo una versión estable y funcional de la lista de aplicaciones para su uso.

19. Acta de Cierre de Proyecto Ágil

Nombre del Proyecto: Imbitis **Fecha de Cierre:** Diciembre 2025

Resumen:

El proyecto Imbitis ha cumplido los objetivos propuestos, entregando un prototipo funcional de aplicación móvil para asistencia en primeros auxilios, desarrollado bajo un enfoque ágil con metodología Scrum.

Validación:

El Product Owner y el equipo de desarrollo confirman que el producto cumple con los criterios de aceptación definidos en el backlog y ha superado las pruebas de validación interna.

Firmas:

- Product Owner: Cristobal Maluenda
- Scrum Master: Matias Rodriguez
- Equipo de Desarrollo: Tomás Olivares, Eugenio Astrosa

20. Informe de Retrospectiva Final

Aspectos positivos:

- Comunicación efectiva entre los integrantes del equipo.
- Cumplimiento constante de los plazos establecidos (cronograma Fase 1, 2 y 3).
- Capacidad técnica para resolver problemas complejos (ej. permisos de Widget).

Aspectos a mejorar:

- Mayor planificación en la gestión de errores y pruebas tempranas.
- Anticipar conflictos de compatibilidad entre versiones de Android.
- Documentar con más detalle las configuraciones de backend durante el desarrollo.

Compromisos:

- Mantener la colaboración y revisión cruzada del código.
- Fortalecer el control de calidad antes de cada liberación.

21. Guía de Despliegue (Entorno de Desarrollo)

Esta guía describe los pasos necesarios para implementar y ejecutar el proyecto Imbitis en Android Studio.

Requisitos previos:

- Android Studio (versión 2023 o superior).
- JDK 11+ instalado.
- SDK de Android configurado (API Nivel 30 o superior).
- Conexión a Internet para dependencias y servicios TTS.
- **Cuenta de Supabase** activa y credenciales de API.

Pasos de instalación:

1. Clonar el repositorio del proyecto desde GitHub.
2. Abrir el proyecto en Android Studio.
3. Cree el archivo `local.properties`y agregue las claves `SUPABASE_URL`y `SUPABASE_KEY`.
4. Sincronizar Gradle y verificar las dependencias.
5. Configurar permisos de llamada y audio en el archivo Manifest (ya incluidos en el repositorio).
6. Ejecutar la aplicación en un emulador o dispositivo físico con Android 9+.

Dependencias principales:

- Librerías AndroidX.
- Cliente de Supabase (Postgrest).
- Servicios de Texto a Voz (Google).

Configuraciones adicionales:

- Asegúrese de que los permisos de llamada y el micrófono estén habilitados.
- Verifique que el dispositivo tenga conexión a internet para descargar las guías desde Supabase.

22. Manual de Usuario

Objetivo: Guiar al usuario en el uso de la aplicación Imbitis durante una situación de emergencia.

1. **Inicio Rápido:** Abra la aplicación pulsando el icono de Imbitis o el Widget de escritorio. La aplicación se cargará directamente en el menú principal **sin pedir registro ni contraseña** .
2. **Menú de Emergencias:** En la pantalla principal, visualice las tarjetas con los diferentes tipos de accidentes (ej. "Atragantamiento", "Choque").
3. **Llamada de emergencias:** Si la situación es crítica, presione el botón rojo de la parte superior con el icono de teléfono. Esto abrirá el marcador de su celular listo para llamar al 131 o 132.
4. **Uso de Guías:**
    - Seleccione la emergencia correspondiente.
    - La guía comenzará inmediatamente. Escuche la narración por voz y observe la imagen de referencia.
    - Utilice los botones "Siguiente" y "Atrás" o los comandos de voz para moverse por los pasos.
5. **Finalización y retroalimentación:** Al terminar la guía, puede usar el formulario opcional para calificar si la ayuda fue útil.

Manual Técnico

Arquitectura del Sistema: El sistema sigue el **Modelo 4+1 de Kruchten** y un estilo arquitectónico **Cliente-Servidor** .

- **Frontend:** Aplicación nativa Android (desarrollada en Android Studio).
- **Backend:** Supabase (BaaS) exponiendo una API REST.

Estructura de Datos (Modelo Físico en Supabase): El sistema utiliza una base de datos PostgreSQL gestionada por Supabase. El esquema final implementado consta de tres tablas principales:

Tabla	***Columnas***	***Descripción***
guías	*guide_id(PAQUETE), title, description, category, image_url,created_at*	*Almacena la información general de cada guía. La columna category agrupa las guías en el menú.*
pasos	*step_id(PK), guide_id(FK), step_number, instruction, image_url,audio_url*	*Detalle paso a paso. Vinculado a guides mediante guide_id.*
comentario	*id(PK), guide_id(FK), rating, comments,created_at*	*Calificaciones y comentarios anónimos vinculados a una guía.*

Dependencias y Servicios:

- **Servicio TTS:** Se utiliza el motor *Text-to-Speech* nativo de Android para la narración.
- **Gestión de Dependencias:** Gradle.
- **Conectividad:** Cliente Supabase para peticiones de datos.

Instrucciones de Despliegue:

1. Clonar el repositorio desde GitHub.
2. Configurar las claves de API de Supabase en el archivo `local.properties`.
3. Sincronizar el proyecto con Gradle.
4. Compilar y generar el APK firmado para distribución.

23. Plan de Continuidad o Monitoreo

Objetivo: Garantizar la disponibilidad de las guías y la operatividad del backend tras la entrega.

Acciones de:

1. **Base de Datos:** Revisión mensual de la integridad de los datos en Supabase y ejecución de copias de seguridad automáticas.
2. **Contenido:** Actualización de los textos y audios de las guías (tabla `steps`) directamente desde el backend sin necesidad de actualizar la aplicación en la tienda.
3. **Compatibilidad:** Revisión semestral ante nuevas versiones de Android (Nivel API) para asegurar que el servicio TTS, el Widget y los permisos de llamada sigan funcionando correctamente.