Manual Técnico de MediControl

Índice

1. Introducción

MediControl es una aplicación multiplataforma desarrollada en Flutter para la gestión de medicamentos, recordatorios de medicación, escaneo de recetas y asistencia mediante IA. Este manual técnico está diseñado para desarrolladores y personal técnico que necesite instalar, configurar, mantener o extender la aplicación.

2. Requisitos del Sistema

Desarrollo

- Flutter SDK 3.1.0 o superior
- Dart SDK 3.0.0 o superior
- Visual Studio Code o Android Studio
- Git (control de versiones)

Dispositivos

- Android 7.0 (API nivel 24) o superior
- iOS 12.0 o superior
- 100 MB de espacio disponible mínimo
- 2 GB de RAM recomendados

Servicios en la nube

- Cuenta activa en Supabase

- API key para servicios de IA
- Variables de entorno configuradas

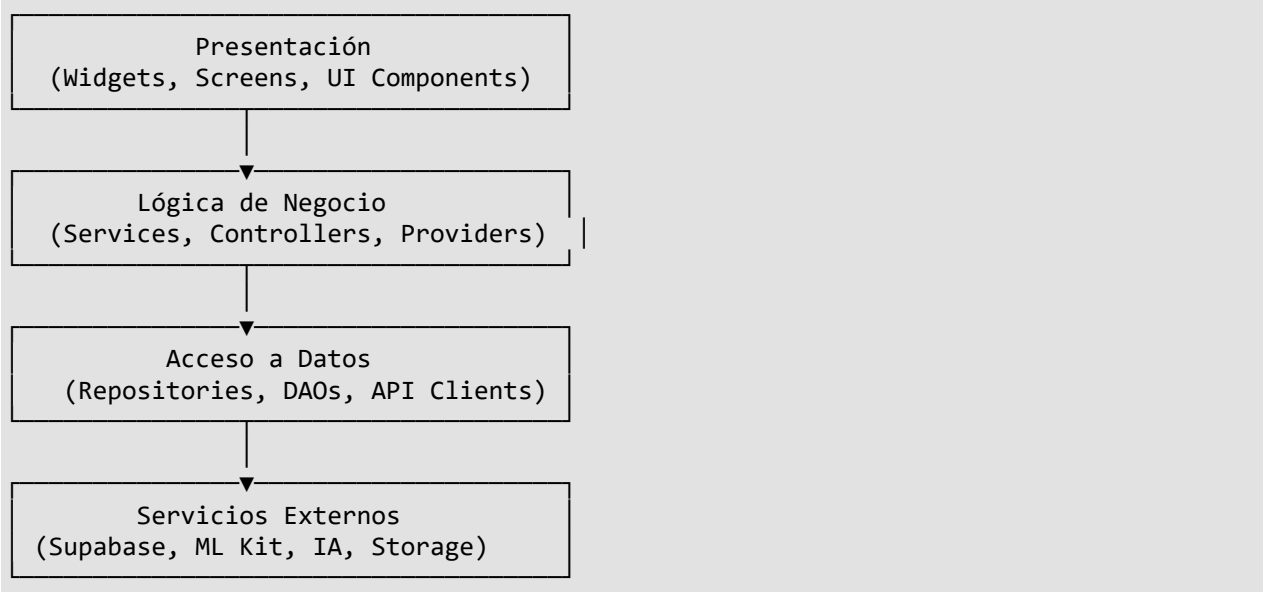
3. Arquitectura de la Aplicación

MediControl sigue una arquitectura basada en el patrón de diseño de separación de responsabilidades, adaptada al contexto de Flutter. Los componentes principales son:

Capas de la Aplicación

1. **Capa de Presentación**: Widgets y pantallas de Flutter
2. **Capa de Lógica de Negocio**: Controladores y servicios que manejan la lógica principal
3. **Capa de Acceso a Datos**: Conexión con Supabase y almacenamiento local
4. **Servicios**: Componentes transversales como notificaciones, reconocimiento de texto, etc.

Diagrama de Arquitectura



4. Instalación y Configuración

Requisitos previos

1. Instalar Flutter SDK
2. Configurar Android Studio o VS Code con las extensiones recomendadas para Flutter
3. Configurar un emulador o conectar un dispositivo físico

Pasos de instalación

1. Clonar el repositorio:

```
git clone https://github.com/username/medicontrol.git
cd medicontrol
```

2. Instalar dependencias:

```
flutter pub get
```

3. Crear archivo `.env` con las variables de entorno necesarias:

```
SUPABASE_URL=your_supabase_url
SUPABASE_ANON_KEY=your_supabase_anon_key
AI_API_KEY=your_ai_api_key
```

4. Ejecutar la aplicación:

```
flutter run
```

Configuración de servicios externos

Supabase

1. Crear un proyecto en Supabase
2. Ejecutar el script SQL de creación de tablas (disponible en `db/schema.sql`)
3. Configurar reglas de Row Level Security (RLS) para proteger los datos
4. Obtener la URL y clave anónima para la conexión

Google ML Kit

La configuración de Google ML Kit para el reconocimiento de texto ya está incluida en el proyecto. Para Android, no se requiere configuración adicional. Para iOS, es necesario:

1. Actualizar `ios/Runner/Info.plist` con los permisos de cámara
2. Asegurarse de que el Pod de MLKit está correctamente configurado en `ios/Podfile`

5. Estructura del Proyecto

```
lib/
├── main.dart              # Punto de entrada de la aplicación
├── login.dart             # Pantalla de inicio de sesión
├── register.dart          # Pantalla de registro
├── home.dart              # Pantalla principal/dashboard
├── medicamentos.dart      # Gestión de medicamentos
├── add_medicamento.dart   # Añadir/editar medicamento
├── historial.dart         # Historial de medicación
├── perfil.dart            # Perfil de usuario
├── ai_assistant.dart      # Asistente virtual con IA
├── scan_prescriptions.dart # Escáner de recetas
├── notification_service.dart # Servicio de notificaciones original
├── notification_service_fixed.dart # Servicio de notificaciones mejorado
├── splash_screen.dart     # Pantalla de carga inicial
├── test_notification.dart # Pantalla de prueba de notificaciones
└── utils/                 # Utilidades y helpers
    └── responsive_utils.dart # Utilidades para diseño responsivo
```

6. Base de Datos

La aplicación utiliza Supabase como backend, que a su vez utiliza PostgreSQL. El esquema principal de la base de datos es el siguiente:

Estructura de tablas

`usuarios`

```
CREATE TABLE usuarios (
  id UUID REFERENCES auth.users NOT NULL PRIMARY KEY,
  nombre TEXT,
  apellidos TEXT,
  email TEXT UNIQUE NOT NULL,
  telefono TEXT,
  fecha_nacimiento DATE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

`medicamentos`

```
CREATE TABLE medicamentos (
  id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
  usuario_id UUID REFERENCES usuarios NOT NULL,
  nombre TEXT NOT NULL,
  descripcion TEXT,
  dosis TEXT NOT NULL,
  unidad TEXT,
  via_administracion TEXT,
  fecha_inicio TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  fecha_fin TIMESTAMP WITH TIME ZONE,
  periodicidad TEXT,
  horarios JSONB,
  notas TEXT,
  color TEXT,
  activo BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

`tomas_medicamentos`

```
CREATE TABLE tomas_medicamentos (
  id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
  medicamento_id UUID REFERENCES medicamentos NOT NULL,
  fecha_programada TIMESTAMP WITH TIME ZONE NOT NULL,
  fecha_toma TIMESTAMP WITH TIME ZONE,
  estado TEXT DEFAULT 'pendiente', -- pendiente, tomado, omitido
  notas TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Índices importantes

```
CREATE INDEX idx_medicamentos_usuario ON medicamentos(usuario_id);
CREATE INDEX idx_tomas_medicamento ON tomas_medicamentos(medicamento_id);
CREATE INDEX idx_tomas_estado ON tomas_medicamentos(estado);
```

Políticas de seguridad (RLS)

```
-- Usuarios solo pueden ver y modificar sus propios datos
CREATE POLICY "Los usuarios pueden ver sus propios datos"
ON usuarios FOR SELECT
USING (auth.uid() = id);

-- Medicamentos solo visibles para su propietario
CREATE POLICY "Los usuarios pueden ver sus propios medicamentos"
ON medicamentos FOR ALL
USING (auth.uid() = usuario_id);
```

7. APIs y Servicios Externos

Supabase

La aplicación utiliza varias funcionalidades de Supabase:

- Autenticación y gestión de usuarios
- Base de datos PostgreSQL
- Storage para almacenamiento de imágenes

Ejemplo de uso:

```
final supabase = Supabase.instance.client;

// Autenticación
final response = await supabase.auth.signInWithPassword(
  email: email,
  password: password,
);

// Consulta a la base de datos
final data = await supabase
  .from('medicamentos')
  .select()
  .eq('usuario_id', supabase.auth.currentUser!.id);
```

Google ML Kit

Se utiliza para el reconocimiento de texto en las recetas médicas:

```
final InputImage inputImage = InputImage.fromFile(imageFile);
final textRecognizer = TextRecognizer();
final RecognizedText recognizedText = await textRecognizer.processImage(inputImage);
textRecognizer.close();
```

APIs de IA

La aplicación se integra con servicios de IA para el asistente virtual. La implementación utiliza una clave de API almacenada en variables de entorno.

8. Seguridad

Autenticación

- Autenticación basada en JWT mediante Supabase
- Almacenamiento seguro de tokens y credenciales
- Sesiones con expiración configurable
- Refresh tokens para renovación automática

Almacenamiento de datos sensibles

- Uso de `flutter_secure_storage` para almacenamiento local seguro
- Cifrado en tránsito mediante HTTPS
- Row Level Security (RLS) en Supabase para protección a nivel de datos

Permisos

La aplicación requiere los siguientes permisos:

- Internet: Para conexión con servicios en la nube
- Cámara: Para escaneo de recetas
- Almacenamiento: Para guardar imágenes temporales
- Notificaciones: Para recordatorios de medicación
- Vibración: Para notificaciones táctiles

9. Solución de Problemas Comunes

Problemas de notificaciones

Si las notificaciones no funcionan correctamente:

1. Verificar que los permisos están concedidos
2. Comprobar que la app no está optimizada para ahorro de batería
3. Validar que las zonas horarias están configuradas correctamente
4. Usar `notification_service_fixed.dart` en lugar de `notification_service.dart`

Problemas con el escaneo de recetas

Si el reconocimiento de texto no funciona adecuadamente:

1. Asegurar buena iluminación y enfoque

2. Verificar que los permisos de cámara están concedidos
3. Comprobar que Google ML Kit está actualizado
4. Probar con una imagen de la galería para descartar problemas de cámara

Errores de conexión con Supabase

Si hay problemas de conexión:

1. Validar conectividad a internet
2. Verificar que las claves API son correctas
3. Comprobar el estado del servicio de Supabase
4. Revisar tokens de autenticación y su validez

10. Mantenimiento

Actualizaciones periódicas

Se recomienda realizar las siguientes acciones periódicamente:

1. Actualizar las dependencias de Flutter: `flutter pub upgrade`
2. Comprobar actualizaciones del SDK de Flutter
3. Mantener al día los servicios de Google ML Kit
4. Revisar el panel de Supabase para actualizaciones y mejoras

Monitorización

Para una monitorización efectiva, se puede:

1. Implementar Firebase Crashlytics para registro de errores
2. Utilizar Firebase Performance Monitoring
3. Configurar alertas en Supabase para uso excesivo de recursos
4. Realizar auditorías de seguridad periódicas

Backup y recuperación

Se recomienda:

1. Configurar backups automáticos en Supabase
2. Exportar periódicamente la estructura de la base de datos
3. Documentar procesos de restauración
4. Realizar simulacros de recuperación ocasionales

Apéndice A: Glosario de términos técnicos

Término	Descripción
Flutter	Framework de desarrollo multiplataforma creado por Google
Supabase	Plataforma de backend como servicio (BaaS) basada en PostgreSQL
ML Kit	Biblioteca de Google para machine learning en dispositivos móviles
RLS	Row Level Security, sistema de seguridad a nivel de registro en PostgreSQL
Flutter Local Notifications	Plugin para gestionar notificaciones locales en Flutter
PostgreSQL	Sistema de gestión de bases de datos relacional

Apéndice B: Referencias y recursos útiles

- Documentación oficial de Flutter
- Documentación de Supabase
- Google ML Kit
- Flutter Local Notifications