Manual Técnico - Sistema de Incidencias Escolares

Índice

1. Introducción

Propósito

Sistema móvil desarrollado en Flutter para la gestión integral de incidencias escolares, permitiendo registro, seguimiento y administración de eventos disciplinarios en instituciones educativas.

Tecnologías Utilizadas

Frontend: Flutter (Dart)

Backend: API REST

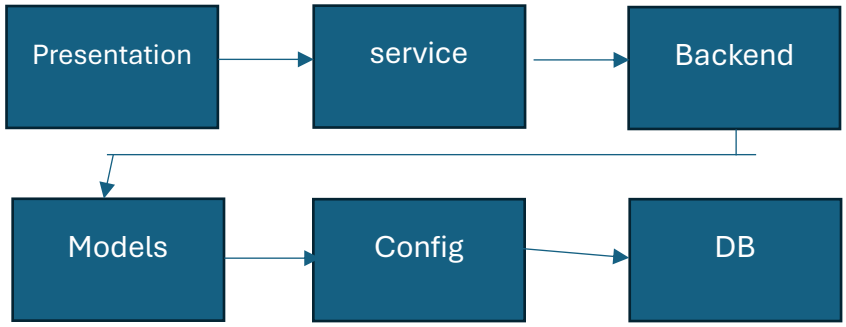
Autenticación: JWT (JSON Web Tokens)

Gestión de Estado: setState (Estado local)

Networking: HTTP package

2. Arquitectura del Sistema

Patrón de Arquitectura



Capas del Sistema

Presentación: Screens y widgets UI

Servicios: Lógica de negocio y comunicación API

Modelos: Estructuras de datos

Configuración: Variables globales y endpoints

3. Estructura del Proyecto

```
lib/
├── config/
│   ├── api_config.dart       # Configuración de endpoints
│   └── global.dart           # Variables globales
├── models/
│   ├── alumno.dart           # Modelo de estudiante
│   ├── grupo.dart            # Modelo de grupo/clase
│   ├── reporte.dart          # Modelo de incidencia
│   ├── seguimiento.dart      # Modelo de seguimiento
│   ├── tipo_reporte.dart     # Modelo de tipos de incidencia
│   └── usuario.dart          # Modelo de usuario
```

```
├── screens/
│   ├── alumno_form_screen.dart
│   ├── alumnos_screen.dart
│   ├── grupo_detalle_screen.dart
│   ├── grupo_form_screen.dart
│   ├── grupos_screen.dart
│   ├── login_screen.dart
│   ├── menu_principal_screen.dart
│   ├── reporte_detail_screen.dart
│   ├── reporte_form_screen.dart
│   ├── reportes_screen.dart
│   ├── tipo_reporte_form_screen.dart
│   ├── tipos_reporte_screen.dart
│   ├── usuario_form_screen.dart
│   └── usuarios_screen.dart
├── services/
│   ├── alumno_service.dart
│   ├── grupo_service.dart
│   ├── reporte_service.dart
│   ├── tipo_reporte_service.dart
│   └── usuario_service.dart
└── main.dart
```

4. Configuración del Entorno

Variables de Configuración

```
// config/api_config.dart
const String apiBaseUrl = 'http://tu-servidor.com/api';

// config/global.dart
String? jwtToken;         // Token de autenticación
String? usuarioRol;       // Rol del usuario logueado
String? notasUsuario;     // Información adicional
```

Dependencias Principales


```
// config/api_config.dart
const String apiBaseUrl = 'http://tu-servidor.com/api';

// config/global.dart
String? jwtToken;         // Token de autenticación
String? usuarioRol;       // Rol del usuario logueado
String? notasUsuario;     // Información adicional
```


5. Modelos de Datos

Usuario

```
class Usuario {
  final int id;
  final String nombre;
  final String apaterno;
  final String amaterno;
  final String correo;
  final String contrasena;
  final String rol;
}
```

Alumno

Reporte

6. Servicios y APIs

Estructura de Servicio Base

```
class BaseService {
  final String baseUrl = apiBaseUrl;

  Map<String, String> get headers => {
    'Authorization': 'Bearer $jwtToken',
    'Content-Type': 'application/json',
  };

  Future<http.Response> get(String endpoint) async {
    return await http.get(Uri.parse('$baseUrl$endpoint'), headers: headers);
  }
}
```

Ejemplos de Endpoints

```
// UsuarioService
GET    /usuarios           # Listar usuarios
POST   /usuarios           # Crear usuario
PUT    /usuarios/{id}      # Actualizar usuario
DELETE /usuarios/{id}      # Eliminar usuario
```

```
// ReporteService
GET    /reportes            # Listar reportes
POST   /reportes            # Crear reporte
PUT    /reportes/{id}       # Actualizar reporte
POST   /reportes/{id}/seguimientos  # Crear seguimiento
```

7. Pantallas y Funcionalidades

Menu Principal

Archivo: menu_principal_screen.dart

Funcionalidad:

Autenticación por roles

Navegación a módulos

Logout con confirmación

```
// Opciones según permisos
if (_permisos.contains('manage_all')) {
  // Admin: acceso completo
  _opciones = todasLasOpciones;
} else {
  // Usuario: solo consulta
  _opciones = opcionesLimitadas;
}
```

Gestión de Reportes

Creación: Validación de campos obligatorios

Seguimiento: Actualización de estatus

Filtrado: Por alumno, tipo y estatus

Búsqueda: Por nombre/matrícula de alumno

Formularios Dinámicos

Validación: Campos marcados con asterisco (*)

Dropdowns: Carga dinámica desde API

Fechas: DatePicker con formato localizado

Confirmaciones: Diálogos de éxito/error

8. Gestión de Estado

Variables Globales

```
// Autenticación
String? jwtToken;
String? usuarioRol;

// Estado de la aplicación
bool _loading = false;
List<Model> _datos = [];
Model? _seleccionado;
```

Patrón de Actualización

```
void _cargarDatos() async {
 setState(() => _loading = true);
 try {
  final datos = await service.obtenerDatos();
  setState(() {
   _datos = datos;
   _loading = false;
  });
 } catch (e) {
  setState(() => _loading = false);
```

```
    _mostrarError(e);
  }
}
```

9. Autenticación y Autorización

Flujo de Login

```
Future<bool> login(String correo, String contrasena) async {
  final response = await http.post(
    Uri.parse('$apiBaseUrl/login'),
    body: json.encode({'correo': correo, 'contrasena': contrasena}),
  );

  if (response.statusCode == 200) {
    final data = json.decode(response.body);
    jwtToken = data['token'];
    usuarioRol = data['rol'];
    return true;
  }
  return false;
}
```

Control de Acceso

```
List<String> _permisosFromRol(String rol) {
  if (rol.toLowerCase() == 'admin') {
    return ['manage_all', 'view_incidencias'];
  }
  return ['view_incidencias'];
}
```

10. Validaciones

Campos Obligatorios

```
// Marcado visual con asterisco
decoration: InputDecoration(
  labelText: 'Campo *',
  hintText: 'Campo obligatorio',
),

// Validación funcional
validator: (value) => value == null || value.isEmpty
  ? 'Campo requerido'
  : null,
```

Validaciones Específicas

Email: Formato de correo electrónico

Duplicados: Usuario existente

Fechas: Rango válido

Selecciones: Elementos requeridos en dropdowns

Manejo de Errores

```
try {
  await service.guardar(datos);
  _mostrarExito();
} catch (e) {
  String mensaje = 'Error genérico';
  if (e.toString().contains('duplicate')) {
    mensaje = 'Ya existe un registro con estos datos';
```

```
 }
 _mostrarError(mensaje);
}
```

11. Mantenimiento y Deployment

Configuración de Entornos

```
// Desarrollo
const String apiBaseUrl = 'http://localhost:3000/api';

// Producción
const String apiBaseUrl = 'https://api.escuela.com/api';
```

Logging y Debug

```
import 'dart:developer' as developer;

developer.log('Mensaje de debug', name: 'ModuloNombre');
```

Build para Producción

```
# Android
flutter build apk --release

# iOS
flutter build ios --release
```

Consideraciones de Seguridad

JWT tokens almacenados en memoria

Validación de entrada en frontend y backend

HTTPS obligatorio en producción

Timeout de sesión automático

Conclusiones

El Sistema de Incidencias Escolares implementa una arquitectura limpia y escalable con:

- Separación clara de responsabilidades
- Gestión robusta de errores
- Interfaz intuitiva con validaciones
- Seguridad mediante JWT
- Flexibilidad para diferentes roles
- Próximas Mejoras
- Offline support con SQLite local
- Push notifications para seguimientos
- Exportación de reportes en PDF
- Dashboard con estadísticas
- Modo oscuro en la interfaz