Manual Técnico

SchoolMeMovil v1.0

Información del Documento

Proyecto: SchoolMeMovil
Versión: 1.0
Autor: Grupo SchoolMe **Ubicación:** Neiva, Huila, Colombia
Fecha: Septiembre 2025
Confidencialidad: Documento interno

Tabla de Contenidos

---

1. Introducción

1.1 Propósito del Documento

Este manual técnico proporciona la documentación completa para desarrolladores, administradores de sistemas y personal técnico responsable de la implementación, mantenimiento y soporte de SchoolMeMovil.

1.2 Alcance

SchoolMeMovil es un aplicativo móvil que digitaliza la agenda escolar, permitiendo a acudientes consultar la agenda de estudio de sus hijos y a todos los

usuarios registrados (docentes, administradores) gestionar su información de perfil.

1.3 Audiencia

Este documento está dirigido a:

- Desarrolladores frontend y backend
- Arquitectos de software
- Administradores de sistemas
- Personal de QA y testing
- Equipos de soporte técnico

1.4 Stack Tecnológico

Tecnología	Versión	Propósito
React Native	Latest	Framework de desarrollo móvil
TypeScript	5.x	Lenguaje de programación tipado
Expo	SDK 49+	Plataforma de desarrollo
React Navigation	6.x	Sistema de navegación
Context API	Native	Gestión de estado global
JWT	-	Autenticación y autorización
AsyncStorage	Latest	Almacenamiento local persistente

---

2. Arquitectura del Sistema

2.1 Arquitectura General

SchoolMeMovil implementa una arquitectura Modular con los siguientes componentes:

Cliente Móvil (React Native) - Interfaz de usuario responsiva - Gestión de estado con Context API - Almacenamiento local con AsyncStorage - Comunicación HTTP con API REST

Servidor Backend (API REST) - Autenticación JWT - Gestión de usuarios y perfiles - Administración de agendas escolares - Control de acceso basado en roles

2.2 Patrón Arquitectónico

El proyecto utiliza una arquitectura modular basada en capas:

```
Presentación (Screens/Modals)
        ↓
Navegación (React Navigation)
```

```
        ↓
Lógica de Negocio (Context/Store)
        ↓
Servicios API (api/services)
        ↓
Backend REST API
```

2.3 Principios de Diseño

- **Separación de responsabilidades:** Cada módulo tiene una función específica
- **Reutilización de componentes:** UI components genéricos y configurables
- **Tipado fuerte:** TypeScript para prevención de errores en tiempo de desarrollo
- **Estado centralizado:** Context API para datos compartidos globalmente
- **Modularidad:** Estructura de carpetas clara y escalable

---

3. Requisitos Técnicos

3.1 Requisitos de Desarrollo

Software Requerido

Componente	Versión Mínima	Propósito
Node.js	18.x o superior	Runtime de JavaScript
npm	9.x o superior	Gestor de paquetes
Git	2.x o superior	Control de versiones
Java JDK	11 o superior	Compilación Android
Android Studio	Última estable	Emulador y herramientas Android
Expo CLI	Global	Herramienta de desarrollo Expo

Sistema Operativo

- **Windows:** 10/11 (64-bit)
- **macOS:** 10.15 o superior
- **Linux:** Ubuntu 20.04+ o equivalente

3.2 Requisitos de Hardware

Mínimos: - Procesador: Intel Core i3 o equivalente - RAM: 8 GB - Almacenamiento: 10 GB disponibles - Conexión a Internet

Recomendados: - Procesador: Intel Core i5/i7 o equivalente - RAM: 16 GB - Almacenamiento: SSD con 20 GB disponibles - Conexión a Internet estable

3.3 Dispositivos Móviles Compatibles

Android: - Versión mínima: Android 6.0 (API 23) - Versión recomendada: Android 10+ (API 29+) - Resoluciones soportadas: 320x480 hasta 1440x3040

iOS (futuro): - iOS 13.0 o superior (preparado para expansión)

---

4. Instalación y Configuración

4.1 Configuración del Entorno de Desarrollo

Paso 1: Instalar Node.js y npm Descargar e instalar desde nodejs.org. Verificar instalación:

```
node --version
npm --version
```

Paso 2: Instalar Expo CLI

```
npm install -g expo-cli
```

Verificar instalación:

```
expo --version
```

Paso 3: Configurar Android Studio

1. Descargar Android Studio desde developer.android.com
2. Instalar Android SDK (API 23 mínimo, API 33 recomendado)
3. Configurar variables de entorno:

Windows:

```
ANDROID_HOME=C:\Users\TuUsuario\AppData\Local\Android\Sdk
Path=%Path%;%ANDROID_HOME%\platform-tools
```

macOS/Linux:

```
export ANDROID_HOME=$HOME/Library/Android/sdk
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

4. Crear emulador Android (AVD) o conectar dispositivo físico

4.2 Clonar e Instalar el Proyecto

Clonar Repositorio

```
git clone https://github.com/JesusCarvajal017/SchoolMeMovil.git
cd SchoolMeMovil
```

Instalar Dependencias

```
npm install
```

Este comando instalará todas las dependencias especificadas en `package.json`.

4.3 Configuración de Variables de Entorno

Opción 1: Archivo de Configuración TypeScript Editar `src/api/constant/Enviroment.ts`:

```typescript
const uri = "IP Local del portatil donde este la api";

export const environment = {
  urlApi: `http://${uri}:5052/api`
};
```

5. Estructura del Proyecto

5.1 Árbol de Directorios

```
SchoolMeMovil/
   .expo/                    # Archivos de configuración Expo
   android/                  # Código nativo Android
   assets/                   # Recursos estáticos (imágenes, fuentes)
   node_modules/             # Dependencias npm
   src/                      # Código fuente principal
       api/                 # Servicios de API
           constant/
               Enviroment.ts
           services/         # Servicios REST
               authService.ts
               userService.ts
               personService.ts
               agendaService.ts
           types/            # Tipos TypeScript para API
               User.ts
               Person.ts
               Agenda.ts
       components/           # Componentes reutilizables
           common/
           forms/
           layout/
       context/              # Contextos de React
           AuthContext.tsx
       navigation/           # Configuración de navegación
           AppNavigator.tsx
           AuthNavigator.tsx
       screens/              # Pantallas principales
```

```
        LoginScreen.tsx
        HomeScreen.tsx
        ProfileScreen.tsx
        EditProfileScreen.tsx
        AgendaScreen.tsx
    modals/               # Diálogos y modales
    util/                # Estado global adicional
App.tsx                   # Punto de entrada principal
Dockerfile                # Contenedor Docker
eas.json                  # Configuración EAS Build
package.json              # Dependencias y scripts
tsconfig.json             # Configuración TypeScript
README.md                 # Documentación básica
```

5.2 Descripción de Módulos Principales

src/api/ Contiene toda la lógica de comunicación con el backend:

- **constant/Enviroment.ts:** Variables de configuración (URLs, timeouts)
- **services/:** Funciones para consumir endpoints REST
- **types/:** Interfaces TypeScript que definen estructuras de datos

src/components/ Componentes de UI reutilizables organizados por categoría:

- **genericos/:** Botones, inputs, etc.
- **Menu/:** Headers, carrusel
- **Narvar/:** Narvar

src/context/ Implementación de Context API para estado global:

- **AuthContext.tsx:** Gestión de autenticación y sesión de usuario

src/navigation/ Configuración de React Navigation:

- **AppNavigator.tsx:** Navegación principal de la app
- **AuthNavigator.tsx:** Flujo de autenticación

src/screens/ Pantallas completas de la aplicación, cada una representa una vista independiente.

---

6. Gestión de Dependencias

6.1 Dependencias de Producción

Instalación de Paquetes Principales

```
# React Native y componentes base
npm install react-native
npm install react-native-safe-area-context
npm install react-native-gesture-handler
npm install react-native-reanimated
npm install react-native-screens
npm install react-native-vector-icons

# Navegación
npm install @react-navigation/native
npm install @react-navigation/native-stack

# Expo
npm install expo
npm install expo-status-bar
npm install expo-linear-gradient
npm install expo-image-picker
npm install expo-file-system

# Autenticación y almacenamiento
npm install jwt-decode
npm install @react-native-async-storage/async-storage

# Validación de formularios
npm install yup
npm install react-hook-form
```

6.2 Dependencias de Desarrollo

```
npm install -D @types/react
npm install -D @types/react-native
npm install -D typescript
npm install -D jest
npm install -D @testing-library/react-native
```

6.3 Herramientas Globales

```
# EAS CLI para builds
npm install -g eas-cli

# Expo CLI
npm install -g expo-cli
```

6.4 Actualización de Dependencias

Para mantener las dependencias actualizadas:

```
# Verificar paquetes desactualizados
npm outdated

# Actualizar paquetes menores
npm update

# Actualizar paquetes mayores (con precaución)
npm install <paquete>@latest
```

Nota: Siempre ejecutar pruebas completas después de actualizar dependencias.

---

7. Autenticación y Seguridad

7.1 Sistema de Autenticación

SchoolMeMovil implementa autenticación basada en JSON Web Tokens (JWT).

Flujo de Autenticación

1. **Login del Usuario:**
   - Usuario ingresa email y contraseña
   - App envía credenciales a `POST /api/auth/login`
   - Backend valida credenciales
2. **Respuesta Exitosa:**

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expiracion": "2025-09-29T10:30:00Z"
}
```

3. **Almacenamiento Seguro:**
   - Token se guarda en AsyncStorage
   - Se programa logout automático al expirar
4. **Obtención de Datos del Usuario:**
   - Se realiza `GET /api/User/{id}` con token
   - Se obtiene `GET /api/Person/data/{personId}`
   - Se recuperan roles con `GET /api/RolUser`
5. **Estado de Sesión:**
   - AuthContext mantiene estado de usuario logueado
   - Token se incluye en header de todas las peticiones

7.2 Gestión de Tokens

Almacenamiento

```
import AsyncStorage from '@react-native-async-storage/async-storage';

// Guardar token
await AsyncStorage.setItem('authToken', token);
await AsyncStorage.setItem('tokenExpiration', expiracion);

// Recuperar token
const token = await AsyncStorage.getItem('authToken');
```

Inclusión en Peticiones

```
const headers = {
  'Content-Type': 'application/json',
  'Authorization': `Bearer ${token}`
};
```

Manejo de Expiración

```
// Programar logout automático
const expirationDate = new Date(expiracion);
const timeout = expirationDate.getTime() - Date.now();

setTimeout(() => {
  logout();
}, timeout);
```

7.3 Sistema de Roles

SchoolMeMovil implementa control de acceso basado en roles (RBAC):

Rol	Permisos
acudiente	Consultar agenda de hijos, ver perfil propio, editar perfil
docente	Cosultar Infomarmacion de perfil, editar perfil
admin	Cosultar Infomarmacion de perfil, editar perfil

Verificación de Roles

```
const hasRole = (requiredRole: string): boolean => {
  return user?.roles?.includes(requiredRole) ?? false;
};

// Uso
```

```
if (hasRole('acudiente')) {
  // Mostrar funcionalidad de acudiente
}
```

7.4 Seguridad de Datos

Mejores Prácticas Implementadas

- **Comunicación HTTPS:** Todas las peticiones usan protocolo seguro
- **No almacenar contraseñas:** Solo tokens en dispositivo
- **Limpieza al logout:** Eliminar todos los datos sensibles de AsyncStorage
- **Validación de entrada:** Uso de Yup para validar formularios
- **Timeout de sesión:** Expiración automática de tokens
- **Manejo seguro de errores:** No exponer detalles técnicos al usuario

---

8. Especificación de API REST

8.1 URL Base

`http://localhost:5052/swagger/index.html`

8.2 Formato de Respuestas

Todas las respuestas siguen el formato JSON estándar.

Respuesta Exitosa:

```
{
  "status": 200,
  "data": { ... }
}
```

Respuesta de Error:

```
{
  "status": 400,
  "error": "Mensaje de error",
  "details": [ ... ]
}
```

8.3 Autenticación (Auth)

POST /api/auth/login Autentica usuario y devuelve token JWT.

Request:

```
{
  "email": "usuario@example.com",
```

```
  "password": "contraseña123"
}
```

Response (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "expiracion": "2025-09-29T10:30:00Z"
}
```

Errores: - `401 Unauthorized`: Credenciales inválidas - `400 Bad Request`: Formato de datos incorrecto

8.4 Endpoints de Usuario

GET /api/User Listar todos los usuarios.

Headers:

`Authorization: Bearer {token}`

Response (200):

```
[
  {
    "id": 1,
    "email": "usuario@example.com",
    "personId": 10,
    "status": "active"
  }
]
```

GET /api/User/{id} Obtener usuario específico por ID.

Headers:

`Authorization: Bearer {token}`

Response (200):

```
{
  "id": 1,
  "email": "usuario@example.com",
  "personId": 10,
  "roles": ["estudiante"],
  "photo": "https://api.schoolme.com/uploads/photo.jpg"
}
```

POST /api/User Crear nuevo usuario.

Headers:

`Authorization: Bearer {token}`

Request:

```
{
  "email": "nuevo@example.com",
  "password": "password123",
  "personId": 15
}
```

Response (201):

```
{
  "id": 25,
  "email": "nuevo@example.com",
  "personId": 15
}
```

PUT /api/User Actualizar usuario completo.

Headers:

`Authorization: Bearer {token}`

Request:

```
{
  "id": 1,
  "email": "actualizado@example.com",
  "personId": 10
}
```

Response (200):

```
{
  "id": 1,
  "email": "actualizado@example.com",
  "personId": 10
}
```

PATCH /api/User Actualización parcial de usuario.

Headers:

`Authorization: Bearer {token}`

Request:

```
{
  "id": 1,
  "email": "nuevo_email@example.com"
}
```

POST /api/User/passwordUpdate Actualizar contraseña de usuario.

Headers:

```
Authorization: Bearer {token}
```

Request:

```
{
  "userId": 1,
  "oldPassword": "password123",
  "newPassword": "newpassword456"
}
```

Response (200):

```
{
  "message": "Contraseña actualizada exitosamente"
}
```

POST /api/User/photoUpdate Actualizar foto de perfil.

Headers:

```
Authorization: Bearer {token}
Content-Type: multipart/form-data
```

Request:

```
FormData:
- userId: 1
- photo: [archivo de imagen]
```

Response (200):

```
{
  "photoUrl": "http://localhost:5052/swagger/user/uploads/user_1_photo.jpg"
}
```

8.5 Endpoints de Persona

GET /api/Person/data/{personId} Obtener datos completos de una persona.

Headers:

```
Authorization: Bearer {token}
```

Response (200):

```
{
  "id": 10,
  "firstName": "Juan",
  "lastName": "Pérez",
```

```
  "identificationType": "CC",
  "identificationNumber": "12345678",
  "birthDate": "1990-05-15",
  "phone": "+57 300 123 4567",
  "address": "Calle 10 #5-20",
  "city": "Neiva",
  "department": "Huila"
}
```

GET /api/Person/PersonBasic/{personId} Obtener datos básicos de una persona.

Headers:

`Authorization: Bearer {token}`

Response (200):

```
{
  "id": 10,
  "firstName": "Juan",
  "lastName": "Pérez",
  "phone": "+57 300 123 4567"
}
```

8.6 Endpoints de Agenda de Estudiante

GET /api/AgendaDayStudent Listar todas las entradas de agenda.

Headers:

`Authorization: Bearer {token}`

Query Parameters: - `studentId` (opcional): Filtrar por estudiante - `date` (opcional): Filtrar por fecha (formato: YYYY-MM-DD)

Response (200):

```
[
  {
    "id": 1,
    "studentId": 5,
    "date": "2025-09-28",
    "subject": "Matemáticas",
    "description": "Tarea páginas 45-50",
    "status": "active",
    "createdBy": 3,
    "createdAt": "2025-09-27T14:30:00Z"
  }
]
```

GET /api/AgendaDayStudent/{id} Obtener entrada específica de agenda.

Headers:

```
Authorization: Bearer {token}
```

Response (200):


```
{
  "id": 1,
  "studentId": 5,
  "date": "2025-09-28",
  "subject": "Matemáticas",
  "description": "Tarea páginas 45-50",
  "status": "active",
  "createdBy": 3,
  "createdAt": "2025-09-27T14:30:00Z",
  "updatedAt": "2025-09-27T14:30:00Z"
}
```


POST /api/AgendaDayStudent Crear nueva entrada de agenda.

Headers:

```
Authorization: Bearer {token}
```

Request:


```
{
  "studentId": 5,
  "date": "2025-09-29",
  "subject": "Ciencias",
  "description": "Estudiar sistema solar",
  "status": "active"
}
```


Response (201):


```
{
  "id": 15,
  "studentId": 5,
  "date": "2025-09-29",
  "subject": "Ciencias",
  "description": "Estudiar sistema solar",
  "status": "active"
}
```


PUT /api/AgendaDayStudent Actualizar entrada completa.

Headers:

`Authorization: Bearer {token}`

Request:

```
{
  "id": 1,
  "studentId": 5,
  "date": "2025-09-28",
  "subject": "Matemáticas",
  "description": "Tarea páginas 45-55 (actualizado)",
  "status": "active"
}
```

PATCH /api/AgendaDayStudent Actualización parcial de entrada.

Headers:

`Authorization: Bearer {token}`

Request:

```
{
  "id": 1,
  "description": "Tarea páginas 45-55 y ejercicios adicionales"
}
```

Cambiar estado de entrada.

Headers:

`Authorization: Bearer {token}`

Response (200):

```
{
  "id": 1,
  "status": "inactive"
}
```

8.7 Endpoints de Roles

GET /api/RolUser Obtener roles del usuario autenticado.

Headers:

`Authorization: Bearer {token}`

Response (200):

```
{
  "userId": 1,
  "roles": ["estudiante", "acudiente"]
}
```

GET /api/RolUser/{userId} Obtener roles de un usuario específico.

Headers:

```
Authorization: Bearer {token}
```

Response (200):

```
{
  "userId": 5,
  "roles": ["docente"]
}
```

8.8 Códigos de Estado HTTP

Código	Significado	Descripción
200	OK	Solicitud exitosa
201	Created	Recurso creado exitosamente
400	Bad Request	Datos de entrada inválidos
401	Unauthorized	Token inválido o expirado
403	Forbidden	Sin permisos para el recurso
404	Not Found	Recurso no encontrado
500	Internal Server Error	Error del servidor

---

9. Modelos de Datos

9.1 Modelo User

Definición TypeScript:

```
export interface User {
  id: number;
  email: string;
  personId?: number;
  roles: string[];
  photo?: string;
  status: 'active' | 'inactive';
  createdAt?: string;
  updatedAt?: string;
}
```

Descripción de Campos:

Campo	Tipo	Requerido	Descripción
token	string	Sí	Token JWT de autenticación

Campo	Tipo	Requerido	Descripción
expiracion	string	Sí	Fecha de expiración (ISO 8601)

---

10. Flujos de Negocio

10.1 Flujo de Autenticación

```
Usuario

     1. Ingresa credenciales


LoginScreen

       2. authService.login()


Backend API

       3. Valida y retorna token


AuthContext

       4. Guarda token en AsyncStorage
       5. Obtiene datos de usuario
       6. Obtiene datos de persona
       7. Obtiene roles


 HomeScreen
```

Pasos Detallados:

1. Usuario ingresa email y contraseña en LoginScreen
2. Se invoca `authService.login(email, password)`
3. Backend valida credenciales y retorna token + expiración
4. AuthContext guarda token en AsyncStorage
5. Se ejecuta `userService.getUserById(userId)` con token
6. Se ejecuta `personService.getPersonData(personId)` con token
7. Se ejecuta `rolUserService.getUserRoles()` para obtener permisos
8. AuthContext actualiza estado global con datos completos

9. Navegación redirige a HomeScreen
10. Se programa logout automático según expiración del token

10.2 Flujo de Consulta de Agenda (Acudiente)

```
Acudiente

      1. Accede a "Mi Agenda"


 AgendaScreen

        2. Verifica rol "acudiente"


 AuthContext

        3. Obtiene lista de hijos


Backend API

        4. GET /api/AgendaDayStudent?studentId={hijo}


 AgendaScreen

        5. Muestra listado de tareas por fecha


Acudiente
```

10.3 Flujo de Edición de Perfil

```
 Usuario

      1. Selecciona "Editar Perfil"


EditProfileScreen

      2. Carga datos actuales
      3. Usuario modifica campos
```

```
     4. Valida con Yup

Validación

       5. Si válido

userService

       6. PATCH /api/User

Backend API

       7. Actualiza BD
       8. Retorna usuario actualizado

AuthContext

       9. Actualiza estado global

ProfileScreen
```

10.4 Flujo de Logout

```
Usuario

     1. Presiona "Cerrar Sesión"

AuthContext

       2. clearTimeout(autoLogout)
       3. AsyncStorage.clear()
       4. setUser(null)
       5. setToken(null)

LoginScreen
```

11. Pruebas y Validación

11.1 Estrategia de Testing

SchoolMeMovil implementa una estrategia de testing en tres niveles:

Pruebas Unitarias **Objetivo:** Validar funciones y componentes individuales

Herramientas: - Jest (framework de testing) - React Native Testing Library

Cobertura esperada: 80%

Áreas críticas: - Servicios de API - Funciones de validación - Context API (AuthContext) - Utilidades y helpers

Pruebas de Integración **Objetivo:** Validar interacción entre módulos

Casos de prueba: - Flujo completo de login - Navegación entre pantallas - Actualización de estado global - Persistencia de datos

Pruebas End-to-End (E2E) **Objetivo:** Validar flujos completos desde perspectiva del usuario

Herramientas opcionales: - Detox - Appium

Flujos críticos: - Login → Consulta de agenda → Logout - Edición de perfil completa - Manejo de errores de red

11.2 Configuración de Jest

jest.config.js:

```
module.exports = {
  preset: 'react-native',
  setupFilesAfterEnv: ['@testing-library/jest-native/extend-expect'],
  transformIgnorePatterns: [
    'node_modules/(?!(react-native|@react-native|expo|@expo|@react-navigation)/)',
  ],
  collectCoverageFrom: [
    'src/**/*.{ts,tsx}',
    '!src/**/*.d.ts',
    '!src/**/index.ts',
  ],
  coverageThreshold: {
    global: {
      branches: 80,
      functions: 80,
```

```
      lines: 80,
      statements: 80,
    },
  },
};
```

11.3 Ejecución de Pruebas

```
# Ejecutar todas las pruebas
npm test

# Ejecutar con cobertura
npm test -- --coverage

# Modo watch (desarrollo)
npm test -- --watch

# Pruebas específicas
npm test AuthContext.test.tsx
```

11.4 Ejemplo de Prueba Unitaria

authService.test.ts:

```
import { authService } from '../api/services/authService';

describe('authService', () => {
  it('debe retornar token al hacer login exitoso', async () => {
    const response = await authService.login(
      'test@example.com',
      'password123'
    );

    expect(response).toHaveProperty('token');
    expect(response).toHaveProperty('expiracion');
    expect(typeof response.token).toBe('string');
  });

  it('debe lanzar error con credenciales inválidas', async () => {
    await expect(
      authService.login('invalid@example.com', 'wrongpass')
    ).rejects.toThrow('Credenciales inválidas');
  });
});
```

11.5 Checklist de Validación

Antes de cada release, verificar:

- Login con credenciales válidas funciona
- Login con credenciales inválidas muestra error apropiado
- Perfil muestra datos dinámicos correctos (user, person, photo, roles)
- Navegación a "Editar perfil" funciona sin errores
- Validaciones de formularios activas y funcionando
- Actualización de perfil persiste cambios
- Logout limpia contexto y AsyncStorage completamente
- Logout redirige a LoginScreen
- Token expirado ejecuta logout automático
- Agenda carga datos correctamente para acudientes
- Roles restringen acceso apropiadamente
- App funciona sin conexión (funcionalidad offline básica)
- Imágenes de perfil se cargan y actualizan correctamente
- Manejo de errores muestra mensajes claros al usuario

---

12. Compilación y Despliegue

12.1 Entornos de Despliegue

SchoolMeMovil maneja tres entornos:

Entorno	Propósito	API URL
Desarrollo	Desarrollo local y pruebas	http://localhost:5052/api EXPO exp://10.3.234.106:8081
Staging	Pruebas pre-producción	http://IP_LOCAL_API:5052/api

12.2 Ejecución en Modo Desarrollo

Inicio rápido

```
# Iniciar servidor de desarrollo
expo start
```

Opciones de ejecución

```
# Abrir en emulador Android
npm run android

# Abrir en dispositivo físico (Expo Go)
npm start
# Escanear QR con Expo Go app
```

```
# Limpiar caché y reiniciar
npm start -c
```

12.3 Compilación APK de Pruebas

Instalación de EAS CLI

```
npm install -g eas-cli
```

Configuración de EAS eas.json:

```
{
  "build": {
    "development": {
      "developmentClient": true,
      "distribution": "internal"
    },
    "preview": {
      "android": {
        "buildType": "apk"
      },
      "distribution": "internal"
    },
    "production": {
      "android": {
        "buildType": "app-bundle"
      }
    }
  }
}
```

Generar APK

```
# Login en Expo
eas login

# Configurar proyecto (primera vez)
eas build:configure

# Generar APK de pruebas
eas build --platform android --profile preview

# Seguir progreso
# URL de descarga aparecerá al completar
```

Nota: El build puede tardar 10-20 minutos. Se recibirá un enlace de descarga al finalizar.

12.4 Generación de App Bundle (Play Store)

Para publicación en Google Play Store:

```
eas build --platform android --profile production
```

Esto genera un archivo `.aab` (Android App Bundle) optimizado para distribución.

12.5 Instalación de APK en Dispositivo

Método 1: Descarga directa

1. Descargar APK desde el enlace proporcionado por EAS
2. Transferir a dispositivo Android
3. Habilitar "Fuentes desconocidas" en configuración
4. Abrir archivo APK y seguir instrucciones

Método 2: ADB

```
# Conectar dispositivo por USB
adb devices

# Instalar APK
adb install schoolme-v1.0.apk
```

12.6 CI/CD Pipeline

Configuración Recomendada (GitHub Actions) .github/workflows/build.yml:

```yaml
name: Build APK

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
```

```yaml
    - name: Install dependencies
      run: npm install

    - name: Run tests
      run: npm test

    - name: Setup Expo
      uses: expo/expo-github-action@v8
      with:
        expo-version: latest
        token: ${{ secrets.EXPO_TOKEN }}

    - name: Build APK
      run: eas build --platform android --profile preview --non-interactive

    - name: Upload artifact
      uses: actions/upload-artifact@v3
      with:
        name: schoolme-apk
        path: '*.apk'
```

12.7 Versionamiento

SchoolMeMovil sigue Semantic Versioning (MAJOR.MINOR.PATCH):

app.json:

```json
{
  "expo": {
    "version": "1.0.0",
    "android": {
      "versionCode": 1
    }
  }
}
```

Incremento de versiones: - **MAJOR (1.x.x):** Cambios incompatibles con versiones anteriores - **MINOR (x.1.x):** Nueva funcionalidad compatible - **PATCH (x.x.1):** Corrección de bugs

---

13. Resolución de Problemas

13.1 Errores Comunes y Soluciones

Error: “StyleSheet doesn’t exist” Síntoma:

`Module '"react-native"' has no exported member 'StyleSheet'`

Solución:

```
// Importar correctamente
import { StyleSheet, View, Text } from 'react-native';
```

Error: Login retorna datos inválidos **Síntoma:** Login parece exitoso pero no carga datos de usuario

Solución: 1. Verificar estructura de respuesta del backend 2. Debe retornar: `{ token, expiracion }` 3. Verificar que endpoints de User y Person estén accesibles 4. Revisar logs de red en DevTools

Verificación:

```
console.log('Response:', JSON.stringify(response, null, 2));
```

Error: TS2724 en tipos **Síntoma:**

`error TS2724: Module '"../types/User"' has no exported member 'User'`

Solución: 1. Verificar que el tipo esté correctamente exportado:

```
// types/User.ts
export interface User {
  // ...
}
```

2. Verificar ruta de importación
3. Reiniciar TypeScript server en VS Code

Error: APK no se genera **Síntoma:** `eas build` falla o se cuelga

Soluciones: 1. Verificar configuración en `eas.json` 2. Asegurar que `app.json` esté completo 3. Verificar conexión a internet 4. Revisar logs: `eas build:list` 5. Limpiar caché: `expo start -c`

Error: Token expirado **Síntoma:** Peticiones fallan con 401 después de cierto tiempo

Solución: Verificar que logout automático esté configurado:

```
const setupAutoLogout = (expiracion: string) => {
  const expirationDate = new Date(expiracion);
  const timeout = expirationDate.getTime() - Date.now();

  if (timeout > 0) {
    setTimeout(() => {
      logout();
    }, timeout);
```

```
  }
};
```

Error: AsyncStorage no persiste datos **Síntoma:** Datos se pierden al reiniciar app

Solución: 1. Verificar await en operaciones:

```
await AsyncStorage.setItem('key', 'value');
```

2. Verificar que no se esté limpiando en lugares no deseados
3. Probar con Storage Debugger de React Native Debugger

13.2 Debugging

Activar Debug Mode

```
# Abrir menu de desarrollo en emulador
# Android: Ctrl + M (Windows/Linux) o Cmd + M (Mac)
# Dispositivo físico: Agitar dispositivo

# Opciones útiles:
# - Enable Remote JS Debugging
# - Show Performance Monitor
# - Toggle Inspector
```

React Native Debugger Herramienta recomendada para debugging avanzado:

```
# Instalar
npm install -g react-native-debugger

# Ejecutar
react-native-debugger
```

Logs en Consola

```
// Logs estructurados
console.log('Info:', data);
console.warn('Warning:', warning);
console.error('Error:', error);

// Logs condicionales (solo en desarrollo)
if (__DEV__) {
  console.log('Development only log');
}
```

Network Inspector Ver peticiones HTTP en tiempo real: 1. Abrir Chrome DevTools 2. Ir a Network tab 3. Filtrar por XHR

13.3 Limpieza de Caché

Si experimenta comportamientos extraños:

```
# Limpiar caché de Expo
expo start -c

# Limpiar node_modules
rm -rf node_modules
npm install

# Limpiar caché de npm
npm cache clean --force

# Android: Limpiar build
cd android
./gradlew clean
cd ..
```

13.4 Recursos de Soporte

Documentación oficial: - React Native: https://reactnative.dev/docs - Expo: https://docs.expo.dev - React Navigation: https://reactnavigation.org/docs

Comunidad: - Stack Overflow: Tag `react-native` - GitHub Issues del proyecto - Expo Forums: https://forums.expo.dev

---

14. Mantenimiento y Monitoreo

14.1 Estrategia de Logging

Niveles de Log SchoolMeMovil implementa logging estructurado:

```
enum LogLevel {
  DEBUG = 'DEBUG',
  INFO = 'INFO',
  WARN = 'WARN',
  ERROR = 'ERROR'
}

const log = (level: LogLevel, message: string, data?: any) => {
  if (__DEV__ || level === LogLevel.ERROR) {
    console.log(`[${level}] ${new Date().toISOString()} - ${message}`, data);
```

```
  }
};
```

Logs Recomendados Autenticación:

```
log(LogLevel.INFO, 'Login attempt', { email });
log(LogLevel.INFO, 'Login successful', { userId });
log(LogLevel.ERROR, 'Login failed', { error });
```

Peticiones API:

```
log(LogLevel.DEBUG, 'API Request', { endpoint, method });
log(LogLevel.ERROR, 'API Error', { endpoint, status, error });
```

14.2 Manejo de Errores en Producción

Error Boundaries Implementar React Error Boundary para capturar errores:

```
class ErrorBoundary extends React.Component {
  state = { hasError: false };

  static getDerivedStateFromError(error) {
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    log(LogLevel.ERROR, 'Unhandled error', { error, errorInfo });
  }

  render() {
    if (this.state.hasError) {
      return <ErrorScreen />;
    }
    return this.props.children;
  }
}
```

Alertas al Usuario Usar `Alert.alert` para errores críticos:

```
const handleError = (error: Error) => {
  Alert.alert(
    'Error',
    'Ocurrió un problema. Por favor intenta nuevamente.',
    [{ text: 'OK' }]
  );
```

```
  log(LogLevel.ERROR, 'User-facing error', { error });
};
```

14.3 Monitoreo (Opcional)

Para producción, considerar integrar:

Sentry:

```
npm install @sentry/react-native
```

```
import * as Sentry from '@sentry/react-native';

Sentry.init({
  dsn: 'YOUR_SENTRY_DSN',
  environment: 'production',
});
```

14.4 Mantenimiento Preventivo

Revisión Mensual

- Actualizar dependencias con vulnerabilidades
- Revisar logs de errores recurrentes
- Verificar uso de almacenamiento
- Validar tamaño de APK (< 50MB recomendado)

Revisión Trimestral

- Actualizar versiones mayores de dependencias
- Revisar y optimizar rendimiento
- Actualizar documentación técnica
- Auditoría de seguridad

14.5 Plan de Rollback

En caso de bugs críticos en producción:

1. **Identificación:** Detectar problema mediante reportes o logs
2. **Evaluación:** Determinar severidad e impacto
3. **Rollback:** Distribuir versión anterior estable
4. **Corrección:** Desarrollar fix en rama separada
5. **Testing:** Validar exhaustivamente la corrección
6. **Re-despliegue:** Publicar nueva versión corregida

APK de respaldo: Mantener últimas 3 versiones estables disponibles para rollback rápido.

---

15. Referencias y Contacto

15.1 Glosario de Términos

Término	Definición
APK	Android Package Kit - formato de instalación Android
AsyncStorage	Sistema de almacenamiento persistente key-value
Context API	Mecanismo de React para estado global
EAS	Expo Application Services - servicio de build
JWT	JSON Web Token - estándar de autenticación
RBAC	Role-Based Access Control - control de acceso por roles
REST	Representational State Transfer - arquitectura de API
TypeScript	Superset tipado de JavaScript

15.2 Enlaces de Referencia

Documentación Técnica: - React Native Docs: https://reactnative.dev - Expo Documentation: https://docs.expo.dev - TypeScript Handbook: https://www.typescriptlang.org/docs - React Navigation: https://reactnavigation.org - JWT.io: https://jwt.io

Repositorios: - Proyecto GitHub: https://github.com/tu-usuario/SchoolMeMovil - Issues: https://github.com/tu-usuario/SchoolMeMovil/issues

15.3 Información de Contacto

Desarrollador Principal: - Nombre: Santiago Chaparro Riaño - Ubicación: Neiva, Huila, Colombia - Email: [Tu email de contacto] - GitHub: [Tu perfil de GitHub]

Soporte Técnico: - Email: soporte@schoolme.com - Horario: Lunes a Viernes, 8:00 AM - 6:00 PM (COT)

15.4 Historial de Versiones

Versión	Fecha	Cambios Principales
1.0.0	Sep 2025	Release inicial - Login, Perfil, Agenda básica

15.5 Licencia

[Especificar tipo de licencia - MIT, Apache, Propietaria, etc.]

Apéndices

Apéndice A: Comandos Útiles de Consola

```
# Gestión de proyecto
npm install                  # Instalar dependencias
npm update                   # Actualizar dependencias
npm audit fix               # Corregir vulnerabilidades

# Desarrollo
expo start                   # Iniciar servidor desarrollo
expo start -c               # Limpiar caché e iniciar
expo start --android        # Abrir en emulador Android

# Testing
npm test                     # Ejecutar pruebas
npm test -- --coverage      # Con cobertura
npm test -- --watch         # Modo watch

# Build
eas build --platform android --profile preview    # APK pruebas
eas build --platform android --profile production # App Bundle

# Utilidades
npm run lint                # Verificar estilo de código
npm run format              # Formatear código
```

Apéndice B: Estructura de Proyecto Ideal

```
src/
  api/
      constant/
      services/
      types/
  components/
      common/          # Botones, inputs, cards
      forms/           # Componentes de formulario
      layout/          # Headers, containers
  context/
      AuthContext.tsx  # Estado de autenticación
  navigation/
      AppNavigator.tsx
      AuthNavigator.tsx
  screens/
      auth/
          LoginScreen.tsx
      home/
```

```
        HomeScreen.tsx
    profile/
        ProfileScreen.tsx
        EditProfileScreen.tsx
    agenda/
        AgendaScreen.tsx
modals/
store/
utils/              # Funciones helper
hooks/              # Custom hooks
styles/             # Estilos globales
```

Fin del Manual Técnico - SchoolMeMovil v1.0

Este documento es propiedad de Santiago Chaparro Riaño y está sujeto a actualizaciones continuas. Última revisión: Septiembre 2025.

Descripción			
id	number	Sí	Identificador único del usuario
email	string	Sí	Correo electrónico (único)
personId	number	No	Referencia a datos de persona
roles	string[]	Sí	Lista de roles asignados
photo	string	No	URL de foto de perfil
status	string	Sí	Estado: 'active' o 'inactive'
createdAt	string	No	Fecha de creación (ISO 8601)
updatedAt	string	No	Fecha de última actualización

9.2 Modelo Person

Definición TypeScript:

```typescript
export interface Person {
  id: number;
  firstName: string;
  lastName: string;
  identificationType?: string;
  identificationNumber?: string;
  birthDate?: string;
  phone?: string;
  address?: string;
  city?: string;
  department?: string;
  createdAt?: string;
  updatedAt?: string;
}
```

Descripción de Campos:

Campo	Tipo	Requerido	Descripción
id	number	Sí	Identificador único de persona
firstName	string	Sí	Nombre(s)
lastName	string	Sí	Apellido(s)
identificationType	string	No	Tipo de documento (CC, TI, CE)
identificationNumber	string	No	Número de documento
birthDate	string	No	Fecha de nacimiento (ISO 8601)
phone	string	No	Número telefónico
address	string	No	Dirección residencial
city	string	No	Ciudad
department	string	No	Departamento

9.3 Modelo AgendaDayStudent

Definición TypeScript:

```typescript
export interface AgendaDayStudent {
  id: number;
  studentId: number;
  date: string;
  subject: string;
  description?: string;
  status: 'active' | 'inactive';
  createdBy?: number;
  createdAt?: string;
```

```
  updatedAt?: string;
}
```

Descripción de Campos:

Campo	Tipo	Requerido	Descripción
id	number	Sí	Identificador único de entrada
studentId	number	Sí	ID del estudiante
date	string	Sí	Fecha de la actividad (YYYY-MM-DD)
subject	string	Sí	Materia o asignatura
description	string	No	Descripción detallada de la tarea
status	string	Sí	Estado: 'active' o 'inactive'
createdBy	number	No	ID del docente que creó la entrada
createdAt	string	No	Fecha de creación
updatedAt	string	No	Fecha de última actualización

9.4 Modelo AuthResponse

Definición TypeScript:

```
export interface AuthResponse {
  token: string;
  expiracion: string;
}
```