Ficha Técnica del Proyecto Móvil

Área de Desarrollo Móvil
Centro De Electricidad Y Automatización Industrial, SENA
3067594: Análisis y Desarrollo de Software
Cómite Evaluador
1 de Junio del 2026.

Cali, Valle del Cauca

Tabla de contenido

Introducción

La presente ficha técnica tiene como propósito documentar de manera integral la estructura del proyecto TAM en el área de desarrollo móvil, proporcionando una visión clara de sus características funcionales, técnicas y arquitectónicas. Este documento sirve como referencia para comprender el funcionamiento de la aplicación, los recursos tecnológicos utilizados y los lineamientos que garantizan su correcto desarrollo e implementación.

En primer lugar, se aborda la descripción funcional del sistema, detallando los procesos, servicios y funcionalidades que ofrece la aplicación móvil a los usuarios. Esto permite identificar el alcance del proyecto y la forma en que cada componente contribuye al cumplimiento de los objetivos planteados por la organización.

Asimismo, se describen los requisitos técnicos necesarios para la operación del sistema, incluyendo los lenguajes de programación, las bases de datos, las herramientas de desarrollo y la infraestructura tecnológica empleada. De igual manera, se presenta la arquitectura de software seleccionada, explicando la organización de los componentes y la comunicación entre ellos para garantizar eficiencia, mantenibilidad y rendimiento.

Finalmente, la ficha técnica contempla las integraciones con APIs y sistemas externos, así como los mecanismos de seguridad y escalabilidad implementados. Estos aspectos son fundamentales para asegurar la protección de la información, la interoperabilidad con otros servicios y la capacidad del sistema para adaptarse al crecimiento futuro y a nuevas necesidades del negocio.

1. **Descripción funcional del sistema**
   La aplicación móvil TAM es una plataforma de comercio electrónico desarrollada para dispositivos Android, que permite a los usuarios explorar, comparar y gestionar productos tecnológicos. Las funcionalidades implementadas incluyen:

   - **Autenticación:** registro de usuarios con verificación de cuenta vía correo electrónico mediante código de 6 dígitos con expiración de 3 minutos (implementado con Nodemailer y SMTP en el backend Node.js), inicio de sesión con control de intentos fallidos (máximo 3 intentos) y cierre de sesión. El sistema incluye limpieza automática de cuentas no verificadas cuyo código ha expirado, evitando correos obsoletos en la base de datos.
   - **Catálogo de productos:** visualización de productos por categorías y subcategorías cargadas dinámicamente desde la API.
   - **Carrito de compras:** gestión de productos (agregar, modificar cantidad, eliminar), resumen de compra con subtotal y costo de envío, persistencia por usuario mediante AsyncStorage (con clave única por usuario_id), fusión de carrito de invitado al iniciar sesión, y validación de stock real antes de confirmar compra.
   - **Comparador de productos:** comparación de celulares, portátiles y consolas con análisis de rendimiento por características técnicas.
   - **Perfil de usuario:** visualización y edición de datos personales (nombre completo, nombre de usuario con validación de unicidad, foto de perfil mediante expo-image-picker y almacenamiento en servidor, y cambio de contraseña con verificación de contraseña actual vía bcrypt).
   - **Menú de navegación (Drawer):** acceso a categorías y subcategorías cargadas dinámicamente desde la API. Permite cerrar sesión cuando un usuario está logueado.

2. **Requisitos técnicos (Lenguajes, BD y infraestructura)**

   2.1 Área de desarrollo Frontend

   1. **Lenguajes y frameworks:**

   - **TypeScript versión 5.9.3** como lenguaje principal.
   - **React Native versión 0.81.5** con Expo como framework de desarrollo móvil.
   - **Expo SDK versión 54**
   - **Expo Router versión 6.0.22** para la navegación basada en archivos.
   - **NativeWind versión 4.2.2** (Tailwind CSS para React Native) para estilos.

   2. **Librerías principales:**

   - **expo-linear-gradient** → gradientes visuales.
   - **expo-font** → fuentes personalizadas (Poppins).

- **expo-splash-screen** → pantalla de carga inicial.
- **@react-native-async-storage/async-storage** → persistencia local del carrito.
- **@react-native-community/datetimepicker** → selector de fecha de nacimiento.
- **@expo/vector-icons** → iconografía (Ionicons).
- **Expo Image Picker** → selección y carga de foto de perfil desde galería del dispositivo
- **React Context API** → manejo del estado global (AuthContext, CartContext, CompareContext).

2.2 Área de desarrollo Backend

3. **Base de Datos:**

- **Motor de Base de Datos: MariaDB** 10.4.32, administrado mediante **phpMyAdmin 5.2.1** e integrado en **XAMPP 8.2.12**.
- **Herramientas de desarrollo:** phpMyAdmin, XAMPP 8.2.12.

4. **Lenguajes:**
- **Typescript 6.0.2:** como lenguaje principal del backend.

5. **Framework:**
- **Express 5.2.1**: framework para la construcción de la API REST.

6. **Librerías principales**
- **mysql2 3.20.0:** conexión y consultas a la base de datos MySQL/MariaDB con queries parametrizados.
- **Nodemailer 9.0.1:** envío de correos electrónicos con código de verificación vía SMTP (Gmail).
- **bcrypt 6.0.0 :** cifrado y verificación de contraseñas.
- **multer 2.2.0 :** procesamiento y almacenamiento de archivos (fotos de perfil).

3. Arquitectura (monolito, microservicios)

La aplicación móvil sigue una arquitectura **cliente-servidor** desacoplada:

- **Frontend móvil (React Native/Expo):** consume servicios REST, maneja estado local con React Context API y persistencia con AsyncStorage. La sesión del usuario persiste en AsyncStorage con restauración automática al reiniciar la app.
- **Backend Node.js (TAM_API):** expone los servicios REST para la gestión de productos, usuarios, autenticación, categorías, carrito de compras, perfil de usuario, subida de archivos (multer) y envío de correos electrónicos de verificación mediante Nodemailer utilizando el protocolo SMTP. Los archivos

estáticos (fotos de perfil) se sirven directamente desde la carpeta uploads/ del servidor.

La **navegación** está organizada en dos grupos principales:

- (auth) → pantallas de autenticación (login, registro, verificación)
- (stack) → pantallas protegidas (home, productos, carrito, perfil, comparador)

El **estado global** se maneja mediante tres contextos:

- **AuthContext** → sesión del usuario (persistida en AsyncStorage, se restaura automáticamente al abrir la app).
- **CartContext** → estado del carrito con sincronización a BD, persistencia por usuario, y fusión de carrito de invitado al iniciar sesión
- **CompareContext** → productos seleccionados para comparar

3.1 Versiones del software

Software/Herramienta	Versión
Node.js	24.13.0
npm	11.6.2
TypeScript (Frontend)	5.9.3
TypeScript (Backend)	6.0.2
React Native	0.81.5
Expo SDK	54.0.32
Expo Go (Android)	54.0.6
Expo Go (Windows)	54.0.8
Expo Router	6.0.22
NativeWind	4.2.2
Express	5.2.1
mysql2	3.20.0
Nodemailer	9.0.1
bcrypt	6.0.0
Multer	2.2.0

MariaDB / MySQL	10.4.32
XAMPP	8.2.12
Apache	2.4.58
PHP	8.2.12
phpMyAdmin	5.2.1
Visual Studio Code	1.199.0
Android Studio	2025.1.2 (Runtime 21.0.8)

3.2 Dispositivos compatibles y requisitos mínimos

La aplicación fue desarrollada y probada principalmente en dispositivos Android. A continuación se detallan las características del dispositivo de prueba y los requisitos mínimos recomendados:

Característica	Detalle
Sistema operativo	Android
Versión Android mínima	mínimo Android 6.0 — requerido por Expo SDK 54
Versión Android recomendada	Android 12 o superior
Versión iOS mínima	iOS 16 o superior (compatible con Expo SDK 54, no probado)
Tamaño de pantalla	Compatible con pantallas de 5.0" a 7" (diseño responsive con NativeWind)
Resolución recomendada	1080 × 1920 px o superior
Densidad	480 dpi (densidad de píxeles, excelente nitidez)
Capacidad de almacenamiento	100 MB libres (para instalación de la app + datos de AsyncStorage + fotos de perfil guardadas localmente)
Memoria RAM	Mínimo 2 GB RAM (recomendado 4 GB o más)

Procesador	ARM64 (64 bits) — compatible con procesadores Snapdragon, MediaTek, Exynos y equivalentes
CPU cores	Mínimo 4
Conexión a red	WiFi o datos móviles activos (la app consume APIs en red local durante desarrollo)

4. **Integraciones (APIs, sistemas externos)**

   La comunicación entre el frontend y la API en TypeScript se estructura bajo el modelo **Cliente-Servidor** mediante los siguientes pilares:

- **Protocolo REST y JSON:** El frontend realiza peticiones HTTP (GET, POST, PUT y DELETE) utilizando Fetch API hacia la API desarrollada en Node.js y Express. El intercambio de información se realiza en formato JSON, garantizando una comunicación eficiente entre cliente y servidor.
- **Contratos de Datos (TypeScript):** Las interfaces y modelos utilizados en el backend se implementan también en el frontend para mantener consistencia en la estructura de los datos y reducir errores durante el desarrollo.
- **Servicio de correo electrónico:** El sistema utiliza la librería Nodemailer integrada en el backend para realizar el envío automático de códigos de verificación mediante el protocolo SMTP (Gmail) durante el proceso de registro y recuperación de cuentas. El flujo incluye: generación del código, almacenamiento en BD (campo token_verificacion en formato "codigo|fechaExpiracion"), envío por correo, y limpieza automática de cuentas no verificadas con código expirado.
- **Carga de imágenes:** La actualización de fotografías de perfil se realiza mediante Expo Image Picker en el frontend y Multer en el backend para el procesamiento y almacenamiento seguro de archivos. Las imágenes sirven como archivos estáticos desde la ruta /uploads/ del servidor TAM_API.
- **Control de errores mediante estados HTTP:** El frontend interpreta las respuestas emitidas por la API utilizando códigos estándar como:

  - 200 y 201: operaciones exitosas.
  - 400: errores de validación.
  - 401: autenticación inválida.
  - 404: recursos no encontrados.
  - 409: conflictos (correo duplicado).
  - 410: códigos expirados.
  - 500: errores internos del servidor.

5. Seguridad y escalabilidad.

Seguridad

- **Autenticación segura:** Uso de JWT (JSON Web Tokens) para gestionar sesiones de usuario. Los tokens se almacenan en AsyncStorage del dispositivo y se persisten entre sesiones.
- **Cifrado de contraseñas:** Almacenamiento seguro en MySQL utilizando hashing de alta seguridad con bcrypt (factor de costo 10). El cambio de contraseña exige verificación de la contraseña actual antes de actualizar.
- **Verificación de cuenta:** Código de 6 dígitos enviado por correo con expiración de 3 minutos, adicionalmente el usuario solo tiene hasta 3 intentos. Limpieza automática de registros no verificados con código expirado para evitar correos obsoletos (cuentas zombie).
- **Control de intentos fallidos:** El login limita a 3 intentos fallidos consecutivos antes de redirigir al usuario al home, previniendo ataques de fuerza bruta.
- **Protección contra Inyección SQL:** Se emplean consultas parametrizadas mediante la librería mysql2, evitando la concatenación directa de datos suministrados por el usuario.
- **Validación de datos en frontend:** Validación del formato del correo (solo Gmail/Hotmail), fortaleza de contraseña (mínimo 8 caracteres, 2 números, 1 carácter especial), edad (18-100 años), nombre de usuario (11-20 caracteres alfanuméricos, sin palabras prohibidas) y documento (6-10 dígitos numéricos).

Escalabilidad

- **Arquitectura modular:** Código estructurado en TypeScript que separa controllers, routes, models, config y middlewares, permitiendo agregar nuevas funcionalidades sin afectar las existentes.
- **Separación de frontend y backend:** La app móvil consume la API REST de forma independiente, permitiendo escalar el servidor sin necesidad de actualizar la app.
- **Optimización de Base de Datos:** Uso de **Pool de conexiones** para reutilizar enlaces a MySQL y diseño de **índices avanzados** para consultas rápidas.
- **Manejo de datos dinámico:** Categorías, subcategorías, productos y usuarios se cargan dinámicamente desde la API, sin datos quemados en el frontend.
- **Posibilidad de crecimiento:** Se pueden agregar fácilmente nuevas funciones como pagos en línea, notificaciones push, recomendaciones de productos o integración con servicios cloud.

Escalabilidad del Proyecto

La app está pensada para crecer sin problemas cuando tenga más usuarios o productos.

1. Estructura modular

- El proyecto está organizado por secciones (auth, carrito, productos, perfil).
- Permite agregar nuevas funciones sin dañar lo existente.

2. Separación de frontend y backend

- La app móvil (frontend) puede conectarse a diferentes servicios o APIs.
- Facilita escalar el servidor sin cambiar la app.

3. Manejo de datos eficiente

- Los productos, carrito y usuarios se manejan de forma dinámica.
- Se pueden conectar a bases de datos grandes sin afectar el rendimiento.

4. Optimización de carga

- Las imágenes y productos se cargan sólo cuando se necesitan.
- Esto hace la app más rápida y ligera.

5. Posibilidad de crecimiento

- Se pueden agregar fácilmente nuevas funciones como pagos en línea, notificaciones push y recomendaciones de producto.

Conclusión

En conclusión, el proyecto TAM es una aplicación que cumple bien con lo que se propone, ya que permite a los usuarios interactuar de forma fácil y segura. Se tuvieron en cuenta aspectos importantes como la protección de la información (cifrado de contraseñas con bcrypt, verificación de cuenta por correo, control de intentos de login) y el buen funcionamiento del sistema (persistencia de sesión, sincronización del carrito, validación de stock en tiempo real).

Además, está pensada para crecer en el futuro, permitiendo agregar nuevas funciones o más usuarios sin que la aplicación se vuelva lenta o complicada. La arquitectura modular con separación de frontend (React Native/Expo) y backend (Node.js/Express) garantiza que cada parte pueda evolucionar de forma independiente. En general, es un proyecto sólido, útil y preparado para seguir mejorando.