Evaluación Unidad 3

INSTITUTO PROFESIONAL SAN SEBASTIAN

“Aplicación móvil con Expo + React Native + TypeScript miAppLoginMultiUsuarioApi-Real,autenticación JWT y CRUD ”

Asignatura:
Desarrollo de aplicaciones móviles
Sección:50
Alumnos:
Efren Tovar
Eduardo Ahumada
Daniel Castro
Jeremy Sanhueza
Profesor:
Boris Belmar

Enlaces

Repositorio GitHub (código fuente):
https://github.com/ejts29/miAppLoginMultiUsuarioApi

Video demostrativo de funcionamiento:
https://www.youtube.com/watch?v=TCWlXRvcXRs

endpoints documentados (producción - profesor):

https://todo-list.dobleb.cl/docs

API utilizada (producción - profesor):
https://todo-list.dobleb.cl/

Integrantes del grupo

EFREN TOVAR

Desarrollo principal de la aplicación, integración completa con la API real del profesor, implementación del flujo de registro y login, configuración de AuthContext, redirección protegida mediante Expo Router, pruebas con Postman, resolución de errores HTTP y adaptación del módulo Todo List para cumplir con las validaciones del backend. Grabación del video demostrativo. Documentación técnica inicial del proyecto.

EDUARDO AHUMADA

Revisión visual de pantallas, refinamiento de estilos, asistencia en la organización del proyecto, revisión del flujo de usuario, apoyo en pruebas funcionales, estructura del README y sugerencias de mejora en la claridad del código.

DANIEL CASTRO

Verificación del flujo general entre pantallas, pruebas del comportamiento del AuthContext, rastreo de errores, sugerencias de arquitectura y validación de las llamadas a la API (GET, POST, PATCH, DELETE). Complemento pruebas con Postman

JEREMY SANHUEZA

Apoyo en validación de rutas protegidas, revisión visual del módulo Todo List, documentación final (texto transcrito a PDF), pruebas finales del flujo completo desde registro hasta CRUD de tareas.

Descripción general de la aplicación

La presente entrega corresponde a la Evaluación 3 del curso de Desarrollo de Aplicaciones Móviles. El proyecto miAppLoginMultiUsuarioApi representa la evolución final del trabajo realizado en las evaluaciones anteriores, integrando ahora un flujo completo de autenticación real (login y registro), manejo de sesión mediante tokens JWT, consumo de una API en producción y operaciones CRUD completas sobre un módulo de tareas (Todo List).

A diferencia de la Evaluación 2, donde la persistencia se realizaba de forma local mediante AsyncStorage y FileSystem, en esta versión toda la información relevante (usuarios y tareas) se gestiona a través de una API REST pública proporcionada por el profesor, alojada en un entorno productivo. La aplicación ahora refleja un flujo de trabajo sólido, realista y propio de un entorno profesional: autenticación remota, validación, llamadas HTTP seguras y manejo de errores a nivel de red y backend.

Asimismo, se mantiene el uso de **Expo Router, React Native, TypeScript** y componentes nativos para experiencia móvil. El proyecto ha sido diseñado siguiendo principios de modularidad, claridad en la arquitectura y separación de responsabilidades.

Resumen Técnico del Proyecto
La aplicación móvil fue desarrollada con React Native y Expo Router, consumiendo principalmente la API REST del profesor (https://todo-list.dobleb.cl/) para gestionar tareas asociadas a un usuario autenticado.

Funcionalidades Clave:

- **Autenticación y Persistencia:** Implementación de Login y Registro de usuarios contra la API , con persistencia de sesión mediante tokens JWT almacenados en AsyncStorage.
- **CRUD Completo:** Operaciones de gestión (Crear, Leer, Actualizar, Eliminar) del módulo de tareas (Todo List).
- **Manejo de Media/Ubicación:** Integración de Expo Location y Expo ImagePicker para coordenadas y selección de fotos, adaptando el envío de la imagen a una URL *string* para cumplir con las restricciones de la API.
- **Arquitectura Sólida:** Uso de AuthContext para el manejo global de la sesión , Servicios API centralizados , y tipado estricto con TypeScript.

Estructura de Rutas y Lógica de Sesión: La navegación utiliza Expo Router y se basa en la lógica de AuthContext:

- app/_layout.tsx: Actúa como punto de entrada que consulta el estado de autenticación para aplicar la **protección de rutas**. Redirige al flujo de autenticación (/auth) si no hay token o a las pantallas internas (/home) si la sesión es válida.
- app/auth/index.tsx: Contiene la lógica para el Registro y Login, interactuando con la API y guardando el token JWT tras una autenticación exitosa.
- app/home/todo-list/: Módulo principal que gestiona las operaciones CRUD de tareas.
- app/home/profile.tsx: Incluye la función signOut() para cerrar la sesión.

Servicios API (src/services/api.ts): Todas las llamadas HTTP se centralizan en este módulo. Esta capa es responsable de implementar las funciones para la autenticación y las tareas (CRUD), asegurar que el token JWT se envíe en el *header* Authorization para las rutas protegidas, y manejar los errores de red y backend (ej. 400, 401).

Funcionalidades implementadas y Requerimientos de la Evaluación 3

Los requerimientos principales de esta evaluación fueron implementados de forma completa:

a) Autenticación real contra backend:

- Registro de usuarios (POST /auth/register).
- Inicio de sesión con validación de credenciales (POST /auth/login).
- Manejo de errores del servidor (401, 400).
- Uso de token JWT para mantener la sesión.
- Almacenar el token en AsyncStorage y cargarlo al iniciar la app.

b) Rutas protegidas mediante Expo Router:

- Acceso a pantallas internas solo si existe sesión válida.
- Redirección automática a login si no hay token.
- Redirección a Home cuando la sesión se inicia correctamente.

c) CRUD completo del módulo de tareas consumiendo la API real:

- Obtener tareas del usuario autenticado (GET /todos).
- Crear tareas con título, ubicación y foto simulada (POST /todos).
- Marcar tareas como completadas o no (PATCH /todos/:id).
- Eliminar tareas (DELETE /todos/:id).
- Manejo visual de carga, errores y estados vacíos.

d) Integración de módulos nativos:

- Expo ImagePicker para seleccionar fotos.
- Expo Location para obtener coordenadas.
- Validación previa antes de enviar datos al backend.
- Adaptación a las limitaciones del backend (la API no admite envío de binarios).

e) Calidad arquitectónica:

- Servicios de API centralizados en src/services/api.ts.
- Manejo global de sesión mediante AuthContext.
- Tipado estricto con TypeScript.
- Componentes y pantallas organizados siguiendo Expo Router.

Arquitectura y estructura del proyecto

La estructura final del proyecto es la siguiente:

```
1. miAppLoginMultiUsuarioApi/
├ app/
| ├ _layout.tsx | ├ auth/
| | └ index.tsx | └ home/
| ├ _layout.tsx | ├ index.tsx | ├ profile.tsx | └ todo-list/
| ├ index.tsx | └ create.tsx ├ src/
| ├ context/
| | └ AuthContext.tsx | ├ services/
| | └ api.ts | ├ storage/
| └ types/
| └ todolist.ts ├ app.json ├ package.json ├ tsconfig.json ├ .env └ README.md
2.
```

Flujo funcional de la aplicación

5.1 Autenticación
La aplicación inicia verificando si existe un token almacenado en AsyncStorage. Esta lógica es gestionada por AuthContext y utilizada por app/_layout.tsx para decidir si el usuario debe ser redirigido hacia la pantalla de login o hacia Home.

El registro exige:

- Email válido.
- Contraseña mínima de 6 caracteres.

Al registrar un nuevo usuario, automáticamente se inicia sesión y se almacena el token JWT.

El login valida credenciales contra la API y notifica errores propios del backend (credenciales inválidas, email inexistente, etc.).

5.2 Navegación protegida
La navegación se gestiona mediante Expo Router.
Las rutas internas (Home, Todo List, Perfil) solo se muestran si existe token.
Si la sesión se cierra, toda la navegación es reseteada.

5.3 Módulo Todo List con API real
La aplicación consulta, crea, modifica y elimina tareas mediante solicitudes HTTP.
Se implementaron validaciones previas y manejo de errores.

Para el campo de imagen, debido a que el backend no soporta multipart/form-data, se optó por enviar una URL simulada en caso de que el usuario seleccione una foto.

5.4 Integración de ubicación
Expo Location obtiene coordenadas que luego se envían en formato JSON { latitude, longitude }. La API las acepta si tienen tipo numérico válido.

Pruebas realizadas

Se realizaron pruebas exhaustivas tanto desde la app móvil como mediante Postman:

- Registro de usuario válido.
- Login con credenciales correctas e incorrectas.
- Acceso autorizado y no autorizado a rutas internas.
- Obtención de lista de tareas vacía y con datos.
- Creación de tareas con o sin ubicación.
- Actualización de estado completed.
- Eliminación de registros.
- Manejo de errores del servidor.

Todas las pruebas fueron exitosas.

Uso de herramientas de apoyo (IA)

Se utilizó inteligencia artificial como herramienta de apoyo para:

- Diagnóstico de errores en llamadas HTTP.
- Revisión de validaciones de email y contraseña.
- Optimización del manejo de tokens y rutas protegidas.
- Apoyo conceptual en arquitectura (Service Layer, Context API).
- Generación de documentación técnica y estructura del informe.

Todas las implementaciones finales fueron revisadas, corregidas y validadas por los integrantes del grupo.

Observaciones finales

El proyecto cumple de manera completa con todos los requerimientos definidos para la Eva3.

La integración con una API real permitió que el equipo adquiriera experiencia en flujos de autenticación, consumo de servicios REST, manejo de errores reales y uso avanzado de Expo Router con rutas protegidas.