Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria
de Ingeniería campus Zacatecas

Desarrollo de Aplicaciones Móviles Nativas

Actividad: Examen 2

Grupo: 7cm1

Alumnos:

Isay Montejano Quirino

Docente:

Efraín Arredondo Morales

26/11/2024

Reporte: Evaluación de Conocimientos en Consumo de APIs REST y Persistencia de Datos

1. Objetivo

El propósito de este proyecto es evaluar los conocimientos en el consumo de APIs REST utilizando Retrofit y la persistencia de datos en una base de datos local con Room. Este desarrollo tiene como objetivo principal garantizar que los datos puedan ser visualizados incluso sin conexión a Internet, ofreciendo una experiencia de usuario fluida y funcional.

2. Introducción

En el mundo actual, muchas aplicaciones móviles dependen del acceso a servicios externos mediante APIs REST. Sin embargo, un desafío importante es garantizar la disponibilidad de datos en caso de pérdida de conectividad. Para resolver esto, se combinan técnicas de consumo de datos mediante Retrofit y almacenamiento local con Room, proporcionando acceso offline y reduciendo la dependencia de la conectividad en tiempo real.

3. Planteamiento del Problema

La necesidad de aplicaciones robustas que manejen datos tanto en línea como fuera de línea motiva este proyecto. El desafío incluye diseñar una solución que pueda consumir datos de la API pública JSONPlaceholder, almacenarlos localmente y gestionarlos de manera eficiente para visualización en cualquier escenario.

4. Requisitos Técnicos

4.1 Consumo de API

1. Utilizar Retrofit para establecer la conexión con la API pública JSONPlaceholder.
2. Descargar y mostrar una lista de usuarios.
3. Visualizar los posts de un usuario seleccionado.
4. Mostrar los comentarios de un post seleccionado.

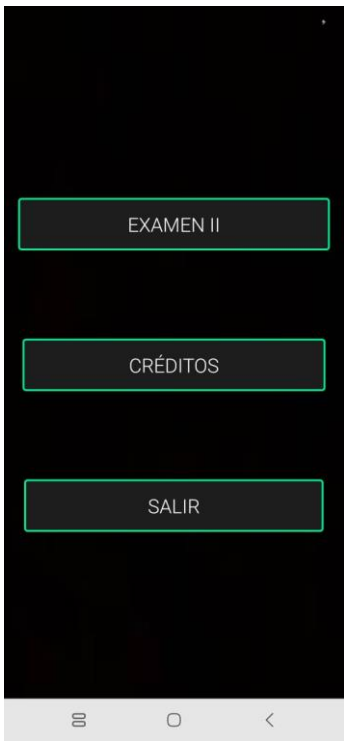
4.2 Persistencia Local

1. Almacenar un usuario descargado en una base de datos local utilizando Room al realizar un clic largo.
2. Al iniciar la aplicación:
- Si hay conexión a Internet, mostrar los usuarios actualizados desde la API.
- Si no hay conexión, mostrar únicamente los datos almacenados localmente.

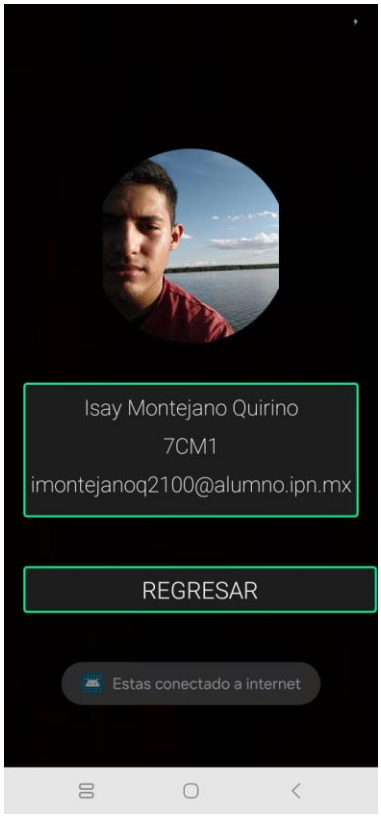
4.3 Interfaz de Usuario

1. Implementar una actividad inicial con tres opciones:

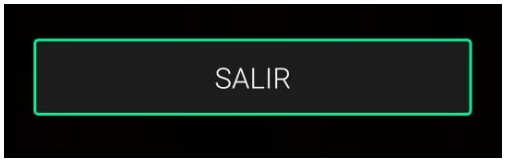
- Usuarios: Para navegar a la lista de usuarios.



- Créditos: Información sobre el desarrollo de la aplicación.



- Salir: Para cerrar la aplicación.



2. Usar RecyclerView para mostrar listas de usuarios, posts y comentarios, asegurando una navegación fluida.

5. Metodología

Se utilizó el enfoque de desarrollo basado en la arquitectura MVVM (Model-View-ViewModel) para estructurar el proyecto. Este modelo facilita la separación de responsabilidades, mejorando la mantenibilidad y escalabilidad del código. Las herramientas principales incluyen Retrofit para el manejo de solicitudes HTTP y Room para la persistencia de datos locales.

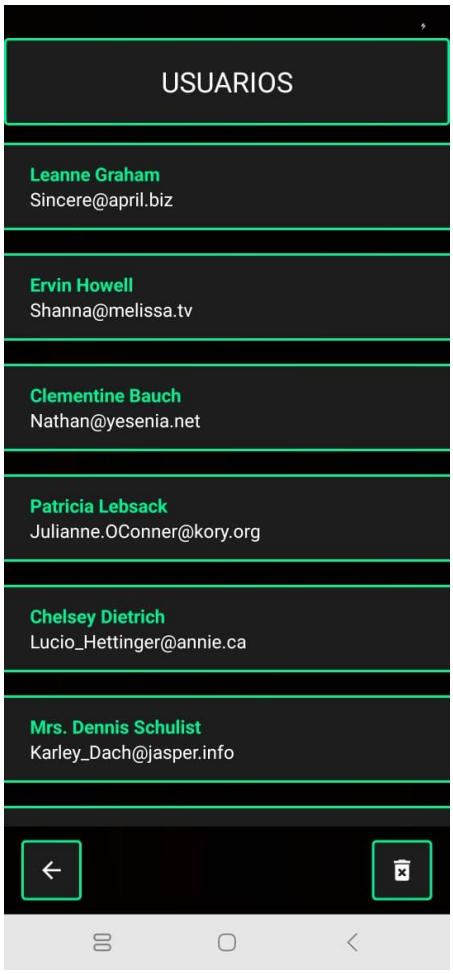
6. Implementación

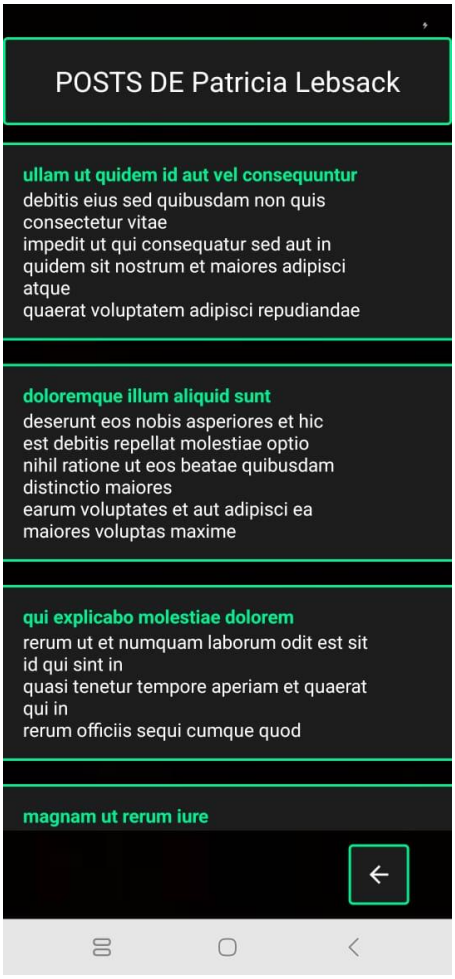
A continuación, se detallan los pasos clave para implementar el proyecto:

1. Configuración de Retrofit:
   - Crear un cliente HTTP para consumir los datos de JSONPlaceholder.
   - Manejar respuestas y errores de la API.
   2. Diseño de la base de datos con Room:
   - Definir entidades para usuarios, posts y comentarios.
   - Configurar DAOs para operaciones CRUD.
   - Configurar la base de datos Room con un builder.
   3. Diseño de la UI:
   - Crear RecyclerViews personalizados para usuarios, posts y comentarios.
   - Diseñar una actividad inicial con opciones claras.
   4. Verificación de conectividad:
   - Implementar un método para verificar si hay conexión a Internet.
   - Decidir si mostrar datos de la API o datos almacenados.

7. Resultados Esperados

Se espera que la aplicación permita a los usuarios:
1. Consultar datos actualizados cuando haya conexión a Internet.
2. Acceder a los datos almacenados localmente cuando no haya conexión.
3. Navegar de manera intuitiva entre usuarios, posts y comentarios.





8. Conclusión

Este proyecto combina habilidades esenciales para el desarrollo de aplicaciones móviles, integrando consumo de APIs REST y almacenamiento local. La implementación de Retrofit y Room asegura una experiencia de usuario fluida y resiliente, demostrando la importancia de manejar datos de forma eficiente y confiable. Esta solución puede ser la base para aplicaciones más complejas con características avanzadas.

9. Referencias

- JSONPlaceholder API: https://jsonplaceholder.typicode.com
- Documentación oficial de Retrofit: https://square.github.io/retrofit/
- Documentación oficial de Room:
https://developer.android.com/jetpack/androidx/releases/room