UNIVERSIDAD DON BOSCO

Entrega segunda evaluacion.

Integrantes: Heysel Guadalupe Argueta Hernandez AH230907

Jaime David Santos Carrillo SC230146

Carlos Amilcar Mejia Cea MC232092

Ahsley Nicole Aguilar Ramirez AR232346

Índice

Introducción

En el desarrollo de aplicaciones móviles y web, la integración con APIs REST es un componente esencial para la comunicación con servidores y servicios en la nube. Las APIs REST permiten que aplicaciones distribuidas interactúen de manera eficiente y flexible mediante el uso de protocolos estándar como HTTP.

En Android, la biblioteca Retrofit se ha posicionado como una de las herramientas más populares para consumir APIs debido a su simplicidad y capacidad para manejar grandes volúmenes de datos. Además, en Kotlin, el uso de corutinas ofrece un enfoque moderno y eficiente para manejar operaciones asíncronas, evitando los problemas tradicionales asociados a los callbacks y los hilos.

Este trabajo tiene como objetivo explicar los conceptos clave de APIs REST, el uso de Retrofit para consumir estas APIs, cómo gestionar errores en las solicitudes y el papel de las corutinas en la programación asíncrona en Kotlin. Además, se comparan diferentes bibliotecas de consumo de APIs y se analizan estrategias efectivas para el manejo de errores en aplicaciones Android.

APIs REST, Retrofit, Manejo de Errores y Corutinas en Kotlin

1. APIs REST

Definición de API REST y su importancia en el desarrollo de aplicaciones modernas

Una API REST (Representational State Transfer) es una arquitectura para diseñar servicios web. En ella, los recursos se representan y se accede a ellos a través de URLs utilizando los métodos HTTP estándar (GET, POST, PUT, DELETE). REST permite que diferentes sistemas se comuniquen entre sí de manera simple y eficiente, lo que es fundamental en el desarrollo de aplicaciones modernas, especialmente en aplicaciones móviles y web que requieren integración con servidores y servicios en la nube.

Principios fundamentales de REST

- Statelessness (Sin Estado): Cada solicitud del cliente al servidor debe contener toda la información necesaria para entender y procesar la solicitud, sin depender del estado almacenado en el servidor.
- Recursos: Cada recurso (por ejemplo, un usuario o un producto) tiene una URI única que lo identifica, y se accede a él mediante los métodos HTTP.
- Uniform Interface (Interfaz uniforme): REST utiliza una interfaz consistente y bien definida para interactuar con los recursos.
- Caching (Caché): Las respuestas de la API pueden ser almacenadas en caché para mejorar el rendimiento y reducir la carga en el servidor.

Ejemplos de APIs REST públicas que se pueden utilizar para pruebas

- OpenWeatherMap: Proporciona datos meteorológicos en tiempo real.
- JSONPlaceholder: Un servicio gratuito para realizar pruebas y prototipos que proporciona datos ficticios para recursos como usuarios, publicaciones, comentarios, etc.

2. Retrofit

¿Qué es Retrofit y por qué se utiliza en el desarrollo de aplicaciones Android?

Retrofit es una biblioteca cliente de tipo HTTP para Android y Java, desarrollada por Square. Facilita la conexión con APIs REST, convirtiendo las respuestas JSON o XML en objetos Java. Es ampliamente utilizada en el desarrollo de aplicaciones Android debido a su simplicidad y facilidad para manejar solicitudes y respuestas HTTP.

Comparación entre Retrofit y otras bibliotecas para consumir APIs

- **Retrofit vs Volley**: Retrofit está más optimizado para manejar grandes cargas de datos y convertir respuestas JSON automáticamente en objetos Java, mientras que Volley es más adecuado para solicitudes de red ligeras.
- **Retrofit vs OkHttp**: OkHttp es la biblioteca subyacente utilizada por Retrofit para manejar las solicitudes HTTP. Retrofit agrega una capa adicional de abstracción, haciendo que sea más sencillo de usar que OkHttp directamente.

Explicación de cómo Retrofit maneja las solicitudes HTTP y las respuestas JSON

Retrofit convierte automáticamente las respuestas HTTP en objetos Java utilizando convertidores como `Gson` o `Moshi`. Las solicitudes HTTP se definen utilizando interfaces y anotaciones como `@GET`, `@POST`, etc., lo que facilita el manejo de operaciones como autenticación, parámetros de consulta y cuerpo de las solicitudes.

3. Manejo de Errores

Estrategias para manejar errores en las solicitudes a la API

1. Validación del cliente: Asegurarse de que los datos sean válidos antes de enviar la solicitud.
2. Reintento: Implementar un sistema de reintentos para errores transitorios.
3. Time-out: Configurar límites de tiempo para evitar solicitudes colgadas.
4. Backoff exponencial: Aumentar progresivamente el tiempo entre cada reintento.
5. Logs de errores: Registrar los errores en archivos o servicios externos para su análisis.

Cómo interpretar diferentes códigos de estado HTTP

- 200 (OK): La solicitud fue exitosa.

- 201 (Created): Un nuevo recurso fue creado exitosamente.
- 400 (Bad Request): La solicitud contiene datos inválidos.
- 401 (Unauthorized): El cliente no tiene las credenciales necesarias.
- 403 (Forbidden): El cliente no tiene permiso para acceder al recurso.
- 404 (Not Found): El recurso solicitado no existe.
- 500 (Internal Server Error): El servidor encontró un error inesperado.

4. Corutinas en Kotlin

Introducción a las corutinas en Kotlin y su uso para manejar operaciones asíncronas

Las corutinas en Kotlin permiten manejar operaciones asíncronas de manera eficiente, evitando el uso excesivo de hilos. Permiten pausar y reanudar funciones sin bloquear el hilo principal, lo que es útil para tareas como solicitudes de red o acceso a bases de datos.

5. Ventajas del uso de corutinas en comparación con otros enfoques como callbacks

1. Código más legible: Las corutinas permiten escribir código asíncrono de manera secuencial.
2. Evitar Callback Hell: Con las corutinas, se elimina la necesidad de anidar múltiples callbacks.
3. Mejor manejo de errores: Las corutinas permiten usar bloques try-catch para manejar excepciones fácilmente.
4. Control preciso: Las corutinas permiten controlar en qué hilo se ejecutan las operaciones sin bloquear los hilos.
5. Eficiencia: Las corutinas son ligeras y consumen menos recursos que los hilos.