**Imagen que contiene Icono

Descripción generada automáticamenteDibujo en blanco y negro

Descripción generada automáticamente con confianza bajaInstituto Politécnico Nacional**

**Escuela Superior De Computo**

**Desarrollo De Aplicaciones Móviles Nativas**

**Práctica 5**

**Consulta de Base de Datos vía APIs**

Nombre Del Alumno:

García Quiroz Gustavo Ivan | 2022630278

Grupo: 7CV3

Nombre Del Profesor: Hurtado Avilés Gabriel

Fecha De Entrega: 12/05/2025

**Índice**

[1 Introducción 4](#_Toc195968450)

[2 Desarrollo 5](#_Toc195968451)

[2.1 Ejercicio 1: Integración de Consultas a la Base de Datos vía API REST 5](#_Toc195968452)

[2.1.1 Implementación de la conexión a la API REST 5](#_Toc195968453)

[2.1.2 Persistencia de la sesión de usuario 6](#_Toc195968454)

[2.1.3 Pruebas de conexión con Retrofit 7](#_Toc195968455)

[2.1.4 Almacenamiento local con SQLite 8](#_Toc195968456)

[2.1.5 Mecanismos de sincronización de datos 10](#_Toc195968457)

[2.2 Ejercicio 2: Consumo de APIs Públicas 11](#_Toc195968458)

[2.2.1 Descripción de la API seleccionada 11](#_Toc195968459)

[2.2.2 Implementación de la integración 12](#_Toc195968460)

[2.2.3 Procesamiento de datos JSON 12](#_Toc195968461)

[2.2.4 Funcionalidades de búsqueda de libros y autores 14](#_Toc195968462)

[2.2.5 Sistema de recomendaciones personalizadas 15](#_Toc195968463)

[2.3 Ejercicio 3: Funcionalidades de Búsqueda, Favoritos y Recomendaciones 17](#_Toc195968464)

[2.3.1 Historial de búsquedas 17](#_Toc195968465)

[2.3.2 Sistema de favoritos 18](#_Toc195968466)

[2.3.3 Algoritmo de recomendaciones 19](#_Toc195968467)

[3 Capturas de Pantalla 19](#_Toc195968468)

[3.1 Inicio de Sesión 19](#_Toc195968469)

[3.2 Registro 20](#_Toc195968470)

[3.3 Perfil de Usuario 20](#_Toc195968471)

[3.4 Funcionalidades de Búsqueda, Favoritos y Recomendaciones 21](#_Toc195968472)

[4 Conclusiones 25](#_Toc195968473)

[5 Bibliografía 26](#_Toc195968474)

# Introducción

La Práctica 5 "Consulta de Base de Datos vía APIs" tiene como objetivo principal desarrollar e integrar consultas avanzadas a bases de datos mediante APIs, añadiendo funcionalidades de búsqueda y recomendaciones en una aplicación móvil Android. Este proyecto permite adquirir experiencia en la implementación de sistemas que utilizan tanto bases de datos locales como servicios remotos, garantizando la funcionalidad continua incluso en situaciones sin conexión a internet.

En el desarrollo de esta práctica, se ha creado la aplicación "SystemBooks", un sistema de gestión y búsqueda de libros que se integra con la API de Open Library para obtener información bibliográfica actualizada. La aplicación permite a los usuarios buscar libros, autores, guardar favoritos y recibir recomendaciones personalizadas basadas en su historial de búsqueda y preferencias.

La arquitectura del proyecto se divide en dos componentes principales: un backend desarrollado con Spring Boot (SistemaRecomendacion2025-2) que gestiona la autenticación, usuarios y persistencia de datos en el servidor, y una aplicación móvil Android (SystemBooks) que implementa la interfaz de usuario y la lógica de consumo de APIs. Esta estructura permite una clara separación de responsabilidades y facilita el mantenimiento y escalabilidad del sistema.

# Desarrollo

## Ejercicio 1: Integración de Consultas a la Base de Datos vía API REST

### Implementación de la conexión a la API REST

La aplicación móvil "SystemBooks" se conecta a dos APIs diferentes para su funcionamiento: la API REST propia desarrollada con Spring Boot (SistemaRecomendacion2025-2) y la API pública de Open Library. Esta arquitectura permite manejar tanto la autenticación y gestión de usuarios como la obtención de datos bibliográficos.

La implementación de la conexión a la API REST se realiza mediante las siguientes clases:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 1 ApiClient.java - Configuración del cliente Retrofit

Texto

El contenido generado por IA puede ser incorrecto.

Figura 2 ApiService.java - Definición de endpoints

La clase ApiRepository sirve como intermediario entre la API y las actividades/fragmentos de la aplicación, proporcionando métodos para realizar operaciones con el servidor.

### Persistencia de la sesión de usuario

La persistencia de la sesión del usuario se logra mediante la clase SessionManager, que utiliza SharedPreferences para almacenar el token JWT y la información del usuario:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 3 SessionManager.java

Esta implementación permite mantener la sesión del usuario activa entre inicios de la aplicación y mostrar la información correspondiente en la interfaz (nombre de usuario, rol) en elementos como el nav\_header.xml.

### Pruebas de conexión con Retrofit

Se utilizó Retrofit como cliente HTTP para conectarse a las APIs. Las pruebas de conexión se realizaron mediante la implementación de callbacks para manejar las respuestas y errores. Para interceptar las peticiones y añadir el token de autenticación, se implementó la clase AuthInterceptor:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4 AuthInterceptor.java

### Almacenamiento local con SQLite

La aplicación implementa una base de datos local mediante SQLite utilizando clases auxiliares como DatabaseHelper, BooksDatabaseHelper, y contratos como FavoritesContract y SearchHistoryContract:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 5 DatabaseHelper.java

Los contratos definen la estructura de las tablas:

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Figura 6 Tablas de favoritos y del historial.

### Mecanismos de sincronización de datos

La sincronización entre la base de datos local y la API REST se implementa mediante repositorios que manejan tanto el almacenamiento local como las llamadas a la API.

Este enfoque garantiza que la aplicación funcione correctamente incluso sin conexión a internet, manteniendo la coherencia de los datos entre la base de datos local y el servidor cuando la conectividad está disponible.

Para optimizar el tráfico de red y mejorar la experiencia del usuario, se implementó un CacheInterceptor que permite el almacenamiento en caché de las respuestas HTTP.

Este sistema completo de persistencia y sincronización asegura que la aplicación pueda funcionar offline y mantener los datos actualizados cuando hay conexión disponible.

## Ejercicio 2: Consumo de APIs Públicas

### Descripción de la API seleccionada

La Open Library API es una interfaz de programación de aplicaciones que proporciona acceso a la extensa base de datos de libros de Open Library, un proyecto de Internet Archive. Esta API permite buscar y recuperar información detallada sobre millones de libros, autores y editoriales. Para esta práctica, se seleccionó esta API debido a su riqueza de datos bibliográficos, facilidad de uso, y su naturaleza gratuita y de código abierto.

**Características principales de la Open Library API:**

* **Acceso gratuito**: No requiere clave de API para consultas básicas.
* **Formato de respuesta**: Devuelve datos en formato JSON, fácilmente procesable por aplicaciones.
* **Endpoints principales**:
  + /search.json: Para búsqueda de libros por título, autor, etc.
  + /books/{id}.json: Para obtener detalles específicos de un libro.
  + /authors/{id}.json: Para obtener información sobre autores.
* **Variedad de datos**: Proporciona títulos, autores, fechas de publicación, ISBN, portadas, y más.

### Implementación de la integración

La integración con la Open Library API se implementó mediante Retrofit, un cliente HTTP para Android que facilita la comunicación con servicios web RESTful. A continuación, se detalla la implementación:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 7 OpenLibraryApi.java - Interface que define los endpoints de la API

### Procesamiento de datos JSON

La respuesta de la API de Open Library viene en formato JSON, por lo que se definieron modelos de datos para facilitar su procesamiento:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 8 SearchResponse.java - Modelo para respuestas de búsqueda

Texto

El contenido generado por IA puede ser incorrecto.

Figura 9 BookResponse.java - Modelo para la información de un libro

### Funcionalidades de búsqueda de libros y autores

La aplicación SystemBooks implementa tres tipos principales de búsqueda:

1. **Búsqueda general de libros**: Permite buscar libros por título, autor o cualquier término.
2. **Búsqueda por autor**: Facilita encontrar libros escritos por un autor específico.
3. **Exploración por categorías**: Permite navegar por libros clasificados en diferentes categorías.

Estas funcionalidades se implementan en SearchFragment.java:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 10 SearchFragment.java - Fragmento para búsqueda de libros

### Sistema de recomendaciones personalizadas

El sistema de recomendaciones se implementa analizando el historial de búsquedas y favoritos del usuario para sugerir libros similares:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 11 RecommendationEngine.java - Sistema de recomendaciones

El sistema de recomendaciones implementado proporciona sugerencias personalizadas basadas en:

1. **Autores favoritos**: Recomienda libros de autores que el usuario ha guardado en favoritos.
2. **Historial de búsqueda**: Utiliza las búsquedas recientes para sugerir libros relacionados.
3. **Categorías populares**: Como respaldo, ofrece recomendaciones de categorías populares.

Esta implementación garantiza que el usuario siempre reciba recomendaciones relevantes, incluso si es nuevo en la aplicación o tiene poca actividad registrada.

Además, el sistema de recomendaciones funciona de manera eficiente tanto online como offline, aprovechando la base de datos local para proporcionar sugerencias incluso sin conexión a internet.

## Ejercicio 3: Funcionalidades de Búsqueda, Favoritos y Recomendaciones

### Historial de búsquedas

El sistema implementa un historial de búsquedas que permite a los usuarios acceder rápidamente a sus consultas anteriores. Esta funcionalidad se gestiona a través de la clase SearchHistoryRepository que interactúa tanto con la base de datos local como con el servidor:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 12

Figura 13 SearchHistoryRepository.java

La visualización del historial se realiza a través del fragmento SearchHistoryFragment, que utiliza el adaptador SearchHistoryAdapter para mostrar las búsquedas previas. Este adaptador permite al usuario repetir búsquedas anteriores con un simple clic.

### Sistema de favoritos

**Perspectiva del usuario**

El sistema de favoritos permite a los usuarios guardar los libros que les interesan para acceder rápidamente a ellos en el futuro. Los usuarios pueden:

* Añadir libros a favoritos desde la vista de detalles del libro o desde los resultados de búsqueda
* Ver su lista de favoritos en el fragmento FavoritesFragment
* Eliminar elementos de la lista de favoritos

La implementación del fragmento de favoritos muestra los libros guardados y permite interacciones:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 14 FavoritesFragment.java

**Perspectiva del administrador**

Los administradores tienen acceso a funcionalidades adicionales a través del fragmento UserManagementFragment, donde pueden:

* Ver la lista de todos los usuarios
* Acceder a los favoritos de cada usuario
* Administrar los favoritos de los usuarios

La implementación permite a los administradores gestionar los datos y obtener información sobre los intereses de los usuarios.

### Algoritmo de recomendaciones

El sistema implementa un algoritmo de recomendaciones basado en los favoritos del usuario y su historial de búsquedas. Este algoritmo analiza patrones en las preferencias del usuario para sugerir nuevos libros que podrían interesarle:

Texto

El contenido generado por IA puede ser incorrecto.

Figura 15 Algoritmo de recomendaciones

# Capturas de Pantalla

## Inicio de Sesión

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 16 Pantalla de Inicio de Sesión.

## Registro

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 17 Pantalla de Registro

## Perfil de Usuario

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 18 Perfil de Usuario

## Funcionalidades de Búsqueda, Favoritos y Recomendaciones

Calendario

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto. Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto. Imagen que contiene Gráfico

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto. Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 19 Funcionalidades de Búsqueda, Favoritos y Recomendaciones

# Conclusiones

El desarrollo de la Práctica 5 ha permitido adquirir competencias avanzadas en la integración de APIs y bases de datos en aplicaciones móviles Android, enfrentando retos significativos como la gestión del estado de conexión, la sincronización de datos entre el sistema local y remoto, y la implementación de mecanismos efectivos de búsqueda y recomendación. La aplicación "SystemBooks" demuestra cómo es posible crear sistemas robustos que funcionan tanto online como offline, utilizando SQLite para el almacenamiento local y Retrofit para comunicarse con servicios externos como la API REST propia y Open Library. Los principales logros incluyen la implementación de un sistema de autenticación seguro, una interfaz de usuario intuitiva y un mecanismo eficiente de persistencia de datos que mantiene la coherencia entre el dispositivo y el servidor. Para futuras iteraciones, sería valioso expandir el sistema de recomendaciones incorporando algoritmos más sofisticados de machine learning que mejoren la personalización de sugerencias, así como añadir soporte para más formatos de libros y fuentes de información bibliográfica adicionales.

# Bibliografía

Android Developers. (2025). *Android Developer Documentation*. Recuperado de <https://developer.android.com/docs>

Spring Framework. (2025). *Spring Boot Reference Documentation*. Recuperado de <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>