Instituto Politécnico Nacional

Escuela Superior De Computo

Desarrollo De Aplicaciones Móviles Nativas

Práctica 2.- Aplicación móvil básica para operaciones CRUD con un servicio REST

Nombre Del Alumno:

García Quiroz Gustavo Ivan | 2022630278

Grupo: 7CV3

Nombre Del Profesor: Hurtado Avilés Gabriel

Fecha De Entrega: 24/03/2025

Índice

Tabla de contenido

Introducción

En el marco de la asignatura de Desarrollo de Aplicaciones Móviles Nativas, se desarrolló una aplicación móvil Android denominada SystemBooks, que implementa un sistema integral de gestión de usuarios con operaciones CRUD (Crear, Leer, Actualizar, Borrar) mediante un servicio REST seguro.

El objetivo principal de esta práctica fue crear una aplicación móvil que permitiera la autenticación de usuarios con roles diferenciados, específicamente Administrador y Usuario, con funcionalidades y permisos distintivos. La aplicación se compone de dos componentes principales:

1. Aplicación móvil Android (SystemBooks): Desarrollada en Java, utiliza una arquitectura modular con fragments, gestión de navegación, y conexión a servicios REST.
2. Servicio REST (SistemaRecomendacion2025-2): Implementado en Spring Boot con Java, proporciona endpoints seguros para autenticación, gestión de usuarios y operaciones CRUD.

La implementación aborda desafíos críticos de seguridad informática, como:

- Autenticación de usuarios
- Encriptación de contraseñas
- Control de acceso basado en roles
- Gestión segura de sesiones

El proyecto integra múltiples tecnologías y conceptos de desarrollo móvil, incluyendo Retrofit para comunicaciones HTTP, fragments para interfaces modulares, y una estructura de proyecto que sigue las mejores prácticas de Android.

Desarrollo

Arquitectura del Proyecto SystemBooks

La arquitectura del proyecto SystemBooks se diseñó siguiendo principios de desarrollo modular, respondiendo a los requisitos de la práctica. El sistema se compone de dos componentes principales: una aplicación móvil Android y un servicio REST desarrollado en Spring Boot.

Estructura de Paquetes en Android

La estructura de paquetes de la aplicación Android se organizó para garantizar una separación de responsabilidades:

- com.example.systembooks.model: Contiene las clases de modelo de datos que representan la estructura fundamental de la información.
- com.example.systembooks.api: Gestiona la configuración del cliente REST y las interfaces de comunicación.
- com.example.systembooks.fragment: Implementa las interfaces de usuario modulares utilizando la arquitectura de fragmentos.
- com.example.systembooks.util: Agrupa utilidades y clases de soporte para funcionalidades transversales.
- com.example.systembooks.repository: Maneja la gestión de datos y la comunicación con la API REST.

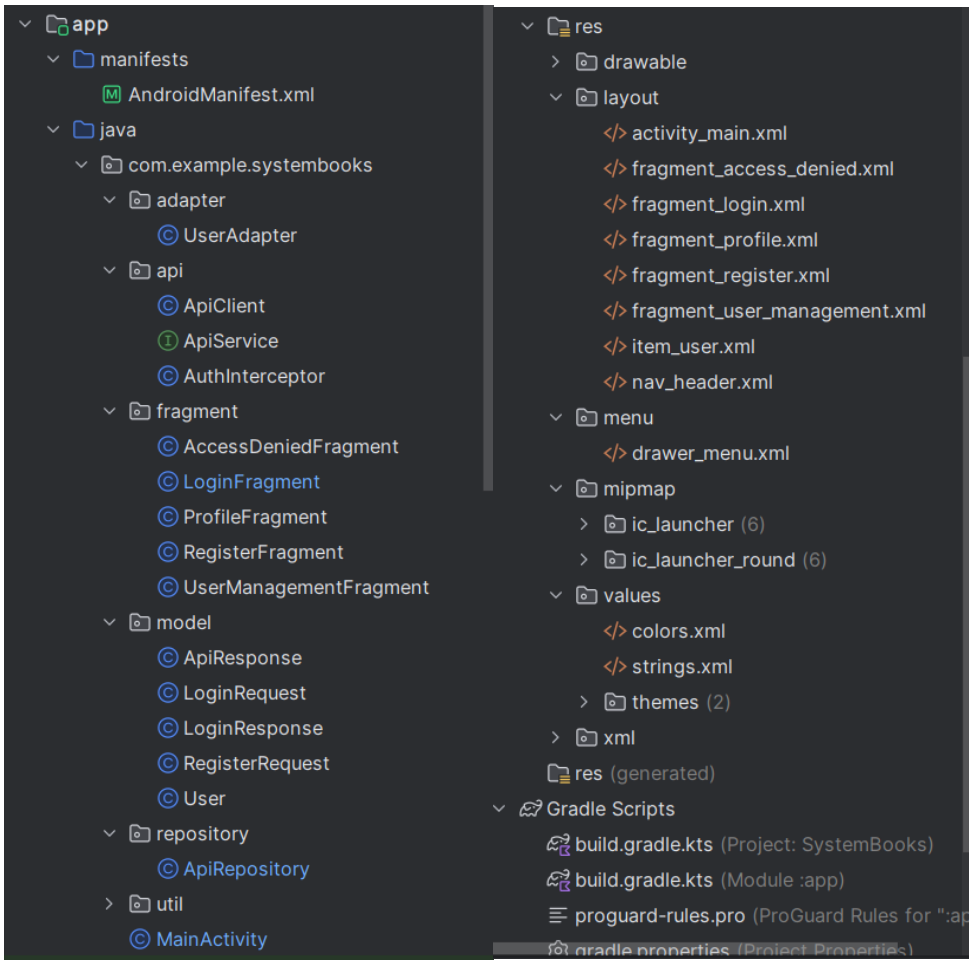

Figura 1 Proyecto SystemBooks

Componentes de Seguridad

Sistema de Autenticación

El sistema de autenticación se implementó considerando múltiples aspectos de seguridad. La autenticación se realiza mediante un proceso robusto que incluye verificación de credenciales, encriptación de contraseñas y gestión de sesiones seguras.

Gestión de Contraseñas

La clase PasswordUtils implementa mecanismos avanzados de encriptación para proteger las credenciales de usuario. Se utilizaron algoritmos de hash seguros para garantizar que las contraseñas nunca se almacenen en texto plano.

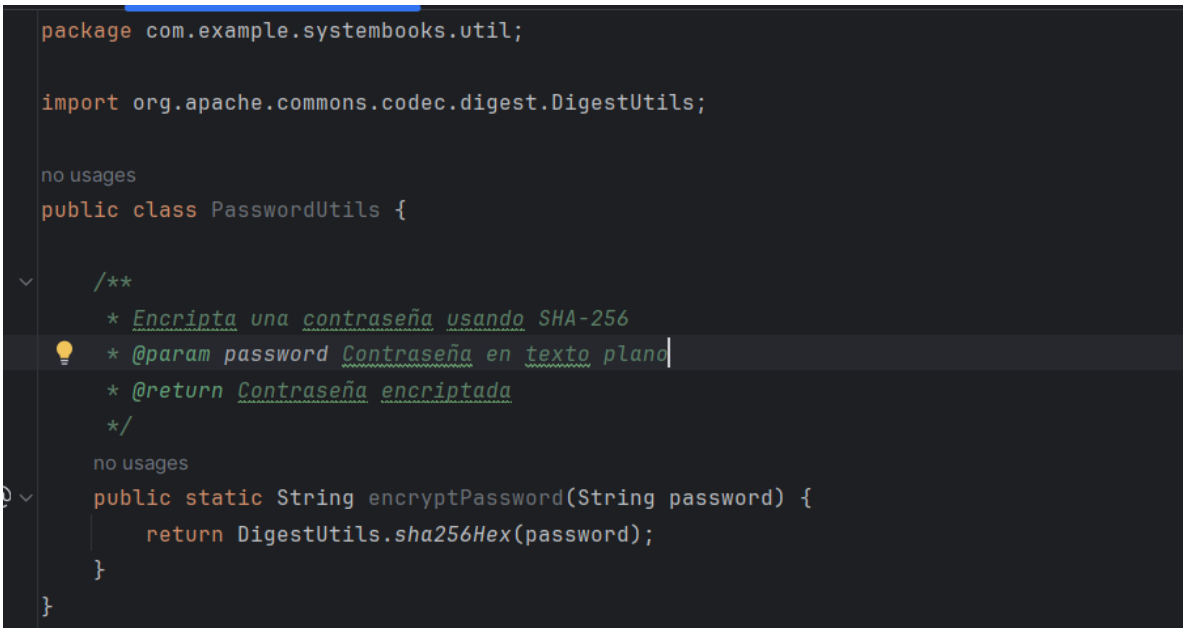

Figura 2 Clase PasswordUtils

Control de Sesiones

El SessionManager gestiona el ciclo de vida de las sesiones de usuario, implementando mecanismos para:

- Crear sesiones seguras
- Validar tokens de autenticación
- Manejar la expiración de sesiones
- Proteger contra vulnerabilidades de robo de sesión

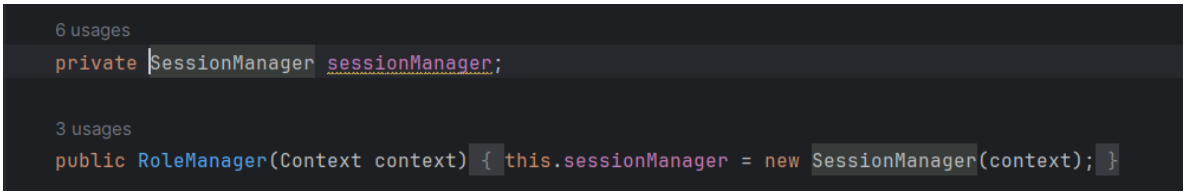

Figura 3 SessionManager

Implementación de Roles

Modelo de Roles

Se desarrolló un sistema de roles con dos niveles de acceso claramente diferenciados:

- **Rol de Administrador**: Acceso completo a todas las operaciones CRUD.
- **Rol de Usuario**: Acceso limitado a su perfil y funcionalidades específicas.

Control de Acceso

La clase RoleManager implementa la lógica de verificación de permisos, asegurando que cada usuario solo pueda acceder a las funcionalidades correspondientes a su rol.

```java
package com.example.systembooks.util;

import android.content.Context;

17 usages
public class RoleManager {

    // Constantes para los diferentes roles
    6 usages
    public static final String ROLE_GUEST = "guest";
    3 usages
    public static final String ROLE_USER = "ROLE_USER";
    5 usages
    public static final String ROLE_ADMIN = "ROLE_ADMIN";

    6 usages
    private SessionManager sessionManager;

    3 usages
    public RoleManager(Context context) { this.sessionManager = new SessionManager(context); }

    /**
     * Verifica si el usuario actual tiene un rol específico
     * @param role Rol a verificar
     * @return true si el usuario tiene el rol, false en caso contrario
     */
    3 usages
    public boolean hasRole(String role) {
        if (!sessionManager.isLoggedIn()) {
            return ROLE_GUEST.equals(role);
        }
```

```java
public class RoleManager {
    public boolean hasRole(String role) {
        String userRole = sessionManager.getUserRole();
        if (userRole == null) {
            return false;
        }

        // Un administrador tiene acceso a todas las funciones
        if (userRole.equals(ROLE_ADMIN)) {
            return true;
        }

        // Si no es admin, se compara con el rol específico
        return userRole.equals(role);
    }

    /**
     * Verifica si el usuario está autorizado para acceder a funciones admin
     * @return true si es administrador, false en caso contrario
     */
    5 usages
    public boolean isAdmin() { return hasRole(ROLE_ADMIN); }

    /**
     * Verifica si el usuario está autorizado para acceder a funciones de usuario
     * @return true si es usuario o administrador, false en caso contrario
     */
    public boolean isUser() { return hasRole(ROLE_USER) || hasRole(ROLE_ADMIN); }

    /**
     * Verifica si el usuario es invitado (no ha iniciado sesión)
     * @return true si es invitado, false en caso contrario
     */
    no usages
    public boolean isGuest() { return !sessionManager.isLoggedIn(); }

    /**
     * Obtiene el rol actual del usuario
     * @return Rol del usuario o "guest" si no está logueado
     */
    1 usage
    public String getCurrentRole() {
        if (!sessionManager.isLoggedIn()) {
            return ROLE_GUEST;
        }

        String role = sessionManager.getUserRole();
        return role != null ? role : ROLE_GUEST;
    }
}
```

Figura 4 RoleManager

Interfaces de Usuario

Se implementaron fragmentos modulares para cada funcionalidad principal:

- LoginFragment: Gestión de inicio de sesión
- RegisterFragment: Registro de nuevos usuarios
- ProfileFragment: Visualización y edición de perfil
- UserManagementFragment: Administración de usuarios (solo para administradores)
-

Comunicación con Servicio REST

Configuración de Retrofit

Se utilizó Retrofit para simplificar la comunicación con el servicio REST, implementando interfaces de comunicación que abstraen la complejidad de las llamadas HTTP.

Interceptores de Autenticación

El AuthInterceptor gestiona la inyección de tokens de autenticación en cada solicitud, garantizando la seguridad de las comunicaciones.

Modelo de Datos

Se definieron modelos de datos:

- User: Representa la información básica del usuario
- LoginRequest: Estructura para solicitudes de inicio de sesión
- LoginResponse: Maneja la respuesta de autenticación
- RegisterRequest: Formato para solicitudes de registro

Gestión de Imágenes de Perfil

Funcionalidad de Imágenes

La clase ImageUtils proporciona métodos para:

- Cargar imágenes de perfil
- Redimensionar y comprimir imágenes
- Gestionar almacenamiento local y remoto de imágenes

Capturas de Pantalla

Inicio de Sesión

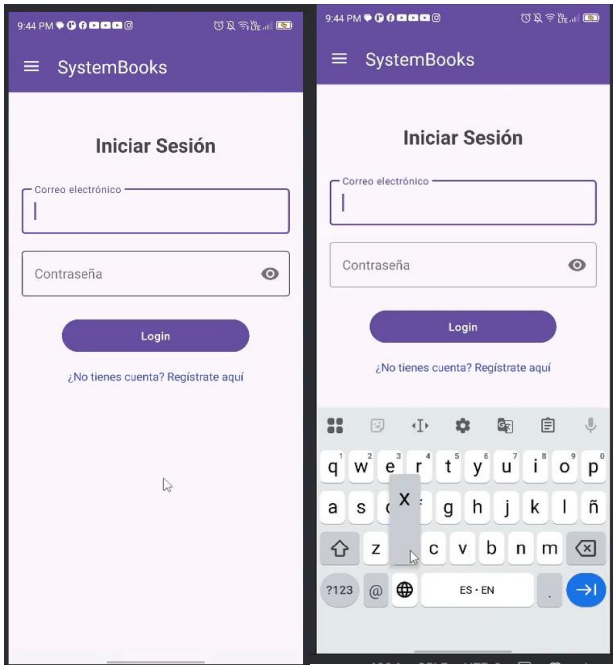

Figura 5 Pantalla de Inicio de Sesión.

Registro

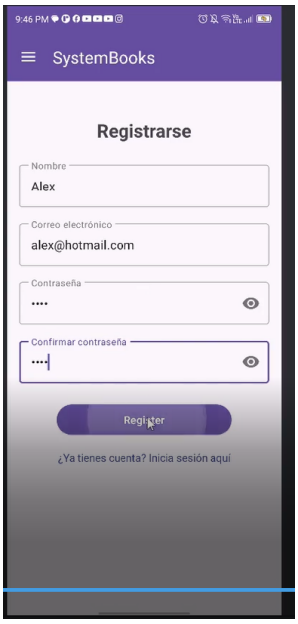

Figura 6 Pantalla de Registro

Perfil de Usuario

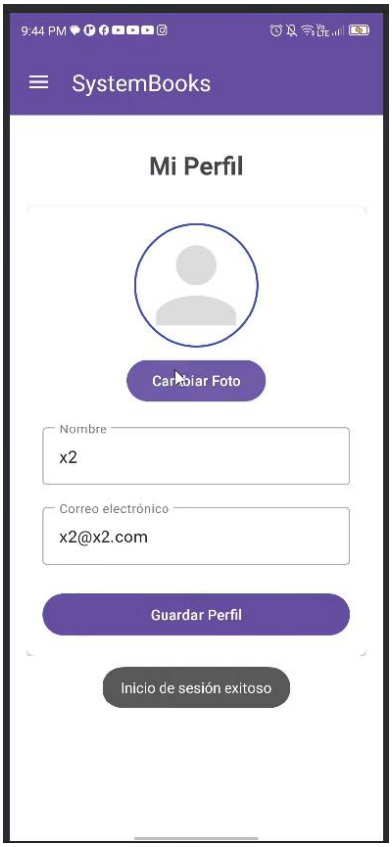

Figura 7 Perfil de Usuario

Conclusiones

El desarrollo de la aplicación SystemBooks permitió explorar y aplicar conceptos de seguridad, arquitectura de software y diseño de interfaces móviles.

La implementación del sistema de autenticación con roles fue uno de los mayores retos del proyecto. Diseñar un mecanismo que garantizara la seguridad de las credenciales de usuario, implementara un control de acceso y mantuviera la integridad de la información requirió un analizar prácticas de seguridad.

La integración entre la aplicación móvil Android y el servicio REST desarrollado en Spring Boot se logró los entre sistemas. La implementación de Retrofit como cliente HTTP simplificó enormemente la gestión de las solicitudes y respuestas entre el cliente móvil y el servidor.

El manejo de sesiones seguras y la utilización de JSON Web Tokens (JWT) proporcionó un gran mecanismo para autenticar y autorizar a los usuarios de manera segura.

El diseño de interfaces siguiendo los principios de Material Design permitió crear una experiencia de usuario intuitiva y consistente. La modularidad de los fragmentos facilitó la navegación y la gestión de diferentes estados de la aplicación.

La implementación del control de acceso basado en roles demostró la complejidad de los sistemas de gestión de permisos. Diferenciar y restringir las funcionalidades entre usuarios administradores y usuarios regulares requirió una lógica de programación precisa y bien estructurada.

El uso de tecnologías como Docker para la contenerización del servicio REST demostró la importancia de las prácticas modernas de desarrollo de software, facilitando la portabilidad y el despliegue de la aplicación.

Este proyecto nos permitió ver los conocimientos teóricos en práctica, bridando una comprensión más profunda de los principios de desarrollo de aplicaciones móviles nativas, seguridad informática y arquitectura de software.

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/