Universidad del País Vasco
Euskal Herriko Unibertsitatea

Ingeniería Informática de Gestión y Sistemas de Información

Desarrollo Avanzado de Software

LibreBook

Estudiante:
Gabiña Barañano, Xabier

15 de marzo de 2025

Índice general

Índice de figuras

Índice de cuadros

Índice de códigos

1. Introducción

LibreBook es una aplicación móvil para Android diseñada para los amantes de la lectura. Es una plataforma que permite a los usuarios crear una biblioteca personal digital donde pueden registrar, organizar y seguir su progreso en los libros que están leyendo o desean leer.

La aplicación está desarrollada completamente en Java y sigue las mejores prácticas de desarrollo para Android, incluyendo el uso de Room para la persistencia de datos, arquitectura MVVM, y componentes de la biblioteca de Material Design para ofrecer una interfaz moderna y funcional.
El repositorio de la aplicación se encuentra en GitHub y está disponible para su descarga y uso bajo la licencia MIT.

Repositorio de la aplicación

Además en el repositorio, tambien esta disponible en GitHub el binario de la aplicación en formato apk para su descarga e instalación en dispositivos Android.

Archivo APK de la aplicación

2. Objetivos

2.1. Elementos obligatorios

- **Uso de ListView+CardView personalizado o de RecyclerView+CardView para mostrar listados de elementos con diferentes características.**
  - En la actividad principal (MainActivity.java) se muestra una lista de cuatro libros aleatorios de la base de datos mediante RecyclerView.
  - En la actividad de búsqueda (SearchActivity.java), dependiendo de la posición del switch, se muestra una lista de libros o de usuarios usando el mismo RecyclerView.
  - En la actividad del perfil (ProfileActivity.java), se muestran tres listas horizontales de libros separadas por categoría (leyendo, por leer, leídos) mediante RecyclerView.
  - Todas estas RecyclerViews usan un adaptador personalizado (BookCardAdapter.java, UsuarioAdapter, LibroAdapter) para mostrar los CardViews con la información de cada elemento.
- **Usar una base de datos local, para listar, añadir y modificar elementos y características de cada elemento.**
  - Se implementa Room para gestionar la base de datos SQLite (AppDatabase.java).
  - Se han creado entidades como Libro.java, Usuario.java y UsuarioLibro.java con sus respectivas relaciones.
  - Los DAOs correspondientes (LibroDao.java, UsuarioDao.java, UsuarioLibroDao.java) contienen métodos para insertar, actualizar, eliminar y consultar datos.
  - Los repositorios (LibroRepository.java, UsuarioRepository.java, BibliotecaRepository.java) proporcionan una capa adicional de abstracción para el acceso a datos.
- **Uso de diálogos.**
  - En SettingsActivity.java se implementan diálogos para cambiar el tema y el idioma de la aplicación.
  - En BookDetailActivity.java se utiliza un diálogo personalizado (dialog_add_book.xml) para añadir libros a la biblioteca y elegir su estado (leído, leyendo, por leer).
  - En BaseActivity.java se muestra un diálogo de confirmación al querer cerrar sesión.
  - En ProfileActivity.java se implementa un diálogo para solicitar permisos de acceso a la galería.
  - Se utilizan diálogos de carga (dialog_loading.xml) durante las operaciones de busqueda que puedan llevar tiempo.
- **Usar notificaciones locales.**
  - En RegisterActivity.java, tras el registro exitoso de un usuario, se muestra una notificación de bienvenida.
  - La clase NotificationUtils.java implementa métodos para crear el canal de notificaciones y mostrar la notificación.
- **Control de la pila de actividades.**

- En BaseActivity.java, método handleNavigationItemSelected(), se controla la navegación entre actividades.
- Se utilizan flags como FLAG_ACTIVITY_CLEAR_TOP y FLAG_ACTIVITY_SINGLE_TOP para gestionar la pila de actividades.
- Se comprueba si ya estamos en la actividad a la que queremos ir para evitar crear instancias innecesarias.
- En los métodos setLanguageMode() y setThemeMode() se reinicia la pila de actividades para aplicar los cambios globalmente.
- En métodos como logoutUser() se controla correctamente el regreso a la actividad principal.

2.2. Elementos opcionales

- **Permitir que una misma funcionalidad se comporte de manera distinta dependiendo de la orientación (o del tamaño) del dispositivo mediante el uso de Fragments.**
  - En la actividad de detalle del libro (BookDetailActivity.java) se utilizan dos fragments: BookInfoFragment y BookActionsFragment.
  - En orientación vertical (layout activity_book_detail.xml), los fragments se muestran uno sobre otro.
  - En orientación horizontal (layout-land/activity_book_detail.xml), los fragments se muestran uno al lado del otro, aprovechando mejor el espacio horizontal.
  - Los fragments permiten una mejor separación de responsabilidades: BookInfoFragment muestra la información detallada del libro, mientras que BookActionsFragment maneja las acciones que el usuario puede realizar.
- **Hacer la aplicación multiidioma y añadir la opción de cambiar de idioma en la propia aplicación.**
  - La aplicación soporta tres idiomas: inglés (values/strings.xml), español (values-es/strings.xml) y euskera (values-eu/strings.xml).
  - En SettingsActivity.java se implementa la opción para cambiar el idioma mediante un diálogo.
  - En BaseActivity.java, los métodos loadLocale(), applyLanguage() y setLanguageMode() gestionan el cambio de idioma.
  - El idioma seleccionado se guarda en SharedPreferences y se aplica en toda la aplicación.
- **Uso de ficheros de texto.**
  - En ProfileActivity.java se implementa la funcionalidad para guardar y cargar imágenes de perfil como archivos en el almacenamiento interno.
  - En el método saveProfileImage() se guarda la imagen de perfil seleccionada como un archivo JPEG.
  - En DatabaseInitializer.java se crean archivos de imagen para los usuarios predeterminados.
  - En el método updateNavigationHeader() de BaseActivity.java se cargan las imágenes de perfil desde archivos.
- **Uso de Preferencias, para guardar las preferencias del usuario en cuanto a mostrar/esconder cierta información, elegir colores para la aplicación, o cualquier otra cosa relacionada con la visualización de la aplicación.**
  - En BaseActivity.java se guardan preferencias para el idioma y el tema seleccionados.
  - En LoginActivity.java y RegisterActivity.java se guardan datos de sesión del usuario.
  - Las preferencias permiten persistir la configuración elegida por el usuario entre sesiones.
- **Crear estilos y temas propios, para personalizar fondos, botones, etc.**

- Se han definido temas personalizados en values/themes.xml y values-night/themes.xml.
- Se han creado estilos específicos para el modo claro (Theme.LibreBook.Light) y el modo oscuro (Theme.LibreBook.Dark).
- Se han definido atributos personalizados en attrs.xml como backgroundColor, cardBackgroundColor y textColor.
- Se utilizan drawables personalizados como rounded_background.xml para dar estilo a elementos visuales.
- Los temas aplican colores consistentes a lo largo de toda la aplicación.

- **Usar intents implícitos para abrir otras aplicaciones, contactos, etc.**
  - En ProfileActivity.java se utiliza un intent implícito para abrir la galería de imágenes (ACTION_PICK).
  - En el método onRequestPermissionsResult() se utiliza un intent implícito para abrir los ajustes de la aplicación cuando se deniegan permisos.
  - El intent Intent(android.provider.Settings.ACTION_APPLICATION_DETAILS_SETTINGS) permite al usuario ir directamente a los ajustes de permisos de la aplicación.
  - En ImageLoader.java se accede implícitamente a recursos de internet para cargar imágenes desde URLs.

- **Añadir una barra de herramientas (ToolBar) personalizada en la aplicación así como un panel de navegación (Navigation Drawer)**
  - En BaseActivity.java se implementa un Toolbar personalizado que se hereda en todas las actividades.
  - El método setupToolbar() configura la barra de herramientas con el título específico de cada actividad.
  - Se implementa un Navigation Drawer (panel lateral de navegación) en el método setupDrawer().
  - El Navigation Drawer incluye un encabezado personalizado (nav_header_main.xml) que muestra información del usuario.
  - El menú del Navigation Drawer (drawer_menu.xml) incluye opciones diferentes según el estado de autenticación del usuario.

3. Descripción de la aplicación

3.1. Clases

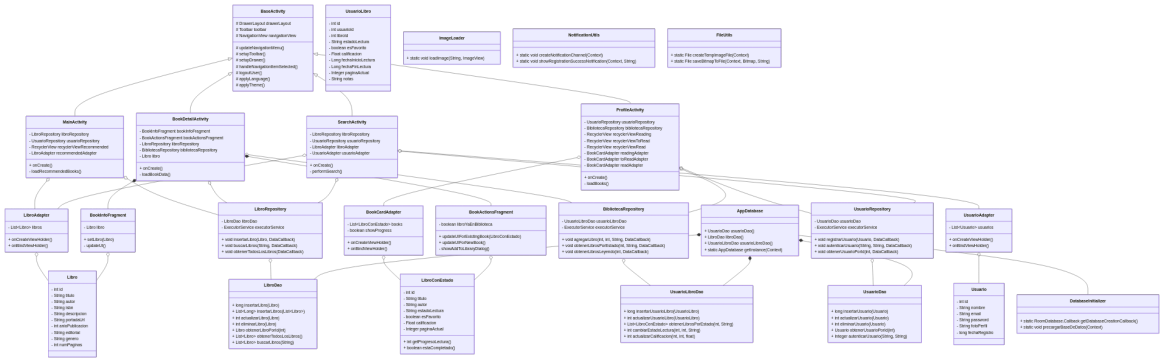

Figura 3.1: Diagrama de clases de la aplicación LibreBook

La aplicación está estructurada siguiendo un patrón arquitectónico basado en capas con una clara separación de responsabilidades:

3.1.1. Activities

- **BaseActivity**: Clase abstracta que implementa la funcionalidad común a todas las actividades, como la barra de herramientas, el menú lateral, la gestión de idiomas y temas, y el control de sesión.
- **MainActivity**: Actividad principal que muestra la pantalla de inicio con libros recomendados y mensajes de bienvenida personalizados.
- **LoginActivity**: Gestiona la autenticación de usuarios existentes, validando credenciales y estableciendo la sesión.
- **RegisterActivity**: Maneja el registro de nuevos usuarios, incluyendo validación de datos y creación de cuentas.
- **BookDetailActivity**: Muestra los detalles de un libro específico mediante dos fragments y permite realizar acciones como añadir a la biblioteca o valorar.
- **ProfileActivity**: Presenta el perfil del usuario con sus estadísticas de lectura y bibliotecas organizadas por estado (leyendo, por leer, leídos).
- **SearchActivity**: Permite buscar libros o usuarios mediante un sistema de filtrado en tiempo real.
- **SettingsActivity**: Ofrece opciones para personalizar la aplicación, como cambiar el idioma o el tema.

3.1.2. Fragments

- **BookInfoFragment**: Muestra la información detallada de un libro (portada, título, autor, descripción, etc.).

- **BookActionsFragment**: Presenta las acciones que el usuario puede realizar con un libro (añadir a biblioteca, calificar, revisar, actualizar estado).

3.1.3. Adaptadores

- **LibroAdapter**: Adaptador para mostrar listas de libros en formato vertical usando CardView.
- **BookCardAdapter**: Adaptador especializado para mostrar libros en formato horizontal con información adicional sobre progreso de lectura.
- **UsuarioAdapter**: Adaptador para mostrar listas de usuarios con su información básica.

3.1.4. Modelos y Entidades

- **Libro**: Entidad que representa un libro con sus atributos (título, autor, ISBN, descripción, etc.).
- **Usuario**: Entidad que representa un usuario de la aplicación (nombre, email, contraseña, foto de perfil).
- **UsuarioLibro**: Entidad de relación entre usuarios y libros que almacena estado de lectura, calificaciones, notas, etc.
- **LibroConEstado**: Clase POJO (Plain Old Java Object) que combina información de un libro y su estado en la biblioteca de un usuario.

3.1.5. Data Access Objects (DAOs)

- **LibroDao**: Interfaz que define operaciones de acceso a datos para la entidad Libro.
- **UsuarioDao**: Interfaz para operaciones de acceso a datos relacionadas con usuarios.
- **UsuarioLibroDao**: Interfaz para operaciones de acceso a datos de la relación usuario-libro.

3.1.6. Repositorios

- **LibroRepository**: Encapsula la lógica de acceso a datos para libros, proporcionando un API limpia a las actividades.
- **UsuarioRepository**: Gestiona el acceso a datos para usuarios, incluyendo operaciones como registro y autenticación.
- **BibliotecaRepository**: Maneja las operaciones relacionadas con la biblioteca personal de un usuario.

3.1.7. Utilidades

- **ImageLoader**: Utilidad para cargar imágenes desde URLs (portadas de libros).
- **FileUtils**: Utilidad para operaciones con archivos, especialmente para gestionar imágenes de perfil.

- **NotificationUtils**: Gestiona la creación y visualización de notificaciones locales.
- **DatabaseInitializer**: Responsable de precargar datos iniciales en la base de datos.

3.2. Base de datos

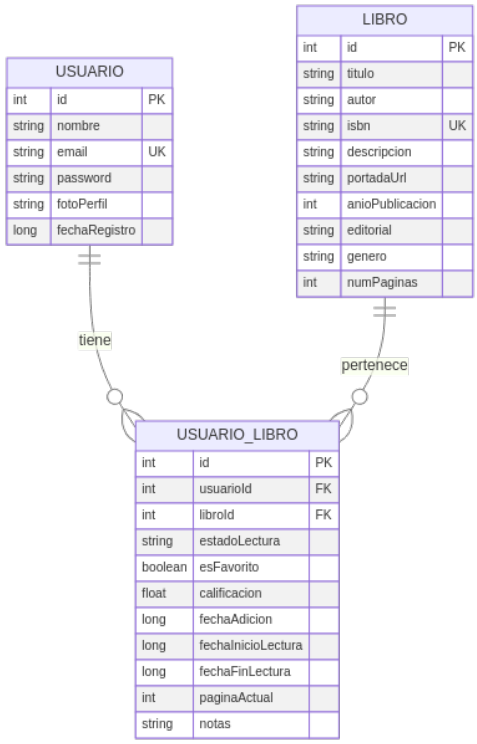

Figura 3.2: Diagrama de la base de datos de la aplicación LibreBook

La aplicación utiliza Room como una capa de abstracción sobre SQLite para proporcionar un acceso más sencillo y seguro a los datos.

3.2.1. Estructura de la base de datos

La base de datos está definida por la clase **AppDatabase**, que extiende de RoomDatabase, y contiene las siguientes entidades:

- **Entidad Libro**: Almacena información completa sobre libros.
  - **Atributos principales**: id, titulo, autor, isbn, descripcion, portadaUrl, anioPublicacion, editorial, genero, numPaginas.
  - **Índices**: ISBN es un campo único indexado para búsquedas eficientes.
- **Entidad Usuario**: Almacena datos de los usuarios registrados.
  - **Atributos principales**: id, nombre, email, password, fotoPerfil, fechaRegistro.
  - **Índices**: email es un campo único indexado para prevenir duplicados y facilitar búsquedas.
- **Entidad UsuarioLibro**: Tabla de relación muchos a muchos que conecta usuarios con libros y almacena metadatos de la relación.
  - **Atributos principales**: id, usuarioId, libroId, estadoLectura, esFavorito, calificacion, fechaAdicion, fechaInicioLectura, fechaFinLectura, paginaActual, notas.

- **Claves foráneas**: usuarioId referencia a Usuario.id, libroId referencia a Libro.id.
- **Índices**: Índice compuesto sobre (usuarioId, libroId) para asegurar unicidad y optimizar consultas.

3.2.2. Operaciones principales

La base de datos soporta las siguientes operaciones a través de los DAOs:

- **Operaciones CRUD básicas**:
  - Inserción, actualización y eliminación de libros, usuarios y relaciones.
  - Consultas por identificador, ISBN, email, etc.
  - Listados completos de entidades.
- **Consultas específicas**:
  - Búsqueda de libros por título o autor (mediante LIKE).
  - Filtrado por género o autor.
  - Obtención de listas de géneros y autores únicos.
  - Autenticación de usuarios (validación de email y contraseña).
  - Búsqueda de usuarios por nombre o email.
- **Consultas relacionales**:
  - Obtención de todos los libros de un usuario, opcionalmente filtrados por estado de lectura.
  - Obtención de libros favoritos.
  - Cálculo de estadísticas de lectura (conteo por estado, calificación promedio).

3.2.3. Inicialización y precarga

La clase **DatabaseInitializer** se encarga de:

- Crear la base de datos si no existe.
- Precargar 10 libros de Dostoievski para tener datos de ejemplo.
- Crear usuarios predeterminados (Administrador y Xabier).
- Establecer relaciones iniciales entre el usuario Xabier y algunos libros con diferentes estados de lectura, calificaciones y notas.

3.2.4. Transacciones y concurrencia

- Todas las operaciones de base de datos se realizan en hilos secundarios mediante ExecutorService para no bloquear la interfaz de usuario.
- Los callbacks permiten notificar a la UI cuando las operaciones se completan.
- Room maneja automáticamente las transacciones para asegurar la integridad de los datos.

4. Manual de usuario

4.1. Inicio

4.2. Registrarse

4.3. Iniciar sesión

4.4. Ver tu biblioteca

4.5. Buscar libros o usuarios

4.6. Añadir libros a tu biblioteca

4.7. Ajustes

5. Dificultades

6. Conclusiones