Documentación del Proyecto: Buscador Inteligente De Libros Descripción General Aplicación Android que permite buscar libros por título, autor o año. Al encontrar una coincidencia, muestra los detalles y la portada del libro en una segunda pantalla. El proyecto sigue el patrón MVVM y utiliza View Binding para una gestión segura de las vistas.

Estructura de Archivos y Módulos

1. MainActivity Archivo: MainActivity.java Función: Pantalla principal de búsqueda. Usa View Binding (ActivityMainBinding) para acceder a los elementos de la interfaz. Instancia el ViewModel (MainViewModelActivity) para la lógica de búsqueda. Al pulsar el botón "Buscar", toma el texto del campo, consulta el ViewModel y navega a ResultadoActivity pasando el libro encontrado. Al volver de la pantalla de resultados, limpia el campo de búsqueda.
2. MainViewModelActivity Archivo: MainViewModelActivity.java Función: ViewModel que gestiona la lista de libros y la lógica de búsqueda. Contiene una lista de objetos Libros con sus datos y la referencia a la imagen de portada (recurso drawable). El método buscarLibro busca coincidencias parciales (mínimo 4 caracteres) en título, autor o año.
3. Libros Archivo: Book/Libros.java Función: Modelo de datos para los libros. Atributos: título, autor, año y el identificador del recurso de la portada (portadaResId). Implementa Serializable para enviar objetos entre Activities. Proporciona constructores y getters.
4. ResultadoActivity Archivo: ResultadoActivity.java Función: Pantalla de resultados que muestra los detalles del libro encontrado. Usa View Binding (ActivityResultadoBinding) para acceder a los elementos de la interfaz. Recibe el objeto Libros por Intent y muestra el título, autor, año y la imagen de portada. Si no se encuentra el libro, muestra un mensaje de error y oculta la portada.

Layouts activity_main.xml TextView con el título de la app. EditText para ingresar el texto de búsqueda. Button para iniciar la búsqueda, con color verde, tamaño grande y borde redondeado. activity_resultado.xml TextView para mostrar "Libro Encontrado:". TextView para mostrar título, autor y año. ImageView grande y centrado para mostrar la portada del libro.

Recursos Drawable Las portadas de los libros deben estar en la carpeta res/drawable con nombres en minúsculas y sin espacios ni tildes (ejemplo: clean_code.jpg). El botón "Buscar" utiliza un fondo personalizado (btn_green_rounded.xml) para el borde redondeado.

View Binding ¿Qué es? View Binding permite acceder a los elementos de la interfaz de usuario de forma segura y eficiente, evitando el uso de findViewById. ¿Cómo funciona en el proyecto? En cada Activity se instancia el objeto de binding correspondiente (ActivityMainBinding o ActivityResultadoBinding). Se accede a los elementos de la vista directamente como propiedades del objeto binding (por ejemplo, binding.etCampoBusqueda).

Flujo de la Aplicación El usuario ingresa al menos 4 caracteres en el campo de búsqueda y pulsa "Buscar". MainActivity consulta al ViewModel por coincidencias parciales en la lista de libros. Si se encuentra un libro, se navega a ResultadoActivity mostrando los detalles y la portada. Si no se encuentra, se muestra un mensaje de error. Al volver a la pantalla principal, el campo de búsqueda se limpia automáticamente.

Buenas Prácticas Aplicadas MVVM: Separación clara entre lógica de negocio y la interfaz. View Binding: Acceso seguro y eficiente a las vistas. Recursos organizados: Imágenes en drawable, layouts claros y personalizados. Código limpio y mantenible: Métodos bien definidos, uso de constructores y getters, manejo de errores.

Documentación Detallada: Buscador Inteligente De Libros

Descripción General Esta aplicación Android permite buscar libros por título, autor o año. Al encontrar una coincidencia, muestra los detalles y la portada del libro en una segunda pantalla. El proyecto sigue el patrón MVVM y utiliza View Binding para una gestión segura y eficiente de las vistas.

Estructura de Archivos y Módulos

1. MainActivity Archivo: MainActivity.java Función: Pantalla principal de búsqueda. Usa View Binding (ActivityMainBinding) para acceder a los elementos de la interfaz. Instancia el ViewModel (MainViewModelActivity) para la lógica de búsqueda. Al pulsar el botón "Buscar", toma el texto del campo, consulta el ViewModel y navega a ResultadoActivity pasando el libro encontrado. Al volver de la pantalla de resultados, limpia el campo de búsqueda automáticamente. Fragmento de código relevante: binding.btnBusqueda.setOnClickListener(v -> { String busqueda = binding.etCampoBusqueda.getText().toString().trim(); Libros libro = viewModel.buscarLibro(busqueda, busqueda, busqueda); Intent intent = new Intent(MainActivity.this, ResultadoActivity.class); intent.putExtra("libro", libro); startActivity(intent); });

2. MainViewModelActivity Archivo: MainViewModelActivity.java Función: ViewModel que gestiona la lista de libros y la lógica de búsqueda. Contiene una lista de objetos Libros con sus datos y la referencia a la imagen de portada (recurso drawable). El método buscarLibro busca coincidencias parciales (mínimo 4 caracteres) en título, autor o año. Fragmento de código relevante: public Libros buscarLibro(String titulo, String autor, String anio) { for (Libros libro : librosList) {

   ```
   if (titulo != null && titulo.length() >= 4 &&
           libro.getTitulo().toLowerCase().contains(titulo.toLowerCase())) {
       return libro;
   }
   if (autor != null && autor.length() >= 4 &&
           libro.getAutor().toLowerCase().contains(autor.toLowerCase())) {
       return libro;
   }
   if (anio != null && anio.length() >= 4 &&
           libro.getAnio().toLowerCase().contains(anio.toLowerCase())) {
       return libro;
   }
   ```

   } return null; }

3. Libros Archivo: Book/Libros.java Función: Modelo de datos para los libros. Atributos: título, autor, año y el identificador del recurso de la portada (portadaResId). Implementa Serializable para enviar objetos entre Activities. Proporciona constructores y getters. Fragmento de código relevante: public class Libros implements Serializable { private String titulo; private String

autor; private String anio; private int portadaResId; // ...constructores y getters... }

4. ResultadoActivity Archivo: ResultadoActivity.java Función: Pantalla de resultados que muestra los detalles del libro encontrado. Usa View Binding (ActivityResultadoBinding) para acceder a los elementos de la interfaz. Recibe el objeto Libros por Intent y muestra el título, autor, año y la imagen de portada. Si no se encuentra el libro, muestra un mensaje de error y oculta la portada. Fragmento de código relevante: Libros libro = (Libros) getIntent().getSerializableExtra("libro"); if (libro != null) { binding.tvTitulo.setText("Título: " + libro.getTitulo()); binding.tvAutor.setText("Autor: " + libro.getAutor()); binding.tvAnio.setText("Año: " + libro.getAnio()); if (libro.getPortadaResId() != 0) {

   ```
   binding.ivPortada.setImageResource(libro.getPortadaResId());
   ```

   } else {

   ```
   binding.ivPortada.setImageResource(android.R.color.transparent);
   ```

   } } else { binding.tvTitulo.setText("Libro no encontrado"); binding.tvAutor.setText(""); binding.tvAnio.setText(""); binding.ivPortada.setImageResource(android.R.color.transparent); }

Layouts activity_main.xml TextView con el título de la app. EditText para ingresar el texto de búsqueda. Button para iniciar la búsqueda, con color verde, tamaño grande y borde redondeado. activity_resultado.xml TextView para mostrar "Libro Encontrado:". TextView para mostrar título, autor y año. ImageView grande y centrado para mostrar la portada del libro.

Recursos Drawable Las portadas de los libros deben estar en la carpeta res/drawable con nombres en minúsculas y sin espacios ni tildes (ejemplo: clean_code.jpg).

View Binding ¿Qué es? View Binding permite acceder a los elementos de la interfaz de usuario de forma segura y eficiente, evitando el uso de findViewById. ¿Cómo funciona en el proyecto? En cada Activity se instancia el objeto de binding correspondiente (ActivityMainBinding o ActivityResultadoBinding). Se accede a los elementos de la vista directamente como propiedades del objeto binding (por ejemplo, binding.etCampoBusqueda). Esto mejora la legibilidad, seguridad y mantenimiento del código.

Flujo de la Aplicación El usuario ingresa al menos 4 caracteres en el campo de búsqueda y pulsa "Buscar". MainActivity consulta al ViewModel por coincidencias parciales en la lista de libros. Si se encuentra un libro, se navega a ResultadoActivity mostrando los detalles y la portada. Si no se encuentra, se muestra un mensaje de error. Al volver a la pantalla principal, el campo de búsqueda se limpia automáticamente.

Buenas Prácticas Aplicadas MVVM: Separación clara entre lógica de negocio y la interfaz. View Binding: Acceso seguro y eficiente a las vistas. Recursos organizados: Imágenes en drawable, layouts claros y personalizados. Código limpio y mantenible: Métodos bien definidos, uso de constructores y getters, manejo de errores.

Recomendaciones Mantén los nombres de las imágenes en drawable en minúsculas y sin espacios/tildes. Si agregas más libros, incluye la portada y actualiza el ViewModel. Personaliza colores, tamaños y estilos en los layouts según tus preferencias.

Diagrama de Flujo (Texto) [MainActivity] | |--(Usuario ingresa búsqueda y pulsa "Buscar") | [MainViewModelActivity] | |--(Busca coincidencia parcial en la lista de libros) | [ResultadoActivity] | |--(Muestra detalles y portada si existe, mensaje de error si no) | [MainActivity] | |--(Campo de búsqueda se limpia al volver)

Ejemplo de Uso Abre la app. Escribe al menos 4 caracteres del título, autor o año de un libro. Pulsa "Buscar". Si existe el libro, verás sus datos y portada en la siguiente pantalla. Pulsa atrás para volver y realizar otra búsqueda.