|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №5**

**«Создание и использование собственных источников данных.**

**Работа со стандартными источниками данных»**

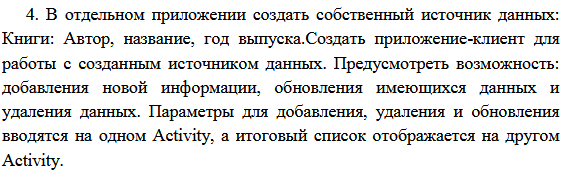
**ДИСЦИПЛИНА: «Разработка мобильного ПО»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-52Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_Боков А. А.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_Прудяк П.Н.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2024 г. | | |

**Цель:** формирование навыков создания собственных источников данных.

**Задачи**

1. Научиться создавать собственные источники данных (ContentProvider).
2. Научиться использовать собственные источники данных.
3. Научиться работать со стандартными источниками данных: аудио файлами, графическими файлами, списком контактов.
4. Разработать эффективные приложения с учетом аппаратных ограничений мобильных устройств.
5. Научиться реализовывать логику работы приложения с учетом специфики платформы Android



**Листинг программы:**

**MainActivity.java(client):**

package com.example.android\_dev\_lab5\_client;  
  
import android.content.ContentValues;  
import android.content.Intent;  
import android.database.Cursor;  
import android.net.Uri;  
import android.os.Bundle;  
import android.text.TextUtils;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainActivity extends AppCompatActivity {  
  
 private static final Uri *BOOKS\_URI* = Uri.*parse*("content://com.example.bookprovider/books");  
  
 private EditText inputAuthor, inputTitle, inputYear, inputId;  
 private TextView outputText;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 // Инициализация UI-компонентов  
 inputAuthor = findViewById(R.id.input\_author);  
 inputTitle = findViewById(R.id.input\_title);  
 inputYear = findViewById(R.id.input\_year);  
 inputId = findViewById(R.id.input\_id);  
 outputText = findViewById(R.id.output\_text);  
  
 Button buttonAdd = findViewById(R.id.button\_add);  
 Button buttonRead = findViewById(R.id.button\_read);  
 Button buttonUpdate = findViewById(R.id.button\_update);  
 Button buttonDelete = findViewById(R.id.button\_delete);  
  
 // Обработчики кнопок  
 buttonAdd.setOnClickListener(view -> addBook());  
 buttonRead.setOnClickListener(view -> readBooks());  
 buttonUpdate.setOnClickListener(view -> updateBook());  
 buttonDelete.setOnClickListener(view -> deleteBook());  
 }  
  
 // Добавление книги  
 private void addBook() {  
 String author = inputAuthor.getText().toString().trim();  
 String title = inputTitle.getText().toString().trim();  
 String year = inputYear.getText().toString().trim();  
  
 if (TextUtils.*isEmpty*(author) || TextUtils.*isEmpty*(title) || TextUtils.*isEmpty*(year)) {  
 outputText.setText("Пожалуйста, заполните все поля!");  
 return;  
 }  
  
 ContentValues values = new ContentValues();  
 values.put("author", author);  
 values.put("title", title);  
 values.put("year", year);  
  
 Uri newUri = getContentResolver().insert(*BOOKS\_URI*, values);  
 outputText.setText("Книга добавлена: " + newUri);  
 }  
  
  
 private void readBooks() {  
 Cursor cursor = getContentResolver().query(*BOOKS\_URI*, null, null, null, null);  
 StringBuilder result = new StringBuilder("Книги в базе данных:\n");  
  
 if (cursor != null) {  
 try {  
 int idIndex = cursor.getColumnIndexOrThrow("\_id");  
 int authorIndex = cursor.getColumnIndexOrThrow("author");  
 int titleIndex = cursor.getColumnIndexOrThrow("title");  
 int yearIndex = cursor.getColumnIndexOrThrow("year");  
  
 while (cursor.moveToNext()) {  
 int id = cursor.getInt(idIndex);  
 String author = cursor.getString(authorIndex);  
 String title = cursor.getString(titleIndex);  
 int year = cursor.getInt(yearIndex);  
  
 result.append("ID: ").append(id)  
 .append(", Автор: ").append(author)  
 .append(", Название: ").append(title)  
 .append(", Год: ").append(year)  
 .append("\n");  
 }  
 } catch (IllegalArgumentException e) {  
 result.append("Ошибка: ").append(e.getMessage());  
 } finally {  
 cursor.close();  
 }  
 } else {  
 result.append("Данные не найдены.");  
 }  
  
 // Переход в BookListActivity с передачей данных  
 Intent intent = new Intent(MainActivity.this, BookListActivity.class);  
 intent.putExtra("BOOK\_DATA", result.toString());  
 startActivity(intent);  
 }  
  
  
  
  
 // Чтение всех книг  
// private void readBooks() {  
// Cursor cursor = getContentResolver().query(BOOKS\_URI, null, null, null, null);  
// if (cursor == null) {  
// outputText.setText("Не удалось получить данные.");  
// return;  
// }  
//  
// StringBuilder result = new StringBuilder("Книги в базе данных:\n");  
// while (cursor.moveToNext()) {  
// int id = cursor.getInt(cursor.getColumnIndex("\_id"));  
// String author = cursor.getString(cursor.getColumnIndex("author"));  
// String title = cursor.getString(cursor.getColumnIndex("title"));  
// int year = cursor.getInt(cursor.getColumnIndex("year"));  
//  
// result.append("ID: ").append(id)  
// .append(", Автор: ").append(author)  
// .append(", Название: ").append(title)  
// .append(", Год: ").append(year)  
// .append("\n");  
// }  
// cursor.close();  
//  
// outputText.setText(result.toString());  
// }  
  
  
  
 // Обновление книги  
 private void updateBook() {  
 String id = inputId.getText().toString().trim();  
 String author = inputAuthor.getText().toString().trim();  
 String title = inputTitle.getText().toString().trim();  
 String year = inputYear.getText().toString().trim();  
  
 if (TextUtils.*isEmpty*(id)) {  
 outputText.setText("Пожалуйста, введите ID для обновления!");  
 return;  
 }  
  
 ContentValues values = new ContentValues();  
 if (!TextUtils.*isEmpty*(author)) values.put("author", author);  
 if (!TextUtils.*isEmpty*(title)) values.put("title", title);  
 if (!TextUtils.*isEmpty*(year)) values.put("year", year);  
  
 int rowsUpdated = getContentResolver().update(  
 Uri.*withAppendedPath*(*BOOKS\_URI*, id), values, null, null);  
 outputText.setText("Обновлено записей: " + rowsUpdated);  
 }  
  
 // Удаление книги  
 private void deleteBook() {  
 String id = inputId.getText().toString().trim();  
 if (TextUtils.*isEmpty*(id)) {  
 outputText.setText("Пожалуйста, введите ID для удаления!");  
 return;  
 }  
  
 int rowsDeleted = getContentResolver().delete(  
 Uri.*withAppendedPath*(*BOOKS\_URI*, id), null, null);  
 outputText.setText("Удалено записей: " + rowsDeleted);  
 }  
}

**BooksListActivity.java:**

package com.example.android\_dev\_lab5\_client;  
import android.os.Bundle;  
import android.widget.TextView;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class BookListActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_book\_list*);  
  
 TextView bookListTextView = findViewById(R.id.*book\_list\_text*);  
  
 // Получение данных из Intent  
 String bookData = getIntent().getStringExtra("BOOK\_DATA");  
  
 // Установка текста в TextView  
 bookListTextView.setText(bookData != null ? bookData : "Нет данных для отображения");  
 }  
}

**activity\_main.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <!-- Поля для ввода данных -->  
 <EditText  
 android:id="@+id/input\_author"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Автор"  
 android:inputType="text" />  
  
 <EditText  
 android:id="@+id/input\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название"  
 android:inputType="text" />  
  
 <EditText  
 android:id="@+id/input\_year"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Год"  
 android:inputType="number" />  
  
 <EditText  
 android:id="@+id/input\_id"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="ID (только для обновления или удаления)"  
 android:inputType="number" />  
  
 <!-- Кнопки для выполнения операций -->  
 <Button  
 android:id="@+id/button\_add"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Добавить книгу" />  
  
 <Button  
 android:id="@+id/button\_read"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Показать все книги" />  
  
 <Button  
 android:id="@+id/button\_update"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Обновить книгу" />  
  
 <Button  
 android:id="@+id/button\_delete"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Удалить книгу" />  
  
 <!-- Поле для отображения данных -->  
 <TextView  
 android:id="@+id/output\_text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:background="#f4f4f4"  
 android:padding="8dp"  
 android:text="Результаты будут здесь"  
 android:textSize="16sp"  
 android:scrollbars="vertical" />  
</LinearLayout>

**activity\_book\_list.xml**

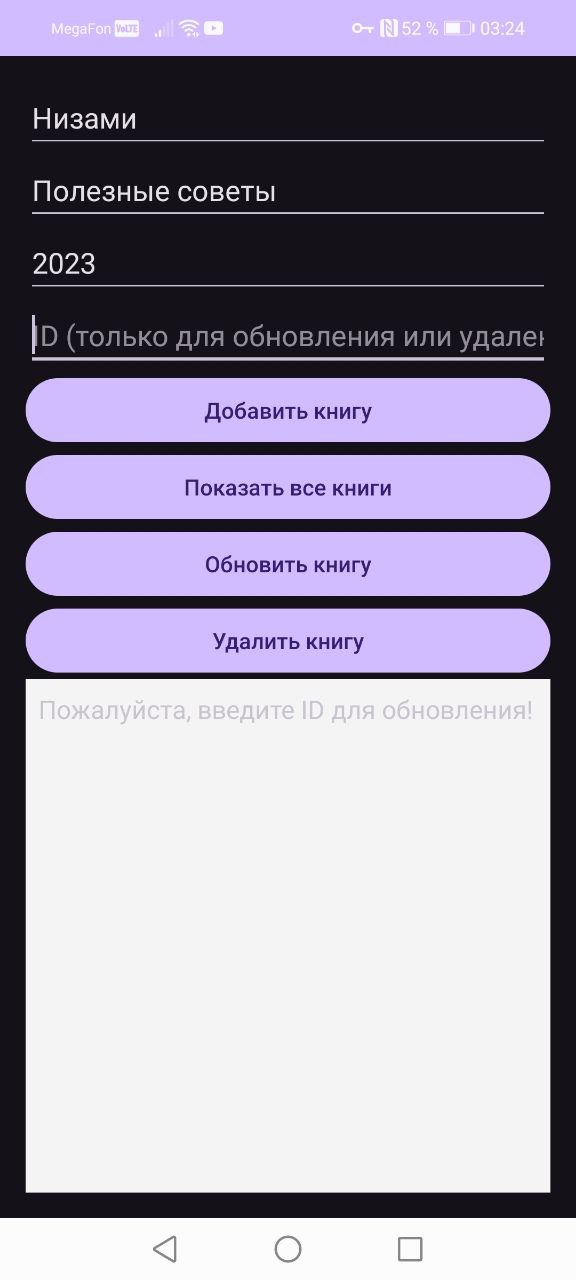
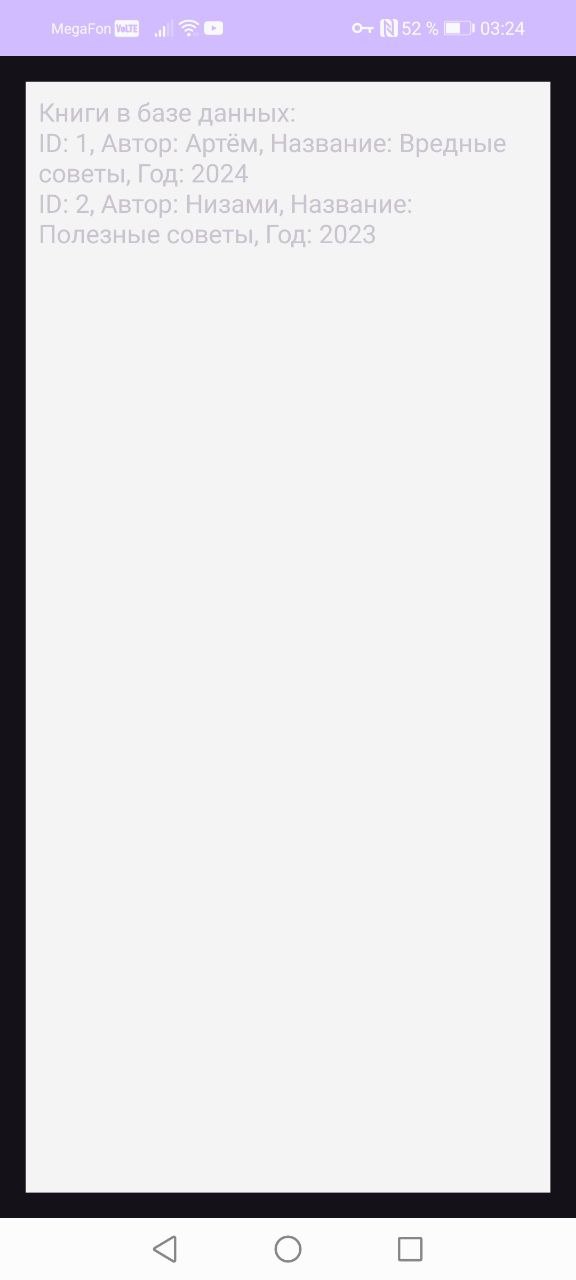
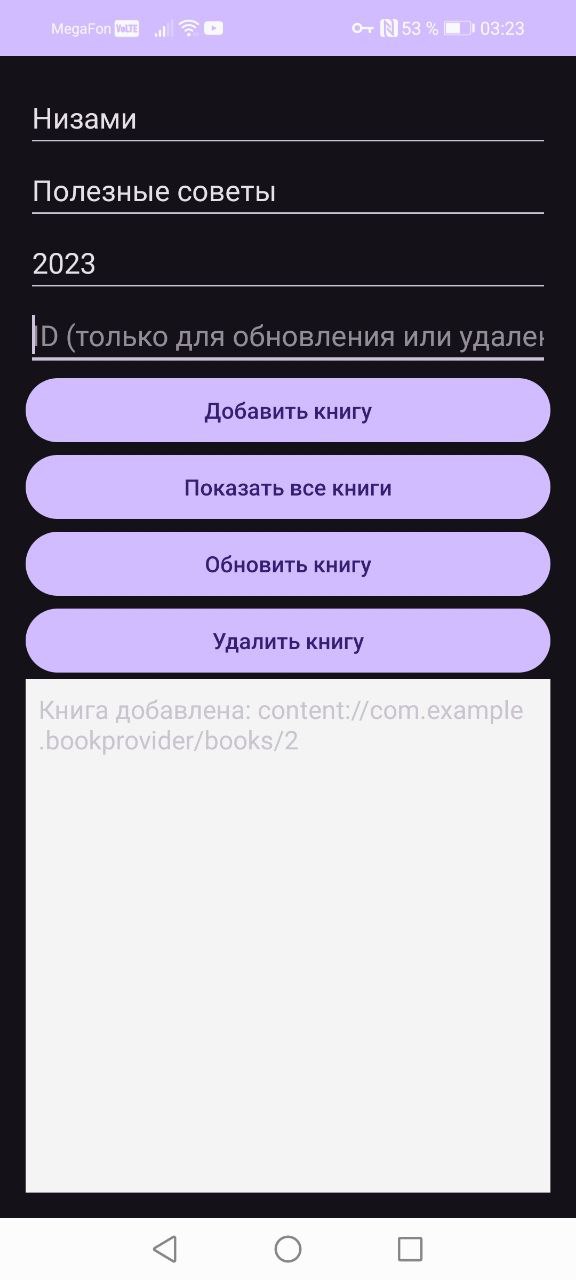
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/book\_list\_text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#f4f4f4"  
 android:padding="8dp"  
 android:textSize="16sp"  
 android:scrollbars="vertical" />  
</LinearLayout>

**AndroidManifest.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Android\_dev\_lab5\_client"  
 tools:targetApi="31">  
 <activity android:name=".BookListActivity" />  
  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**BookProvider.java**

package com.example.android\_dev\_lab5;  
import android.content.ContentProvider;  
import android.content.ContentUris;  
import android.content.ContentValues;  
import android.content.Context;  
import android.content.UriMatcher;  
import android.database.Cursor;  
import android.database.SQLException;  
import android.database.sqlite.SQLiteDatabase;  
import android.net.Uri;  
  
public class BookProvider extends ContentProvider {  
  
 private static final String *AUTHORITY* = "com.example.bookprovider";  
 private static final String *PATH\_BOOKS* = "books";  
 public static final Uri *CONTENT\_URI* = Uri.*parse*("content://" + *AUTHORITY* + "/" + *PATH\_BOOKS*);  
  
 private static final int *BOOKS* = 1;  
 private static final int *BOOK\_ID* = 2;  
  
 private static final UriMatcher *uriMatcher* = new UriMatcher(UriMatcher.*NO\_MATCH*);  
  
 static {  
 *uriMatcher*.addURI(*AUTHORITY*, *PATH\_BOOKS*, *BOOKS*);  
 *uriMatcher*.addURI(*AUTHORITY*, *PATH\_BOOKS* + "/#", *BOOK\_ID*);  
 }  
  
 private DatabaseHelper dbHelper;  
  
 @Override  
 public boolean onCreate() {  
 dbHelper = new DatabaseHelper(getContext());  
 return true;  
 }  
  
 @Override  
 public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor;  
  
 switch (*uriMatcher*.match(uri)) {  
 case *BOOKS*:  
 cursor = db.query(DatabaseHelper.*TABLE\_NAME*, projection, selection, selectionArgs, null, null, sortOrder);  
 break;  
 case *BOOK\_ID*:  
 selection = DatabaseHelper.*COL\_ID* + "=?";  
 selectionArgs = new String[]{String.*valueOf*(ContentUris.*parseId*(uri))};  
 cursor = db.query(DatabaseHelper.*TABLE\_NAME*, projection, selection, selectionArgs, null, null, sortOrder);  
 break;  
 default:  
 throw new IllegalArgumentException("Unknown URI: " + uri);  
 }  
  
 cursor.setNotificationUri(getContext().getContentResolver(), uri);  
 return cursor;  
 }  
  
 @Override  
 public Uri insert(Uri uri, ContentValues values) {  
 if (*uriMatcher*.match(uri) != *BOOKS*) {  
 throw new IllegalArgumentException("Invalid URI for insert: " + uri);  
 }  
  
 SQLiteDatabase db = dbHelper.getWritableDatabase();  
 long id = db.insert(DatabaseHelper.*TABLE\_NAME*, null, values);  
  
 if (id > 0) {  
 Uri newUri = ContentUris.*withAppendedId*(*CONTENT\_URI*, id);  
 getContext().getContentResolver().notifyChange(newUri, null);  
 return newUri;  
 } else {  
 throw new SQLException("Failed to insert row into " + uri);  
 }  
 }  
  
 @Override  
 public int delete(Uri uri, String selection, String[] selectionArgs) {  
 SQLiteDatabase db = dbHelper.getWritableDatabase();  
 int rowsDeleted;  
  
 switch (*uriMatcher*.match(uri)) {  
 case *BOOKS*:  
 rowsDeleted = db.delete(DatabaseHelper.*TABLE\_NAME*, selection, selectionArgs);  
 break;  
 case *BOOK\_ID*:  
 selection = DatabaseHelper.*COL\_ID* + "=?";  
 selectionArgs = new String[]{String.*valueOf*(ContentUris.*parseId*(uri))};  
 rowsDeleted = db.delete(DatabaseHelper.*TABLE\_NAME*, selection, selectionArgs);  
 break;  
 default:  
 throw new IllegalArgumentException("Unknown URI: " + uri);  
 }  
  
 if (rowsDeleted > 0) {  
 getContext().getContentResolver().notifyChange(uri, null);  
 }  
  
 return rowsDeleted;  
 }  
  
 @Override  
 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {  
 SQLiteDatabase db = dbHelper.getWritableDatabase();  
 int rowsUpdated;  
  
 switch (*uriMatcher*.match(uri)) {  
 case *BOOKS*:  
 rowsUpdated = db.update(DatabaseHelper.*TABLE\_NAME*, values, selection, selectionArgs);  
 break;  
 case *BOOK\_ID*:  
 selection = DatabaseHelper.*COL\_ID* + "=?";  
 selectionArgs = new String[]{String.*valueOf*(ContentUris.*parseId*(uri))};  
 rowsUpdated = db.update(DatabaseHelper.*TABLE\_NAME*, values, selection, selectionArgs);  
 break;  
 default:  
 throw new IllegalArgumentException("Unknown URI: " + uri);  
 }  
  
 if (rowsUpdated > 0) {  
 getContext().getContentResolver().notifyChange(uri, null);  
 }  
  
 return rowsUpdated;  
 }  
  
 @Override  
 public String getType(Uri uri) {  
 switch (*uriMatcher*.match(uri)) {  
 case *BOOKS*:  
 return "vnd.android.cursor.dir/vnd." + *AUTHORITY* + "." + *PATH\_BOOKS*;  
 case *BOOK\_ID*:  
 return "vnd.android.cursor.item/vnd." + *AUTHORITY* + "." + *PATH\_BOOKS*;  
 default:  
 throw new IllegalArgumentException("Unknown URI: " + uri);  
 }  
 }  
}

**Результаты работы:**

**Вывод:** в ходе лабораторной работы было реализовано получение информации о книгах (автор, название и год) с использованием встроенного content provider и выборка треков.