|  |  |
| --- | --- |
| 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 №4**

**«Использование БД а Andoid приложениях»**

**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 практических навыков разработки приложений с использованием СУБД SQLite, списков и файлов при разработке Android-приложений с несколькими Activity.

Задачи:

1. Научиться работать с СУБД SQLite.

2. Научиться сохранять результаты выполнения запросов к базе

данных в списки, файлы и LogCat.

3. Понять особенности реализации Android-приложений с

использованием списков и СУБД SQLite

Формулировка задания:



**Листинг:**

**MainActivity.java**

package com.example.android\_dev\_lab4new;  
import static com.example.android\_dev\_lab4new.R.\*;  
  
import android.annotation.SuppressLint;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import java.io.BufferedReader;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStreamReader;  
  
public class MainActivity extends AppCompatActivity {  
  
 private DBHelper dbHelper;  
 private TextView tvResults;  
 private Button btnSort, btnGroup, btnSum, btnAvg, btnMax, btnGreaterThan, btnLessThanAvg, btnTypeGreaterThan, btnReadFromFile, btnShowBooks;  
  
  
 private void writeToFile(String data) {  
 try {  
 FileOutputStream fos = openFileOutput("books\_results.txt", *MODE\_PRIVATE*);  
 fos.write(data.getBytes());  
 fos.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private String readFromFile() {  
 StringBuilder stringBuilder = new StringBuilder();  
 try {  
 FileInputStream fis = openFileInput("books\_results.txt");  
 InputStreamReader isr = new InputStreamReader(fis, "UTF-8"); // Указываем кодировку  
 BufferedReader reader = new BufferedReader(isr);  
  
 String line;  
 while ((line = reader.readLine()) != null) {  
 stringBuilder.append(line).append("\n");  
 }  
 reader.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return stringBuilder.toString();  
 }  
  
  
  
  
 @SuppressLint("Range")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 dbHelper = new DBHelper(this);  
 tvResults = findViewById(id.*tvResults*);  
 btnSort = findViewById(R.id.*btnSort*);  
 btnGroup = findViewById(R.id.*btnGroup*);  
 btnSum = findViewById(R.id.*btnSum*);  
 btnAvg = findViewById(R.id.*btnAvg*);  
 btnMax = findViewById(R.id.*btnMax*);  
 btnGreaterThan = findViewById(R.id.*btnGreaterThan*);  
 btnLessThanAvg = findViewById(R.id.*btnLessThanAvg*);  
 btnTypeGreaterThan = findViewById(R.id.*btnTypeGreaterThan*);  
 btnReadFromFile = findViewById(R.id.*btnReadFromFile*);  
 btnShowBooks = findViewById(R.id.*btnShowBooks*);  
  
 SQLiteDatabase dbd = dbHelper.getWritableDatabase();  
 dbd.execSQL("DELETE FROM " + DBHelper.*TABLE\_BOOKS*);  
 dbd.execSQL("VACUUM"); // Это очищает пространство в базе данных  
  
  
 dbHelper.addBook("Учебник", "Издательство 1", 2023, 500, "Твердая");  
 dbHelper.addBook("Энциклопедия", "Издательство 1", 2022, 1500, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 2", 2024, 300, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 1", 2021, 450, "Мягкая");  
 dbHelper.addBook("Справочник", "Издательство 2", 2023, 600, "Твердая");  
 dbHelper.addBook("Словарь", "Издательство 1", 2022, 1000, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 3", 2022, 340, "Твердая");  
 dbHelper.addBook("Худ. литература", "Издательство 4", 2020, 700, "Мягкая");  
 dbHelper.addBook("Учебник", "Издательство 2", 2021, 500, "Мягкая");  
 dbHelper.addBook("Энциклопедия", "Издательство 4", 2020, 1200, "Твердая");  
 dbHelper.addBook("Словарь", "Издательство 1", 2024, 800, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 2", 2023, 670, "Мягкая");  
 dbHelper.addBook("Справочник", "Издательство 3", 2023, 270, "Твердая");  
 dbHelper.addBook("Худ. литература", "Издательство 1", 2022, 640, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 3", 2019, 500, "Мягкая");  
  
 // Кнопка для сортировки  
// btnSort.setOnClickListener(v -> displayCursorResults(dbHelper.sortBooksByPages()));  
  
 btnSort.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT \* FROM books ORDER BY pages", null);  
  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 result.append("Тип: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", Издательство: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", Год: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", Страницы: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", Обложка: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
 } while (cursor.moveToNext());  
  
 // Отображаем в логе  
 Log.*d*("SortBooks", result.toString());  
  
 // Записываем в файл  
 writeToFile(result.toString());  
 tvResults.setText("Вывод только в файл и в лог");  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для группировки  
// btnGroup.setOnClickListener(v -> displayCursorResults(dbHelper.groupBooksByTypeAndPublisher()));  
  
 btnGroup.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT type, publisher, COUNT(\*) FROM books GROUP BY type, publisher", null);  
  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 result.append("Тип: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", Издательство: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", Количество: ").append(cursor.getInt(cursor.getColumnIndex("COUNT(\*)")))  
 .append("\n");  
 } while (cursor.moveToNext());  
  
 // Отображаем в логе  
 Log.*d*("GroupBooks", result.toString());  
 tvResults.setText(result.toString());  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для суммы страниц  
// btnSum.setOnClickListener(v -> displayCursorResults(dbHelper.sumPages()));  
  
 btnSum.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT SUM(pages) FROM books", null);  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int sum = cursor.getInt(0);  
  
 // Выводим в лог  
 Log.*d*("SumPages", "Сумма страниц: " + sum);  
  
 // Записываем в файл  
 writeToFile("Сумма страниц: " + sum);  
 tvResults.setText("Вывод только в файл и в лог");  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для среднего значения страниц  
// btnAvg.setOnClickListener(v -> displayCursorResults(dbHelper.averagePagesByType()));  
  
 btnAvg.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT type, AVG(pages) FROM books GROUP BY type", null);  
  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 result.append("Тип: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", Среднее количество страниц: ").append(cursor.getFloat(cursor.getColumnIndex("AVG(pages)")))  
 .append("\n");  
 } while (cursor.moveToNext());  
  
 // Отображаем в логе  
 Log.*d*("AvgPages", result.toString());  
  
 // Записываем в файл  
 writeToFile(result.toString());  
  
 tvResults.setText(result.toString());  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для максимального значения страниц  
// btnMax.setOnClickListener(v -> displayCursorResults(dbHelper.maxPages()));  
  
 btnMax.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT MAX(pages) FROM books", null);  
  
 if (cursor != null && cursor.moveToFirst()) {  
 int max = cursor.getInt(0);  
  
 // Отображаем в логе  
 Log.*d*("MaxPages", "Максимальное количество страниц: " + max);  
 tvResults.setText("Вывод только в лог");  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для количества страниц больше 300  
// btnGreaterThan.setOnClickListener(v -> displayCursorResults(dbHelper.booksWithPagesGreaterThan(300)));  
  
 btnGreaterThan.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT \* FROM books WHERE pages > 300", null);  
  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 result.append("Тип: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", Издательство: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", Год: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", Страницы: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", Обложка: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
 } while (cursor.moveToNext());  
  
 // Отображаем в логе  
 Log.*d*("GreaterThan300", result.toString());  
 tvResults.setText(result.toString());  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для количества страниц меньше средней  
// btnLessThanAvg.setOnClickListener(v -> displayCursorResults(dbHelper.booksWithPagesLessThanAverage()));  
  
 btnLessThanAvg.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT \* FROM books WHERE pages < (SELECT AVG(pages) FROM books)", null);  
  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 result.append("Тип: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", Издательство: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", Год: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", Страницы: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", Обложка: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
 } while (cursor.moveToNext());  
  
 // Отображаем в логе  
 Log.*d*("LessThanAvgPages", result.toString());  
 tvResults.setText(result.toString());  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для типа книг с страницами больше 300  
// btnTypeGreaterThan.setOnClickListener(v -> displayCursorResults(dbHelper.bookTypesWithPagesGreaterThan(300)));  
  
 btnTypeGreaterThan.setOnClickListener(v -> {  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT type FROM books WHERE pages > 300", null);  
  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 result.append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*))).append("\n");  
 } while (cursor.moveToNext());  
  
 // Отображаем в логе  
 Log.*d*("TypeGreaterThan300", result.toString());  
 tvResults.setText("Вывод только в лог");  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 });  
  
  
 // Кнопка для добавления книги  
 btnReadFromFile.setOnClickListener(v -> {  
  
 tvResults.setText(readFromFile());  
 });  
  
 // Кнопка для вывода всех книг  
 btnShowBooks.setOnClickListener(v -> displayCursorResults(dbHelper.getAllBooks()));  
 }  
  
 // Метод для отображения результатов запроса  
  
 @SuppressLint("Range")  
 private void displayCursorResults(Cursor cursor) {  
 StringBuilder result = new StringBuilder();  
 if (cursor != null && cursor.moveToFirst()) {  
 try {  
 // Проверка наличия столбцов перед получением данных  
 int typeIndex = cursor.getColumnIndexOrThrow(DBHelper.*COLUMN\_TYPE*);  
 int publisherIndex = cursor.getColumnIndexOrThrow(DBHelper.*COLUMN\_PUBLISHER*);  
 int yearIndex = cursor.getColumnIndexOrThrow(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*);  
 int pagesIndex = cursor.getColumnIndexOrThrow(DBHelper.*COLUMN\_PAGES*);  
 int coverIndex = cursor.getColumnIndexOrThrow(DBHelper.*COLUMN\_COVER\_TYPE*);  
  
 do {  
 result.append("Тип: ").append(cursor.getString(typeIndex))  
 .append(", Издательство: ").append(cursor.getString(publisherIndex))  
 .append(", Год: ").append(cursor.getInt(yearIndex))  
 .append(", Страницы: ").append(cursor.getInt(pagesIndex))  
 .append(", Обложка: ").append(cursor.getString(coverIndex))  
 .append("\n");  
 } while (cursor.moveToNext());  
  
 tvResults.setText(result.toString());  
// Log.d("DbMethod", result.toString());  
//  
// // Записываем в файл  
// writeToFile(result.toString());  
  
 } catch (IllegalArgumentException e) {  
 Log.*e*("DbMethod", "One or more columns are missing: " + e.getMessage());  
 tvResults.setText("Ошибка при обработке данных");  
 }  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 }  
  
  
  
  
// @SuppressLint("Range")  
// private void displayCursorResults(Cursor cursor) {  
// StringBuilder result = new StringBuilder();  
// if (cursor != null && cursor.moveToFirst()) {  
// do {  
// result.append("Тип: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN\_TYPE)))  
// .append(", Издательство: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN\_PUBLISHER)))  
// .append(", Год: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.COLUMN\_YEAR\_OF\_PUBLICATION)))  
// .append(", Страницы: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.COLUMN\_PAGES)))  
// .append(", Обложка: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.COLUMN\_COVER\_TYPE)))  
// .append("\n");  
// } while (cursor.moveToNext());  
// tvResults.setText(result.toString());  
// Log.d("DbMethod", result.toString());  
//  
// // Записываем в файл  
// writeToFile(result.toString());  
// } else {  
// tvResults.setText("Нет данных");  
// }  
// }  
}

**DBHelper.java**

package com.example.android\_dev\_lab4new;  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
  
public class DBHelper extends SQLiteOpenHelper {  
  
 private static final String *DATABASE\_NAME* = "books.db";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 public static final String *TABLE\_BOOKS* = "books";  
 public static final String *COLUMN\_ID* = "id";  
 public static final String *COLUMN\_TYPE* = "type";  
 public static final String *COLUMN\_PUBLISHER* = "publisher";  
 public static final String *COLUMN\_YEAR\_OF\_PUBLICATION* = "year\_of\_publication";  
 public static final String *COLUMN\_PAGES* = "pages";  
 public static final String *COLUMN\_COVER\_TYPE* = "cover\_type";  
  
 public DBHelper(Context context) {  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String CREATE\_TABLE = "CREATE TABLE " + *TABLE\_BOOKS* + " ("  
 + *COLUMN\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT, "  
 + *COLUMN\_TYPE* + " TEXT, "  
 + *COLUMN\_PUBLISHER* + " TEXT, "  
 + *COLUMN\_YEAR\_OF\_PUBLICATION* + " INTEGER, "  
 + *COLUMN\_PAGES* + " INTEGER, "  
 + *COLUMN\_COVER\_TYPE* + " TEXT)";  
 db.execSQL(CREATE\_TABLE);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_BOOKS*);  
 onCreate(db);  
 }  
  
// public Cursor sortBooksByPages() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " ORDER BY " + DBHelper.COLUMN\_PAGES, null);  
// }  
//  
// public Cursor groupBooksByTypeAndPublisher() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT type, publisher, COUNT(\*) FROM " + DBHelper.TABLE\_BOOKS + " GROUP BY type, publisher", null);  
// }  
// public Cursor sumPages() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT SUM(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS, null);  
// }  
//  
// public Cursor averagePagesByType() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT type, AVG(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS + " GROUP BY type", null);  
// }  
//  
// public Cursor maxPages() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " = (SELECT MAX(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS + ")", null);  
// }  
// public Cursor booksWithPagesGreaterThan(int threshold) {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " > ?", new String[]{String.valueOf(threshold)});  
// }  
//  
//  
// public Cursor booksWithPagesLessThanAverage() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " < (SELECT AVG(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS + ")", null);  
// }  
// public Cursor bookTypesWithPagesGreaterThan(int threshold) {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT " + DBHelper.COLUMN\_TYPE + " FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " > ?", new String[]{String.valueOf(threshold)});  
// }  
  
 public void addBook(String type, String publisher, int year, int pages, String coverType) {  
 SQLiteDatabase db = getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put(DBHelper.*COLUMN\_TYPE*, type);  
 values.put(DBHelper.*COLUMN\_PUBLISHER*, publisher);  
 values.put(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*, year);  
 values.put(DBHelper.*COLUMN\_PAGES*, pages);  
 values.put(DBHelper.*COLUMN\_COVER\_TYPE*, coverType);  
 db.insert(DBHelper.*TABLE\_BOOKS*, null, values);  
 db.close();  
 }  
  
 public Cursor getAllBooks() {  
 SQLiteDatabase db = getReadableDatabase();  
 return db.rawQuery("SELECT \* FROM " + DBHelper.*TABLE\_BOOKS*, null);  
 }  
  
  
}

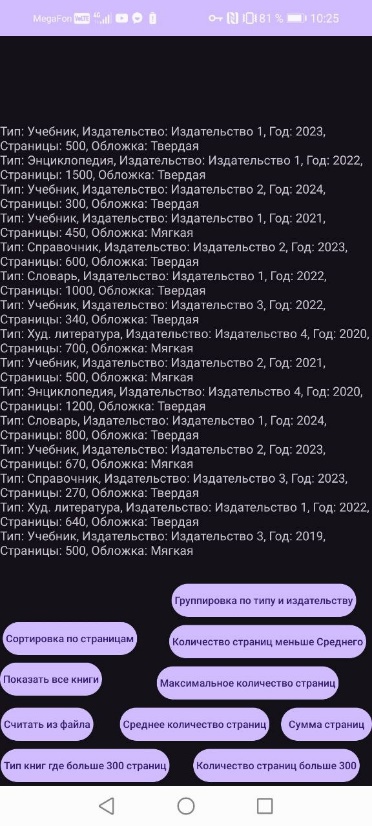
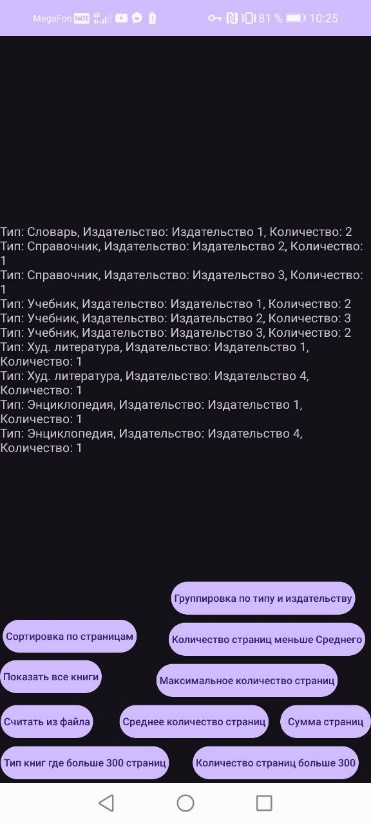
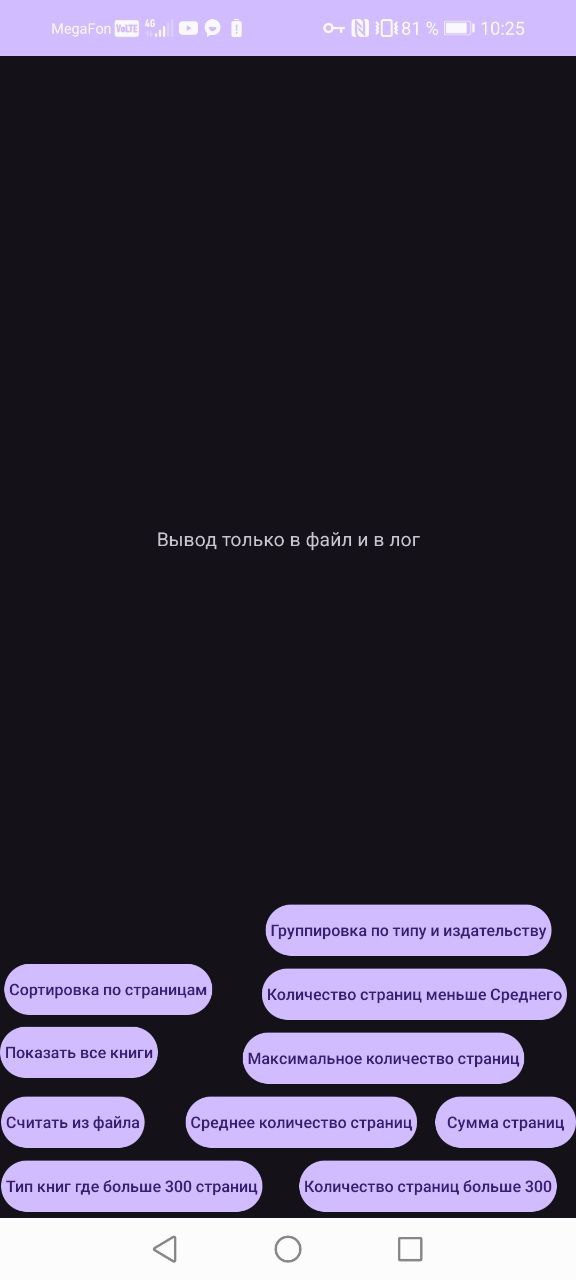
**activity\_main.xml**

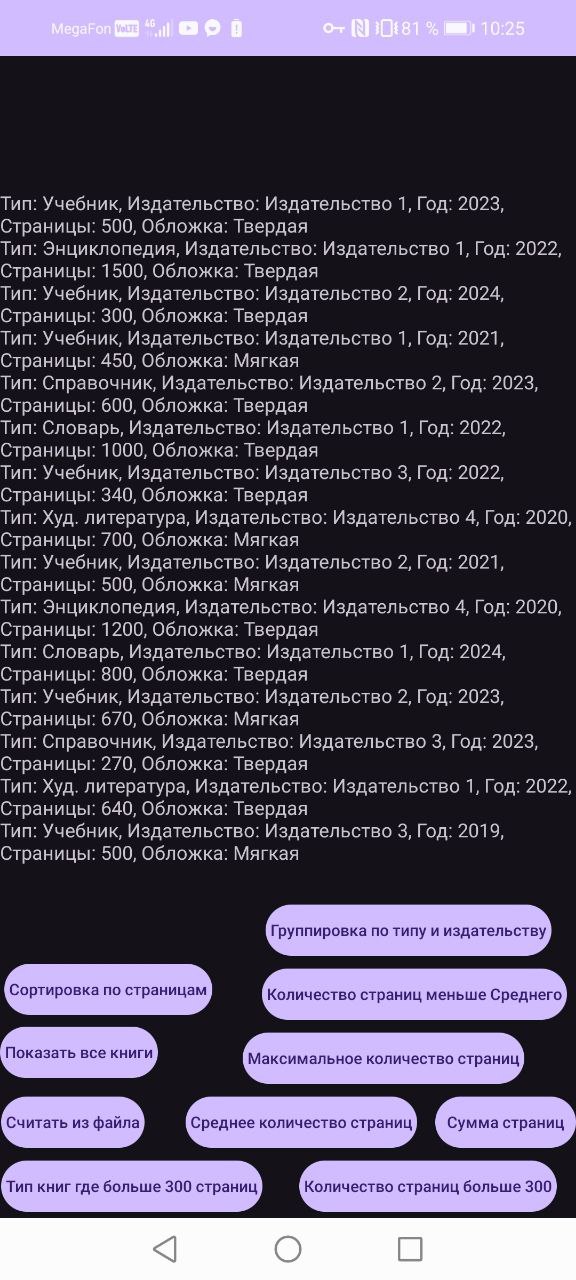
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <!-- Кнопки для запросов -->  
  
 <!-- Кнопки для добавления книги и вывода всех книг -->  
  
 <!-- TextView для отображения результатов -->  
  
 <Button  
 android:id="@+id/btnSort"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Сортировка по страницам"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.011"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.821" />  
  
 <Button  
 android:id="@+id/btnShowBooks"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Показать все книги"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@id/btnReadFromFile"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.994" />  
  
 <Button  
 android:id="@+id/btnLessThanAvg"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Количество страниц меньше Среднего"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnMax"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.966"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnSum"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Сумма страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnGreaterThan"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnReadFromFile"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Считать из файла"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@id/btnTypeGreaterThan"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.003"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnMax"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Максимальное количество страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnAvg"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.825"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnTypeGreaterThan"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Тип книг где больше 300 страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.004"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnGreaterThan"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Количество страниц больше 300"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.939"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnAvg"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Среднее количество страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnGreaterThan"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.538"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnGroup"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Группировка по типу и издательству"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnLessThanAvg"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.916"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <TextView  
 android:id="@+id/tvResults"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Результаты будут здесь"  
 android:textSize="12sp"  
 app:layout\_constraintBottom\_toTopOf="@id/btnShowBooks"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**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\_lab4new"  
 tools:targetApi="31">  
 <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>

**Результаты выполнения работы:**

**  **

****

**Вывод:** в ходе лабораторной работы было разработано приложение, взаимодействующее с базой данных SQLite.