ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Кафедра прикладной информатики**

Лабораторная работа №1

ПО ДИСЦИПЛИНЕ «Программирование мобильных устройств»

НА ТЕМУ:

«**Знакомство с Android Studio**»

**Выполнил:**

студент группы ПИН-Б-З-22-1

Пензев Константин Сергеевич

**Проверил:**

Старший преподаватель института перспективной инженерии

Мартыновская Александра Сергеевна

Ставрополь, 2024 г.

**Лабораторная работа №4. Использование базы данных**

**Цель работы:** Научиться использовать встроенную базу данных SQLite в Android-приложениях для хранения и управления данными.

**1. Введение**

В данной лабораторной работе мы изучили основы работы с базой данных SQLite в Android. SQLite — это встроенная реляционная база данных, которая позволяет хранить данные локально на устройстве. Мы создали класс для работы с базой данных, реализовали методы для создания, обновления и удаления данных, а также научились отображать данные из базы на экране приложения.

**2. Описание работы**

**2.1. Создание нового проекта**

1. **Создание проекта:**
   * Запустили Android Studio и создали новый проект, выбрав шаблон **Empty Activity**.
   * Указали имя проекта — **DatabaseApp** и выбрали язык программирования — **Java**.
   * Установили минимальную версию SDK — **API 24: Android 7.0 (Nougat)**.
2. **Структура проекта:**
   * После создания проекта изучили структуру файлов:
     + **app/java/com.example.databaseapp** — папка с Java-кодом.
     + **app/res/layout** — папка с XML-файлами, отвечающими за интерфейс приложения.
     + **app/res/values** — папка для хранения строковых ресурсов и констант.

**2.2. Создание класса для работы с базой данных**

1. **Создание класса SQLHelper:**
   * В папке **app/java/com.example.databaseapp** создали новый класс **SQLHelper**, который наследуется от **SQLiteOpenHelper**.
   * Пример кода:

java

public class SQLHelper extends SQLiteOpenHelper {

private static final String DATABASE\_NAME = "my\_database.db";

private static final int DATABASE\_VERSION = 1;

public SQLHelper(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

db.execSQL("CREATE TABLE table1 (\_id INTEGER PRIMARY KEY, name TEXT, age INTEGER)");

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS table1");

onCreate(db);

}

}

1. **Методы onCreate и onUpgrade:**
   * Метод **onCreate** вызывается при первом создании базы данных. В нём мы создали таблицу **table1** с тремя столбцами: **\_id**, **name** и **age**.
   * Метод **onUpgrade** вызывается при обновлении базы данных. В нём мы удалили старую таблицу и создали новую.

**2.3. Работа с базой данных**

1. **Добавление данных:**
   * В классе **SQLHelper** создали метод для добавления данных в таблицу.
   * Пример кода:

java

public void addData(String name, int age) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put("name", name);

values.put("age", age);

db.insert("table1", null, values);

db.close();

}

1. **Получение данных:**
   * Создали метод для получения всех данных из таблицы.
   * Пример кода:

java

public Cursor getData() {

SQLiteDatabase db = this.getReadableDatabase();

return db.query("table1", null, null, null, null, null, null);

}

1. **Обновление данных:**
   * Создали метод для обновления данных в таблице.
   * Пример кода:

java

public void updateData(int id, String name, int age) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put("name", name);

values.put("age", age);

db.update("table1", values, "\_id = ?", new String[]{String.valueOf(id)});

db.close();

}

1. **Удаление данных:**
   * Создали метод для удаления данных из таблицы.
   * Пример кода:

java

public void deleteData(int id) {

SQLiteDatabase db = this.getWritableDatabase();

db.delete("table1", "\_id = ?", new String[]{String.valueOf(id)});

db.close();

}

**2.4. Отображение данных на экране**

1. **Создание интерфейса:**
   * В файле **activity\_main.xml** создали интерфейс для отображения данных из базы.
   * Добавили **ListView** для отображения списка данных и кнопку для добавления новых записей.
   * Пример кода:

xml

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<Button

android:id="@+id/addButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Добавить запись" />

<ListView

android:id="@+id/dataListView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

1. **Заполнение ListView данными:**
   * В классе **MainActivity** получили данные из базы и отобразили их в **ListView**.
   * Пример кода:

java

SQLHelper dbHelper = new SQLHelper(this);

Cursor cursor = dbHelper.getData();

List<String> dataList = new ArrayList<>();

while (cursor.moveToNext()) {

String name = cursor.getString(cursor.getColumnIndex("name"));

int age = cursor.getInt(cursor.getColumnIndex("age"));

dataList.add(name + " - " + age);

}

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, dataList);

ListView listView = findViewById(R.id.dataListView);

listView.setAdapter(adapter);

1. **Добавление новых записей:**
   * Реализовали обработку нажатия на кнопку для добавления новых записей в базу данных.
   * Пример кода:

java

Button addButton = findViewById(R.id.addButton);

addButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

SQLHelper dbHelper = new SQLHelper(MainActivity.this);

dbHelper.addData("Новая запись", 25);

updateListView();

}

});

**2.5. Тестирование приложения**

1. **Запуск приложения:**
   * Запустили приложение на эмуляторе и убедились, что данные из базы отображаются корректно.
   * Проверили добавление, обновление и удаление записей.

**3. Результаты**

* Создан новый проект с использованием шаблона **Empty Activity**.
* Реализован класс **SQLHelper** для работы с базой данных SQLite.
* Созданы методы для добавления, получения, обновления и удаления данных.
* Данные из базы успешно отображаются на экране приложения.
* Приложение успешно запущено на эмуляторе, все функции работают корректно.

**4. Выводы**

* SQLite предоставляет удобный способ хранения данных локально на устройстве.
* Освоены базовые операции с базой данных: создание, чтение, обновление и удаление записей.
* Данные из базы можно легко отображать на экране с помощью **ListView**.