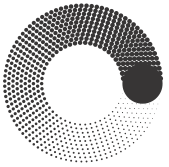
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.04.02 «Информационные системы и технологии»,**

**ЛАБОРАТОРНАЯ РАБОТА № 8**

**Дисциплина: Разработка мобильных приложений для Android**

**Тема: Работа с СУБД SQLite**

**Выполнил(а): студент(ка) группы 224-371** Лейн Ф.Е.

(Фамилия И.О.)

**Проверил(а):** Якубовский К.И.

(Фамилия И.О. )

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**ЦЕЛЬ РАБОТЫ**

Разработать приложение, позволяющее создавать базу данных, заполнять начальными значениями, считывать данные из базы, сохранять новые записи, удалять записи и очищать базу.

**ХОД РАБОТЫ**

В начале была выполнена разработка макета (формы) приложения, результат показан на рисунке 1. Сверху находятся 4 кнопки, реализующие запрос на отображение (обновление) базы данных, добавление строки, удаление строки и очистку всей базы данных. Далее расположены 3 текстовых поля (имя, телефон и заметки). Внизу Activity находится ScrollView, содержащий TextView для отображения данных в таблице.

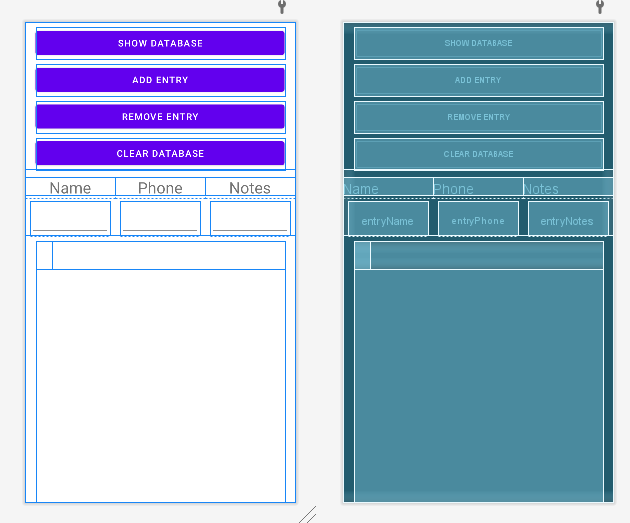


Рисунок 1 – Разработанный макет приложения

После открытия Activity, происходит инициализация TableLayout, а именно – добавление соответствующих TextView для отображения данных. Всего создаётся 11 строк (10 касаний + заголовок) и 4 колонки (индекс, ID, X, Y), что показано в листинге 1.

Листинг 1 – инициализация таблицы данных

*// Create 11 rows of text (header + 10 IDs)***for** (**int** row = 0; row < 11; row++) {  
 *// Create a new TableRow* TableRow tableRow = **new** TableRow(**this**);  
  
 *// Add 4 TextViews to the TableRow* **for** (**int** column = 0; column < 4; column++) {  
 TextView textView = **new** TextView(**this**);  
  
 *// Header* **if** (row == 0) {  
 **switch** (column) {  
 **case** 0:  
 textView.setText(**"Index"**);  
 **break**;  
 **case** 1:  
 textView.setText(**"ID"**);  
 **break**;  
 **case** 2:  
 textView.setText(**"X"**);  
 **break**;  
 **case** 3:  
 textView.setText(**"Y"**);  
 **break**;  
 }  
 }  
  
 *// Other ID* **else** {  
 **switch** (column) {  
 *// First column (Index)* **case** 0:  
 textView.setText(String.*valueOf*(row));  
 **break**;  
  
 *// ID* **case** 1:  
 **IDTextViews**.add(textView);  
 textView.setText(**""**);  
 **break**;  
  
 *// X* **case** 2:  
 **XTextViews**.add(textView);  
 textView.setText(**""**);  
 **break**;  
  
 *// Y* **case** 3:  
 **YTextViews**.add(textView);  
 textView.setText(**""**);  
 **break**;  
 }  
 }  
  
 *// Add TextView to the TableRow* tableRow.addView(textView);  
 }  
  
 *// Add the TableRow to the TableLayout* tableLayout.addView(tableRow);  
}

Далее, производится назначение метода, обрабатывающего все касания тестового поля, что показано в листинге 2.

Листинг 2 – назначение обработчика событий касаний

*// Connect onTouch to TextView*findViewById(R.id.***textView***).setOnTouchListener(**this**);

Метод onTouch реализует обработку касаний. А именно, обработку таких событий как: ***ACTION\_DOWN***, ***ACTION\_POINTER\_DOWN***, ***ACTION\_UP***, ***ACTION\_POINTER\_UP*** и ***ACTION\_MOVE***. Данные записываются в соответствующие объекты TextView, что показано в листинге 3.

Листинг 3 – обработка касаний

@Override  
**public boolean** onTouch(View v, MotionEvent event) {  
 **int** actionMask = event.getActionMasked();  
 **int** pointerIndex = event.getActionIndex();  
 **int** pointerCount = event.getPointerCount();  
  
 **switch** (actionMask) {  
 **case** MotionEvent.***ACTION\_DOWN***:  
 **touchFlag** = **true**;  
  
 *// Set last touch index* **case** MotionEvent.***ACTION\_POINTER\_DOWN***:  
 **touchLastIndex**.setText(String.*valueOf*(pointerIndex));  
 **break**;  
  
 *// Clear all rows and stats if we released all touches* **case** MotionEvent.***ACTION\_UP***:  
 **touchFlag** = **false**;  
 **for** (**int** i = 0; i < **IDTextViews**.size(); i++) {  
 **IDTextViews**.get(i).setText(**""**);  
 **XTextViews**.get(i).setText(**""**);  
 **YTextViews**.get(i).setText(**""**);  
 }  
 **touches**.setText(**"0"**);  
  
 *// Set last release index* **case** MotionEvent.***ACTION\_POINTER\_UP***:  
 **releaseLastIndex**.setText(String.*valueOf*(pointerIndex));  
 **break**;  
  
 *// Show current touch coordinates* **case** MotionEvent.***ACTION\_MOVE***:  
 **for** (**int** i = 0; i < 10; i++) {  
 **if** (i < pointerCount) {  
 **IDTextViews**.get(i).setText(String.*valueOf*(event.getPointerId(i)));  
 **XTextViews**.get(i).setText(String.*valueOf*((**int**) event.getX(i)));  
 **YTextViews**.get(i).setText(String.*valueOf*((**int**) event.getY(i)));  
 } **else** {  
 **IDTextViews**.get(i).setText(**""**);  
 **XTextViews**.get(i).setText(**""**);  
 **YTextViews**.get(i).setText(**""**);  
 }  
 }  
 **break**;  
 }  
  
 *// Set number of touches* **if** (**touchFlag**)  
 **touches**.setText(String.*valueOf*(pointerCount));  
 **return true**;  
  
}

Для операций с базой данных был создан класс DBHandler, листинг которого представлен в приложении Б. Данный класс расширяет класс SQLiteOpenHelper и наследует класс BaseColumns.

В методе onCreate производится создание таблицы (если она не существует), что показано в листинге 4.

Листинг 4 – создание таблицы

*/\*\*  
 \* Creates table if not exists  
 \** ***@param db*** *SQLiteDatabase object  
 \*/*@Override  
**public void** onCreate(SQLiteDatabase db) {  
 String createTableSQL =  
 String.*format*(**"CREATE TABLE IF NOT EXISTS %s (%s INTEGER PRIMARY KEY AUTOINCREMENT, %s TEXT, %s TEXT, %s TEXT);"**,  
 ***DB\_TABLE\_NAME***,  
 ***DB\_COLUMN\_ID***,  
 ***DB\_COLUMN\_NAME***,  
 ***DB\_COLUMN\_PHONE***,  
 ***DB\_COLUMN\_NOTES***);  
 db.execSQL(createTableSQL);  
}

Основной класс работает непосредственно с классом DBHandler. В листинге 5 показан процесс чтения данных из базы данных. Для этого создаётся объект Cursor, который последовательно считывает все строки таблицы и добавляет их в соответствующий TextView.

Листинг 5 – чтение базы данных

*/\*\*  
 \* Shows all table entries  
 \*/***private void** databaseShow() {  
 *// Clear list of entries* **textViewEntries**.setText(**""**);  
  
 String[] projection = {  
 DBHandler.***DB\_COLUMN\_NAME***,  
 DBHandler.***DB\_COLUMN\_PHONE***,  
 DBHandler.***DB\_COLUMN\_NOTES***,  
 };  
 Cursor cursor = **dbHandler**.getWritableDatabase().query(  
 DBHandler.***DB\_TABLE\_NAME***,  
 projection,  
 **null**,  
 **null**,  
 **null**,  
 **null**,  
 **null** );  
  
 *// Add all entries* **int** nameIndex = cursor.getColumnIndex(DBHandler.***DB\_COLUMN\_NAME***);  
 **int** phoneIndex = cursor.getColumnIndex(DBHandler.***DB\_COLUMN\_PHONE***);  
 **int** notesIndex = cursor.getColumnIndex(DBHandler.***DB\_COLUMN\_NOTES***);  
 **while** (cursor.moveToNext()) {  
 String nameValue = cursor.getString(nameIndex);  
 String phoneValue = cursor.getString(phoneIndex);  
 String notesValue = cursor.getString(notesIndex);  
 **textViewEntries**.append(String.*format*(**"\n%s\n%s\n%s\n"**,  
 nameValue, phoneValue, notesValue));  
 }  
 cursor.close();  
}

Добавление данных в базу данных производится при помощи метода getWritableDatabase().insert(), в который передаются все 3 текстовых поля (имя, телефон и заметки).

Листинг 6 – добавление строки в базу данных

*/\*\*  
 \* Adds entry to table  
 \*/***private void** databaseAddEntry() {  
 *// Read entry values* ContentValues values = **new** ContentValues();  
 values.put(DBHandler.***DB\_COLUMN\_NAME***, **editTextName**.getText().toString());  
 values.put(DBHandler.***DB\_COLUMN\_PHONE***, **editTextPhone**.getText().toString());  
 values.put(DBHandler.***DB\_COLUMN\_NOTES***, **editTextNotes**.getText().toString());  
  
 *// Add to database* **dbHandler**.getWritableDatabase().insert(DBHandler.***DB\_TABLE\_NAME***, **null**, values);  
  
 *// Show entries (update)* databaseShow();  
}

Удаление строки производится по имени. Для этого используется условие поиска name=? и заданное имя для удаления, что показано в листинге 7. Соответственно, если в таблице имеется несколько строк с одинаковым именем, все этим строки будут удалены. Также, можно сделать удаление уникальных строк по id, однако для этого необходимо использовать иной подход к отображению данных, например, отдельных TextView, или, объект RecycleView.

Листинг 7 – Удаление строки из базы данных

*/\*\*  
 \* Deletes table entry by Name  
 \*/***private void** databaseDeleteEntry() {  
 *// Delete rows with selected name* String selection = DBHandler.***DB\_COLUMN\_NAME*** + **"=?"**;  
 **dbHandler**.getWritableDatabase().delete(DBHandler.***DB\_TABLE\_NAME***,  
 selection,  
 **new** String[] {**editTextName**.getText().toString()});  
  
 *// Show entries (update)* databaseShow();  
}

Очистка всей таблицы производится при помощи метода getWritableDatabase().delete(), что удалит все имеющие строки в таблице внутри базы данных.

**ВЫВОД**

В результате выполнения данной лабораторной работы были изучены навыки работы с базой данных SQLite. Были изучены навыки создания базы данных, таблицы, добавление и удаление строк, поиск по ключу и т.д.

**Приложение А. Листинг кода разметки Activity приложения**

<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**Button  
 android:id="@+id/buttonShow"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Show database"** />  
  
 <**Button  
 android:id="@+id/buttonAdd"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Add entry"** />  
  
 <**Button  
 android:id="@+id/buttonDel"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Remove entry"** />  
  
 <**Button  
 android:id="@+id/buttonClear"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Clear database"** />  
  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center"**>  
  
 <**TextView  
 android:text="Name"  
 android:gravity="center"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:layout\_weight="1"  
 android:textSize="24sp"** />  
  
 <**TextView  
 android:text="Phone"  
 android:gravity="center"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:layout\_weight="1"  
 android:textSize="24sp"** />  
  
 <**TextView  
 android:text="Notes"  
 android:gravity="center"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:layout\_weight="1"  
 android:textSize="24sp"** />  
 </**LinearLayout**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"**>  
  
 <**EditText  
 android:id="@+id/entryName"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_weight="1"  
 android:textSize="24sp"** />  
  
 <**EditText  
 android:id="@+id/entryPhone"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_weight="1"  
 android:textSize="24sp"** />  
  
 <**EditText  
 android:id="@+id/entryNotes"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_weight="1"  
 android:textSize="24sp"** />  
 </**LinearLayout**>  
  
  
 <**ScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginTop="8dp"**>  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"** >  
 <**TextView  
 android:id="@+id/entriesList"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:paddingStart="24dp"  
 android:paddingTop="16dp"  
 android:textSize="20sp"** />  
 </**LinearLayout**>  
 </**ScrollView**>  
</**LinearLayout**>

**Приложение Б. Листинг кода класса DBHandler**

**package** com.labs.lr8;  
  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
**import** android.provider.BaseColumns;  
  
**public class** DBHandler **extends** SQLiteOpenHelper **implements** BaseColumns {  
 *// Database .db file name and version* **public static final** String ***DB\_FILE\_NAME*** = **"database.db"**;  
 **public static final int *DB\_VERSION*** = 1;  
  
 *// Database columns* **public static final** String ***DB\_TABLE\_NAME*** = **"people"**;  
 **public static final** String ***DB\_COLUMN\_ID*** = **"\_id"**;  
 **public static final** String ***DB\_COLUMN\_NAME*** = **"name"**;  
 **public static final** String ***DB\_COLUMN\_PHONE*** = **"phone"**;  
 **public static final** String ***DB\_COLUMN\_NOTES*** = **"notes"**;  
  
 **public** DBHandler(Context context) {  
 **super**(context, ***DB\_FILE\_NAME***, **null**, ***DB\_VERSION***);  
 }  
  
 */\*\*  
 \* Creates table if not exists  
 \** ***@param db*** *SQLiteDatabase object  
 \*/* @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 String createTableSQL =  
 String.*format*(**"CREATE TABLE IF NOT EXISTS %s (%s INTEGER PRIMARY KEY AUTOINCREMENT, %s TEXT, %s TEXT, %s TEXT);"**,  
 ***DB\_TABLE\_NAME***,  
 ***DB\_COLUMN\_ID***,  
 ***DB\_COLUMN\_NAME***,  
 ***DB\_COLUMN\_PHONE***,  
 ***DB\_COLUMN\_NOTES***);  
 db.execSQL(createTableSQL);  
 }  
  
 */\*\*  
 \* Deletes current table if DB\_VERSION changed and creates new one  
 \** ***@param db*** *SQLiteDatabase object  
 \** ***@param oldVersion*** *Old DB\_VERSION  
 \** ***@param newVersion*** *New DB\_VERSION  
 \*/* @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
 *// Delete old table* db.execSQL(String.*format*(**"DROP TABLE IF EXISTS %s"**, ***DB\_TABLE\_NAME***));  
  
 *// Create new one* onCreate(db);  
 }  
}

**Приложение В. Листинг кода основного класса приложения**

**package** com.labs.lr8;  
  
**import** android.content.ContentValues;  
**import** android.database.Cursor;  
**import** android.os.Bundle;  
**import** android.widget.EditText;  
**import** android.widget.TextView;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 *// Entry fields* **private** EditText **editTextName**, **editTextPhone**, **editTextNotes**;  
  
 *// "List" of entries* **private** TextView **textViewEntries**;  
  
 *// Database helper* DBHandler **dbHandler**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 *// Connect buttons* findViewById(R.id.***buttonShow***).setOnClickListener(v -> databaseShow());  
 findViewById(R.id.***buttonAdd***).setOnClickListener(v -> databaseAddEntry());  
 findViewById(R.id.***buttonDel***).setOnClickListener(v -> databaseDeleteEntry());  
 findViewById(R.id.***buttonClear***).setOnClickListener(v -> databaseClear());  
  
 *// Initialize EditTexts and TextViews* **editTextName** = findViewById(R.id.***entryName***);  
 **editTextPhone** = findViewById(R.id.***entryPhone***);  
 **editTextNotes** = findViewById(R.id.***entryNotes***);  
 **textViewEntries** = findViewById(R.id.***entriesList***);  
  
 *// Initialize DBHandler class* **dbHandler** = **new** DBHandler(**this**);  
 }  
  
 */\*\*  
 \* Shows all table entries  
 \*/* **private void** databaseShow() {  
 *// Clear list of entries* **textViewEntries**.setText(**""**);  
  
 String[] projection = {  
 DBHandler.***DB\_COLUMN\_NAME***,  
 DBHandler.***DB\_COLUMN\_PHONE***,  
 DBHandler.***DB\_COLUMN\_NOTES***,  
 };  
 Cursor cursor = **dbHandler**.getWritableDatabase().query(  
 DBHandler.***DB\_TABLE\_NAME***,  
 projection,  
 **null**,  
 **null**,  
 **null**,  
 **null**,  
 **null** );  
  
 *// Add all entries* **int** nameIndex = cursor.getColumnIndex(DBHandler.***DB\_COLUMN\_NAME***);  
 **int** phoneIndex = cursor.getColumnIndex(DBHandler.***DB\_COLUMN\_PHONE***);  
 **int** notesIndex = cursor.getColumnIndex(DBHandler.***DB\_COLUMN\_NOTES***);  
 **while** (cursor.moveToNext()) {  
 String nameValue = cursor.getString(nameIndex);  
 String phoneValue = cursor.getString(phoneIndex);  
 String notesValue = cursor.getString(notesIndex);  
 **textViewEntries**.append(String.*format*(**"\n%s\n%s\n%s\n"**,  
 nameValue, phoneValue, notesValue));  
 }  
 cursor.close();  
 }  
  
 */\*\*  
 \* Adds entry to table  
 \*/* **private void** databaseAddEntry() {  
 *// Read entry values* ContentValues values = **new** ContentValues();  
 values.put(DBHandler.***DB\_COLUMN\_NAME***, **editTextName**.getText().toString());  
 values.put(DBHandler.***DB\_COLUMN\_PHONE***, **editTextPhone**.getText().toString());  
 values.put(DBHandler.***DB\_COLUMN\_NOTES***, **editTextNotes**.getText().toString());  
  
 *// Add to database* **dbHandler**.getWritableDatabase().insert(DBHandler.***DB\_TABLE\_NAME***, **null**, values);  
  
 *// Show entries (update)* databaseShow();  
 }  
  
 */\*\*  
 \* Deletes table entry by Name  
 \*/* **private void** databaseDeleteEntry() {  
 *// Delete rows with selected name* String selection = DBHandler.***DB\_COLUMN\_NAME*** + **"=?"**;  
 **dbHandler**.getWritableDatabase().delete(DBHandler.***DB\_TABLE\_NAME***,  
 selection,  
 **new** String[] {**editTextName**.getText().toString()});  
  
 *// Show entries (update)* databaseShow();  
 }  
  
 */\*\*  
 \* Removes entire table  
 \*/* **private void** databaseClear() {  
 *// Delete table* **dbHandler**.getWritableDatabase().delete(DBHandler.***DB\_TABLE\_NAME***,  
 **null**, **null**);  
  
 *// Show entries (update)* databaseShow();  
 }  
}