МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Кафедра прикладной информатики**

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

ДИСЦИПЛИНЫ «Программирование мобильных устройств»

НА ТЕМУ:

**Использование баз данных**

**Выполнил:**

студент группы ПИН-б-о-21-1

Стороженко Артем Владимирович

Проверил: А. А. Щеголев

Ставрополь, 2023

**Цель:** научиться использовать встроенную базу данных.

**Ход работы**

**Задание 1**

1. Вначале я создал класс SQLHelper, необходимый для взаимодействия с базой данных и переопределил его методы onUpgrade() и onCreate(), а также создал метод getFullTable(). Код класса представлен на листинге 1.1.
2. Далее, в классе MainActivity я создал метод getList(), позволяющий получить доступ к таблице в базе данных. Код метода представлен на листинге 1.2
3. В метод onCreate() класса MainActivity были добавлены команды для отображения содержимого таблицы базы данных через TextView элементы. Код показан на листинге 1.3.

public class SQLHelper extends SQLiteOpenHelper {  
  
 public SQLHelper(Context context) {  
 super(context, "TrainBase", null, 1);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL("create table " + "table1 ( \_id integer primary key autoincrement, name text not null, Age integer not null);");  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int i, int i1) {  
 db.execSQL("DROP TABLE IF EXISTS table1");  
 onCreate(db);  
 }  
  
 public Cursor getFullTable(){  
 SQLiteDatabase db = this.getWritableDatabase();  
 return db.query("table1", new String[] {" \_id", "name", "Age"}, null, null, null, null, null);  
 }  
}

Листинг 1.1 – класс SQLHelper

public ArrayList<Person> getList(){  
 ArrayList<Person> persons = new ArrayList<>();  
 Cursor cursor = db.getFullTable();  
 if (cursor != null){  
 while (cursor.moveToNext()) {  
 persons.add(new Person(cursor.getInt(0), cursor.getString(1), cursor.getInt(2)));  
 }  
 }  
 return persons;  
}

Листинг 1.2 – код метода getList()

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 db = new SQLHelper(this);  
 LinearLayout ll = (LinearLayout)findViewById(R.id.*ll*);  
 LinearLayout row;  
 ArrayList<Person> persons = getList();  
 for (Person p: persons){  
 row = new LinearLayout(this);  
 row.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 TextView id = new TextView(this);  
 id.setText(Integer.*toString*(p.\_id));  
 id.setWidth(120);  
  
  
 TextView name = new TextView(this);  
 name.setText(p.name);  
 name.setWidth(200);  
  
 TextView Age = new TextView(this);  
 Age.setText(Integer.*toString*(p.Age));  
 Age.setWidth(120);  
  
 row.addView(id);  
 row.addView(name);  
 row.addView(Age);  
  
 ll.addView(row);  
 }  
}

Листинг 1.3 – метод onCreate() класса MainActivity

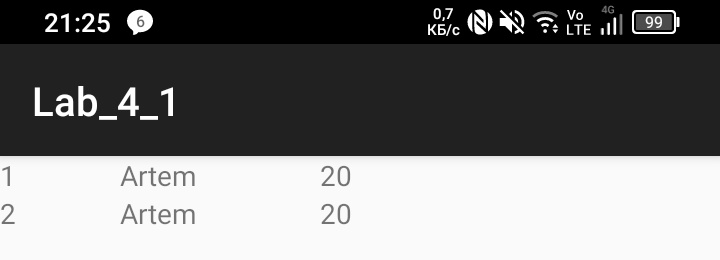


Рисунок 1 – Результат выполнения задания 1

**Задание 2**

В рамках задания 2 лабораторной работы необходимо было создать базу данных и методы для отображения и редактирования данных в ней соответственно варианту (мой вариант 8 - ЖКХ).

1. В первую очередь, мной был создан класс, описывающий строку в таблице базы данных dbRow. Конструктор этого класса представлен на листинге 2.1.
2. Далее, в файле разметки были созданы необходимые графические элементы, а именно «шапка» для таблицы и для поля ввода данных, само поле для ввода данных, менеджер размещения LinearLayout для хранения строк таблицы базы данных и кнопки для осуществления вызова методов. Интерфейс приложения представлен на рисунке 2.1.
3. Далее, был реализован метод showFullTable(), выводящий все строки таблицы с помощью TextView. Код метода представлен на листинге 2.2.
4. Следующим шагом был реализован механизм выбора строки таблицы.

Для этого для каждого созданного TextView назначался onClickListener, реализующий при нажатии на нужный LinearLayout, представляющий одну строу таблицы, ввод его значения в соответствующие элементы EditText. Код слушателя представлен на листинге 2.3.

1. Далее, были реализованы методы получения данных из полей ввода данных (метод getEditsData() ), очистки этих полей (метод editsClear() ) и преобразования этих данных для ввода в базу данных (метод getContentValue() ). Код методов представлен на листингах 2.4, 2.5 и 2.6 соответственно.
2. Последним шагом, были реализованы методы для кнопок работы с данными, а именно для добавления строки в таблицу (метод addRow() ), изменения строки в таблице (метод updateRow() ) и удаления строки из таблицы (метод deleteRow() ). Код методов представлен на листингах 2.7, 2.8 и 2.9 соответственно.

public dbRow(int Id, int PA, String Fio, String Street\_name, int HN, int AN,  
 String ST, double sum, String PB){  
 id = Id;  
 PersonalAccount = PA;  
 FIO = Fio;  
 Street = Street\_name;  
 HouseNumber = HN;  
 ApartmentNumber = AN;  
 ServiceType = ST;  
 Sum = sum;  
 PayBefore = PB;  
}

Листинг 2.1 – конструктор класса dbRow

public void showFullTable() {  
 LinearLayout row;  
 HorizontalScrollView scrollview;  
 LinearLayout table\_rows = (LinearLayout) findViewById(R.id.*table\_rows*);  
 ArrayList<dbRow> dbRows = getList();  
 LinearLayout table = findViewById(R.id.*table\_rows*);  
 table.removeAllViews();  
 for (dbRow new\_row : dbRows) {  
 row = new LinearLayout(this);  
 row.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 TextView id = new TextView(this);  
 id.setText(Integer.*toString*(new\_row.id));  
 id.setTextSize(12);  
 id.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 id.setWidth(72);  
 id.setHeight(40);  
 id.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView personal\_account = new TextView(this);  
 personal\_account.setText(Integer.*toString*(new\_row.PersonalAccount));  
 personal\_account.setTextSize(12);  
 personal\_account.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 personal\_account.setWidth(200);  
 personal\_account.setHeight(40);  
 personal\_account.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView fio = new TextView(this);  
 fio.setText(new\_row.FIO);  
 fio.setTextSize(12);  
 fio.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 fio.setWidth(400);  
 fio.setHeight(40);  
 fio.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView street = new TextView(this);  
 street.setText(new\_row.Street);  
 street.setTextSize(12);  
 street.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 street.setWidth(260);  
 street.setHeight(40);  
 street.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView house\_number = new TextView(this);  
 house\_number.setText(Integer.*toString*(new\_row.HouseNumber));  
 house\_number.setTextSize(12);  
 house\_number.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 house\_number.setWidth(144);  
 house\_number.setHeight(40);  
 house\_number.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView apartment\_number = new TextView(this);  
 apartment\_number.setText(Integer.*toString*(new\_row.ApartmentNumber));  
 apartment\_number.setTextSize(12);  
 apartment\_number.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 apartment\_number.setWidth(170);  
 apartment\_number.setHeight(40);  
 apartment\_number.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView service\_type = new TextView(this);  
 service\_type.setText(new\_row.ServiceType);  
 service\_type.setTextSize(12);  
 service\_type.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 service\_type.setWidth(240);  
 service\_type.setHeight(40);  
 service\_type.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView sum = new TextView(this);  
 sum.setText(Double.*toString*(new\_row.Sum));  
 sum.setTextSize(12);  
 sum.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 sum.setWidth(160);  
 sum.setHeight(40);  
 sum.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 TextView pay\_before = new TextView(this);  
 pay\_before.setText(new\_row.PayBefore);  
 pay\_before.setTextSize(12);  
 pay\_before.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 pay\_before.setWidth(240);  
 pay\_before.setHeight(40);  
 pay\_before.setBackground(getResources().getDrawable(R.drawable.*cell*));  
  
 row.addView(id);  
 row.addView(personal\_account);  
 row.addView(fio);  
 row.addView(street);  
 row.addView(house\_number);  
 row.addView(apartment\_number);  
 row.addView(service\_type);  
 row.addView(sum);  
 row.addView(pay\_before);  
  
 row.setClickable(true);  
 LinearLayout local\_row = row;

Листинг 2.2 – метод showFullTable()

row.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 EditText[] edits = new EditText[9];  
 EditText col\_1 = findViewById(R.id.*id\_edit*);  
 edits[0] = col\_1;  
 EditText col\_2 = findViewById(R.id.*personal\_account\_edit*);  
 edits[1] = col\_2;  
 EditText col\_3 = findViewById(R.id.*fio\_edit*);  
 edits[2] = col\_3;  
 EditText col\_4 = findViewById(R.id.*street\_edit*);  
 edits[3] = col\_4;  
 EditText col\_5 = findViewById(R.id.*house\_number\_edit*);  
 edits[4] = col\_5;  
 EditText col\_6 = findViewById(R.id.*apartment\_number\_edit*);  
 edits[5] = col\_6;  
 EditText col\_7 = findViewById(R.id.*service\_type\_edit*);  
 edits[6] = col\_7;  
 EditText col\_8 = findViewById(R.id.*sum\_edit*);  
 edits[7] = col\_8;  
 EditText col\_9 = findViewById(R.id.*pay\_before\_edit*);  
 edits[8] = col\_9;  
 for (int i = 0; i < 9; i++){  
 TextView child = (TextView)local\_row.getChildAt(i);  
 edits[i].setText(child.getText());  
 }  
 }

Листинг 2.3 – код слушателя

public dbRow getEditsData(){  
 EditText edit\_id = findViewById(R.id.*id\_edit*);  
 int id\_value = Integer.*parseInt*(edit\_id.getText().toString());

EditText edit\_personal\_account = findViewById(R.id.*personal\_account\_edit*);  
 int personal\_account\_value = Integer.*parseInt*(edit\_personal\_account.getText().toString());

EditText edit\_fio = findViewById(R.id.*fio\_edit*);  
 String FIO\_value = edit\_fio.getText().toString();

EditText edit\_street = findViewById(R.id.*street\_edit*);  
 String street\_value = edit\_street.getText().toString();

EditText edit\_house\_number = findViewById(R.id.*house\_number\_edit*);  
 int house\_number\_value = Integer.*parseInt*(edit\_house\_number.getText().toString());

EditText edit\_apartment\_number = findViewById(R.id.*apartment\_number\_edit*);  
 int apartment\_number\_value = Integer.*parseInt*(edit\_apartment\_number.getText().toString());

EditText edit\_service\_type = findViewById(R.id.*service\_type\_edit*);  
 String service\_type\_value = edit\_service\_type.getText().toString();

EditText edit\_sum = findViewById(R.id.*sum\_edit*);  
 Double sum\_value = Double.*parseDouble*(edit\_sum.getText().toString());

EditText edit\_pay\_before = findViewById(R.id.*pay\_before\_edit*);  
 String pay\_before\_value = edit\_pay\_before.getText().toString();

dbRow row = new dbRow(id\_value, personal\_account\_value, FIO\_value, street\_value, house\_number\_value, apartment\_number\_value, service\_type\_value, sum\_value, pay\_before\_value);  
  
 return row;  
}

Листинг 2.4 – метод getEditsData()

public void editsClear(){  
 LinearLayout edits = findViewById(R.id.*edits*);  
 int child\_count = edits.getChildCount();  
 for (int child\_index = 0; child\_index < child\_count; child\_index++){  
 EditText edittext = (EditText) edits.getChildAt(child\_index);  
 edittext.setText("");  
 }

Листинг 2.5 – метод editsClear()

public ContentValues getContentValues(dbRow row){  
 ContentValues cv = new ContentValues();  
 cv.put("\_id", row.id);  
 cv.put("personal\_account", row.PersonalAccount);  
 cv.put("FIO", row.FIO);  
 cv.put("street\_name", row.Street);  
 cv.put("house\_number", row.HouseNumber);  
 cv.put("apartment\_number", row.ApartmentNumber);  
 cv.put("service\_type", row.ServiceType);  
 cv.put("sum", row.Sum);  
 cv.put("pay\_before", row.PayBefore);  
 return cv;  
}

Листинг 2.6 – метод getContentValues()

public void addRow(View view){  
 try{  
 dbRow row = getEditsData();  
 SQLiteDatabase db = helper\_2.getWritableDatabase();  
 ContentValues cv = new ContentValues();  
 showFullTable();  
 editsClear();  
 }  
 catch (Exception e) {  
 Toast.*makeText*(this, "Incorrect input", Toast.*LENGTH\_SHORT*).show();  
 }  
}

Листинг 2.7 – метод addRow()

public void updateRow(View v) {  
 try {  
 dbRow row = getEditsData();  
 SQLiteDatabase db = helper\_2.getWritableDatabase();  
 ContentValues cv = getContentValues(row);  
 db.update("table\_JKH\_1", cv, "\_id = ?", new String[]{Integer.*toString*(row.id)});  
 showFullTable();  
 editsClear();  
 }  
 catch(Exception e){  
 Toast.*makeText*(this, "Incorrect input", Toast.*LENGTH\_SHORT*).show();  
 }  
  
}

Листинг 2.8 – метод updateRow()

public void deleteRow(View v){  
 try  
 {  
 dbRow row = getEditsData();  
 SQLiteDatabase db = helper\_2.getWritableDatabase();  
 ContentValues cv = getContentValues(row);  
 db.delete("table\_JKH\_2","\_id = ?", new String[]{Integer.*toString*(row.id)});  
 showFullTable();  
 editsClear();  
 }  
 catch(Exception e){  
 Toast.*makeText*(this, "Incorrect input", Toast.*LENGTH\_SHORT*).show();  
 }  
  
}

Листинг 2.9 – метод deleteRow()



Рисунок 2.1 – Интерфейс приложения

**Вывод:** выполнив задания лабораторной работы я получил навыки использования баз данных SQLite, а также опыт создания интерфейса для пользователя базы данных.