Министерство образования и науки 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 "Фундаментальные науки"

КАФЕДРА "Программное обеспечение ЭВМ, информационные технологии и прикладная математика"

О Т Ч Е Т

Лабораторная работа №6

ДИСЦИПЛИНА: " Мобильные операционные системы "

ТЕМА: " СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ СОБСТВЕННЫХ ИСТОЧНИКОВ ДАННЫХ. РАБОТА СО СТАНДАРТНЫМИ ИСТОЧНИКАМИ ДАННЫХ "

Выполнил: студент гр. ИТД-81 Турченко С.А._____

Проверил: Гришунов С.С. _____

Дата сдачи (защиты) лабораторной работы:

Результаты сдачи (защиты):

Количество рейтинговых баллов

Оценка

Калуга, 2018 г.

Цель:

1. Научиться создавать собственные источники данных (Content Provider).
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.

Задача:

Вариант 14

На первом Activity вводится необходимая фамилия (или часть фамилии), на втором Activity выводится искомый список (Имя, Фамилия, моб. телефон). Фамилии во втором списке соответствуют запросу. Второй список отсортировать по имени.

Код программы

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        this.requestPermission();
    }

    public void requestPermission() {
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.READ_CONTACTS},
                0);
    }

    public void onApply(View view) {
        Intent intent = new Intent(MainActivity.this, OutputActivity.class);

        intent.putExtra("name", ((EditText)
findViewById(R.id.activity_main_name)).getText().toString());

        startActivity(intent);
    }

    private void insertItem(String firstName, String secondName, String phone) {
        ContentValues values = new ContentValues();

        values.put(PeopleContentProvider.FIRST_NAME_FIELD, firstName);
        values.put(PeopleContentProvider.SECOND_NAME_FIELD, secondName);
        values.put(PeopleContentProvider.PHONE_FIELD, phone);

        Uri uri = getContentResolver().insert(PeopleContentProvider.URI, values);

        Log.d("insert_item", uri.toString());
    }

    private void addDummy() {
        insertItem("Дожрдж", "Оруэлл", "84842277313");
        insertItem("Станислав", "Турченко", "89066547898");
        insertItem("Елена", "Турченко", "89066555643");
        insertItem("Аднрей", "Турченко", "89035446464");
        insertItem("Ростислав", "Турников", "89066554896");
    }

    public void onDummy(View view) {
        this.addDummy();
    }
}
```

OutputActivity.java

```
public class OutputActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_output);
```

```
        Cursor cursor = this.getCursor(this.getName());

        if ((cursor == null) || (cursor.getCount() == 0)) {
            this.nothingFound();
        } else {
            this.bind(cursor);
        }

    }

    private String getName() {
        return getIntent().getExtras().getString("name");
    }

    private Cursor getCursor(String name) {
        String selection = null;

        if (!TextUtils.isEmpty(name)) {
            selection = PeopleContentProvider.SECOND_NAME_FIELD + " like '%" + name
+ "%'";
        }


        Uri peopleUri = Uri.parse(PeopleContentProvider.URL);
        return getContentResolver().query(
                peopleUri,
                new String[] {
                    PeopleContentProvider.ID_FIELD,
                    PeopleContentProvider.FIRST_NAME_FIELD,
                    PeopleContentProvider.SECOND_NAME_FIELD,
                    PeopleContentProvider.PHONE_FIELD
                },
                selection,
                null,
                PeopleContentProvider.FIRST_NAME_FIELD);
    }

    private void bind(Cursor cursor) {
        ListView view = (ListView) findViewById(R.id.output);
        PeopleAdapter adapter = new PeopleAdapter(this, cursor);
        view.setAdapter(adapter);
    }

    private void nothingFound() {
        View view = findViewById(R.id.output_nothing_found);
        view.setVisibility(View.VISIBLE);
    }

}
```

DatabaseHelper.java

```
public class DatabaseHelper extends SQLiteOpenHelper {

    public DatabaseHelper(Context context) {
        super(
                context,
                PeopleContentProvider.DATABASE_NAME,
                null,
                PeopleContentProvider.DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase database) {
```

```
        database.execSQL(PeopleContentProvider.CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion)
{
        database.execSQL("DROP TABLE IF EXISTS " +
PeopleContentProvider.PEOPLE_TABLE_NAME);
        onCreate(database);
    }

}
```

PeopleContentProvider.java

```
public class PeopleContentProvider extends ContentProvider {

    public static final String DATABASE_NAME = "people";
    public static final int DATABASE_VERSION = 1;
    public static final String PEOPLE_TABLE_NAME = "people";

    public static final String ID_FIELD = "_id";
    public static final String FIRST_NAME_FIELD = "first_name";
    public static final String SECOND_NAME_FIELD = "second_name";
    public static final String PHONE_FIELD = "phone";

    public static final String CREATE_TABLE = "CREATE TABLE " + PEOPLE_TABLE_NAME +
" (" +
            ID_FIELD + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            FIRST_NAME_FIELD + " TEXT, " +
            SECOND_NAME_FIELD +" TEXT, " +
            PHONE_FIELD + " TEXT );";

    public static final String AUTHORITY = "edu.bmstu.stas.provider.database";

    public static final int URI_DEFAULT = 1;
    public static final int URI_PEOPLE = 2;
    public static final int URI_ITEM = 3;

    public static final String URL = "content://" + AUTHORITY + "/default";
    public static final Uri URI = Uri.parse(URL);

    private static HashMap<String, String> PROJECTION_MAP;

    private static final UriMatcher uriMatcher;
    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(AUTHORITY, "default/", URI_DEFAULT);
        uriMatcher.addURI(AUTHORITY, "people/*", URI_PEOPLE);
        uriMatcher.addURI(AUTHORITY, "item/#", URI_ITEM);
    }

    private SQLiteDatabase database;

    public boolean onCreate() {
        Context context = getContext();
        DatabaseHelper helper = new DatabaseHelper(context);

        database = helper.getWritableDatabase();
        return database != null;
    }

    @Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)) {
```

```
            case URI_PEOPLE:
                return "vndr.android.cursor.dir/vnd.bmstu.people";
            case URI_ITEM:
                return "vndr.android.cursor.item/vnd.bmstu.people";
            default:
                return null;
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        long id = this.database.insert(PEOPLE_TABLE_NAME, null, values);

        // https://www.sqlite.org/autoinc.html - 1 is the first id;
        if (id <= 0)
            return null;

        Uri newUri = ContentUris.withAppendedId(URI, id);
        getContext().getContentResolver().notifyChange(newUri, null);
        return  newUri;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[]
selectionArgs) {
        int count = 0;

        switch (uriMatcher.match(uri)) {
            case URI_PEOPLE:
                count = this.database.update(PEOPLE_TABLE_NAME, values, selection,
selectionArgs);
                break;
            case URI_ITEM:
                String newSelection = ID_FIELD + " = " +
uri.getPathSegments().get(1);
                if (!TextUtils.isEmpty(selection))
                    newSelection += " AND (" + selection + ")";
                count = this.database.update(PEOPLE_TABLE_NAME, values,
newSelection, selectionArgs);
        }

        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        int count = 0;

        switch (uriMatcher.match(uri)) {
            case URI_PEOPLE:
                count = this.database.delete(PEOPLE_TABLE_NAME, selection,
selectionArgs);
                break;
            case URI_ITEM:
                String newSelection = ID_FIELD + " = " +
uri.getPathSegments().get(1);
                if (!TextUtils.isEmpty(selection))
                    newSelection += " AND (" + selection + ")";
                count = this.database.delete(PEOPLE_TABLE_NAME, selection,
selectionArgs);
                break;
        }

        getContext().getContentResolver().notifyChange(uri, null);
```

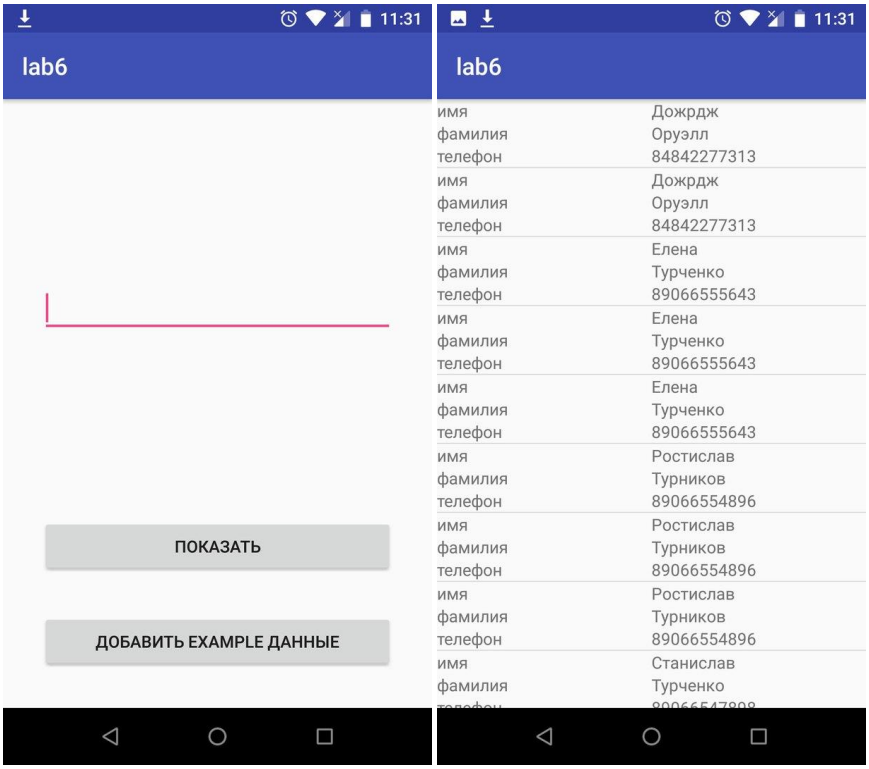
```
        return count;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
        SQLiteQueryBuilder builder = new SQLiteQueryBuilder();
        builder.setTables(PEOPLE_TABLE_NAME);

        switch (uriMatcher.match(uri)) {
            case URI_PEOPLE:
                builder.setProjectionMap(PROJECTION_MAP);
                break;
            case URI_ITEM:
                builder.appendWhere(ID_FIELD + " = " +
uri.getPathSegments().get(1));
            break;
        }

        if (TextUtils.isEmpty(sortOrder))
            sortOrder = ID_FIELD;

        Cursor cursor = builder.query(
                this.database,
                projection,
                selection,
                selectionArgs,
                null,
                null,
                sortOrder);

        cursor.setNotificationUri(getContext().getContentResolver(), uri);
        return cursor;
    }

}
```

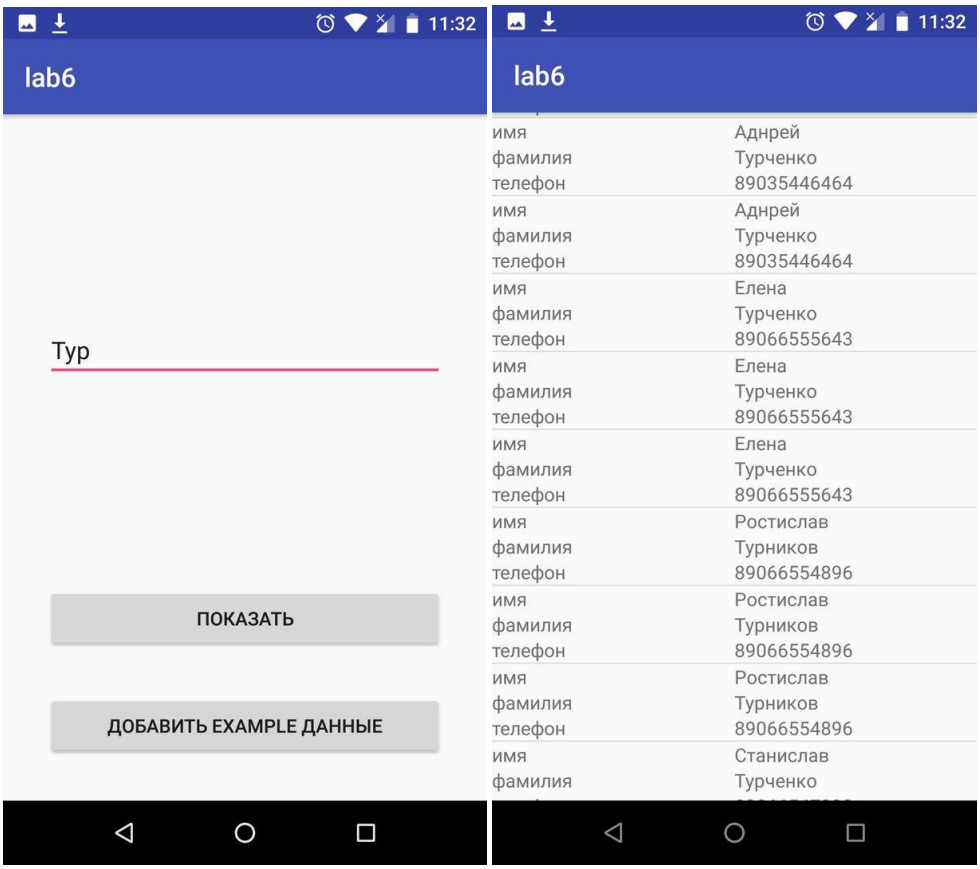
PeopleAdapter.java

```
public class PeopleAdapter extends CursorAdapter {


    public PeopleAdapter (Context context, Cursor cursor) {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.people_record, parent,
false);
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        String first_name = cursor.getString(cursor.getColumnIndexOrThrow(
                PeopleContentProvider.FIRST_NAME_FIELD
        ));
        TextView firstNameView = view.findViewById(R.id.people_first_name_content);
        firstNameView.setText(first_name);

        String second_name = cursor.getString(cursor.getColumnIndexOrThrow(
                PeopleContentProvider.SECOND_NAME_FIELD
        ));
        TextView secondNameView =
view.findViewById(R.id.people_second_name_content);
```

```
        secondNameView.setText(second_name);

        String phone = cursor.getString(cursor.getColumnIndexOrThrow(
                PeopleContentProvider.PHONE_FIELD
        ));
        TextView phoneView = view.findViewById(R.id.people_phone_content);
        phoneView.setText(phone);

    }

}
```

Результат





Вывод

В ходе выполнения лабораторной работы были получены навыки создания собственных источников данных.