**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский Томский политехнический университет»**

Инженерная школа информационных технологий и робототехники

Направление «Информационные системы и технологии»

Отчёт по лабораторной работе №4

**«Взаимодействие с файловой системой, SQLite»**

по дисциплине **«Программирование мобильных устройств и встраиваемых систем»**

Выполнил:

Студент группы 8И6А Козловский В. Е. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

Преподаватель ОИТ Погребной А.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Томск 2020

# Задание

Вариант 4. Хранить группы в БД. Сохранять в SharedPreferences позицию списка, чтобы после перезапуска приложения список сохранял своё состояние.

# Ход работы

Для хранения групп в БД был создан класс “GroupDao”:

@Dao  
**public interface** GroupDao {  
 @Query(**"SELECT \* FROM `group`"**)  
 List<Group> getAll();  
  
 @Query(**"SELECT \* FROM `group` WHERE "**+  
 **"group\_name = :groupName"**)  
 Group selectByName(@NonNull String groupName);  
  
 @Query(**"SELECT \* FROM `group` WHERE "**+  
 **"id = :groupId"**)  
 Group selectById(@NonNull **int** groupId);  
  
 @Insert  
 **void** insert(@NonNull Group group);  
  
 @Query(  
 **"SELECT COUNT(\*) FROM `group` WHERE "** +  
 **"group\_name = :groupName"** )  
 **int** count(@NonNull String groupName);  
}

Схема базы была изменена с использованием миграций. Ниже представлена миграция изменения таблицы “student” и добавления таблицы “group”:

**public static final** Migration ***MIGRATION\_1\_2*** = **new** Migration(1,2) {  
 @Override  
 **public void** migrate(@NonNull SupportSQLiteDatabase database) {  
 database.execSQL(**"CREATE TABLE `group` ("** +  
 **"id INTEGER PRIMARY KEY NOT NULL,"** +  
 **"group\_name TEXT NOT NULL)"**);  
  
 database.execSQL(**"CREATE TABLE new\_Student ("** +  
 **"id INTEGER PRIMARY KEY NOT NULL,"** +  
 **"first\_name TEXT NOT NULL,"** +  
 **"second\_name TEXT NOT NULL,"** +  
 **"last\_name TEXT NOT NULL,"** +  
 **"group\_id INTEGER NOT NULL DEFAULT 0)"**);  
  
 database.execSQL(**"INSERT INTO new\_Student (id, first\_name,second\_name,last\_name) "** +  
 **"SELECT id, first\_name, second\_name, last\_name FROM student"**);  
 database.execSQL(**"DROP TABLE student"**);  
 database.execSQL(**"ALTER TABLE new\_Student RENAME TO student"**);  
 }  
};

Адаптер “StudentAdapter” был заменен новым “GroupStudentAdapter”:

**package** ru.tpu.courses.lab4.adapter;  
  
**import** android.view.ViewGroup;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.recyclerview.widget.RecyclerView;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**import** ru.tpu.courses.lab4.db.Student;  
**import** ru.tpu.courses.lab4.db.Group;  
**import** ru.tpu.courses.lab4.db.StudentGroupListItem;  
  
**public class** GroupStudentAdapter **extends** RecyclerView.Adapter<RecyclerView.ViewHolder> {  
  
 **public static final int *TYPE\_GROUP*** = 0;  
 **public static final int *TYPE\_STUDENT*** = 1;  
  
  
 **private** List<StudentGroupListItem> **studentGroups** = **new** ArrayList<>();  
 **private** onStudentClickListener **mOnStudentClickListener**;  
  
 **public** GroupStudentAdapter(onStudentClickListener listener){  
 **mOnStudentClickListener** = listener;  
 }  
 @Override  
 @NonNull  
 **public** RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 **switch** (viewType) {  
 **case *TYPE\_STUDENT***:  
 **return new** StudentHolder(parent, **mOnStudentClickListener**);  
 **case *TYPE\_GROUP***:  
 **return new** GroupHolder(parent);  
 }  
 **throw new** IllegalArgumentException(**"unknown viewType = "** + viewType);  
 }  
  
  
 @Override  
 **public int** getItemCount() {  
 **return studentGroups**.size();  
 }  
  
 @Override  
 **public int** getItemViewType(**int** position) {  
 **return studentGroups**.get(position).getType();  
 }  
  
 **public void** setData(List<Student> students, List<Group> groups){  
 **this**.**studentGroups**.clear();  
 **for** (Group group: groups  
 ) {  
 **this**.**studentGroups**.add(group);  
 **for** (Student student:students  
 ) {  
 **if**(student.**groupId** == group.**id**){  
 **this**.**studentGroups**.add(student);  
 }  
 }  
 }  
 }  
  
  
 **public interface** onStudentClickListener{  
 **void** onStudentClick(**int** id);  
 }  
}

Для хранения положения списка был реализован класс “TempStudentPref”:

**public class** TempActivityPref {  
  
 **private static final** String ***PREF\_POSITION*** = **"position"**;  
  
 **private final** SharedPreferences **prefs**;  
  
 **public** TempActivityPref(@NonNull Context context) {  
 **prefs** = context.getSharedPreferences(**"temp\_activity"**, Context.***MODE\_PRIVATE***);  
 }  
  
 @Nullable  
 **public int** getPosition() {  
 **return prefs**.getInt(***PREF\_POSITION***, 0);  
 }  
  
 **public void** set(  
 @Nullable **int** position  
 ) {  
 **prefs**.edit()  
 .putInt(***PREF\_POSITION***, position)  
 .apply();  
 }  
  
 **public void** clear() {  
 **prefs**.edit().clear().apply();  
 }  
}

В классе “Lab4Activity” объект TempActivityPref используется для записи измененного положения списка (в onPause) и его восстановления при повторном запуске (в onCreate).

# Результат

При переходе к модулю 4 лабораторной работы на экран устройства выводится список студентов без группы, изначально пустой. При нажатии на кнопку слева, появляется экран добавления, где пользователь вводит название новой группы. Теперь при добавлении нового студента можно выбрать новую группу. Добавленные студенты отображаются в разных группах (рис. 1).

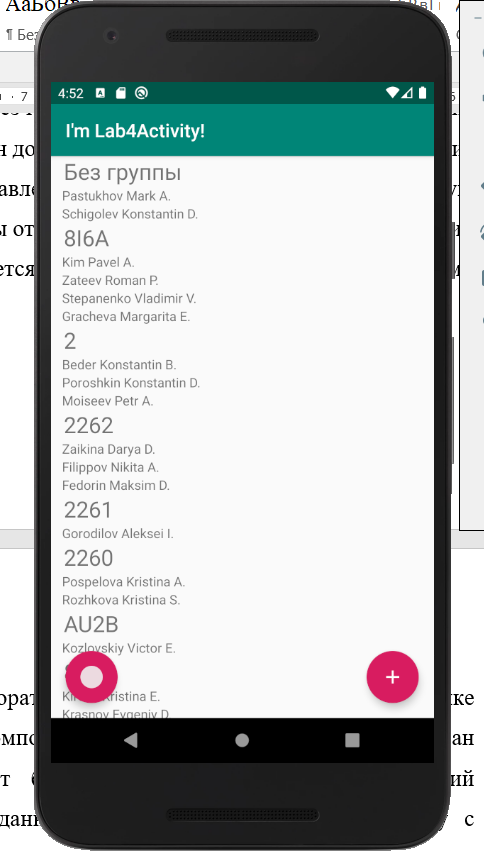


Рисунок 1 – Список студентов в группах

При нажатии на запись студента, открывается окно создания студента с заполненными данными.

Положение прокрутки списка (рис. 2) сохраняется при повторном запуске модуля.

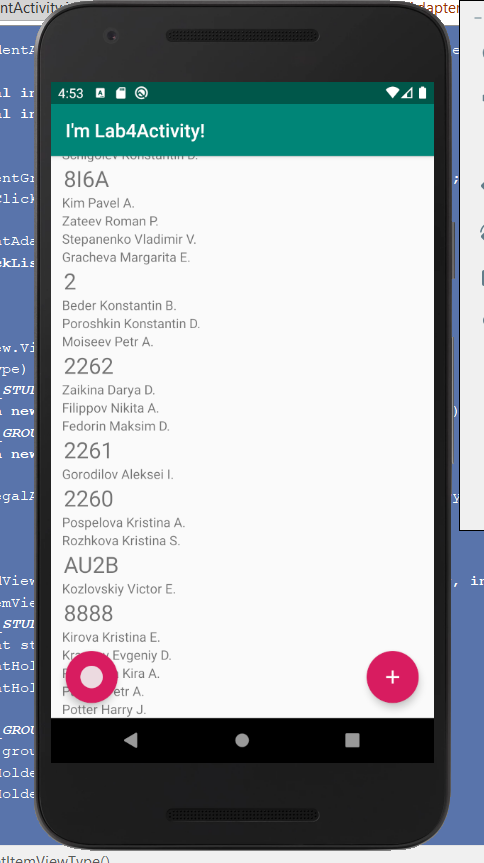


Рисунок 2 – Список студентов пролистан до конца

# Вывод

В ходе выполнения лабораторной работы были закреплены на практике знания принципов работы компонентов SharedPreferences. Был реализован Data-Access-Object компонент библиотеки Android-Room, позволяющий взаимодействовать с базой данных, в том числе изменять её схему с использованием миграций.