ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Преподаватель департамента программной инженерии факультета компьютерных наук

_____ Н. К. Чуйкин
«___» _____ 2021 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук

_____ В.В. Шилов
«___» _____ 2021 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ПЛАНИРОВАНИЯ СПОРТИВНЫХ ТРЕНИРОВОК С ЭЛЕМЕНТАМИ СОЦИАЛЬНОЙ СЕТИ

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.06.05-01 12 01-1-ЛУ

Исполнитель
студент группы БПИ185
_____ / Д. А. Чертанов/
«____»_____ 2021 г.

Москва 2021

УТВЕРЖДЕН
RU.17701729.06.05-01 12 01-1-ЛУ

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ПЛАНИРОВАНИЯ СПОРТИВНЫХ ТРЕНИРОВОК С ЭЛЕМЕНТАМИ СОЦИАЛЬНОЙ СЕТИ

Текст программы

RU.17701729.06.05-01 12 01-1

Листов 240

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

Москва 2021

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ 6
1.1. ExerciseOverviewAdapter.java 6
1.2. FriendOverviewAdapter.java 8
1.3. TimeTableDayAdapter.java 10
1.4. TrainingAdapter.java 13
1.5. ExerciseOverview.java 15
1.6. ExerciseTemp.java 16
1.7. TimeTableDay.java 16
1.8. TimeTableExercise.java 17
1.9. ExerciseInTrainingEditing.java 18
1.10. InsertSetJSON 19
1.11. TrainingDayOfWeekHelper.java 19
1.12. TrainingEditingHelper.java 20
1.13. TrainingExerciseInstance.java 20
1.14. AccountEditingFragment.java 21
1.15. AccountFragment.java 24
1.16. AddExerciseDescriptionFragment.java 30
1.17. AddExerciseMuscleFragment.java 32
1.18. AddExerciseTagFragment.java 35
1.19. AddExerciseVideoFragment.java 37
1.20. AddSetFragment.java 39
1.21. CopyTrainingFragment.java 42
1.22. CreatingTrainingFragment.java 43
1.23. EditingTrainingExerciseFragment.java 45
1.24. ExerciseDescriptionFragment.java 47
1.25. ExerciseSetEditingFragment.java 50
1.26. FilterFragment.java 52
1.27. FriendsListFragment.java 55
1.28. LoginFragment.java 59
1.29. LogOutFragment.java 61
1.30. MainFragment.java 63
1.31. ReconnectFragment.java 66
1.32. SearchFragment.java 67

1.33. SignUpFragment.java 72
1.34. TimeTableFragment.java 75
1.35. TrainingEditingFragment.java 78
1.36. TrainingExerciseSetsFragment.java 80
1.37. TrainingExerciseSuccessFragment.java 85
1.38. TrainingsFragment.java 90
1.39. AccountEditingLogic.java 94
1.40. AddExerciseFromDescriptionLogic.java 94
1.41. AddExerciseInTrainingLogic.java 96
1.42. AddExerciseSearchLogic.java 97
1.43. AddFriendLogic.java 98
1.44. AddSetSaveLogic.java 99
1.45. AddTrainingLogic.java 102
1.46. CopyTrainingLogic.java 103
1.47. DeleteTrainingExerciseLogic.java 105
1.48. DoneNoteLogic.java 107
1.49. EditTrainingTitleLogic.java 108
1.50. ExerciseSetLongClickLogic.java 110
1.51. FilterApplyLogic.java 111
1.52. FilterCancelLogic.java 114
1.53. MoveTrainingExerciseLogic.java 115
1.54. NewExerciseDescriptionLogic.java 119
1.55. NewExerciseFilterLogic.java 120
1.56. NewExerciseTagsLogic.java 122
1.57. NewExerciseVideoLogic.java 123
1.58. NewTrainingCreationLogic.java 125
1.59. OpenEditingTrainingExeciseFragmentLogic.java 127
1.60. OpenExerciseFromQuestionLogic.java 128
1.61. RemoveAccountLogic.java 128
1.62. RemoveFriendLogic.java 130
1.63. RemoveTrainingLogic.java 131
1.64. SaveAccountEditingLogic.java 133
1.65. SetCurrentTrainingLogic.java 136
1.66. SetEditLogic.java 139
1.67. SetRemoveLogic.java 142

1.68. SignInLogic.java 144
1.69. SignUpLogic.java 145
1.70. TimeTableExerciseClickLogic.java 150
1.71. TrainingExerciseSetClickLogic.java 151
1.72. TrainingExerciseSuccessCreateSetLogic.java 151
1.73. TrainingLongClickLogic.java 152
1.74. DeleteAccountTask.java 153
1.75. DeleteFriendshipTask.java 154
1.76. DeleteMoveTrainingExeciseTask.java 155
1.77. DeleteTrainingExerciseSuccessTask.java 157
1.78. DeleteTrainingExerciseTask.java 158
1.79. DeleteTrainingTask.java 159
1.80. InsertFriendshipTask.java 160
1.81. InsertNewExerciseTask.java 161
1.82. InsertTrainingExerciseSuccessTask.java 163
1.83. InsertTrainingExerciseTask.java 164
1.84. InsertTrainingTask.java 165
1.85. MoveTrainingExerciseTask.java 166
1.86. PullExerciseTask.java 169
1.87. PullRemoteDBTask.java 170
1.88. SignUpTask.java 173
1.89. UpdateAccountInfoTask.java 174
1.90. UpdateCurrentTrainingTask.java 176
1.91. UpdateNoteTask.java 177
1.92. UpdateTrainingExerciseSuccessTask.java 178
1.93. UpdateTrainingExerciseTask.java 179
1.94. UpdateTrainingTask.java 181
1.95. Account.java 182
1.96. BodyCondition.java 185
1.97. Exercise.java 187
1.98. ExerciseTag.java 189
1.99. ExerciseToTag.java 190
1.100. ExerciseTrainingType.java 191
1.101. Friendship.java 191
1.102. HttpWork.java 192

1.103. Muscle.java 210
1.104. TargetMuscle.java 211
1.105. Training.java 212
1.106. TrainingExercise.java 213
1.107. TrainingExerciseNote.java 215
1.108. TrainingExerciseSuccess.java 217
1.109. TrainingType.java 220
1.110. ExerciseSetResult.java 221
1.111. MainActivity.java 221
1.112. SQLiteHelper,java 222
1.113. TrainingExerciseSetsObj.java 238
1.114. TrainingExerciseSuccessObj.java 238
ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 240

1. ТЕКСТ ПРОГРАММЫ

1.1. ExerciseOverviewAdapter.java

```
package com.example.testbottomnavigationbar.adapters;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.db.ExerciseOverview;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.fragments.ExerciseDescriptionFragment;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.ArrayList;
import java.util.List;

public class ExerciseOverviewAdapter extends RecyclerView.Adapter<ExerciseOverviewAdapter.ExerciseOverviewViewHolder> implements View.OnClickListener {
    private final List<ExerciseOverview> exercises;
    private final int type;
    TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public ExerciseOverviewAdapter(List<ExerciseOverview> exercises, int type, TrainingDayOfWeekHelper trainingDayOfWeekHelper) {
        this.exercises = exercises;
        this.type = type;
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    @NonNull
    @Override
    public ExerciseOverviewViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
```

```
    View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.exercise_overview,
viewGroup, false);
    view.setOnClickListener(this);
    return new ExerciseOverviewViewHolder(view);
  }

  @Override
  public void onBindViewHolder(@NonNull ExerciseOverviewViewHolder viewHolder, int i) {
    viewHolder.bind(exercises.get(i));
  }

  @Override
  public int getItemCount() {
    return exercises.size();
  }

  @Override
  public void onClick(View v) {
    ConstraintLayout constraintLayout = (ConstraintLayout) v;
    TextView textView = (TextView) constraintLayout.getChildAt(0);
    String exerciseTitle = (String) textView.getText();

    if (MainActivity.LOG) {
      Log.d(MainActivity.TEG, exerciseTitle);
    }

    Fragment fragment = new ExerciseDescriptionFragment(exerciseTitle, type,
trainingDayOfWeekHelper);
    FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
    fTrans.replace(R.id.fragment_cv, fragment).commit();
    fTrans.addToBackStack(null);
  }

  static final class ExerciseOverviewViewHolder extends RecyclerView.ViewHolder {
    private final TextView titleTextView;
    private final LinearLayout tagsLinearLayout;

    public ExerciseOverviewViewHolder(@NonNull View itemView) {
      super(itemView);
      titleTextView = itemView.findViewById(R.id.exercise_title_textview);
      tagsLinearLayout = itemView.findViewById(R.id.exercise_title_linearlayout);
    }

    private void bind(@NonNull ExerciseOverview exerciseOverview) {
      titleTextView.setText(exerciseOverview.getTitle());
```

```
            tagsLinearLayout.removeAllViews();
            ArrayList<String> tags = exerciseOverview.getTags();
            for (String tag : tags) {
                ConstraintLayout constraintLayout = (ConstraintLayout)
LayoutInflater.from(tagsLinearLayout.getContext()).inflate(R.layout.tag_layout, tagsLinearLayout, false);
                TextView textView = (TextView) constraintLayout.getChildAt(0);
                textView.setText(tag);

                tagsLinearLayout.addView(constraintLayout);
            }
        }
    }
}
```

1.2. FriendOverviewAdapter.java

```
package com.example.testbottomnavigationbar.adapters;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AccountFragment;
import com.example.testbottomnavigationbar.remote_db.Account;

import java.util.List;

public class FriendOverviewAdapter extends
RecyclerView.Adapter<FriendOverviewAdapter.FriendOverviewViewHolder> {
    private final int type;
    private final int accountId;
    private final List<Account> friendsList;

    public FriendOverviewAdapter(int type, int accountId, List<Account> friendsList) {
        this.type = type;
        this.accountId = accountId;
        this.friendsList = friendsList;
    }
```

```
    @NonNull
    @Override
    public FriendOverviewAdapter.FriendOverviewViewHolder onCreateViewHolder(@NonNull
ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.friend_overview,
viewGroup, false);

        return new FriendOverviewAdapter.FriendOverviewViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull FriendOverviewAdapter.FriendOverviewViewHolder
viewHolder, int i) {
        viewHolder.bind(friendsList.get(i));
    }

    @Override
    public int getItemCount() {
        return friendsList.size();
    }

    static final class FriendOverviewViewHolder extends RecyclerView.ViewHolder {
        private final TextView friendName;
        private final View parentView;

        public FriendOverviewViewHolder(@NonNull View itemView) {
            super(itemView);
            friendName = itemView.findViewById(R.id.friend_overview_name);
            parentView = itemView;
        }

        private void bind(@NonNull Account friendAccount) {
            friendName.setText(friendAccount.getFirstName() + " " + friendAccount.getSecondName());

            parentView.setOnClickListener(v -> {
                Fragment fragment = new AccountFragment(1, friendAccount.getAccountId());
                FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
                fTrans.replace(R.id.fragment_cv, fragment, null).commit();
                fTrans.addToBackStack(null);
            });
        }
    }
}
```

1.3. TimeTableDayAdapter.java

```
package com.example.testbottomnavigationbar.adapters;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.db.TimeTableDay;
import com.example.testbottomnavigationbar.db.TimeTableExercise;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.listeners.AddExerciseInTrainingLogic;
import com.example.testbottomnavigationbar.listeners.OpenEditingTrainingExeciseFragmentLogic;
import com.example.testbottomnavigationbar.listeners.TimeTableExerciseClickLogic;
import com.example.testbottomnavigationbar.listeners.TrainingExerciseSetClickLogic;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.ArrayList;
import java.util.List;

public class TimeTableDayAdapter extends RecyclerView.Adapter<TimeTableDayAdapter.TimeTableDayViewHolder> {
    private List<TimeTableDay> days;
    private final int type;
    private final int accountId;
    private final Training training;

    public TimeTableDayAdapter(List<TimeTableDay> days, int type, int accountId, Training training) {
        this.type = type;
        this.accountId = accountId;
        this.days = new ArrayList<>();
        this.training = training;

        if (type == 0) {
```

```
            for (TimeTableDay timeTableDay : days) {
                if (timeTableDay.getExercises().size() != 0) {
                    this.days.add(timeTableDay);
                }
            }
        } else {
            this.days = days;
        }
    }

    @NonNull
    @Override
    public TimeTableDayViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.timetable_layout, parent,
false);

        if (type != 1) {
            // hide add exercise button
            ConstraintLayout constraintLayout = (ConstraintLayout) ((ConstraintLayout) view).getChildAt(2);
            constraintLayout.setVisibility(View.GONE);
        }

        return new TimeTableDayAdapter.TimeTableDayViewHolder(view, type, accountId, training);
    }

    @Override
    public void onBindViewHolder(@NonNull TimeTableDayViewHolder holder, int position) {
        holder.bind(days.get(position));
    }

    @Override
    public int getItemCount() {
        return days.size();
    }

    static final class TimeTableDayViewHolder extends RecyclerView.ViewHolder {
        private final TextView dayOfWeek;
        private final LinearLayout exerciseLinearLayout;
        private final ConstraintLayout addExerciseCL;
        private final Training training;
        private final int type;
        private final int accountId;

        public TimeTableDayViewHolder(@NonNull View itemView, int type, int accountId, Training
training) {
            super(itemView);
```

```
        this.type = type;
        this.accountId = accountId;
        this.training = training;

        dayOfWeek = itemView.findViewById(R.id.timetable_layout_dayofweek);
        exerciseLinearLayout = itemView.findViewById(R.id.timetable_Layout_linearlayout);
        addExerciseCL = itemView.findViewById(R.id.timetable_layout_add_exercise_cl);
    }

    private void bind(@NonNull TimeTableDay timeTableDay) {
        dayOfWeek.setText(timeTableDay.getStringDayOfWeek());

        exerciseLinearLayout.removeAllViews();
        List<TimeTableExercise> exercises = timeTableDay.getExercises();
        for (TimeTableExercise exercise : exercises) {
            ConstraintLayout constraintLayout = (ConstraintLayout)
LayoutInflater.from(exerciseLinearLayout.getContext()).inflate(R.layout.timetable_exercise,
exerciseLinearLayout, false);
            TextView textView = (TextView) constraintLayout.getChildAt(0);
            textView.setText(exercise.getExerciseTitle());

            if (exercise.isLastExercise()) {
                TextView bottom_line = (TextView) constraintLayout.getChildAt(2);
                bottom_line.setVisibility(View.GONE);
            }

            exerciseLinearLayout.addView(constraintLayout);

            if (type == 0) {
                constraintLayout.setOnClickListener(new TimeTableExerciseClickLogic(new
TrainingExerciseInstance(exercise.getExerciseTitle(), timeTableDay.getTrainingId(),
timeTableDay.getDayOfWeek(), exercise.getOrderNumber())));
            } else {
                constraintLayout.setOnClickListener(new TrainingExerciseSetClickLogic(type, accountId,
new TrainingExerciseInstance(exercise.getExerciseTitle(), timeTableDay.getTrainingId(),
timeTableDay.getDayOfWeek(), exercise.getOrderNumber())));
            }

            if (type != 2) {
                constraintLayout.setOnLongClickListener(new
OpenEditingTrainingExeciseFragmentLogic(type, new
TrainingExerciseInstance(exercise.getExerciseTitle(), timeTableDay.getTrainingId(),
timeTableDay.getDayOfWeek(), exercise.getOrderNumber())));
            }
        }

        // open search fragment
```

```
            addExerciseCL.setOnClickListener(new AddExerciseInTrainingLogic(new
TrainingDayOfWeekHelper(training.getTrainingId(), timeTableDay.getDayOfWeek())));
        }
    }
}
```

1.4. TrainingAdapter.java

```
package com.example.testbottomnavigationbar.adapters;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.entities.TrainingEditingHelper;
import com.example.testbottomnavigationbar.fragments.AddExerciseTagFragment;
import com.example.testbottomnavigationbar.fragments.CopyTrainingFragment;
import com.example.testbottomnavigationbar.fragments.TimeTableFragment;
import com.example.testbottomnavigationbar.listeners.TrainingLongClickLogic;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.List;

public class TrainingAdapter extends RecyclerView.Adapter<TrainingAdapter.TrainingViewHolder> {
    private final int type;
    private final int accountId;
    private final List<Training> trainings;

    public TrainingAdapter(int type, int accountId, List<Training> trainings) {
        this.type = type;
        this.accountId = accountId;
        this.trainings = trainings;
```

```
    }

    @NonNull
    @Override
    public TrainingAdapter.TrainingViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup,
int i) {
        View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.training_overview,
viewGroup, false);
        return new TrainingAdapter.TrainingViewHolder(view, type, accountId);
    }

    @Override
    public void onBindViewHolder(@NonNull TrainingAdapter.TrainingViewHolder viewHolder, int i) {
        viewHolder.bind(trainings.get(i));
    }

    @Override
    public int getItemCount() {
        return trainings.size();
    }

    static final class TrainingViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
        private final int type;
        private final int accountId;
        private final TextView titleTextView;
        private Training training;

        public TrainingViewHolder(@NonNull View itemView, int type, int accountId) {
            super(itemView);
            titleTextView = itemView.findViewById(R.id.training_overview_textview);
            this.type = type;
            this.accountId = accountId;
        }

        private void bind(@NonNull Training training) {
            titleTextView.setText(training.getTitle());
            this.training = training;

            ConstraintLayout constraintLayout = (ConstraintLayout) titleTextView.getParent();
            constraintLayout.setOnClickListener(this);

            if (type == 1) {
                constraintLayout.setOnLongClickListener(new TrainingLongClickLogic(new
TrainingEditingHelper(training.getTrainingId(), training.getTitle())));
            } else {
                // Сделать копирование тренировки
```

```
            constraintLayout.setOnLongClickListener(v -> {
                Fragment fragment = new CopyTrainingFragment(accountId, training);
                FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
                fTrans.replace(R.id.fragment_cv, fragment).commit();
                fTrans.addToBackStack(null);
                return true;
            });
        }
    }

    @Override
    public void onClick(View v) {
        SQLiteDatabase db = v.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);

        Fragment fragment = new TimeTableFragment(SQLiteHelper.getTrainingIdFromTitle(db,
titleTextView.getText().toString(), accountId), type, accountId, training);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
  }
}
```

1.5. ExerciseOverview.java

```
package com.example.testbottomnavigationbar.db;

import java.util.ArrayList;

public class ExerciseOverview {
  private String get_all_sorted_exercises_titles;
  private final ArrayList<String> tags;

  public ExerciseOverview(String title, ArrayList<String> tags) {
    get_all_sorted_exercises_titles = title;
    this.tags = tags;
  }

  public String getTitle() {
    return get_all_sorted_exercises_titles;
  }

  public ArrayList<String> getTags() {
    return tags;
```

```
    }
}
```

1.6. ExerciseTemp.java

```
package com.example.testbottomnavigationbar.db;

public class ExerciseTemp {
    private final String title;
    private final int orderNumber;

    public ExerciseTemp(String title, int orderNumber) {
        this.title = title;
        this.orderNumber = orderNumber;
    }

    public String getTitle() {
        return title;
    }

    public int getOrderNumber() {
        return orderNumber;
    }
}
```

1.7. TimeTableDay.java

```
package com.example.testbottomnavigationbar.db;

import java.util.ArrayList;
import java.util.List;

public class TimeTableDay {
    private final int trainingId;
    private final int dayOfWeek;
    private final List<TimeTableExercise> exercises;

    public TimeTableDay(int trainingId, int dayOfWeek, List<TimeTableExercise> exercises) {
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
        this.exercises = exercises;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }
```

```
    public String getStringDayOfWeek() {
        String[] ans = new String[] {"Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота", "Воскресенье"};
        return ans[dayOfWeek];
    }

    public List<TimeTableExercise> getExercises() {
        return exercises;
    }

    public int getTrainingId() {
        return trainingId;
    }
}
```

1.8. TimeTableExercise.java

```
package com.example.testbottomnavigationbar.db;

public class TimeTableExercise {
    private final String exerciseTitle;
    private final boolean isLastExercise;
    private final int orderNumber;

    public TimeTableExercise(String exerciseTitle, boolean isLastExercise, int orderNumber) {
        this.exerciseTitle = exerciseTitle;
        this.isLastExercise = isLastExercise;
        this.orderNumber = orderNumber;
    }

    public String getExerciseTitle() {
        return exerciseTitle;
    }

    public boolean isLastExercise() {
        return isLastExercise;
    }

    public int getOrderNumber() {
        return orderNumber;
    }
}
```

1.9. ExerciseInTrainingEditing.java

```
package com.example.testbottomnavigationbar.entities;

public class ExerciseInTrainingEditing {
    private final int trainingId;
    private final int exerciseId;
    private final int orderNumber;
    private final int setNumber;
    private final int accountId;
    private final int dayOfWeek;

    public ExerciseInTrainingEditing(int trainingId, int exerciseId, int orderNumber, int setNumber, int accountId, int dayOfWeek) {
        this.trainingId = trainingId;
        this.exerciseId = exerciseId;
        this.orderNumber = orderNumber;
        this.setNumber = setNumber;
        this.accountId = accountId;
        this.dayOfWeek = dayOfWeek;
    }

    public int getTrainingId() {
        return trainingId;
    }

    public int getExerciseId() {
        return exerciseId;
    }

    public int getOrderNumber() {
        return orderNumber;
    }

    public int getSetNumber() {
        return setNumber;
    }

    public int getAccountId() {
        return accountId;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }
}
```

1.10. InsertSetJSON

```
package com.example.testbottomnavigationbar.entities;

public class InsertSetJSON {
    private final int weight;
    private final int reps;
    private final int timer;

    public InsertSetJSON(int weight, int reps, int timer) {
        this.weight = weight;
        this.reps = reps;
        this.timer = timer;
    }

    public InsertSetJSON(int reps, int timer) {
        weight = -1;
        this.reps = reps;
        this.timer = timer;
    }

    public String getJSON() {
        if (weight != -1) {
            return "{ " +
                    "\"weight\" : " + weight + ", " +
                    "\"repsnum\" : " + reps + ", " +
                    "\"timer\" : " + timer +
                    " }";
        } else {
            return "{ " +
                    "\"repsnum\" : " + reps + ", " +
                    "\"timer\" : " + timer +
                    " }";
        }
    }
}
```

1.11. TrainingDayOfWeekHelper.java

```
package com.example.testbottomnavigationbar.entities;

public class TrainingDayOfWeekHelper {
    private final int trainingId;
    private final int dayOfWeek;

    public TrainingDayOfWeekHelper(int trainingId, int dayOfWeek) {
        this.trainingId = trainingId;
```

```
        this.dayOfWeek = dayOfWeek;
    }

    public int getTrainingId() {
        return trainingId;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }
}
```

1.12. TrainingEditingHelper.java

```
package com.example.testbottomnavigationbar.entities;

public class TrainingEditingHelper {
    private final int trainingId;
    private final String title;

    public TrainingEditingHelper(int trainingId, String title) {
        this.trainingId = trainingId;
        this.title = title;
    }

    public int getTrainingId() {
        return trainingId;
    }

    public String getTitle() {
        return title;
    }
}
```

1.13. TrainingExerciseInstance.java

```
package com.example.testbottomnavigationbar.entities;

public class TrainingExerciseInstance {
    private final String exerciseTitle;
    private final int trainingId;
    private final int dayOfWeek;
    private final int orderNumber;

    public TrainingExerciseInstance(String exerciseTitle, int trainingId, int dayOfWeek, int orderNumber)
{
        this.exerciseTitle = exerciseTitle;
```

```
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
        this.orderNumber = orderNumber;
    }

    public String getExerciseTitle() {
        return exerciseTitle;
    }

    public int getTrainingId() {
        return trainingId;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }

    public int getOrderNumber() {
        return orderNumber;
    }
}
```

1.14. AccountEditingFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.SaveAccountEditingLogic;
import com.example.testbottomnavigationbar.remote_db.Account;
```

```
import com.example.testbottomnavigationbar.remote_db.BodyCondition;

import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.Locale;

public class AccountEditingFragment extends Fragment {
    private final Account account;
    private final BodyCondition bodyCondition;

    public AccountEditingFragment(Account account, BodyCondition bodyCondition) {
        super(R.layout.account_editing);
        this.account = account;
        this.bodyCondition = bodyCondition;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.account_editing, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneET();
        tuneSaveButton();
    }

    private void tuneSaveButton() {
        TextView saveButton = getView().findViewById(R.id.account_editing_save);
        saveButton.setOnClickListener(new SaveAccountEditingLogic(account, bodyCondition));
    }

    private void tuneET() {
        tuneFirstName();
        tuneSecondName();
        tuneUsername();
        tuneAge();
        tuneWeight();
        tuneHeight();
        tuneBodyFatShare();
    }

    private void tuneFirstName() {
        EditText firstName = getView().findViewById(R.id.account_editing_name_edit);
        firstName.setText(account.getFirstName());
```

```
        setHideKeyBoard(firstName);
    }

    private void tuneSecondName() {
        EditText secondName = getView().findViewById(R.id.account_editing_second_name_edit);
        secondName.setText(account.getSecondName());

        setHideKeyBoard(secondName);
    }

    private void tuneUsername() {
        EditText username = getView().findViewById(R.id.account_editing_second_username_edit);
        username.setText(account.getUserName());

        setHideKeyBoard(username);
    }

    private void tuneAge() {
        EditText age = getView().findViewById(R.id.account_editing_age_edit);
        age.setText(Integer.valueOf(bodyCondition.getAge()).toString());

        setHideKeyBoard(age);
    }

    private void tuneWeight() {
        EditText width = getView().findViewById(R.id.account_editing_weight_edit);
        DecimalFormat decimalFormat = new DecimalFormat("###.##", new
DecimalFormatSymbols(Locale.US));
        width.setText(decimalFormat.format(bodyCondition.getWeight()));

        setHideKeyBoard(width);
    }

    private void tuneHeight() {
        EditText height = getView().findViewById(R.id.account_editing_height_edit);
        DecimalFormat decimalFormat = new DecimalFormat("###.##", new
DecimalFormatSymbols(Locale.US));
        height.setText(decimalFormat.format(bodyCondition.getHeight()));

        setHideKeyBoard(height);
    }

    private void tuneBodyFatShare() {
        EditText bodyFatShare = getView().findViewById(R.id.account_editing_percent_edit);
        DecimalFormat decimalFormat = new DecimalFormat("###.##", new
DecimalFormatSymbols(Locale.US));
        bodyFatShare.setText(decimalFormat.format(bodyCondition.getBodyFatShare()));
```

```
        setHideKeyBoard(bodyFatShare);
    }

    private void setHideKeyBoard(EditText editText) {
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_toolbar);
        toolbar.setVisibility(View.VISIBLE);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.15. AccountFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.listeners.AccountEditingLogic;
import com.example.testbottomnavigationbar.listeners.AddFriendLogic;
import com.example.testbottomnavigationbar.listeners.RemoveFriendLogic;
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;

import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.Locale;

public class AccountFragment extends Fragment {
    private final int type;
    private final int accountId;
    private Account account;
    private BodyCondition bodyCondition;

    public AccountFragment(int type, int accountId) {
        super(R.layout.fragment_account);
        this.type = type;
        this.accountId = accountId;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_account, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        account = SQLiteHelper.getAccountFromId(db, accountId);
        bodyCondition = SQLiteHelper.getBodyConditionFromAccountId(db, accountId);

        tuneToolbar(view);
        tuneEditButton();
        tuneLogOutButton();
```

```
        tuneTrainingsButton();
        tuneFriendsButton();
        tuneAddFriend();
        fillInfo();
    }

    private void tuneAddFriend() {
        SQLiteDatabase db = getView().getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
getView().getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
        ImageView imageView = getView().findViewById(R.id.fragment_account_add_friend);
        ImageView imageView1 = getView().findViewById(R.id.fragment_account_remove_friend);

        if (type == 0) {
            imageView.setVisibility(View.GONE);
            imageView1.setVisibility(View.GONE);
        } else if (!SQLiteHelper.isFriend(db, currentAccountDB, accountId)) {
            imageView1.setVisibility(View.GONE);
            imageView.setVisibility(View.VISIBLE);
            imageView.setOnClickListener(new
AddFriendLogic(SQLiteHelper.getCurrentAccountId(currentAccountDB), accountId));
        } else {
            imageView.setVisibility(View.GONE);
            imageView1.setVisibility(View.VISIBLE);
            imageView1.setOnClickListener(new
RemoveFriendLogic(SQLiteHelper.getCurrentAccountId(currentAccountDB), accountId));
        }
    }

    private void tuneTrainingsButton() {
        ConstraintLayout constraintLayout =
getView().findViewById(R.id.fragment_account_training_constraintlayout);

        if (type == 0) {
            constraintLayout.setVisibility(View.GONE);

            TextView textView = getView().findViewById(R.id.fragment_account_trainings_bottom_stroke);
            textView.setVisibility(View.GONE);
        } else {
            constraintLayout.setOnClickListener(v -> {
                Fragment fragment = new TrainingsFragment(2, accountId);
                FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
                fTrans.replace(R.id.fragment_cv, fragment, null).commit();
                fTrans.addToBackStack(null);
```

```
        });

        TextView friendsCount = getView().findViewById(R.id.fragment_account_trainings_count);
        int trainingsCnt = getTrainingCnt();
        friendsCount.setText(Integer.valueOf(trainingsCnt).toString());
    }
}

private void tuneFriendsButton() {
    TextView friendsCount = getView().findViewById(R.id.fragment_account_friends_count);
    int friendsCnt = getFriendCnt();
    friendsCount.setText(Integer.valueOf(friendsCnt).toString());

    ConstraintLayout constraintLayout = getView().findViewById(R.id.fragment_account_friends_cl);
    constraintLayout.setOnClickListener(v -> {
        Fragment fragment = new FriendsListFragment(type, accountId,
FriendsListFragment.generateFriendsListFromAccountId(v.getContext(), accountId));
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment, null).commit();
        fTrans.addToBackStack(null);
    });
}

private int getFriendCnt() {
    SQLiteDatabase db = getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
    return SQLiteHelper.getFriendsCnt(db, accountId);
}

private int getTrainingCnt() {
    SQLiteDatabase db = getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
    return SQLiteHelper.getTrainingCnt(db, accountId);
}

private void tuneEditButton() {
    TextView editButton = getView().findViewById(R.id.fragment_account_edit);

    if (type != 0) {
        editButton.setVisibility(View.GONE);
    } else {
        editButton.setOnClickListener(new AccountEditingLogic(account, bodyCondition));
    }
}

private void fillInfo() {
```

```
        fillName();
        fillUsername();
        fillAge();
        fillHeight();
        fillWidth();
        fillBodyFatShare();
    }

    private void fillName() {
        TextView name = getView().findViewById(R.id.fragment_account_name);
        String strName = account.getFirstName() + " " + account.getSecondName();
        name.setText(strName);
    }

    private void fillUsername() {
        TextView username = getView().findViewById(R.id.fragment_account_username);
        String strUsername = "@" + account.getUserName();
        username.setText(strUsername);
    }

    private void fillAge() {
        TextView age = getView().findViewById(R.id.fragment_account_age);
        String strAge = "Возраст: " + Integer.valueOf(bodyCondition.getAge()).toString();
        age.setText(strAge);
    }

    private void fillHeight() {
        TextView height = getView().findViewById(R.id.fragment_account_height);
        DecimalFormat decimalFormat = new DecimalFormat("###.##", new
DecimalFormatSymbols(Locale.US));
        String strHeight = "Рост: " + decimalFormat.format(bodyCondition.getHeight()) + " см";
        height.setText(strHeight);
    }

    private void fillWidth() {
        TextView width = getView().findViewById(R.id.fragment_account_weight);
        DecimalFormat decimalFormat = new DecimalFormat("###.##", new
DecimalFormatSymbols(Locale.US));
        String strWidth = "Вес: " + decimalFormat.format(bodyCondition.getWeight()) + " кг";
        width.setText(strWidth);
    }

    private void fillBodyFatShare() {
        TextView bodyFatShare = getView().findViewById(R.id.fragmnet_account_percent);
        DecimalFormat decimalFormat = new DecimalFormat("###.##", new
DecimalFormatSymbols(Locale.US));
```

```
        String strBodyFatShare = "Процент жира: " +
decimalFormat.format(bodyCondition.getBodyFatShare()) + " %";
        bodyFatShare.setText(strBodyFatShare);
    }

    private void tuneLogOutButton() {
        ImageView logOutButton = getView().findViewById(R.id.fragment_account_logout_iv);

        if (type == 0) {
            logOutButton.setOnClickListener(v -> {
//                SQLiteDatabase currentAccountDb =
getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
//                currentAccountDb.execSQL("DELETE FROM CurrentAccount");
//
//                Fragment fragment = new LoginFragment();
//                FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
//                ((FragmentActivity)
v.getContext()).getSupportFragmentManager().popBackStackImmediate(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
//                fTrans.replace(R.id.activity_main_fragment_cv, fragment, "loginFragment").commit();
//                fTrans.addToBackStack(null);

                Fragment fragment = new LogOutFragment(accountId);
                FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
                fTrans.replace(R.id.activity_main_fragment_cv, fragment, "loginFragment").commit();
                fTrans.addToBackStack(null);
            });
        } else {
            logOutButton.setVisibility(View.GONE);
        }
    }

    private void tuneToolbar(View view) {
        if (type == 0) {
            return;
        }

        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_account_toolbar);
        toolbar.setVisibility(View.VISIBLE);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
```

```
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.16. AddExerciseDescriptionFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.NewExerciseDescriptionLogic;

import java.util.HashSet;

import static androidx.core.content.ContextCompat.getSystemService;

public class AddExerciseDescriptionFragment extends Fragment {
    private HashSet<String> tags;
    private HashSet<String> filters;
    private String title;

    public AddExerciseDescriptionFragment(HashSet<String> tags, HashSet<String> filters, String title) {
        super(R.layout.fragment_add_exercise_description);
        this.tags = tags;
        this.filters = filters;
```

```
        this.title = title;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_add_exercise_description, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneDone();

        EditText editText = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_add_exercise_description_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneDone() {
        getView().findViewById(R.id.done).setOnClickListener(new NewExerciseDescriptionLogic(tags, filters, title));
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_add_exercise_description_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity) v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity) view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.17. AddExerciseMuscleFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.NewExerciseFilterLogic;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.remote_db.TargetMuscle;

import java.util.HashSet;
import java.util.List;

public class AddExerciseMuscleFragment extends Fragment implements View.OnClickListener {
    NewExerciseFilterLogic newExerciseFilterLogic;

    public AddExerciseMuscleFragment(HashSet<String> tags) {
        super(R.layout.fragment_add_exercise_muscle);
        newExerciseFilterLogic = new NewExerciseFilterLogic(tags);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_add_exercise_muscle, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        handleFilterTypes();
```

```
        handleDoneButton();
        tuneToolbar(view);
        tuneEditText(view);
    }

    private void tuneEditText(View view) {
        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_title_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void handleDoneButton() {
        getView().findViewById(R.id.apply).setOnClickListener(newExerciseFilterLogic);
    }

    private void handleFilterTypes() {
//        ConstraintLayout constraintLayout = getView().findViewById(R.id.training_type_cl);
//        setCLChildsListener(constraintLayout);

        ConstraintLayout constraintLayout = getView().findViewById(R.id.chest_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.biceps_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.triceps_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.back_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.shoulders_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.hip_biceps_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.thigh_quads_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.press_cl);
        setCLChildsListener(constraintLayout);
    }
```

```
    private void setCLChildsListener(ConstraintLayout constraintLayout) {
        for (int i = 1; i < constraintLayout.getChildCount() - 1; ++i) {
            constraintLayout.getChildAt(i).setOnClickListener(this);
        }
    }

    @Override
    public void onClick(View v) {
        if (v instanceof TextView) {
            if (((TextView) v).getText().equals("Все")) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Все  " + ((ConstraintLayout) v.getParent()).getChildCount());
                }

                ConstraintLayout constraintLayout = (ConstraintLayout) v.getParent();
                for (int i = 2; i < constraintLayout.getChildCount() - 1; ++i) {
                    TextView textView = (TextView) constraintLayout.getChildAt(i);

                    if (textView.getCurrentTextColor() == Color.WHITE) {
                        newExerciseFilterLogic.addFilter((String) textView.getText());
                        textView.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_select_type_background));
                        textView.setTextColor(Color.BLACK);
                    }
                }
            } else {
                if (((TextView) v).getCurrentTextColor() == Color.WHITE) {
                    newExerciseFilterLogic.addFilter((String) ((TextView) v).getText());
                    v.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_select_type_background));
                    ((TextView) v).setTextColor(Color.BLACK);

                    if (MainActivity.LOG) {
                        Log.d(MainActivity.TEG, "Add filter");
                    }
                } else {
                    newExerciseFilterLogic.removeFilter((String) ((TextView) v).getText());
                    v.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_type_background));
                    ((TextView) v).setTextColor(Color.WHITE);

                    if (MainActivity.LOG) {
                        Log.d(MainActivity.TEG, "Remove filter");
                    }
                }
            }
```

```
        }
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_add_exercise_muscle_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity) v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity) view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.18. AddExerciseTagFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.NewExerciseTagsLogic;

public class AddExerciseTagFragment extends Fragment implements View.OnClickListener {
    NewExerciseTagsLogic newExerciseTagsLogic;
```

```
    public AddExerciseTagFragment() {
        super(R.layout.fragment_add_exercise_tag);
        newExerciseTagsLogic = new NewExerciseTagsLogic();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_add_exercise_tag, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        handleFilterTypes();
        handleDoneButton();
        tuneToolbar(view);
    }

    private void handleDoneButton() {
        getView().findViewById(R.id.apply).setOnClickListener(newExerciseTagsLogic);
    }

    private void handleFilterTypes() {
        ConstraintLayout constraintLayout = getView().findViewById(R.id.tag_cl);
        setCLChildsListener(constraintLayout);
    }

    private void setCLChildsListener(ConstraintLayout constraintLayout) {
        for (int i = 3; i < constraintLayout.getChildCount() - 1; ++i) {
            constraintLayout.getChildAt(i).setOnClickListener(this);
        }
    }

    @Override
    public void onClick(View v) {
        if (v instanceof TextView) {
            if (((TextView) v).getCurrentTextColor() == Color.WHITE) {
                newExerciseTagsLogic.addTag((String) ((TextView) v).getText());
                v.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_select_type_background));
                ((TextView) v).setTextColor(Color.BLACK);

                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Add tag");
                }
            } else {
                newExerciseTagsLogic.removeTag((String) ((TextView) v).getText());
```

```
            v.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_type_background));
            ((TextView) v).setTextColor(Color.WHITE);

            if (MainActivity.LOG) {
               Log.d(MainActivity.TEG, "Remove tag");
            }
         }
      }
   }

   private void tuneToolbar(View view) {
      Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_tag_toolbar);
      ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

      if (MainActivity.LOG) {
         Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
      }

      toolbar.setNavigationIcon(R.drawable.ic_back);
      toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
      ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
   }
}
```

1.19. AddExerciseVideoFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
```

```
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.NewExerciseDescriptionLogic;
import com.example.testbottomnavigationbar.listeners.NewExerciseVideoLogic;

import java.util.HashSet;

public class AddExerciseVideoFragment extends Fragment {
    private HashSet<String> tags;
    private HashSet<String> filters;
    private String title;
    private String description;

    public AddExerciseVideoFragment(HashSet<String> tags, HashSet<String> filters, String title, String
description) {
        super(R.layout.fragment_add_exercise_videopath);
        this.tags = tags;
        this.filters = filters;
        this.title = title;
        this.description = description;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_add_exercise_videopath, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneDone();

        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_video_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneDone() {
        getView().findViewById(R.id.done).setOnClickListener(new NewExerciseVideoLogic(tags, filters,
title, description));
    }

    private void tuneToolbar(View view) {
```

```
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_video_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.20. AddSetFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.AddSetSaveLogic;

public class AddSetFragment extends Fragment {
    private final int trainingId;
    private final int dayOfWeek;
    private final int orderNumber;
    private final int exerciseId;
    private final int setNumber;
```

```
    private final int type;

    public AddSetFragment(int trainingId, int dayOfWeek, int orderNumber, int exerciseId, int setNumber,
int type) {
        super(R.layout.add_set_layout);

        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
        this.orderNumber = orderNumber;
        this.exerciseId = exerciseId;
        this.setNumber = setNumber;
        this.type = type;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.add_set_layout, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        if (type == 1) {
            hideWeight(view);
        }

        tuneET();
        tuneToolbar(view);
        tuneSaveButton(view);
    }

    private void tuneET() {
        EditText editText = getView().findViewById(R.id.add_set_layout_weight_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        editText = getView().findViewById(R.id.add_set_layout_count_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        editText = getView().findViewById(R.id.add_set_layout_timer_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
```

```
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void hideWeight(View view) {
        ImageView imageView = ((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_weight_ic);
        imageView.setVisibility(View.INVISIBLE);

        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_weight_et);
        editText.setVisibility(View.INVISIBLE);

        TextView textView = ((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_weight_tv);
        textView.setVisibility(View.INVISIBLE);
    }

    private void tuneSaveButton(View view) {
        TextView textView =
((FragmentActivity)view.getContext()).findViewById(R.id.add_set_layout_save);
        textView.setOnClickListener(new AddSetSaveLogic(trainingId, dayOfWeek, orderNumber,
exerciseId, setNumber, type));
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.21. CopyTrainingFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.CopyTrainingLogic;
import com.example.testbottomnavigationbar.remote_db.Training;

public class CopyTrainingFragment extends Fragment {
    private final int friendAccountId;
    private final Training training;

    public CopyTrainingFragment(int friendAccountId, Training training) {
        super(R.layout.fragment_copy_training);
        this.friendAccountId = friendAccountId;
        this.training = training;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_copy_training, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneET();
        tuneCopyButton();
    }

    private void tuneCopyButton() {
```

```
        TextView copyButton = getView().findViewById(R.id.done);
        copyButton.setOnClickListener(new CopyTrainingLogic(friendAccountId, training));
    }

    private void tuneET() {
        EditText editText = getView().findViewById(R.id.fragment_add_training_title_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_copy_training_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.22. CreatingTrainingFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
```

```
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.adapters.TrainingAdapter;
import com.example.testbottomnavigationbar.listeners.NewTrainingCreationLogic;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.List;

public class CreatingTrainingFragment extends Fragment {
    public CreatingTrainingFragment() {
        super(R.layout.fragment_creating_training);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_creating_training, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneEditText(view);
        handleDoneButton();
    }

    private void handleDoneButton() {
        getView().findViewById(R.id.done).setOnClickListener(new NewTrainingCreationLogic());
    }

    private void tuneEditText(View view) {
        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_training_title_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_creating_training_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);
```

```
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.23. EditingTrainingExerciseFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.listeners.DeleteTrainingExerciseLogic;
import com.example.testbottomnavigationbar.listeners.MoveTrainingExerciseLogic;

public class EditingTrainingExerciseFragment extends Fragment {
    private final int type;
    private final TrainingExerciseInstance trainingExerciseInstance;

    public EditingTrainingExerciseFragment(int type, TrainingExerciseInstance trainingExerciseInstance) {
        super(R.layout.fragment_editing_training_exercise);
        this.type = type;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_editing_training_exercise, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneEditText(getView().findViewById(R.id.fragment_editing_training_exercise_et));
        tuneMoveButton();
        tuneDeleteButton();
    }

    private void tuneEditText(EditText editText) {
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneMoveButton() {
        TextView moveButton = getView().findViewById(R.id.fragment_training_editing_edit);
        moveButton.setOnClickListener(new MoveTrainingExerciseLogic(type, trainingExerciseInstance));
    }

    private void tuneDeleteButton() {
        TextView deleteButton = getView().findViewById(R.id.fragment_editing_training_exercise_delete);
        deleteButton.setOnClickListener(new DeleteTrainingExerciseLogic(type, trainingExerciseInstance));
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_editing_training_exercise_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.24. ExerciseDescriptionFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.listeners.AddExerciseFromDescriptionLogic;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.ArrayList;
import java.util.List;

public class ExerciseDescriptionFragment extends Fragment {
    private final String exerciseTitle;
    private int type;
    private TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public ExerciseDescriptionFragment(String title, int type, TrainingDayOfWeekHelper
trainingDayOfWeekHelper) {
        super(R.layout.fragment_exercise_description);
        exerciseTitle = title;
        this.type = type;
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    @Override
```

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_exercise_description, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        addContent(view.getContext());

        if (type == 1) {
            handleType();
        }
    }

    private void handleType() {
        ImageView imageView = getView().findViewById(R.id.fragment_exercise_description_add);
        imageView.setVisibility(View.VISIBLE);
        imageView.setOnClickListener(new
AddExerciseFromDescriptionLogic(trainingDayOfWeekHelper, exerciseTitle));
    }

    private void addContent(Context context) {
        String description = "", videoPath = "";
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        Cursor cursor = db.rawQuery("SELECT * FROM Exercise WHERE Title = '" + exerciseTitle + "';",
null);
        if (cursor != null && cursor.moveToFirst()) {
            description = cursor.getString(2);
            videoPath = cursor.getString(3);
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "videoPath   " + videoPath);
        }

        TextView title = ((FragmentActivity)
context).findViewById(R.id.fragment_exercise_description_title);
        title.setText(exerciseTitle);

        TextView textDescription = ((FragmentActivity)
context).findViewById(R.id.fragment_exercise_description_description);
        textDescription.setText(description);

        TextView textVideoPath = ((FragmentActivity)
context).findViewById(R.id.fragment_exercise_description_video);
        textVideoPath.setText(videoPath);
```

```
        TextView textTargetMuscles = ((FragmentActivity) context).findViewById(R.id.fragment_exercise_description_target_muscles);
        List<String> targetMuscles = getExerciseMuscles(db, SQLiteHelper.getExerciseIdFromTitle(db, exerciseTitle));
        for (String targetMuscle : targetMuscles) {
            String text = (String) textTargetMuscles.getText();
            if (text.length() != 0) {
                text += "\n";
            }
            text += "- " + targetMuscle;

            textTargetMuscles.setText(text);
        }
    }

    private List<String> getExerciseMuscles(SQLiteDatabase db, int exerciseId) {
        List<String> targetMuscles = new ArrayList<>();

        Cursor muscleIdCursor = db.rawQuery("SELECT MuscleId FROM TargetMuscle WHERE ExerciseId = " + exerciseId + ";", null);
        if (muscleIdCursor != null && muscleIdCursor.moveToFirst()) {
            do {
                int muscleId = muscleIdCursor.getInt(0);

                Cursor muscleTitleCursor = db.rawQuery("SELECT Title FROM Muscle WHERE MuscleId = " + muscleId + ";", null);
                if (muscleTitleCursor != null && muscleTitleCursor.moveToFirst()) {
                    targetMuscles.add(muscleTitleCursor.getString(0));
                }
            } while (muscleIdCursor.moveToNext());
        }

        return targetMuscles;
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity) view.getContext()).findViewById(R.id.my_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity) v1.getContext()).getSupportFragmentManager().popBackStack());
```

```
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.25. ExerciseSetEditingFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.listeners.AddSetSaveLogic;
import com.example.testbottomnavigationbar.listeners.SetEditLogic;
import com.example.testbottomnavigationbar.listeners.SetRemoveLogic;

public class ExerciseSetEditingFragment extends Fragment {
    private final int type;
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public ExerciseSetEditingFragment(int type, ExerciseInTrainingEditing exerciseInTrainingEditing) {
        super(R.layout.fragment_exercise_set_editing);
        this.type = type;
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
```

```
        return inflater.inflate(R.layout.fragment_exercise_set_editing, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneEditButton(view);
        tuneRemoveButton(view);

        tuneEditText(((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_weight_et));
        tuneEditText(((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_count_et));
        tuneEditText(((FragmentActivity)
view.getContext()).findViewById(R.id.add_set_layout_timer_et));

        if (type == 1) {
            // hide weight
            getView().findViewById(R.id.add_set_layout_weight_ic).setVisibility(View.GONE);
            getView().findViewById(R.id.add_set_layout_weight_et).setVisibility(View.GONE);
            getView().findViewById(R.id.add_set_layout_weight_tv).setVisibility(View.GONE);
        }
    }

    private void tuneEditText(EditText editText) {
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneEditButton(View view) {
        TextView                                    textView                                    =
((FragmentActivity)view.getContext()).findViewById(R.id.add_set_layout_save);
        textView.setOnClickListener(new SetEditLogic(type, exerciseInTrainingEditing));
    }

    private void tuneRemoveButton(View view) {
        TextView                                    textView                                    =
((FragmentActivity)view.getContext()).findViewById(R.id.add_set_layout_delete);

        if (!needToShow(view)) {
            textView.setVisibility(View.GONE);
        }

        textView.setOnClickListener(new SetRemoveLogic(type, exerciseInTrainingEditing));
```

```
    }

    private boolean needToShow(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        int maxSetNumber = -1;

        if (type == 0) {
            maxSetNumber = SQLiteHelper.getMaxSetNumberFromTrainingExerciseSuccess(db,
                    exerciseInTrainingEditing.getAccountId(), exerciseInTrainingEditing.getTrainingId(),
                    exerciseInTrainingEditing.getDayOfWeek(), exerciseInTrainingEditing.getOrderNumber());
        }
        else {
            maxSetNumber = SQLiteHelper.getMaxSetNumberFromTrainingExercise(db,
                    exerciseInTrainingEditing.getAccountId(), exerciseInTrainingEditing.getTrainingId(),
                    exerciseInTrainingEditing.getDayOfWeek(), exerciseInTrainingEditing.getOrderNumber());
        }

        return (maxSetNumber == exerciseInTrainingEditing.getSetNumber());
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_exercise_set_editing_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.26. FilterFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.listeners.FilterApplyLogic;
import com.example.testbottomnavigationbar.listeners.FilterCancelLogic;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.remote_db.Training;

public class FilterFragment extends Fragment implements View.OnClickListener {
    private FilterApplyLogic filterLogic;
    private int type;
    private final TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public FilterFragment(int type, TrainingDayOfWeekHelper trainingDayOfWeekHelper) {
        super(R.layout.filters_layout);
        this.type = type;
        filterLogic = new FilterApplyLogic(type, trainingDayOfWeekHelper);
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.filters_layout, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        handleCancelButton();
        handleFilterTypes();
        handleApplyButton();
    }

    private void handleApplyButton() {
        getView().findViewById(R.id.apply).setOnClickListener(filterLogic);
    }

    private void handleCancelButton() {
```

```
        getView().findViewById(R.id.filter_layout_cancel).setOnClickListener(new FilterCancelLogic(type,
trainingDayOfWeekHelper));
    }

    private void handleFilterTypes() {
//        ConstraintLayout constraintLayout = getView().findViewById(R.id.training_type_cl);
//        setCLChildsListener(constraintLayout);

        ConstraintLayout constraintLayout = getView().findViewById(R.id.chest_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.biceps_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.triceps_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.back_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.shoulders_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.hip_biceps_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.thigh_quads_cl);
        setCLChildsListener(constraintLayout);

        constraintLayout = getView().findViewById(R.id.press_cl);
        setCLChildsListener(constraintLayout);
    }

    private void setCLChildsListener(ConstraintLayout constraintLayout) {
        for (int i = 1; i < constraintLayout.getChildCount() - 1; ++i) {
            constraintLayout.getChildAt(i).setOnClickListener(this);
        }
    }

    @Override
    public void onClick(View v) {
        if (v instanceof TextView) {
            if (((TextView) v).getText().equals("Все")) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Все  " + ((ConstraintLayout) v.getParent()).getChildCount());
                }
```

```
            ConstraintLayout constraintLayout = (ConstraintLayout) v.getParent();
            for (int i = 2; i < constraintLayout.getChildCount() - 1; ++i) {
               TextView textView = (TextView) constraintLayout.getChildAt(i);

               if (textView.getCurrentTextColor() == Color.WHITE) {
                  filterLogic.addFilter((String) textView.getText());
                  textView.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_select_type_background));
                  textView.setTextColor(Color.BLACK);
               }
            }
         } else {
            if (((TextView) v).getCurrentTextColor() == Color.WHITE) {
               filterLogic.addFilter((String) ((TextView) v).getText());
               v.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_select_type_background));
               ((TextView) v).setTextColor(Color.BLACK);

               if (MainActivity.LOG) {
                  Log.d(MainActivity.TEG, "Add filter");
               }
            } else {
               filterLogic.removeFilter((String) ((TextView) v).getText());
               v.setBackground(ContextCompat.getDrawable(v.getContext(),
R.drawable.filter_type_background));
               ((TextView) v).setTextColor(Color.WHITE);

               if (MainActivity.LOG) {
                  Log.d(MainActivity.TEG, "Remove filter");
               }
            }
         }
      }
   }
}
```

1.27. FriendsListFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.adapters.ExerciseOverviewAdapter;
import com.example.testbottomnavigationbar.adapters.FriendOverviewAdapter;
import com.example.testbottomnavigationbar.remote_db.Account;

import java.util.ArrayList;
import java.util.List;

public class FriendsListFragment extends Fragment {
    private final int type;
    private final int accountId;
    private List<Account> friendsList;

    public FriendsListFragment(int type, int accountId, List<Account> friendsList) {
        super(R.layout.fragment_friends_search);
        this.type = type;
        this.accountId = accountId;
        this.friendsList = friendsList;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_friends_search, container, false);
    }

    private void fillRecyclerView() {
        RecyclerView recyclerView =
getView().findViewById(R.id.fragment_friends_search_recyclerview);
        FriendOverviewAdapter friendOverviewAdapter = new FriendOverviewAdapter(type, accountId,
friendsList);
```

```
        recyclerView.setAdapter(friendOverviewAdapter);

        LinearLayoutManager mLayoutManager = new LinearLayoutManager(getContext(),
RecyclerView.VERTICAL, false);
        recyclerView.setLayoutManager(mLayoutManager);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneEditText(view.getContext());

        fillRecyclerView();
    }

    private void tuneEditText(Context context) {
        EditText editText = getView().findViewById(R.id.fragment_friends_search_et);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        editText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "EditText " + s);
                }

                friendsList = generateFriendsListFromPrefix(context, s.toString(), accountId);
                fillRecyclerView();
            }

            @Override
            public void afterTextChanged(Editable s) {
            }
        });
    }

    public static List<Account> generateFriendsListFromAccountId(Context context, int accountId) {
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
```

```
        SQLiteDatabase currentAccountDB = context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(), Context.MODE_PRIVATE, null);
        int currentAccountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        List<Account> friendsList = new ArrayList<>();

        Cursor friendsCursor = db.rawQuery("SELECT FriendId FROM Friendship WHERE AccountId = " + accountId + " AND FriendId <> " + currentAccountId + ";", null);
        if (friendsCursor != null && friendsCursor.moveToFirst()) {
            do {
                int friendId = friendsCursor.getInt(0);

                Cursor friendAccountCursor = db.rawQuery("SELECT * FROM Account WHERE AccountId = " + friendId + ";", null);
                if (friendAccountCursor != null && friendAccountCursor.moveToFirst()) {
                    friendsList.add(new Account(
                            friendId,
                            friendAccountCursor.getString(1),
                            friendAccountCursor.getString(2),
                            friendAccountCursor.getString(3),
                            friendAccountCursor.getString(4),
                            friendAccountCursor.getString(5),
                            (friendAccountCursor.isNull(6) ? null : friendAccountCursor.getInt(6)),
                            friendAccountCursor.getString(7),
                            friendAccountCursor.getString(8)
                    ));
                }
            } while (friendsCursor.moveToNext());
        }

        return friendsList;
    }

    public static List<Account> generateFriendsListFromPrefix(Context context, String prefix, int accountId) {
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);

        SQLiteDatabase currentAccountDB = context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(), Context.MODE_PRIVATE, null);
        int currentAccountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        List<Account> friendsList = new ArrayList<>();
```

```
//        Cursor friendsCursor = db.rawQuery("SELECT FriendId FROM Friendship WHERE AccountId =
" + accountId + " AND Username LIKE '" + prefix + "%';", null);
        Cursor friendAccountCursor = db.rawQuery("SELECT * FROM Account WHERE AccountId <> " +
accountId + " AND Username LIKE '" + prefix + "%' AND AccountId <> " + currentAccountId +";", null);
        if (friendAccountCursor != null && friendAccountCursor.moveToFirst()) {
            do {
                friendsList.add(new Account(
                        friendAccountCursor.getInt(0),
                        friendAccountCursor.getString(1),
                        friendAccountCursor.getString(2),
                        friendAccountCursor.getString(3),
                        friendAccountCursor.getString(4),
                        friendAccountCursor.getString(5),
                        (friendAccountCursor.isNull(6) ? null : friendAccountCursor.getInt(6)),
                        friendAccountCursor.getString(7),
                        friendAccountCursor.getString(8)
                ));
            } while (friendAccountCursor.moveToNext());
        }

        return friendsList;
    }

    private void tuneToolbar(View view) {
        Toolbar                    toolbar                    =                    ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_friends_search_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1                    ->                    ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.28. LoginFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.SignInLogic;

public class LoginFragment extends Fragment {
    public LoginFragment() {
        super(R.layout.login_layout);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.login_layout, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneUsername();
        tunePassword();
        tuneSignInButton();
        tuneSignUpButton();
    }

    private void tuneUsername() {
        EditText username = getView().findViewById(R.id.login_layout_login_et);
        username.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tunePassword() {
        EditText password = getView().findViewById(R.id.login_layout_password_et);
        password.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
```

```
        }
      });
   }

   private void tuneSignInButton() {
      TextView signInButton = getView().findViewById(R.id.login_layout_login);
      signInButton.setOnClickListener(new SignInLogic());
   }

   private void tuneSignUpButton() {
      TextView sigUpButton = getView().findViewById(R.id.login_layout_sign_up);
      sigUpButton.setOnClickListener(v -> {
         Fragment fragment = new SignUpFragment();
         FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
         fTrans.replace(R.id.activity_main_fragment_cv, fragment, "signUpFragment").commit();
         fTrans.addToBackStack(null);
      });
   }
}
```

1.29. LogOutFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.listeners.RemoveAccountLogic;
```

```
public class LogOutFragment extends Fragment {
    private final int accountId;

    public LogOutFragment(int accountId) {
        super(R.layout.fragment_log_out);
        this.accountId = accountId;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_log_out, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneLogOutButton();
        tuneDeleteButton();
        tuneToolbar(view);
    }

    private void tuneLogOutButton() {
        TextView logOutButton = getView().findViewById(R.id.fragment_log_out_log_out);
        logOutButton.setOnClickListener(v -> {
            SQLiteDatabase currentAccountDb =
getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
            currentAccountDb.execSQL("DELETE FROM CurrentAccount");

            Fragment fragment = new LoginFragment();
            FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
            ((FragmentActivity)
v.getContext()).getSupportFragmentManager().popBackStackImmediate(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
            fTrans.replace(R.id.activity_main_fragment_cv, fragment, "loginFragment").commit();
            fTrans.addToBackStack(null);
        });
    }

    private void tuneDeleteButton() {
        TextView deleteButton = getView().findViewById(R.id.fragment_log_out_delete);
        deleteButton.setOnClickListener(new RemoveAccountLogic(accountId));
    }

    private void tuneToolbar(View view) {
```

```
        Toolbar toolbar = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_log_out_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity) v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity) view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.30. MainFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.annotation.SuppressLint;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.google.android.material.bottomnavigation.BottomNavigationView;

public class MainFragment extends Fragment {
    public MainFragment() {
        super(R.layout.fragment_main);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
```

```
        return inflater.inflate(R.layout.fragment_main, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneBottomNavigationBar();
        setStartFragment(getContext());
    }

    @SuppressLint("NonConstantResourceId")
    private void tuneBottomNavigationBar() {
        BottomNavigationView bottomNavigationView = getView().findViewById(R.id.bottom_nav);
        bottomNavigationView.setOnNavigationItemSelectedListener(item -> {
            switch (item.getItemId()) {
                case R.id.menu_bottom_nav__search:
                    if (isFragmentVisible("searchFragment")) {
                        break;
                    }
                    setCurrentFragment(new
SearchFragment(SearchFragment.generateExerciseOverviewList(getContext()), 0, null),
"searchFragment");
                    break;
                case R.id.menu_bottom_nav__trainings:
                    if (isFragmentVisible("trainingsFragment")) {
                        break;
                    }
                    SQLiteDatabase currentAccountDB0 =
getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
                    int accountId0 = SQLiteHelper.getCurrentAccountId(currentAccountDB0);
                    setCurrentFragment(new TrainingsFragment(1, accountId0), "trainingsFragment");
                    break;
                case R.id.menu_bottom_nav__timetable:
                    if (isFragmentVisible("timetableFragment")) {
                        break;
                    }
                    SQLiteDatabase db = getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
                    SQLiteDatabase currentAccountDB =
getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

                    int trainingId = SQLiteHelper.getCurrentTrainingId(db, currentAccountDB);
                    String trainingTitle = SQLiteHelper.getTrainingTitleFromId(db, trainingId);
                    int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
```

```
                setCurrentFragment(new TimeTableFragment(SQLiteHelper.getCurrentTrainingId(db,
currentAccountDB), 0, accountId, new Training(trainingId, trainingTitle, accountId)),
"timetableFragment");
                break;
            case R.id.menu_bottom_nav__account:
                if (isFragmentVisible("accountFragment")) {
                    break;
                }
                SQLiteDatabase currentAccountDB1 =
getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
                setCurrentFragment(new AccountFragment(0,
SQLiteHelper.getCurrentAccountId(currentAccountDB1)), "accountFragment");
                break;
            }
            return true;
        });
    }

    private void setStartFragment(Context context) {
        Fragment fragment = new SearchFragment(SearchFragment.generateExerciseOverviewList(context),
0, null);
        FragmentTransaction fTrans = ((FragmentActivity)
context).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment, "searchFragment").commit();
        fTrans.addToBackStack(null);
    }

    private void setCurrentFragment(Fragment fragment, String fragmentName) {
        clearBackStack();
        FragmentTransaction fTrans = ((FragmentActivity)
getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment, fragmentName).commit();
        fTrans.addToBackStack(null);
    }

    private boolean isFragmentVisible(String fragmentName) {
        Fragment fragment = ((FragmentActivity)
getContext()).getSupportFragmentManager().findFragmentByTag(fragmentName);
        return (fragment != null && fragment.isVisible());
    }

    private void clearBackStack() {
        ((FragmentActivity)
getContext()).getSupportFragmentManager().popBackStackImmediate("mainFragment",
FragmentManager.POP_BACK_STACK_INCLUSIVE);
    }
```

```
}
```

1.31. ReconnectFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;

import java.io.IOException;

public class ReconnectFragment extends Fragment {
    public ReconnectFragment() {
        super(R.layout.fragment_reconnect);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_reconnect, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneReconnectButton();
    }

    private void tuneReconnectButton() {
        ProgressBar progressBar = getView().findViewById(R.id.fragment_reconnect_progressbar);

        TextView reconnectButton = getView().findViewById(R.id.fragment_reconnect_reconnect);
        reconnectButton.setVisibility(View.VISIBLE);
        reconnectButton.setOnClickListener(v -> {
            v.setVisibility(View.GONE);
            try {
```

```
            SQLiteHelper.handleDBAbsence(v.getContext(), progressBar);
        } catch (IOException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with load DB:  " + e, e);
            }
        }
    });
  }
}
```

1.32. SearchFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.adapters.ExerciseOverviewAdapter;
import com.example.testbottomnavigationbar.db.ExerciseOverview;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.listeners.AddExerciseSearchLogic;
import com.example.testbottomnavigationbar.remote_db.Training;
```

```
import java.util.ArrayList;
import java.util.List;

public class SearchFragment extends Fragment implements View.OnClickListener {
    private List<ExerciseOverview> exerciseOverviewList;
    private int type;
    private final TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public SearchFragment(List<ExerciseOverview> exerciseOverviewList, int type,
TrainingDayOfWeekHelper trainingDayOfWeekHelper) {
        super(R.layout.fragment_search);
        this.exerciseOverviewList = exerciseOverviewList;
        this.type = type;
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_search, container, false);
    }

    private void fillRecyclerView() {
        RecyclerView recyclerView = getView().findViewById(R.id.fragment_search_recyclerview);
        ExerciseOverviewAdapter exerciseOverviewAdapter = new
ExerciseOverviewAdapter(exerciseOverviewList, type, trainingDayOfWeekHelper);
        recyclerView.setAdapter(exerciseOverviewAdapter);

        LinearLayoutManager mLayoutManager = new LinearLayoutManager(getContext(),
RecyclerView.VERTICAL, false);
        recyclerView.setLayoutManager(mLayoutManager);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        if (type == 1) {
            handleType(view);
        }

        fillRecyclerView();
        tuneEditText(view.getContext());
        tuneFilter();
        tuneAddExercise();
    }

    @Override
    public void onClick(View v) {
        if (v instanceof ConstraintLayout) {
```

```
        if (type == 0) {
            Fragment fragment = new FilterFragment(type, trainingDayOfWeekHelper);
            FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
            fTrans.replace(R.id.fragment_cv, fragment).commit();
            fTrans.addToBackStack(null);
        } else {
            ((FragmentActivity) getView().getContext()).getSupportFragmentManager().popBackStack();

            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Start filter");
            }

            Fragment fragment = new FilterFragment(type, trainingDayOfWeekHelper);
            FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
            fTrans.replace(R.id.fragment_cv, fragment).commit();
            fTrans.addToBackStack(null);
        }
    }
}

private void handleType(View view) {
    tuneToolbar(view);
    hideAddExerciseButton();
}

private void hideAddExerciseButton() {
    ConstraintLayout constraintLayout =
getView().findViewById(R.id.fragment_search_add_exercise_cl);
    constraintLayout.setVisibility(View.GONE);
}

private void tuneToolbar(View view) {
    Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_search_toolbar);
    toolbar.setVisibility(View.VISIBLE);
    ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

    if (MainActivity.LOG) {
        Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
    }

    toolbar.setNavigationIcon(R.drawable.ic_back);
    toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
```

```
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }

    private void tuneAddExercise() {
        ConstraintLayout                                constraintLayout                                =
getView().findViewById(R.id.fragment_search_add_exercise_cl);
        constraintLayout.setOnClickListener(new AddExerciseSearchLogic());
    }

    private void tuneFilter() {
        ConstraintLayout constraintLayout = getView().findViewById(R.id.fragment_search_filter_cl);
        constraintLayout.setOnClickListener(this);
    }

    private void tuneEditText(Context context) {
        EditText editText = getView().findViewById(R.id.fragment_search_edittext);
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        editText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) { }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "EditText " + s);
                }

                exerciseOverviewList = generateExerciseOverviewListFromTitle(context, s.toString());
                fillRecyclerView();
            }

            @Override
            public void afterTextChanged(Editable s) { }
        });
//
//        editText.setOnFocusChangeListener((v, hasFocus) -> {
//            if(v.getId() == R.id.fragment_search_edittext && !hasFocus) {
//                                                    InputMethodManager    imm    =    (InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE);
//                imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
//            }
```

```
//      });
    }

    public static List<ExerciseOverview> generateExerciseOverviewListFromTitle(Context context, String prefix) {
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);

        List<ExerciseOverview> exerciseOverviewList = new ArrayList<>();

        Cursor exercises = db.rawQuery("SELECT ExerciseId, Title FROM Exercise WHERE Title LIKE '" + prefix + "%';", null);
        while (exercises.moveToNext()) {
            int id = exercises.getInt(0);
            String title = exercises.getString(1);

            ArrayList<String> tags = new ArrayList<>();
            Cursor tagsId = db.rawQuery("SELECT TagId FROM ExerciseToTag WHERE ExerciseId = " + id + ";", null);
            while (tagsId.moveToNext()) {
                int tagId = tagsId.getInt(0);

                Cursor tagTitle = db.rawQuery("SELECT Title FROM ExerciseTag WHERE TagId = " + tagId + ";", null);
                while (tagTitle.moveToNext()) {
                    tags.add(tagTitle.getString(0));
                }
            }

            exerciseOverviewList.add(new ExerciseOverview(title, tags));

            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, title + " " + id + " " + tags.size());
            }
        }

        return exerciseOverviewList;
    }

    public static List<ExerciseOverview> generateExerciseOverviewList(Context context) {
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);

        List<ExerciseOverview> exerciseOverviewList = new ArrayList<>();

        Cursor exercises = db.rawQuery("SELECT ExerciseId, Title FROM Exercise;", null);
        while (exercises.moveToNext()) {
```

```
            int id = exercises.getInt(0);
            String title = exercises.getString(1);

            ArrayList<String> tags = new ArrayList<>();
            Cursor tagsId = db.rawQuery("SELECT TagId FROM ExerciseToTag WHERE ExerciseId = " + id
+ ";", null);
            while (tagsId.moveToNext()) {
                int tagId = tagsId.getInt(0);

                Cursor tagTitle = db.rawQuery("SELECT Title FROM ExerciseTag WHERE TagId = " + tagId
+ ";", null);
                while (tagTitle.moveToNext()) {
                    tags.add(tagTitle.getString(0));
                }
            }

            exerciseOverviewList.add(new ExerciseOverview(title, tags));

            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, title + " " + id + " " + tags.size());
            }
        }

        return exerciseOverviewList;
    }
}
```

1.33. SignUpFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
```

```
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.listeners.SignUpLogic;

public class SignUpFragment extends Fragment {
    public SignUpFragment() {
        super(R.layout.fragment_sign_up);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_sign_up, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneET();
        tuneSignUpButton();
    }

    private void tuneSignUpButton() {
        TextView signUpButton = getView().findViewById(R.id.login_layout_sign_up);
        signUpButton.setOnClickListener(new SignUpLogic());
    }

    private void tuneET() {
        EditText firstName = getView().findViewById(R.id.account_editing_name_edit);
        firstName.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        EditText secondName = getView().findViewById(R.id.account_editing_second_name_edit);
        secondName.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        EditText username = getView().findViewById(R.id.account_editing_second_username_edit);
        username.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
```

```
        EditText password = getView().findViewById(R.id.fragment_sign_up_password_edit);
        password.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        EditText age = getView().findViewById(R.id.account_editing_age_edit);
        age.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        EditText weight = getView().findViewById(R.id.account_editing_weight_edit);
        weight.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        EditText height = getView().findViewById(R.id.account_editing_height_edit);
        height.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });

        EditText bodyFatShare = getView().findViewById(R.id.account_editing_percent_edit);
        bodyFatShare.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_sign_up_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
```

```
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.34. TimeTableFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.adapters.TimeTableDayAdapter;
import com.example.testbottomnavigationbar.db.ExerciseTemp;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.db.TimeTableDay;
import com.example.testbottomnavigationbar.db.TimeTableExercise;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.ArrayList;
import java.util.List;
import java.util.TreeMap;

public class TimeTableFragment extends Fragment {
    private final int trainingId;
    private final int type;
```

```
    private final int accountId;
    private final Training training;

    public TimeTableFragment(int trainingId, int type, int accountId, Training training) {
        super(R.layout.fragment_timetable);
        this.trainingId = trainingId;
        this.type = type;
        this.accountId = accountId;
        this.training = training;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_timetable, container, false);
    }

    private void fillRecyclerView(List<TimeTableDay> days) {
        RecyclerView recyclerView = getView().findViewById(R.id.fragment_timetable_recyclerview);
        TimeTableDayAdapter timeTableDayAdapter = new TimeTableDayAdapter(days, type, accountId,
training);
        recyclerView.setAdapter(timeTableDayAdapter);

        LinearLayoutManager mLayoutManager = new LinearLayoutManager(getContext(),
RecyclerView.VERTICAL, false);
        recyclerView.setLayoutManager(mLayoutManager);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        if (type != 0) {
            handleType(view);
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "TimeTableFragment creating");
        }

        fillRecyclerView(generateTimeTableDayList(view.getContext()));
    }

    private void handleType(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_timetable_toolbar);
        toolbar.setVisibility(View.VISIBLE);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
```

```
        Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
    }

    toolbar.setNavigationIcon(R.drawable.ic_back);
    toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
    ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
  }

  private List<TimeTableDay> generateTimeTableDayList(Context context) {
    SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
//                                                  SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

    ArrayList<TimeTableDay> list = new ArrayList<>();

//      int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
    if (MainActivity.LOG) {
      Log.d(MainActivity.TEG, "out " + accountId + " " + trainingId);
    }

    if (trainingId != -1) {
      TreeMap<Integer, ArrayList<ExerciseTemp>> map = new TreeMap<>();
      Cursor cursor = null;

      if (type == 0) {
        cursor = db.rawQuery("SELECT DISTINCT DayOfWeek, OrderNumber, ExerciseId FROM
TrainingExerciseSuccess WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + ";",
null);
      } else {
        cursor = db.rawQuery("SELECT DISTINCT DayOfWeek, OrderNumber, ExerciseId FROM
TrainingExercise WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + ";", null);
      }

      if (cursor != null && cursor.moveToFirst()) {
        do {
          int dayOfWeek = cursor.getInt(0);
          int orderNumber = cursor.getInt(1);
          String title = getExerciseTitleFromId(db, cursor.getInt(2));

          if (!map.containsKey(dayOfWeek)) {
            map.put(dayOfWeek, new ArrayList<>());
          }
          ArrayList<ExerciseTemp> exercises = map.get(dayOfWeek);
```

```
                exercises.add(new ExerciseTemp(title, orderNumber));
                map.put(dayOfWeek, exercises);
            } while (cursor.moveToNext());
        }


        for (int i = 0; i < 7; ++i) {
            if (!map.containsKey(i)) {
                map.put(i, new ArrayList<>());
            }
        }

        for (Integer key : map.keySet()) {
            ArrayList<ExerciseTemp> exercises = map.get(key);
            exercises.sort((o1, o2) -> (o1.getOrderNumber() < o2.getOrderNumber() ? -1 : 1));

            ArrayList<TimeTableExercise> texercises = new ArrayList<>();
            for (int i = 0; i < exercises.size(); ++i) {
                texercises.add(new TimeTableExercise(exercises.get(i).getTitle(), (i == exercises.size() - 1),
exercises.get(i).getOrderNumber()));
            }

            list.add(new TimeTableDay(trainingId, key, texercises));
        }
    }

    return list;
  }

  private String getExerciseTitleFromId(SQLiteDatabase db, int exerciseId) {
    Cursor cursor = db.rawQuery("SELECT Title FROM Exercise WHERE ExerciseId = " + exerciseId
+ ";", null);

    if (cursor != null && cursor.moveToFirst()) {
      return cursor.getString(0);
    } else {
      return null;
    }
  }
}
```

1.35. TrainingEditingFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
```

```
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingEditingHelper;
import com.example.testbottomnavigationbar.listeners.EditTrainingTitleLogic;
import com.example.testbottomnavigationbar.listeners.RemoveTrainingLogic;
import com.example.testbottomnavigationbar.listeners.SetCurrentTrainingLogic;

public class TrainingEditingFragment extends Fragment {
    private final TrainingEditingHelper trainingEditingHelper;

    public TrainingEditingFragment(TrainingEditingHelper trainingEditingHelper) {
        super(R.layout.fragment_training_editing);

        this.trainingEditingHelper = trainingEditingHelper;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_training_editing, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        tuneToolbar(view);
        tuneEditButton(view);
        tuneRemoveButton(view);
        tuneSetCurrentTrainingButton(view);

        tuneEditText(((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_training_editing_et));
    }
```

```
    private void tuneEditText(EditText editText) {
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
    }

    private void tuneEditButton(View view) {
        TextView textView = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_training_editing_edit);
        textView.setOnClickListener(new EditTrainingTitleLogic(trainingEditingHelper));
    }

    private void tuneRemoveButton(View view) {
        TextView textView = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_training_editing_delete);
        textView.setOnClickListener(new RemoveTrainingLogic(trainingEditingHelper));
    }

    private void tuneSetCurrentTrainingButton(View view) {
        TextView textView = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_training_editing_current_training);
        textView.setOnClickListener(new SetCurrentTrainingLogic(trainingEditingHelper));
    }

    private void tuneToolbar(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_training_editing_toolbar);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
}
```

1.36. TrainingExerciseSetsFragment.java

```
package com.example.testbottomnavigationbar.fragments;
```

```
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.media.Image;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.ExerciseSetResult;
import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.TrainingExerciseSetsObj;
import com.example.testbottomnavigationbar.TrainingExerciseSuccessObj;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.listeners.ExerciseSetLongClickLogic;
import com.example.testbottomnavigationbar.listeners.OpenExerciseFromQuestionLogic;
import com.example.testbottomnavigationbar.listeners.TrainingExerciseSuccessCreateSetLogic;

import java.util.List;

public class TrainingExerciseSetsFragment extends Fragment {
    private final int type;
    private final int accountId;
    private final TrainingExerciseInstance trainingExerciseInstance;
    private TrainingExerciseSetsObj trainingExerciseSetsObj;

    public TrainingExerciseSetsFragment(int type, int accountId, TrainingExerciseInstance
trainingExerciseInstance) {
        super(R.layout.training_exercise_success_frame);
```

```
        this.type = type;
        this.accountId = accountId;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.training_exercise_success_frame, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        addBackButton(view);
        generateTrainingExerciseSets(view.getContext());
        addContent(view);
        tuneAddSet(view);
    }

    private void tuneAddSet(View view) {
        ConstraintLayout constraintLayout = ((FragmentActivity)
view.getContext()).findViewById(R.id.training_exercise_success_addset_constrainlayout);

        if (type != 1) {
            constraintLayout.setVisibility(View.GONE);
            return;
        }

        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        constraintLayout.setOnClickListener(new
TrainingExerciseSuccessCreateSetLogic(trainingExerciseInstance.getTrainingId(),
                trainingExerciseInstance.getDayOfWeek(), trainingExerciseInstance.getOrderNumber(),
                SQLiteHelper.getExerciseIdFromTitle(db, trainingExerciseInstance.getExerciseTitle()),
                trainingExerciseSetsObj.getSetsInfo().size() + 1, 1));
    }

    private void addContent(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout)
LayoutInflater.from(view.getContext()).inflate(R.layout.training_exercise_success, (ViewGroup) view,
false);
        LinearLayout linearLayout = (LinearLayout) constraintLayout.getChildAt(2);

        fillExerciseSets(linearLayout);
        setExerciseTitle((TextView) constraintLayout.getChildAt(0));

        ScrollView scrollView = ((FragmentActivity)
view.getContext()).findViewById(R.id.training_exercise_success_scrollview);
```

```
        scrollView.addView(constraintLayout);

        // Hide note
        TextView textView = (TextView) constraintLayout.getChildAt(4);
        textView.setVisibility(View.GONE);
        EditText editText = (EditText) constraintLayout.getChildAt(5);
        editText.setVisibility(View.GONE);
        TextView textView2 = (TextView) constraintLayout.getChildAt(6);
        textView2.setVisibility(View.GONE);

        tuneQuestionButton((ImageView) constraintLayout.getChildAt(1));
    }

    private void tuneQuestionButton(ImageView imageView) {
        imageView.setOnClickListener(new
OpenExerciseFromQuestionLogic(trainingExerciseInstance.getExerciseTitle()));
    }

    private void setExerciseTitle(TextView textView) {
        textView.setText(trainingExerciseInstance.getExerciseTitle());
    }

    private void fillExerciseSets(LinearLayout linearLayout) {
        SQLiteDatabase db = getView().getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
//                                                  SQLiteDatabase currentAccountDB =
getView().getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        List<ExerciseSetResult> list = trainingExerciseSetsObj.getSetsInfo();
        for (int i = 0; i < list.size(); ++i) {
            ConstraintLayout constraintLayout = (ConstraintLayout)
LayoutInflater.from(linearLayout.getContext()).inflate(R.layout.set_info, linearLayout, false);

            if (type == 1) {
                constraintLayout.setOnLongClickListener(new ExerciseSetLongClickLogic(1,
                        new ExerciseInTrainingEditing(trainingExerciseInstance.getTrainingId(),
                                SQLiteHelper.getExerciseIdFromTitle(db,
trainingExerciseInstance.getExerciseTitle()),
                                trainingExerciseInstance.getOrderNumber(),
                                i + 1,
                                accountId,
                                trainingExerciseInstance.getDayOfWeek())));
            }

            TextView setNum = (TextView) constraintLayout.getChildAt(0);
            setNum.setText("#" + (i + 1));
```

```
            TextView weight = (TextView) constraintLayout.getChildAt(5);
            if (list.get(i).getWeight() != 0) {
                weight.setText(list.get(i).getWeight() + " кг");
            } else {
                weight.setText("-  кг");
            }

            TextView repetitions = (TextView) constraintLayout.getChildAt(6);
            if (list.get(i).getRepetitions() != 0) {
                repetitions.setText(list.get(i).getRepetitions() + " раз");
            } else {
                repetitions.setText("-  раз");
            }

            TextView timer = (TextView) constraintLayout.getChildAt(7);
            if (list.get(i).getTimer() != 0) {
                timer.setText(list.get(i).getTimer() + " сек");
            } else {
                timer.setText("-  сек");
            }

            linearLayout.addView(constraintLayout);
        }
    }

    private void addBackButton(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.my_toolbar_training_exercise_success);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }

    private void generateTrainingExerciseSets(Context context) {
        trainingExerciseSetsObj = new
TrainingExerciseSetsObj(trainingExerciseInstance.getExerciseTitle());
```

```
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
//                                  SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

//        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
        Cursor cursor = db.rawQuery("SELECT RepsNum, Timer, SetNumber FROM TrainingExercise
WHERE AccountId = " + accountId + " AND TrainingId = " + trainingExerciseInstance.getTrainingId() +
" AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + " AND OrderNumber = " +
trainingExerciseInstance.getOrderNumber() + " ORDER BY SetNumber;", null);
        if (cursor != null && cursor.moveToFirst()) {
            do {
                ExerciseSetResult exerciseSetResult = new ExerciseSetResult(0, cursor.getInt(0),
cursor.getInt(1));
                trainingExerciseSetsObj.addExerciseSetResult(exerciseSetResult);
            } while (cursor.moveToNext());
        }
    }
}
```

1.37. TrainingExerciseSuccessFragment.java

```
package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
```

```
import com.example.testbottomnavigationbar.ExerciseSetResult;
import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.TrainingExerciseSuccessObj;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.listeners.DoneNoteLogic;
import com.example.testbottomnavigationbar.listeners.ExerciseSetLongClickLogic;
import com.example.testbottomnavigationbar.listeners.OpenExerciseFromQuestionLogic;
import com.example.testbottomnavigationbar.listeners.TrainingExerciseSuccessCreateSetLogic;

import java.util.List;

public class TrainingExerciseSuccessFragment extends Fragment {
    private final TrainingExerciseInstance trainingExerciseInstance;
    private TrainingExerciseSuccessObj trainingExerciseSuccessObj;

    public TrainingExerciseSuccessFragment(TrainingExerciseInstance trainingExerciseInstance) {
        super(R.layout.training_exercise_success_frame);

        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.training_exercise_success_frame, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        addBackButton(view);
        generateTrainingExerciseSuccess(view.getContext());
        addContent(view);
        tuneAddSet(view);
    }

    private void tuneAddSet(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);

        ConstraintLayout constraintLayout = ((FragmentActivity)
view.getContext()).findViewById(R.id.training_exercise_success_addset_constrainlayout);
        constraintLayout.setOnClickListener(new
TrainingExerciseSuccessCreateSetLogic(trainingExerciseInstance.getTrainingId(),
                trainingExerciseInstance.getDayOfWeek(), trainingExerciseInstance.getOrderNumber(),
                SQLiteHelper.getExerciseIdFromTitle(db, trainingExerciseInstance.getExerciseTitle()),
```

```
            trainingExerciseSuccessObj.getSetsInfo().size() + 1, 0));
    }

    private void addBackButton(View view) {
        Toolbar toolbar = ((FragmentActivity)
view.getContext()).findViewById(R.id.my_toolbar_training_exercise_success);
        ((AppCompatActivity) view.getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
view.getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }

    private void addContent(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout)
LayoutInflater.from(view.getContext()).inflate(R.layout.training_exercise_success, (ViewGroup) view,
false);
        LinearLayout linearLayout = (LinearLayout) constraintLayout.getChildAt(2);

        fillExerciseSets(linearLayout);
        setExerciseTitle((TextView) constraintLayout.getChildAt(0));

        ScrollView scrollView = ((FragmentActivity)
view.getContext()).findViewById(R.id.training_exercise_success_scrollview);
        scrollView.addView(constraintLayout);

        tuneQuestionButton((ImageView) constraintLayout.getChildAt(1));

        tuneNotes(constraintLayout);
    }

    private void tuneNotes(ConstraintLayout constraintLayout) {
        SQLiteDatabase db = getView().getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
getView().getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        EditText editText = (EditText) constraintLayout.getChildAt(5);
```

```
        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if (!hasFocus) {
                MainActivity.hideSoftKeyboard(v.getContext(), v);
            }
        });
        fillNote(editText, db, accountId);

        TextView doneButton = (TextView) constraintLayout.getChildAt(6);
        doneButton.setOnClickListener(new DoneNoteLogic(accountId, trainingExerciseInstance));
    }

    private void fillNote(EditText editText, SQLiteDatabase db, int accountId) {
        Cursor cursor = db.rawQuery("SELECT Note FROM TrainingExerciseNote " +
                "WHERE TrainingId = " + trainingExerciseInstance.getTrainingId() +
                " AND Accountid = " + accountId +
                " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() +
                " AND OrderNumber = " + trainingExerciseInstance.getOrderNumber() + ";", null);

        String res = "";
        if (cursor != null && cursor.moveToFirst()) {
            String note = cursor.getString(0);
            if (note != null) {
                res = note;
            }
        }

        editText.setText(res);
    }

    private void tuneQuestionButton(ImageView imageView) {
        imageView.setOnClickListener(new
OpenExerciseFromQuestionLogic(trainingExerciseInstance.getExerciseTitle()));
    }

    private void setExerciseTitle(TextView textView) {
        textView.setText(trainingExerciseInstance.getExerciseTitle());
    }

    private void fillExerciseSets(LinearLayout linearLayout) {
        SQLiteDatabase db = getView().getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
getView().getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        List<ExerciseSetResult> list = trainingExerciseSuccessObj.getSetsInfo();
        for (int i = 0; i < list.size(); ++i) {
```

```
        ConstraintLayout constraintLayout = (ConstraintLayout)
LayoutInflater.from(linearLayout.getContext()).inflate(R.layout.set_info, linearLayout, false);
        constraintLayout.setOnLongClickListener(new ExerciseSetLongClickLogic(0,
              new ExerciseInTrainingEditing(trainingExerciseInstance.getTrainingId(),
                    SQLiteHelper.getExerciseIdFromTitle(db, trainingExerciseInstance.getExerciseTitle()),
                    trainingExerciseInstance.getOrderNumber(),
                    i + 1,
                    SQLiteHelper.getCurrentAccountId(currentAccountDB),
                    trainingExerciseInstance.getDayOfWeek())));

        TextView setNum = (TextView) constraintLayout.getChildAt(0);
        setNum.setText("#" + (i + 1));

        TextView weight = (TextView) constraintLayout.getChildAt(5);
        if (list.get(i).getWeight() != 0) {
           weight.setText(list.get(i).getWeight() + " кг");
        } else {
           weight.setText("-  кг");
        }

        TextView repetitions = (TextView) constraintLayout.getChildAt(6);
        if (list.get(i).getRepetitions() != 0) {
           repetitions.setText(list.get(i).getRepetitions() + " раз");
        } else {
           repetitions.setText("-  раз");
        }

        TextView timer = (TextView) constraintLayout.getChildAt(7);
        if (list.get(i).getTimer() != 0) {
           timer.setText(list.get(i).getTimer() + " сек");
        } else {
           timer.setText("-  сек");
        }

        linearLayout.addView(constraintLayout);
     }
  }

  private void generateTrainingExerciseSuccess(Context context) {
     trainingExerciseSuccessObj = new
TrainingExerciseSuccessObj(trainingExerciseInstance.getExerciseTitle());

     SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
     SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
```

```
        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
        Cursor cursor = db.rawQuery("SELECT Weight, RepsNum, Timer, SetNumber FROM
TrainingExerciseSuccess WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() + " AND DayOfWeek = " +
trainingExerciseInstance.getDayOfWeek() + " AND OrderNumber = " +
trainingExerciseInstance.getOrderNumber() + " ORDER BY SetNumber;", null);
        if (cursor != null && cursor.moveToFirst()) {
            do {
                ExerciseSetResult exerciseSetResult = new ExerciseSetResult(cursor.getInt(0), cursor.getInt(1),
cursor.getInt(2));
                trainingExerciseSuccessObj.addExerciseSetResult(exerciseSetResult);
            } while (cursor.moveToNext());
        }
    }
}
```

1.38. TrainingsFragment.java

```
 package com.example.testbottomnavigationbar.fragments;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
```

```
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.adapters.TrainingAdapter;
import com.example.testbottomnavigationbar.listeners.AddTrainingLogic;
import com.example.testbottomnavigationbar.remote_db.Training;

import java.util.ArrayList;
import java.util.List;

public class TrainingsFragment extends Fragment {
    private final int type;
    private final int accountId;

    public TrainingsFragment(int type, int accountId) {
        super(R.layout.fragment_trainings);
        this.type = type;
        this.accountId = accountId;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceSate) {
        return inflater.inflate(R.layout.fragment_trainings, container, false);
    }

    private void fillRecyclerView(List<Training> trainings) {
        RecyclerView recyclerView = getView().findViewById(R.id.fragment_trainings_recyclerview);
        TrainingAdapter trainingAdapter = new TrainingAdapter(type, accountId, trainings);
        recyclerView.setAdapter(trainingAdapter);

        LinearLayoutManager mLayoutManager = new LinearLayoutManager(getContext(),
RecyclerView.VERTICAL, false);
        recyclerView.setLayoutManager(mLayoutManager);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        if (type != 1) {
            handleType();
        }

        fillRecyclerView(generateTrainingsList(view.getContext(), accountId));
        tuneEditText(view.getContext());
        tuneAddTraining();
    }

    private void handleType() {
        TextView textView = getView().findViewById(R.id.fragment_trainings_header);
        textView.setVisibility(View.GONE);
```

```
        textView = getView().findViewById(R.id.fragment_trainins_title_line);
        textView.setVisibility(View.GONE);

        Toolbar toolbar = ((FragmentActivity)
getView().getContext()).findViewById(R.id.fragment_trainings_toolbar);
        toolbar.setVisibility(View.VISIBLE);
        ((AppCompatActivity) getView().getContext()).setSupportActionBar(toolbar);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "toolbar " + (toolbar == null));
        }

        toolbar.setNavigationIcon(R.drawable.ic_back);
        toolbar.setNavigationOnClickListener(v1 -> ((FragmentActivity)
v1.getContext()).getSupportFragmentManager().popBackStack());
        ((AppCompatActivity)
getView().getContext()).getSupportActionBar().setDisplayShowTitleEnabled(false);
    }

    private void tuneAddTraining() {
        ImageView imageView = getView().findViewById(R.id.fragment_trainings_add_button);

        if (type == 1) {
            imageView.setOnClickListener(new AddTrainingLogic());
        } else {
            imageView.setVisibility(View.GONE);
        }
    }

    private void tuneEditText(Context context) {
        EditText editText = getView().findViewById(R.id.fragment_trainings_edittext);
        editText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) { }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                fillRecyclerView(generateTrainingsListFromTitle(context, s.toString(), accountId));
            }

            @Override
            public void afterTextChanged(Editable s) { }
        });

        editText.setOnFocusChangeListener((v, hasFocus) -> {
            if(v.getId() == R.id.fragment_trainings_edittext && !hasFocus) {
```

```
            InputMethodManager imm = (InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
        }
      });
   }

   private List<Training> generateTrainingsListFromTitle(Context context, String prefix, int accountId) {
      SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
//                                                    SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

      ArrayList<Training> list = new ArrayList<>();

//       int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
      Cursor cursor = db.rawQuery("SELECT Title FROM Training WHERE Title LIKE '" + prefix + "%'
AND AccountId = " + accountId + ";", null);
      if (cursor != null && cursor.moveToFirst()) {
         do {
            String title = cursor.getString(0);
            list.add(new Training(SQLiteHelper.getTrainingIdFromTitle(db, title, accountId), title,
accountId));
         } while(cursor.moveToNext());
      }

      return list;
   }

   private List<Training> generateTrainingsList(Context context, int accountId) {
      SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
//                                                    SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

      ArrayList<Training> list = new ArrayList<>();

//       int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
      Cursor cursor = db.rawQuery("SELECT Title FROM Training WHERE AccountId = " + accountId +
";", null);
      if (cursor != null && cursor.moveToFirst()) {
         do {
            String title = cursor.getString(0);
            list.add(new Training(SQLiteHelper.getTrainingIdFromTitle(db, title, accountId), title,
accountId));
```

```
        } while(cursor.moveToNext());
      }

      return list;
   }
}
```

1.39. AccountEditingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AccountEditingFragment;
import com.example.testbottomnavigationbar.fragments.AddExerciseTagFragment;
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;

public class AccountEditingLogic implements View.OnClickListener {
   private final Account account;
   private final BodyCondition bodyCondition;

   public AccountEditingLogic(Account account, BodyCondition bodyCondition) {
      this.account = account;
      this.bodyCondition = bodyCondition;
   }

   @Override
   public void onClick(View v) {
      Fragment fragment = new AccountEditingFragment(account, bodyCondition);
      FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
      fTrans.replace(R.id.fragment_cv, fragment).commit();
      fTrans.addToBackStack(null);
   }
}
```

1.40. AddExerciseFromDescriptionLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
```

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingExerciseTask;

import java.io.IOException;

public class AddExerciseFromDescriptionLogic implements View.OnClickListener {
    private final TrainingDayOfWeekHelper trainingDayOfWeekHelper;
    private final String exerciseTitle;

    public AddExerciseFromDescriptionLogic(TrainingDayOfWeekHelper trainingDayOfWeekHelper,
String exerciseTitle) {
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
        this.exerciseTitle = exerciseTitle;
    }

    @Override
    public void onClick(View v) {
        try {
            addExerciseInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem in adding exercise from description   " + e, e);
            }
        }

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void addExerciseInDB(View view) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Trying adding exercise in DB");
        }

        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
```

```
        SQLiteDatabase currentAccountDB = view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
        int trainingId = trainingDayOfWeekHelper.getTrainingId();
        int dayOfWeek = trainingDayOfWeekHelper.getDayOfWeek();
        int orderNumber = getOrderNumber(db, trainingId, accountId, dayOfWeek);
        int exerciseId = SQLiteHelper.getExerciseIdFromTitle(db, exerciseTitle);

        db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum, Timer, AccountId)\n" +
                "VALUES ('" + trainingId + "', '" + dayOfWeek + "', '" + orderNumber +
                "', '" + exerciseId + "', '" + 1 + "', '" + 0 + "', '" + 0 + "', '" + accountId + "');");

        addExerciseInRemoteDB(accountId, trainingId, dayOfWeek, orderNumber, exerciseId);
    }

    private void addExerciseInRemoteDB(int accountId, int trainingId, int dayOfWeek, int orderNumber,
int exerciseId) {
        new InsertTrainingExerciseTask(new TrainingExercise(trainingId, dayOfWeek, orderNumber,
exerciseId, 1,
                0, 0, accountId)).execute();
    }

    private int getOrderNumber(SQLiteDatabase db, int trainingId, int accountId, int dayOfWeek) {
        int orderNumber = 1;

        Cursor cursor = db.rawQuery("SELECT MAX(OrderNumber) FROM (SELECT OrderNumber
FROM TrainingExercise " +
                "WHERE TrainingId = " + trainingId + " AND AccountId = " + accountId + " AND DayOfWeek
= " + dayOfWeek + ") AS OrderTable;", null);
        if (cursor != null && cursor.moveToFirst()) {
            orderNumber = cursor.getInt(0) + 1;
        }

        return orderNumber;
    }
}
```

1.41. AddExerciseInTrainingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
```

```
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.fragments.SearchFragment;

public class AddExerciseInTrainingLogic implements View.OnClickListener {
    private final TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public AddExerciseInTrainingLogic(TrainingDayOfWeekHelper trainingDayOfWeekHelper) {
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    @Override
    public void onClick(View v) {
        Fragment fragment = new SearchFragment(SearchFragment.generateExerciseOverviewList(v.getContext()), 1, trainingDayOfWeekHelper);
        FragmentTransaction fTrans = ((FragmentActivity) v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.42. AddExerciseSearchLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AddExerciseMuscleFragment;
import com.example.testbottomnavigationbar.fragments.AddExerciseTagFragment;

public class AddExerciseSearchLogic implements View.OnClickListener {
    @Override
    public void onClick(View v) {
        Fragment fragment = new AddExerciseTagFragment();
        FragmentTransaction fTrans = ((FragmentActivity) v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
```

```
        fTrans.addToBackStack(null);
    }
}
```

1.43. AddFriendLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.AccountFragment;
import com.example.testbottomnavigationbar.remote_db.Friendship;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertFriendshipTask;

public class AddFriendLogic implements View.OnClickListener {
    private final int accountId;
    private final int friendId;

    public AddFriendLogic(int accountId, int friendId) {
        this.accountId = accountId;
        this.friendId = friendId;
    }

    @Override
    public void onClick(View v) {
        addFriendInDb(v);

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
        Fragment fragment = new AccountFragment(1, friendId);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment, null).commit();
        fTrans.addToBackStack(null);
    }

    private void addFriendInDb(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
```

```
        db.execSQL("INSERT INTO Friendship(AccountId, FriendId)\n" +
                "VALUES ('" +  accountId + "', '" + friendId + "');");

        addFriendInRemoteDB();
    }

    private void addFriendInRemoteDB() {
        new InsertFriendshipTask(new Friendship(accountId, friendId)).execute();
    }
}
```

1.44. AddSetSaveLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingExerciseSuccessTask;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingExerciseTask;

import java.io.IOException;
import java.net.MalformedURLException;

public class AddSetSaveLogic implements View.OnClickListener {
    private final int trainingId;
    private final int dayOfWeek;
    private final int orderNumber;
    private final int exerciseId;
    private final int setNumber;
    private final int type;

    public AddSetSaveLogic(int trainingId, int dayOfWeek, int orderNumber, int exerciseId, int setNumber,
int type) {
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
```

```
        this.orderNumber = orderNumber;
        this.exerciseId = exerciseId;
        this.setNumber = setNumber;
        this.type = type;
    }


    @Override
    public void onClick(View v) {
        if (!checkInput(v)) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Bad input value to add set");
            }

            return;
        }


        try {
            addSetInDB(v);
        } catch (IOException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Add set value problems");
            }
        }
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void addSetInDB(View view) throws IOException {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        if (type == 0) {
            db.execSQL("INSERT INTO TrainingExerciseSuccess(AccountId, TrainingId, DayOfWeek,
OrderNumber, ExerciseId, SetNumber, RepsNum, Weight, Timer)\n" +
                    "VALUES ('" + accountId + "', '" + trainingId + "', '" + dayOfWeek + "', '" + orderNumber +
                    "', '" + exerciseId + "', '" + setNumber + "', '" + getRepsNum(view) + "', '" + getWeight(view)
+ "', '" + getTimer(view) + "');");
        } else {
            db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber,
ExerciseId, SetNumber, RepsNum, Timer, AccountId)\n" +
                    "VALUES ('" + trainingId + "', '" + dayOfWeek + "', '" + orderNumber +
```

```
                "', '" + exerciseId + "', '" + setNumber + "', '" + getRepsNum(view) + "', '" + getTimer(view) +
"', '" + accountId + "');");
        }

        addSetInRemoteDB(view, accountId);
    }

    private void addSetInRemoteDB(View view, int accountId) throws IOException {
        if (type == 0) {
            new InsertTrainingExerciseSuccessTask(new TrainingExerciseSuccess(accountId, trainingId,
dayOfWeek, orderNumber,
                    exerciseId, setNumber, getRepsNum(view), getWeight(view), getTimer(view))).execute();
        } else {
            new InsertTrainingExerciseTask(new TrainingExercise(trainingId, dayOfWeek, orderNumber,
exerciseId, setNumber,
                    getRepsNum(view), getTimer(view), accountId)).execute();
        }
    }

    private boolean checkInput(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText repsNum = (EditText) constraintLayout.getChildAt(4);
        EditText weight = (EditText) constraintLayout.getChildAt(1);
        EditText timer = (EditText) constraintLayout.getChildAt(7);

        if (!isNumber(repsNum.getText().toString()) || getRepsNum(view) < 0) {
            return false;
        }
        if (type == 0 && (!isNumber(weight.getText().toString()) || getWeight(view) < 0)) {
            return false;
        }
        if (!isNumber(timer.getText().toString()) || getTimer(view) < 0) {
            return false;
        }

        return true;
    }

    private boolean isNumber(String s) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "isNumber   " + s);
        }

        if (s.length() == 0) {
            return false;
        }
```

```
        for (int i = 0; i < s.length(); ++i) {
            if (s.charAt(i) - '0' < 0 || s.charAt(i) - '0' > 9) {
                return false;
            }
        }

        return true;
    }

    private int getRepsNum(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText editText = (EditText) constraintLayout.getChildAt(4);

        return Integer.parseInt(editText.getText().toString());
    }

    private int getWeight(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText editText = (EditText) constraintLayout.getChildAt(1);

        return Integer.parseInt(editText.getText().toString());
    }

    private int getTimer(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText editText = (EditText) constraintLayout.getChildAt(7);

        return Integer.parseInt(editText.getText().toString());
    }
}
```

1.45. AddTrainingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.CreatingTrainingFragment;

public class AddTrainingLogic implements View.OnClickListener {
    @Override
    public void onClick(View v) {
```

```
        Fragment fragment = new CreatingTrainingFragment();
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.46. CopyTrainingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingTask;

import java.util.ArrayList;
import java.util.List;

public class CopyTrainingLogic implements View.OnClickListener {
    private final int friendAccountId;
    private final Training training;
    private String strTrainingTitle;

    public CopyTrainingLogic(int friendAccountId, Training training) {
        this.friendAccountId = friendAccountId;
        this.training = training;
    }

    @Override
    public void onClick(View v) {
        if (!accessName(v)) {
            return;
        }

        copyToDB(v);
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
```

```
    }

    private void copyToDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
        int currentAccount = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        db.execSQL("INSERT INTO Training(Title, AccountId)\n" +
                "VALUES ('" + strTrainingTitle + "', '" + currentAccount + "');");

        List<TrainingExercise> trainingExerciseList = new ArrayList<>();
        int newTrainingId = SQLiteHelper.getTrainingIdFromTitle(db, strTrainingTitle, currentAccount);
        Training newTraining = new Training(newTrainingId, strTrainingTitle, currentAccount);

        Cursor cursor = db.rawQuery("SELECT * FROM TrainingExercise WHERE TrainingId = " +
training.getTrainingId() + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            do {
                TrainingExercise trainingExercise = new TrainingExercise(
                        newTrainingId,
                        cursor.getInt(1),
                        cursor.getInt(2),
                        cursor.getInt(3),
                        cursor.getInt(4),
                        cursor.getInt(5),
                        cursor.getInt(6),
                        currentAccount
                );

                db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber,
ExerciseId, SetNumber, RepsNum, Timer, AccountId)\n" +
                        "VALUES ('" + newTrainingId + "', '" + trainingExercise.getDayOfWeek() + "', '" +
                        trainingExercise.getOrderNumber() + "', '" + trainingExercise.getExerciseId() + "', '" +
trainingExercise.getSetNumber() + "', '" +
                        trainingExercise.getRepsNum() + "', '" + trainingExercise.getTimer() + "', '" +
currentAccount + "');");

                trainingExerciseList.add(trainingExercise);
            } while (cursor.moveToNext());
        }

        copyToRemoteDB(trainingExerciseList, newTraining);
    }
```

```
    private void copyToRemoteDB(List<TrainingExercise> trainingExerciseList, Training training) {
        new InsertTrainingTask(training, 1, trainingExerciseList).execute();
    }

    private boolean accessName(View view) {
        EditText trainingTitle = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_add_training_title_et);
        strTrainingTitle = trainingTitle.getText().toString();

        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB = view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(), Context.MODE_PRIVATE, null);
        int currentAccount = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        return (!strTrainingTitle.equals("") && SQLiteHelper.getTrainingIdFromTitle(db, strTrainingTitle, currentAccount) == -1);
    }
}
```

1.47. DeleteTrainingExerciseLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.remote_db.tasks.DeleteMoveTrainingExeciseTask;

public class DeleteTrainingExerciseLogic implements View.OnClickListener {
    private final int type;
    private final TrainingExerciseInstance trainingExerciseInstance;
    private int maxOrderNumber;

    public DeleteTrainingExerciseLogic(int type, TrainingExerciseInstance trainingExerciseInstance) {
        this.type = type;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
```

```
    public void onClick(View v) {
        deleteTrainingExerciseInDB(v);

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void deleteTrainingExerciseInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        if (type == 0) {
            deleteInDB(db, accountId, "TrainingExerciseSuccess");
            deleteInDB(db, accountId, "TrainingExerciseNote");
        } else {
            deleteInDB(db, accountId, "TrainingExercise");
        }

        deleteTrainingExerciseInRemoteDB(accountId);
    }

    private void deleteInDB(SQLiteDatabase db, int accountId, String tableName) {
        // удаляем все сеты тренировки
        db.execSQL("PRAGMA foreign_keys=ON;");
        db.execSQL("DELETE FROM " + tableName +
                " WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + " AND OrderNumber =
" + trainingExerciseInstance.getOrderNumber() + ";");

        // сдвинаем все сеты упражнений выше вниз
        getMaxOrderNumber(db, accountId, tableName);
        for (int i = trainingExerciseInstance.getOrderNumber() + 1; i <= maxOrderNumber; ++i) {
            db.execSQL("UPDATE " + tableName +
                    " SET OrderNumber = " + (i - 1) +
                    " WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                    " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() +
                    " AND OrderNumber = " + i + ";");
        }
    }

    private void deleteTrainingExerciseInRemoteDB(int accountId) {
```

```
        new DeleteMoveTrainingExeciseTask(type, accountId, maxOrderNumber,
trainingExerciseInstance).execute();
    }

    private void getMaxOrderNumber(SQLiteDatabase db, int accountId, String tableName) {
        maxOrderNumber = -1;

        Cursor cursor = db.rawQuery("SELECT MAX(OrderNumber) " +
                "FROM " + tableName +
                " WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + ";", null);

        if (cursor != null && cursor.moveToFirst()) {
            maxOrderNumber = cursor.getInt(0);
        }
    }
}
```

1.48. DoneNoteLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateNoteTask;

public class DoneNoteLogic implements View.OnClickListener {
    private final int accountId;
    private final TrainingExerciseInstance trainingExerciseInstance;
    private String note;

    public DoneNoteLogic(int accountId, TrainingExerciseInstance trainingExerciseInstance) {
        this.accountId = accountId;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }
```

```
    @Override
    public void onClick(View v) {
        try {
            findNote(v);
            updateNoteInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Update note value problems");
            }
        }
    }

    private boolean findNote(View view) {
        EditText editText = ((FragmentActivity) view.getContext()).findViewById(R.id.training_exercise_success_note);
        note = editText.getText().toString();

        return true;
    }

    private void updateNoteInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);
        int trainingId = trainingExerciseInstance.getTrainingId();
        int dayOfWeek = trainingExerciseInstance.getDayOfWeek();
        int orderNumber = trainingExerciseInstance.getOrderNumber();

        db.execSQL("UPDATE TrainingExerciseNote\n" +
                "SET Note = '" + note + "'\n" +
                "WHERE TrainingId = " + trainingId +
                " AND AccountId = " + accountId +
                " AND DayOfWeek = " + dayOfWeek +
                " AND OrderNumber = " + orderNumber + ";");

        updateNoteInRemoteDB(db, trainingId, dayOfWeek, orderNumber);
    }

    private void updateNoteInRemoteDB(SQLiteDatabase db, int trainingId, int dayOfWeek, int orderNumber) {
        new UpdateNoteTask(accountId, trainingId, dayOfWeek, orderNumber, note).execute();
    }
}
```

1.49. EditTrainingTitleLogic.java

```
package com.example.testbottomnavigationbar.listeners;
```

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingEditingHelper;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateTrainingTask;

public class EditTrainingTitleLogic implements View.OnClickListener {
    private String title = "";
    private final TrainingEditingHelper trainingEditingHelper;

    public EditTrainingTitleLogic(TrainingEditingHelper trainingEditingHelper) {
        this.trainingEditingHelper = trainingEditingHelper;
    }

    @Override
    public void onClick(View v) {
        if (!findTitle(v)) {
            return;
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "onClick: title  " + title);
        }

        try {
            updateTrainingInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Update training value problems");
            }
        }

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private boolean findTitle(View view) {
        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_training_editing_et);
```

```
        title = editText.getText().toString();

        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int trainingId = SQLiteHelper.getTrainingIdFromTitle(db, title,
SQLiteHelper.getCurrentAccountId(currentAccountDB));

        return (!title.equals("") && trainingId == -1);
    }

    private void updateTrainingInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int trainingId = trainingEditingHelper.getTrainingId();

        db.execSQL("UPDATE Training\n" +
                "SET Title = '" + title + "'\n" +
                "WHERE TrainingId = " + trainingId + ";");

        updateTrainingInRemoteDB(currentAccountDB);
    }

    private void updateTrainingInRemoteDB(SQLiteDatabase currentAccountDB) {
        new UpdateTrainingTask(
                new Training(trainingEditingHelper.getTrainingId(), title,
SQLiteHelper.getCurrentAccountId(currentAccountDB)).getTitleJSON(),
                trainingEditingHelper.getTrainingId()).execute();
    }
}
```

1.50. ExerciseSetLongClickLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
```

```
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.fragments.ExerciseSetEditingFragment;

public class ExerciseSetLongClickLogic implements View.OnLongClickListener {
    private final int type;
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public ExerciseSetLongClickLogic(int type, ExerciseInTrainingEditing exerciseInTrainingEditing) {
        this.type = type;
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    public boolean onLongClick(View v) {
        Fragment fragment = new ExerciseSetEditingFragment(type, exerciseInTrainingEditing);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);

        return true;
    }
}
```

1.51. FilterApplyLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.db.ExerciseOverview;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.fragments.SearchFragment;
import com.example.testbottomnavigationbar.remote_db.Training;
```

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

public class FilterApplyLogic implements View.OnClickListener {
    private HashSet<String> filters;
    private int type;
    private final TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public FilterApplyLogic(int type, TrainingDayOfWeekHelper trainingDayOfWeekHelper) {
        filters = new HashSet<>();
        this.type = type;
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    public void addFilter(String filter) {
        filters.add(filter);
    }

    public void removeFilter(String filter) {
        filters.remove(filter);
    }

    @Override
    public void onClick(View v) {
        if (MainActivity.LOG) {
            String res = "";
            for (String filter : filters) {
                res += filter + " ";
            }

            Log.d(MainActivity.TEG, res);
        }

        if (type == 0) {
            Fragment fragment = new SearchFragment(generateExerciseOverviewList(v.getContext()), 0,
trainingDayOfWeekHelper);
            FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
            fTrans.replace(R.id.fragment_cv, fragment, "searchFragment").commit();
            fTrans.addToBackStack(null);
        } else {
            ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();

            Fragment fragment = new SearchFragment(generateExerciseOverviewList(v.getContext()), 1,
trainingDayOfWeekHelper);
```

```
            FragmentTransaction fTrans = ((FragmentActivity) v.getContext()).getSupportFragmentManager().beginTransaction();
            fTrans.replace(R.id.fragment_cv, fragment).commit();
            fTrans.addToBackStack(null);
        }
    }

    private List<ExerciseOverview> generateExerciseOverviewList(Context context) {
        if (filters.size() == 0) {
            return SearchFragment.generateExerciseOverviewList(context);
        }

        List<ExerciseOverview> exerciseOverviewList = new ArrayList<>();
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);

        List<Integer> selectedMusclesId = new ArrayList<>();
        Cursor cursor = db.rawQuery(getSQLQueryWithAllFilterMuscleId(), null);
        if (cursor != null && cursor.moveToFirst()) {
            do {
                selectedMusclesId.add(cursor.getInt(0));
            } while (cursor.moveToNext());
        }

        cursor = db.rawQuery(getSQLQueryWithAllNeededExercisesId(selectedMusclesId), null);
        if (cursor != null && cursor.moveToFirst()) {
            do {
                int exerciseId = cursor.getInt(0);

                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "exerciseId " + exerciseId);
                }

                String title;
                Cursor titleCursor = db.rawQuery("SELECT Title FROM Exercise WHERE ExerciseId = " + exerciseId + ";", null);
                titleCursor.moveToFirst();
                title = titleCursor.getString(0);

                ArrayList<String> tags = new ArrayList<>();
                Cursor tagsId = db.rawQuery("SELECT TagId FROM ExerciseToTag WHERE ExerciseId = " + exerciseId + ";", null);
                while (tagsId.moveToNext()) {
                    int tagId = tagsId.getInt(0);

                    Cursor tagTitle = db.rawQuery("SELECT Title FROM ExerciseTag WHERE TagId = " + tagId + ";", null);
```

```
                while (tagTitle.moveToNext()) {
                    tags.add(tagTitle.getString(0));
                }
            }

            exerciseOverviewList.add(new ExerciseOverview(title, tags));
        } while (cursor.moveToNext());
    }

    return exerciseOverviewList;
}

private String getSQLQueryWithAllNeededExercisesId(List<Integer> selectedMusclesId) {
    StringBuilder query = new StringBuilder();
    query.append("SELECT ExerciseId FROM TargetMuscle WHERE MuscleId IN (");
    for (Integer muscleId : selectedMusclesId) {
        query.append(muscleId);
        query.append(", ");
    }
    query.delete(query.length() - 2, query.length());
    query.append(") GROUP BY ExerciseId HAVING COUNT(*) = ");
    query.append(selectedMusclesId.size());
    query.append(";");

    return query.toString();
}

private String getSQLQueryWithAllFilterMuscleId() {
    StringBuilder query = new StringBuilder();
    query.append("SELECT MuscleId FROM Muscle WHERE Title IN (");
    for (String filter : filters) {
        query.append("'");
        query.append(filter);
        query.append("', ");
    }
    query.delete(query.length() - 2, query.length());
    query.append(");");

    return query.toString();
}
}
```

1.52. FilterCancelLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;
```

```
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingDayOfWeekHelper;
import com.example.testbottomnavigationbar.fragments.SearchFragment;
import com.example.testbottomnavigationbar.remote_db.Training;

public class FilterCancelLogic implements View.OnClickListener {
    private int type;
    private final TrainingDayOfWeekHelper trainingDayOfWeekHelper;

    public FilterCancelLogic(int type, TrainingDayOfWeekHelper trainingDayOfWeekHelper) {
        this.type = type;
        this.trainingDayOfWeekHelper = trainingDayOfWeekHelper;
    }

    @Override
    public void onClick(View v) {
        if (type == 0) {
            ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
        } else {
            ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();

            Fragment fragment = new
SearchFragment(SearchFragment.generateExerciseOverviewList(v.getContext()), 1,
trainingDayOfWeekHelper);
            FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
            fTrans.replace(R.id.fragment_cv, fragment).commit();
            fTrans.addToBackStack(null);
        }
    }
}
```

1.53. MoveTrainingExerciseLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
```

```
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.remote_db.tasks.MoveTrainingExerciseTask;

public class MoveTrainingExerciseLogic implements View.OnClickListener {
    private final int type;
    private final TrainingExerciseInstance trainingExerciseInstance;
    private int orderNumber = -1;

    public MoveTrainingExerciseLogic(int type, TrainingExerciseInstance trainingExerciseInstance) {
        this.type = type;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    public void onClick(View v) {
        if (!checkOrderNumber(v)) {
            return;
        }

        moveTrainingExerciseInDB(v);

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void moveTrainingExerciseInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        if (type == 0) {
            updateInDB(db, accountId, "TrainingExerciseSuccess");
            updateInDB(db, accountId, "TrainingExerciseNote");
        } else {
            updateInDB(db, accountId, "TrainingExercise");
        }

        moveTrainingExerciseInRemoteDB(accountId);
```

```
    }

    private void updateInDB(SQLiteDatabase db, int accountId, String tableName) {
        // Перемещаем все сеты далеко вперед
        db.execSQL("UPDATE " + tableName +
                " SET OrderNumber = 100 " +
                "WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + " AND OrderNumber =
" + trainingExerciseInstance.getOrderNumber() + ";");

        // Сдвигаем те, что между
        if (orderNumber > trainingExerciseInstance.getOrderNumber()) {
            for (int i = trainingExerciseInstance.getOrderNumber() + 1; i <= orderNumber; ++i) {
                db.execSQL("UPDATE " + tableName +
                        " SET OrderNumber = " + (i - 1) +
                        " WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                        " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() +
                        " AND OrderNumber = " + i + ";");
            }
        } else {
            for (int i = trainingExerciseInstance.getOrderNumber() - 1; i >= orderNumber; --i) {
                db.execSQL("UPDATE " + tableName +
                        " SET OrderNumber = " + (i + 1) +
                        " WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                        " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() +
                        " AND OrderNumber = " + i + ";");
            }
        }

        // Ставим нужное на новое место
        db.execSQL("UPDATE " + tableName +
                " SET OrderNumber = " + orderNumber +
                " WHERE AccountId = " + accountId + " AND TrainingId = " +
trainingExerciseInstance.getTrainingId() +
                " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + " AND OrderNumber =
100;");
    }

    private void moveTrainingExerciseInRemoteDB(int accountId) {
        new MoveTrainingExerciseTask(type, accountId, orderNumber, trainingExerciseInstance).execute();
    }

    private boolean checkOrderNumber(View view) {
```

```
    EditText orderNumberET = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_editing_training_exercise_et);

    if (isNumber(orderNumberET.getText().toString())) {
      int orderNumber = Integer.parseInt(orderNumberET.getText().toString());
      if (check(view, orderNumber)) {
        this.orderNumber = orderNumber;
        return true;
      }
    }

    return false;
  }

  private boolean check(View view, int orderNumber) {
    return (orderNumber > 0 && orderNumber <= getMaxOrderNumber(view) && orderNumber != trainingExerciseInstance.getOrderNumber());
  }

  private int getMaxOrderNumber(View view) {
    SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);
    SQLiteDatabase currentAccountDB = view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(), Context.MODE_PRIVATE, null);

    int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

    if (type == 0) {
      Cursor cursor = db.rawQuery("SELECT MAX(OrderNumber) " +
          "FROM TrainingExerciseSuccess " +
          "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingExerciseInstance.getTrainingId() +
          " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + ";", null);

      if (cursor != null && cursor.moveToFirst()) {
        return cursor.getInt(0);
      }
    } else {
      Cursor cursor = db.rawQuery("SELECT MAX(OrderNumber) " +
          "FROM TrainingExercise " +
          "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingExerciseInstance.getTrainingId() +
          " AND DayOfWeek = " + trainingExerciseInstance.getDayOfWeek() + ";", null);

      if (cursor != null && cursor.moveToFirst()) {
        return cursor.getInt(0);
```

```
            }
        }

        return -1;
    }

    private boolean isNumber(String s) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "isNumber   " + s);
        }

        if (s.length() == 0) {
            return false;
        }

        for (int i = 0; i < s.length(); ++i) {
            if (s.charAt(i) - '0' < 0 || s.charAt(i) - '0' > 9) {
                return false;
            }
        }

        return true;
    }
}
```

1.54. NewExerciseDescriptionLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AddExerciseDescriptionFragment;
import com.example.testbottomnavigationbar.fragments.AddExerciseVideoFragment;

import java.util.HashSet;

public class NewExerciseDescriptionLogic implements View.OnClickListener {
    private HashSet<String> tags;
    private HashSet<String> filters;
```

```
    private String title;
    private String description;

    public NewExerciseDescriptionLogic(HashSet<String> tags, HashSet<String> filters, String title) {
        this.tags = tags;
        this.filters = filters;
        this.title = title;
    }

    @Override
    public void onClick(View v) {
        findDescription(v);
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "description   " + description);
        }

        Fragment fragment = new AddExerciseVideoFragment(tags, filters, title, description);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }

    private void findDescription(View view) {
        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_description_et);
        description = editText.getText().toString();
    }
}
```

1.55. NewExerciseFilterLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.AddExerciseDescriptionFragment;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
```

```
import androidx.fragment.app.FragmentTransaction;

import java.util.HashSet;

public class NewExerciseFilterLogic implements View.OnClickListener {
    private HashSet<String> tags;
    private HashSet<String> filters;
    private String title;

    public NewExerciseFilterLogic(HashSet<String> tags) {
        filters = new HashSet<>();
        this.tags = tags;
    }

    public void addFilter(String filter) {
        filters.add(filter);
    }

    public void removeFilter(String filter) {
        filters.remove(filter);
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @Override
    public void onClick(View v) {
        if (!findTitle(v)) {
            return;
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "title   " + title);
        }

        Fragment fragment = new AddExerciseDescriptionFragment(tags, filters, title);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }

    private boolean findTitle(View view) {
        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_title_et);
        title = editText.getText().toString();
```

```
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        int titleId = SQLiteHelper.getExerciseIdFromTitle(db, title);

        return (!title.equals("") && titleId == -1);
    }
}
```

1.56. NewExerciseTagsLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.util.Log;
import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AddExerciseDescriptionFragment;
import com.example.testbottomnavigationbar.fragments.AddExerciseMuscleFragment;

import java.util.HashSet;

public class NewExerciseTagsLogic implements View.OnClickListener {
    private HashSet<String> tags;

    public NewExerciseTagsLogic() {
        tags = new HashSet<>();
    }

    public void addTag(String tag) {
        tags.add(tag);
    }

    public void removeTag(String tag) {
        tags.remove(tag);
    }

    @Override
    public void onClick(View v) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "tags   " + tags.size());
        }
```

```
        if (tags.size() > 3) {
            return;
        }

        Fragment fragment = new AddExerciseMuscleFragment(tags);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.57. NewExerciseVideoLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.SearchFragment;
import com.example.testbottomnavigationbar.fragments.TrainingExerciseSuccessFragment;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.remote_db.ExerciseTag;
import com.example.testbottomnavigationbar.remote_db.ExerciseToTag;
import com.example.testbottomnavigationbar.remote_db.TargetMuscle;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertNewExerciseTask;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingExerciseSuccessTask;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

public class NewExerciseVideoLogic implements View.OnClickListener {
    private HashSet<String> tags;
    private HashSet<String> filters;
```

```
    private String title;
    private String description;
    private String videoPath;

    public NewExerciseVideoLogic(HashSet<String> tags, HashSet<String> filters, String title, String
description) {
        this.tags = tags;
        this.filters = filters;
        this.title = title;
        this.description = description;
    }

    @Override
    public void onClick(View v) {
        findVideoPath(v);
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "tags   " + tags.size());
            Log.d(MainActivity.TEG, "video   " + videoPath);
            Log.d(MainActivity.TEG, "description   " + description);
            Log.d(MainActivity.TEG, "title   " + title);
            Log.d(MainActivity.TEG, "filters   " + filters.size());
        }

        addInDB(v.getContext());

        ((FragmentActivity)
v.getContext()).getSupportFragmentManager().popBackStackImmediate("mainFragment",
FragmentManager.POP_BACK_STACK_INCLUSIVE);
        Fragment fragment = new
SearchFragment(SearchFragment.generateExerciseOverviewList(v.getContext()), 0, null);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment, "searchFragment").commit();
        fTrans.addToBackStack(null);
    }

    private void findVideoPath(View view) {
        EditText editText = ((FragmentActivity)
view.getContext()).findViewById(R.id.fragment_add_exercise_video_et);
        videoPath = editText.getText().toString();
    }

    private void addInDB(Context context) {
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);

        db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
```

```
            "VALUES ('" + title + "', '" + description + "', '" + videoPath + "');");

        int exerciseId = SQLiteHelper.getExerciseIdFromTitle(db, title);
        List<ExerciseToTag> exerciseToTagList = new ArrayList<>();
        List<TargetMuscle> targetMuscleList = new ArrayList<>();

        for (String tag : tags) {
            ExerciseToTag exerciseToTag = new ExerciseToTag(exerciseId,
SQLiteHelper.getTagIdFromTitle(db, tag));
            exerciseToTagList.add(exerciseToTag);

            db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
                    "VALUES ('" + exerciseId + "', '" + exerciseToTag.getTagId() + "');");
        }

        for (String filter : filters) {
            TargetMuscle targetMuscle = new TargetMuscle(exerciseId,
SQLiteHelper.getMuscleIdFromTitle(db, filter));
            targetMuscleList.add(targetMuscle);

            db.execSQL("INSERT INTO TargetMuscle(ExerciseId, MuscleId)\n" +
                    "VALUES ('" + exerciseId + "', '" + targetMuscle.getMuscleId() + "');");
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "new exerciseId   " + SQLiteHelper.getExerciseIdFromTitle(db, title));
        }


        addInRemoteDB(exerciseId, exerciseToTagList, targetMuscleList);
    }

    private void addInRemoteDB(int exerciseId, List<ExerciseToTag> exerciseTagList,
List<TargetMuscle> targetMuscleList) {
        new InsertNewExerciseTask(new Exercise(exerciseId, title, description, videoPath), exerciseTagList,
targetMuscleList).execute();
    }
}
```

1.58. NewTrainingCreationLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
```

```
import android.widget.EditText;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingTask;


public class NewTrainingCreationLogic implements View.OnClickListener {
    private String title = "";

    @Override
    public void onClick(View v) {
        if (!findTitle(v)) {
            return;
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "title   " + title);
        }

        addTrainingInDB(v);

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private boolean findTitle(View view) {
        EditText editText = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_add_training_title_et);
        title = editText.getText().toString();

        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB = view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(), Context.MODE_PRIVATE, null);

        int titleId = SQLiteHelper.getTrainingIdFromTitle(db, title, SQLiteHelper.getCurrentAccountId(currentAccountDB));

        return (!title.equals("") && titleId == -1);
    }

    private void addTrainingInDB(View view) {
```

```
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB = view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        db.execSQL("INSERT INTO Training(Title, AccountId)\n" +
                "VALUES ('" + title + "', '" + accountId + "');");

        addTrainingInRemoteDB(accountId, db);
    }

    private void addTrainingInRemoteDB(int accountId, SQLiteDatabase db) {
        new InsertTrainingTask(new Training(SQLiteHelper.getTrainingIdFromTitle(db, title, accountId),
title, accountId), 0, null).execute();
    }
}
```

1.59. OpenEditingTrainingExeciseFragmentLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.fragments.EditingTrainingExerciseFragment;

public class OpenEditingTrainingExeciseFragmentLogic implements View.OnLongClickListener {
    private final int type;
    private final TrainingExerciseInstance trainingExerciseInstance;

    public OpenEditingTrainingExeciseFragmentLogic(int type, TrainingExerciseInstance
trainingExerciseInstance) {
        this.type = type;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    public boolean onLongClick(View v) {
        Fragment fragment = new EditingTrainingExerciseFragment(type, trainingExerciseInstance);
```

```
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);

        return true;
    }
}
```

1.60. OpenExerciseFromQuestionLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AddExerciseTagFragment;
import com.example.testbottomnavigationbar.fragments.ExerciseDescriptionFragment;

public class OpenExerciseFromQuestionLogic implements View.OnClickListener {
    private final String title;

    public OpenExerciseFromQuestionLogic(String title) {
        this.title = title;
    }

    @Override
    public void onClick(View v) {
        Fragment fragment = new ExerciseDescriptionFragment(title, 0, null);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.61. RemoveAccountLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;
```

```
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.LoginFragment;
import com.example.testbottomnavigationbar.remote_db.tasks.DeleteAccountTask;

public class RemoveAccountLogic implements View.OnClickListener {
    private final int accountId;

    public RemoveAccountLogic(int accountId) {
        this.accountId = accountId;
    }

    @Override
    public void onClick(View v) {
        removeAccountFromDB(v);

        SQLiteDatabase currentAccountDb =
v.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);
        currentAccountDb.execSQL("DELETE FROM CurrentAccount");

        Fragment fragment = new LoginFragment();
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStackImmediate(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
        fTrans.replace(R.id.activity_main_fragment_cv, fragment, "loginFragment").commit();
        fTrans.addToBackStack(null);
    }

    private void removeAccountFromDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        db.execSQL("PRAGMA foreign_keys=ON;");

        db.execSQL("DELETE FROM Account WHERE AccountId = " + accountId + ";");

        removeFriendFromRemoteDB();
    }

    private void removeFriendFromRemoteDB() {
```

```
        new DeleteAccountTask(accountId).execute();
    }
}
```

1.62. RemoveFriendLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.AccountFragment;
import com.example.testbottomnavigationbar.remote_db.Friendship;
import com.example.testbottomnavigationbar.remote_db.tasks.DeleteFriendshipTask;

public class RemoveFriendLogic implements View.OnClickListener {
    private final int accountId;
    private final int friendId;

    public RemoveFriendLogic(int accountId, int friendId) {
        this.accountId = accountId;
        this.friendId = friendId;
    }

    @Override
    public void onClick(View v) {
        addFriendInDb(v);

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
        Fragment fragment = new AccountFragment(1, friendId);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment, null).commit();
        fTrans.addToBackStack(null);
    }

    private void addFriendInDb(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        db.execSQL("PRAGMA foreign_keys=ON;");
```

```
        db.execSQL("DELETE FROM Friendship WHERE AccountId = " + accountId + " AND FriendId =
" + friendId + ";");

        addFriendInRemoteDB();
    }

    private void addFriendInRemoteDB() {
        new DeleteFriendshipTask(new Friendship(accountId, friendId)).execute();
    }
}
```

1.63. RemoveTrainingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingEditingHelper;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.remote_db.tasks.DeleteTrainingTask;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateTrainingTask;

public class RemoveTrainingLogic implements View.OnClickListener{
    private final TrainingEditingHelper trainingEditingHelper;
    private boolean needSetNullCurrentTraining = false;

    public RemoveTrainingLogic(TrainingEditingHelper trainingEditingHelper) {
        this.trainingEditingHelper = trainingEditingHelper;
    }

    @Override
    public void onClick(View v) {
        try {
            removeTrainingInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Remove training value problems");
            }
```

```
        }

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void removeTrainingInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int trainingId = trainingEditingHelper.getTrainingId();

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Delete trainingId:  " + trainingId);
            Log.d(MainActivity.TEG, "Delete local trainingId:  " + SQLiteHelper.getTrainingIdFromTitle(db,
trainingEditingHelper.getTitle(), SQLiteHelper.getCurrentAccountId(currentAccountDB)));
        }

        db.execSQL("PRAGMA foreign_keys=ON;");
        db.execSQL("DELETE FROM Training\n" +
                "WHERE TrainingId = " + trainingId + ";");

        if (MainActivity.LOG) {
            Cursor cursor = db.rawQuery("SELECT * FROM TrainingExercise WHERE TrainingId = " +
trainingId + ";", null);
            Log.d(MainActivity.TEG, "After delete training:  " + (cursor != null && cursor.moveToFirst()));
        }

        setNullCurrentTraining(db, currentAccountDB);
        removeTrainingInRemoteDB(db, currentAccountDB);
    }

    private void removeTrainingInRemoteDB(SQLiteDatabase db, SQLiteDatabase currentAccountDB) {
        new DeleteTrainingTask(trainingEditingHelper.getTrainingId(), needSetNullCurrentTraining,
SQLiteHelper.getCurrentAccountId(currentAccountDB)).execute();
    }

    private void setNullCurrentTraining(SQLiteDatabase db, SQLiteDatabase currentAccountDB) {
        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
        int currentTrainingId = SQLiteHelper.getCurrentTrainingId(db, currentAccountDB);

        if (trainingEditingHelper.getTrainingId() == currentTrainingId) {
            needSetNullCurrentTraining = true;

            db.execSQL("UPDATE Account\n" +
```

```
                "SET TrainingId = null\n" +
                "WHERE AccountId = " + accountId + ";");
        }
    }
}
```

1.64. SaveAccountEditingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateAccountInfoTask;

public class SaveAccountEditingLogic implements View.OnClickListener {
    private final Account account;
    private final BodyCondition bodyCondition;
    private Account newAccount = new Account();
    private BodyCondition newBodyCondition = new BodyCondition();

    public SaveAccountEditingLogic(Account account, BodyCondition bodyCondition) {
        this.account = account;
        this.bodyCondition = bodyCondition;
    }

    @Override
    public void onClick(View v) {
        if (!collectData(v)) {
            return;
        }

        addInfoInDB(v);
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void addInfoInDB(View view) {
```

```
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);

        db.execSQL("UPDATE Account" +
                " SET FirstName = '" + newAccount.getFirstName() + "', SecondName = '" +
newAccount.getSecondName() + "', UserName = '" + newAccount.getUserName() + "'" +
                " WHERE AccountId = " + account.getAccountId() + ";");

        db.execSQL("UPDATE BodyCondition" +
                " SET Age = " + newBodyCondition.getAge() + ", Weight = " + newBodyCondition.getWeight()
+ ", Height = " + newBodyCondition.getHeight() + ", BodyFatShare = " +
newBodyCondition.getBodyFatShare() +
                " WHERE AccountId = " + account.getAccountId() + ";");

        addInfoInRemoteDB();
    }

    private void addInfoInRemoteDB() {
        new UpdateAccountInfoTask(account, bodyCondition, newAccount, newBodyCondition).execute();
    }

    private boolean collectData(View view) {
        return (collectFirstName(view) && collectSecondName(view) && collectUsername(view) &&
                collectAge(view) && collectWeight(view) && collectHeight(view) &&
collectBodyFatShare(view));
    }

    private boolean collectFirstName(View view) {
        EditText firstName = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_name_edit);
        String strFirstName = firstName.getText().toString();
        newAccount.setFirstName(strFirstName);

        return !strFirstName.equals("");
    }

    private boolean collectSecondName(View view) {
        EditText secondName = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_second_name_edit);
        String strSecondName = secondName.getText().toString();
        newAccount.setSecondName(strSecondName);

        return !strSecondName.equals("");
    }

    private boolean collectUsername(View view) {
```

```
        EditText username = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_second_username_edit);
        String strUsername = username.getText().toString();
        newAccount.setUserName(strUsername);
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);

        return (strUsername.equals(account.getUserName()) || (!strUsername.equals("") && SQLiteHelper.getAccountIdFromUsername(db, strUsername) == -1));
    }

    private boolean collectAge(View view) {
        EditText age = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_age_edit);
        String strAge = age.getText().toString();

        try {
            newBodyCondition.setAge(Integer.parseInt(strAge));
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "edittext:  " + e, e);
            }
            return false;
        }

        return (newBodyCondition.getAge() > 0);
    }

    private boolean collectWeight(View view) {
        EditText weight = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_weight_edit);
        String strWeight = weight.getText().toString();

        try {
            newBodyCondition.setWeight(Float.parseFloat(strWeight));
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "edittext:  " + e, e);
            }
            return false;
        }

        return (newBodyCondition.getWeight() > 0);
    }

    private boolean collectHeight(View view) {
```

```
        EditText height = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_height_edit);
        String strHeight = height.getText().toString();

        try {
            newBodyCondition.setHeight(Float.parseFloat(strHeight));
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "edittext:  " + e, e);
            }
            return false;
        }

        return (newBodyCondition.getHeight() > 0);
    }

    private boolean collectBodyFatShare(View view) {
        EditText bodyFatShare = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_percent_edit);
        String strBodyFatShare = bodyFatShare.getText().toString();

        try {
            newBodyCondition.setBodyFatShare(Float.parseFloat(strBodyFatShare));
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "edittext:  " + e, e);
            }
            return false;
        }

        return (newBodyCondition.getBodyFatShare() > 0);
    }
}
```

1.65. SetCurrentTrainingLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
```

```
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.TrainingEditingHelper;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseNote;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateCurrentTrainingTask;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateTrainingTask;

import java.util.ArrayList;
import java.util.List;

public class SetCurrentTrainingLogic implements View.OnClickListener {
    private final TrainingEditingHelper trainingEditingHelper;
    private List<TrainingExerciseSuccess> trainingExerciseSuccessList = new ArrayList<>();
    private List<TrainingExerciseNote> trainingExerciseNoteList = new ArrayList<>();

    public SetCurrentTrainingLogic(TrainingEditingHelper trainingEditingHelper) {
        this.trainingEditingHelper = trainingEditingHelper;
    }

    @Override
    public void onClick(View v) {
        try {
            updateCurrentTrainingInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Update current training value problems" + e, e);
            }
        }

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void updateCurrentTrainingInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int trainingId = trainingEditingHelper.getTrainingId();
        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);

        db.execSQL("UPDATE Account\n" +
                "SET TrainingId = " + trainingId + "\n" +
                "WHERE AccountId = " + accountId + ";");
```

```
        removeTrainingExerciseSuccessFromDB(db, accountId, trainingId);
        removeTrainingExerciseNoteFromDB(db, accountId, trainingId);
        addNewTrainingExerciseSuccess(db, accountId, trainingId);
        addNewTrainingExerciseNote(db, accountId, trainingId);

        updateCurrentTrainingInRemoteDB(accountId, trainingId);
    }

    private void updateCurrentTrainingInRemoteDB(int accountId, int trainingId) {
        new UpdateCurrentTrainingTask(accountId, trainingId, trainingExerciseSuccessList,
trainingExerciseNoteList).execute();
    }

    private void removeTrainingExerciseSuccessFromDB(SQLiteDatabase db, int accountId, int trainingId)
{
        db.execSQL("DELETE FROM TrainingExerciseSuccess\n" +
                "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + ";");
    }

    private void removeTrainingExerciseNoteFromDB(SQLiteDatabase db, int accountId, int trainingId) {
        db.execSQL("DELETE FROM TrainingExerciseNote\n" +
                "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + ";");
    }

    private void addNewTrainingExerciseSuccess(SQLiteDatabase db, int accountId, int trainingId) {
        Cursor cursor = db.rawQuery("SELECT * FROM TrainingExercise " +
                "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + ";", null);

        if (cursor != null && cursor.moveToFirst()) {
            do {
                int dayOfWeek = cursor.getInt(1);
                int orderNumber = cursor.getInt(2);
                int exerciseId = cursor.getInt(3);
                int setNumber = cursor.getInt(4);
                int repsNum = cursor.getInt(5);
                int timer = cursor.getInt(6);

                trainingExerciseSuccessList.add(new TrainingExerciseSuccess(accountId, trainingId,
dayOfWeek, orderNumber, exerciseId, setNumber, repsNum, 0, timer));

                db.execSQL("INSERT INTO TrainingExerciseSuccess(AccountId, TrainingId, DayOfWeek,
OrderNumber, ExerciseId, SetNumber, RepsNum, Weight, Timer)\n" +
                        "VALUES ('" + accountId + "', '" + trainingId + "', '" + dayOfWeek +
                        "', '" + orderNumber + "', '" + exerciseId + "', '" + setNumber + "', '" + repsNum + "', '" + 0
+ "', '" + timer + "');");
            } while (cursor.moveToNext());
        }
```

```
    }

    private void addNewTrainingExerciseNote(SQLiteDatabase db, int accountId, int trainingId) {
        Cursor cursor = db.rawQuery("SELECT DISTINCT DayOfWeek, OrderNumber, ExerciseId  FROM
TrainingExercise " +
                "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + ";", null);

        if (cursor != null && cursor.moveToFirst()) {
            do {
                int dayOfWeek = cursor.getInt(0);
                int orderNumber = cursor.getInt(1);
                int exerciseId = cursor.getInt(2);

                trainingExerciseNoteList.add(new TrainingExerciseNote(trainingId, accountId, dayOfWeek,
orderNumber, exerciseId, ""));

                db.execSQL("INSERT INTO TrainingExerciseNote(TrainingId, AccountId, DayOfWeek,
OrderNumber, ExerciseId, Note)\n" +
                        "VALUES ('" + trainingId + "', '" + accountId + "', '" + dayOfWeek +
                        "', '" + orderNumber + "', '" + exerciseId + "', '');");
            } while (cursor.moveToNext());
        }
    }
}
```

1.66. SetEditLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.InsertSetJSON;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingExerciseSuccessTask;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertTrainingExerciseTask;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateTrainingExerciseSuccessTask;
```

```
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateTrainingExerciseTask;

import java.io.IOException;

public class SetEditLogic implements View.OnClickListener {
    private final int type;
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public SetEditLogic(int type, ExerciseInTrainingEditing exerciseInTrainingEditing) {
        this.type = type;
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    public void onClick(View v) {
        if (!checkInput(v)) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Bad input value to add set");
            }

            return;
        }

        try {
            addSetInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Add set value problems");
            }
        }
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void addSetInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
        int trainingId = exerciseInTrainingEditing.getTrainingId();
        int dayOfWeek = exerciseInTrainingEditing.getDayOfWeek();
        int orderNumber = exerciseInTrainingEditing.getOrderNumber();
        int exerciseId = exerciseInTrainingEditing.getExerciseId();
        int setNumber = exerciseInTrainingEditing.getSetNumber();
```

```
        if (type == 0) {
            db.execSQL("UPDATE TrainingExerciseSuccess\n" +
                    "SET RepsNum = " + getRepsNum(view) + ", Weight = " + getWeight(view) + ", Timer = " +
getTimer(view) + "\n" +
                    "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + " " +
                    "AND DayOfWeek = " + dayOfWeek + " AND OrderNumber = " + orderNumber + " AND
SetNumber = " + setNumber + ";");
        } else {
            db.execSQL("UPDATE TrainingExercise\n" +
                    "SET RepsNum = " + getRepsNum(view) + ", Timer = " + getTimer(view) + "\n" +
                    "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + " " +
                    "AND DayOfWeek = " + dayOfWeek + " AND OrderNumber = " + orderNumber + " AND
SetNumber = " + setNumber + ";");
        }

        addSetInRemoteDB(view);
    }

    private void addSetInRemoteDB(View view) {
        if (type == 0) {
            new UpdateTrainingExerciseSuccessTask(new InsertSetJSON(getWeight(view),
getRepsNum(view), getTimer(view)), exerciseInTrainingEditing).execute();
        } else {
            new UpdateTrainingExerciseTask(new InsertSetJSON(-1, getRepsNum(view), getTimer(view)),
exerciseInTrainingEditing).execute();
        }
    }

    private boolean checkInput(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText repsNum = (EditText) constraintLayout.getChildAt(4);
        EditText weight = (EditText) constraintLayout.getChildAt(1);
        EditText timer = (EditText) constraintLayout.getChildAt(7);

        if (!isNumber(repsNum.getText().toString()) || getRepsNum(view) < 0) {
            return false;
        }
        if (type == 0 && (!isNumber(weight.getText().toString()) || getWeight(view) < 0)) {
            return false;
        }
        if (!isNumber(timer.getText().toString()) || getTimer(view) < 0) {
            return false;
        }

        return true;
    }
```

```
    private boolean isNumber(String s) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "isNumber   " + s);
        }

        if (s.length() == 0) {
            return false;
        }

        for (int i = 0; i < s.length(); ++i) {
            if (s.charAt(i) - '0' < 0 || s.charAt(i) - '0' > 9) {
                return false;
            }
        }

        return true;
    }

    private int getRepsNum(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText editText = (EditText) constraintLayout.getChildAt(4);

        return Integer.parseInt(editText.getText().toString());
    }

    private int getWeight(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText editText = (EditText) constraintLayout.getChildAt(1);

        return Integer.parseInt(editText.getText().toString());
    }

    private int getTimer(View view) {
        ConstraintLayout constraintLayout = (ConstraintLayout) view.getParent();
        EditText editText = (EditText) constraintLayout.getChildAt(7);

        return Integer.parseInt(editText.getText().toString());
    }
}
```

1.67. SetRemoveLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
```

```
import android.view.View;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.remote_db.tasks.DeleteTrainingExerciseSuccessTask;
import com.example.testbottomnavigationbar.remote_db.tasks.DeleteTrainingExerciseTask;

public class SetRemoveLogic implements View.OnClickListener {
    private final int type;
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public SetRemoveLogic(int type, ExerciseInTrainingEditing exerciseInTrainingEditing) {
        this.type = type;
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    public void onClick(View v) {
        try {
            removeSetInDB(v);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Remove set value problems");
            }
        }

        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStack();
    }

    private void removeSetInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB =
view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

        int accountId = SQLiteHelper.getCurrentAccountId(currentAccountDB);
        int trainingId = exerciseInTrainingEditing.getTrainingId();
        int dayOfWeek = exerciseInTrainingEditing.getDayOfWeek();
        int orderNumber = exerciseInTrainingEditing.getOrderNumber();
        int exerciseId = exerciseInTrainingEditing.getExerciseId();
        int setNumber = exerciseInTrainingEditing.getSetNumber();

        if (type == 0) {
```

```
            db.execSQL("DELETE FROM TrainingExerciseSuccess\n" +
                    "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + " " +
                    "AND DayOfWeek = " + dayOfWeek + " AND OrderNumber = " + orderNumber + " AND
SetNumber = " + setNumber + ";");
        } else {
            db.execSQL("DELETE FROM TrainingExercise\n" +
                    "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + " " +
                    "AND DayOfWeek = " + dayOfWeek + " AND OrderNumber = " + orderNumber + " AND
SetNumber = " + setNumber + ";");
        }

        removeSetInRemoteDB();
    }

    private void removeSetInRemoteDB() {
        if (type == 0) {
            new DeleteTrainingExerciseSuccessTask(exerciseInTrainingEditing).execute();
        } else {
            new DeleteTrainingExerciseTask(exerciseInTrainingEditing).execute();
        }
    }
}
```

1.68. SignInLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.MainFragment;

public class SignInLogic implements View.OnClickListener {
    @Override
    public void onClick(View v) {
        if (checkInfo(v)) {
```

```
        ((FragmentActivity)
v.getContext()).getSupportFragmentManager().popBackStackImmediate(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);

        Fragment fragment = new MainFragment();
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.activity_main_fragment_cv, fragment, "mainFragment").commit();
        fTrans.addToBackStack(null);
      }
   }

   private boolean checkInfo(View view) {
      EditText username = ((FragmentActivity)
view.getContext()).findViewById(R.id.login_layout_login_et);
      String strUsername = username.getText().toString();

      EditText password = ((FragmentActivity)
view.getContext()).findViewById(R.id.login_layout_password_et);
      String strPassword = password.getText().toString();

      return checkAccountInDb(view.getContext(), strUsername, strPassword);
   }

   private boolean checkAccountInDb(Context context, String username, String password) {
      SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
      SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(),
Context.MODE_PRIVATE, null);

      Cursor cursor = db.rawQuery("SELECT AccountId FROM Account WHERE Username = '" +
username + "' AND HashPassword = '" + password + "';", null);
      if (cursor != null && cursor.moveToFirst()) {
         int accountId = cursor.getInt(0);

         currentAccountDB.execSQL("INSERT INTO CurrentAccount(AccountId)\n" +
               "VALUES ('" + accountId + "')");
         return true;
      }

      return false;
   }
}
```

1.69. SignUpLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.MainFragment;
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;
import com.example.testbottomnavigationbar.remote_db.tasks.SignUpTask;
import com.example.testbottomnavigationbar.remote_db.tasks.UpdateAccountInfoTask;

public class SignUpLogic implements View.OnClickListener {
    private Account newAccount = new Account();
    private BodyCondition newBodyCondition = new BodyCondition();

    @Override
    public void onClick(View v) {
        if (!collectData(v)) {
            return;
        }

        addInfoInDB(v);

        Fragment fragment = new MainFragment();
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        ((FragmentActivity) v.getContext()).getSupportFragmentManager().popBackStackImmediate(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
        fTrans.replace(R.id.activity_main_fragment_cv, fragment, "mainFragment").commit();
        fTrans.addToBackStack(null);
    }

    private void addInfoInDB(View view) {
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(),
Context.MODE_PRIVATE, null);
```

```
        SQLiteDatabase currentAccountDB = view.getContext().openOrCreateDatabase(SQLiteHelper.getCurrentAccountDBName(), Context.MODE_PRIVATE, null);

        db.execSQL("INSERT INTO Account(Username, HashPassword, BirthDate, RegistrationDate, Sex, TrainingId, FirstName, SecondName)\n" +
                "VALUES ('" + newAccount.getUserName() + "', '" + newAccount.getHashPassword() + "', '2020-01-01', '2020-01-01', 'Мужской', null, '" + newAccount.getFirstName() + "', '" + newAccount.getSecondName() + "');");

        int accountId = -1;
        Cursor cursor = db.rawQuery("SELECT AccountId FROM Account WHERE Username = '" + newAccount.getUserName() + "';", null);
        if (cursor != null && cursor.moveToFirst()) {
            accountId = cursor.getInt(0);
            newAccount.setAccountId(accountId);
            newBodyCondition.setAccountId(accountId);
        }

        db.execSQL("INSERT INTO BodyCondition(AccountId, Weight, Height, Age, BodyFatShare)\n" +
                "VALUES ('" + accountId + "', '" + newBodyCondition.getWeight() + "', '" + newBodyCondition.getHeight() + "', '" + newBodyCondition.getAge() + "', '" + newBodyCondition.getBodyFatShare() + "');");

        currentAccountDB.execSQL("INSERT INTO CurrentAccount(AccountId)\n" +
                "VALUES ('" + accountId + "');");

        newAccount.setBirthDate("2020-01-01");
        newAccount.setRegistrationDate("2020-01-01");
        newAccount.setSex("Мужской");
        newAccount.setTrainingId(null);
        addInfoInRemoteDB();
    }

    private void addInfoInRemoteDB() {
        new SignUpTask(newAccount, newBodyCondition).execute();
    }

    private boolean collectData(View view) {
        return (collectFirstName(view) && collectSecondName(view) && collectUsername(view) && collectPassword(view) &&
                collectAge(view) && collectWeight(view) && collectHeight(view) && collectBodyFatShare(view));
    }

    private boolean collectFirstName(View view) {
```

```
        EditText firstName = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_name_edit);
        String strFirstName = firstName.getText().toString();
        newAccount.setFirstName(strFirstName);

        return !strFirstName.equals("");
    }

    private boolean collectSecondName(View view) {
        EditText secondName = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_second_name_edit);
        String strSecondName = secondName.getText().toString();
        newAccount.setSecondName(strSecondName);

        return !strSecondName.equals("");
    }

    private boolean collectUsername(View view) {
        EditText username = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_second_username_edit);
        String strUsername = username.getText().toString();
        newAccount.setUserName(strUsername);
        SQLiteDatabase db = view.getContext().openOrCreateDatabase(SQLiteHelper.getDbName(), Context.MODE_PRIVATE, null);

        return (!strUsername.equals("") && SQLiteHelper.getAccountIdFromUsername(db, strUsername) == -1);
    }

    private boolean collectPassword(View view) {
        EditText password = ((FragmentActivity) view.getContext()).findViewById(R.id.fragment_sign_up_password_edit);
        String strPassword = password.getText().toString();
        newAccount.setHashPassword(strPassword);

        return !strPassword.equals("");
    }

    private boolean collectAge(View view) {
        EditText age = ((FragmentActivity) view.getContext()).findViewById(R.id.account_editing_age_edit);
        String strAge = age.getText().toString();

        try {
            newBodyCondition.setAge(Integer.parseInt(strAge));
        } catch (Exception e) {
            if (MainActivity.LOG) {
```

```
            Log.d(MainActivity.TEG, "edittext:  " + e, e);
        }
        return false;
    }

    return (newBodyCondition.getAge() > 0);
}

private boolean collectWeight(View view) {
    EditText weight = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_weight_edit);
    String strWeight = weight.getText().toString();

    try {
        newBodyCondition.setWeight(Float.parseFloat(strWeight));
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "edittext:  " + e, e);
        }
        return false;
    }

    return (newBodyCondition.getWeight() > 0);
}

private boolean collectHeight(View view) {
    EditText height = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_height_edit);
    String strHeight = height.getText().toString();

    try {
        newBodyCondition.setHeight(Float.parseFloat(strHeight));
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "edittext:  " + e, e);
        }
        return false;
    }

    return (newBodyCondition.getHeight() > 0);
}

private boolean collectBodyFatShare(View view) {
    EditText bodyFatShare = ((FragmentActivity)
view.getContext()).findViewById(R.id.account_editing_percent_edit);
    String strBodyFatShare = bodyFatShare.getText().toString();
```

```
        try {
            newBodyCondition.setBodyFatShare(Float.parseFloat(strBodyFatShare));
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "edittext:  " + e, e);
            }
            return false;
        }

        return (newBodyCondition.getBodyFatShare() > 0);
    }
}
```

1.70. TimeTableExerciseClickLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.util.Log;
import android.view.View;
import android.widget.TextView;

import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.fragments.TrainingExerciseSuccessFragment;

public class TimeTableExerciseClickLogic implements View.OnClickListener {
    private final TrainingExerciseInstance trainingExerciseInstance;

    public TimeTableExerciseClickLogic(TrainingExerciseInstance trainingExerciseInstance) {
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    public void onClick(View v) {
        Fragment fragment = new TrainingExerciseSuccessFragment(trainingExerciseInstance);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.71. TrainingExerciseSetClickLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.fragments.TrainingExerciseSetsFragment;
import com.example.testbottomnavigationbar.fragments.TrainingExerciseSuccessFragment;

public class TrainingExerciseSetClickLogic implements View.OnClickListener {
    private final int type;
    private final int accountId;
    private final TrainingExerciseInstance trainingExerciseInstance;

    public TrainingExerciseSetClickLogic(int type, int accountId, TrainingExerciseInstance
trainingExerciseInstance) {
        this.type = type;
        this.accountId = accountId;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    public void onClick(View v) {
        Fragment fragment = new TrainingExerciseSetsFragment(type, accountId, trainingExerciseInstance);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.72. TrainingExerciseSuccessCreateSetLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;
```

```
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.fragments.AddSetFragment;

public class TrainingExerciseSuccessCreateSetLogic implements View.OnClickListener {
    private final int trainingId;
    private final int dayOfWeek;
    private final int orderNumber;
    private final int exerciseId;
    private final int setNumber;
    private final int type;

    public TrainingExerciseSuccessCreateSetLogic(int trainingId, int dayOfWeek, int orderNumber, int exerciseId, int setNumber, int type) {
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
        this.orderNumber = orderNumber;
        this.exerciseId = exerciseId;
        this.setNumber = setNumber;
        this.type = type;
    }


    @Override
    public void onClick(View v) {
        Fragment fragment = new AddSetFragment(trainingId, dayOfWeek, orderNumber, exerciseId, setNumber, type);
        FragmentTransaction fTrans = ((FragmentActivity) v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);
    }
}
```

1.73. TrainingLongClickLogic.java

```
package com.example.testbottomnavigationbar.listeners;

import android.view.View;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.entities.TrainingEditingHelper;
import com.example.testbottomnavigationbar.fragments.TrainingEditingFragment;
```

```
public class TrainingLongClickLogic implements View.OnLongClickListener {
    private final TrainingEditingHelper trainingEditingHelper;

    public TrainingLongClickLogic(TrainingEditingHelper trainingEditingHelper) {
        this.trainingEditingHelper = trainingEditingHelper;
    }

    @Override
    public boolean onLongClick(View v) {
        Fragment fragment = new TrainingEditingFragment(trainingEditingHelper);
        FragmentTransaction fTrans = ((FragmentActivity)
v.getContext()).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.fragment_cv, fragment).commit();
        fTrans.addToBackStack(null);

        return true;
    }
}
```

1.74. DeleteAccountTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.io.IOException;
import java.net.MalformedURLException;

public class DeleteAccountTask extends AsyncTask<Void, Void, Integer> {
    private final int accountId;

    public DeleteAccountTask(int accountId) {
        this.accountId = accountId;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
```

```
                Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
            }
        }
        try {
            worker.deleteAccount(accountId);
        } catch (IOException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with removing Friendship:   " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error remove in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.75. DeleteFriendshipTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.Friendship;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.io.IOException;
import java.net.MalformedURLException;

public class DeleteFriendshipTask extends AsyncTask<Void, Void, Integer> {
  private final Friendship friendship;

  public DeleteFriendshipTask(Friendship friendship) {
    this.friendship = friendship;
  }

  @Override
  protected Integer doInBackground(Void... voids) {
    try {
      HttpWork worker = null;
      try {
```

```
            worker = new HttpWork();
        } catch (MalformedURLException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
            }
        }
        try {
            worker.deleteFriendship(friendship);
        } catch (IOException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with removing Friendship:   " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error remove in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.76. DeleteMoveTrainingExeciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class DeleteMoveTrainingExeciseTask extends AsyncTask<Void, Void, Integer> {
   private final int type;
   private final int accountId;
   private final int maxOrderNumber;
   private final TrainingExerciseInstance trainingExerciseInstance;

   public DeleteMoveTrainingExeciseTask(int type, int accountId, int maxOrderNumber,
TrainingExerciseInstance trainingExerciseInstance) {
       this.type = type;
       this.accountId = accountId;
```

```
        this.maxOrderNumber = maxOrderNumber;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }

            try {
                if (type == 0) {
                    worker.deleteTrainingExerciseSuccessWithOrderNumber(accountId,
trainingExerciseInstance);
                    worker.deleteTrainingExerciseNoteWithOrderNumber(accountId, trainingExerciseInstance);

                    for (int i = trainingExerciseInstance.getOrderNumber() + 1; i <= maxOrderNumber; ++i) {
                        worker.updateTrainingExerciseSuccessWithNewOrderNumber(accountId, i - 1,
                                new TrainingExerciseInstance(null,
                                        trainingExerciseInstance.getTrainingId(),
                                        trainingExerciseInstance.getDayOfWeek(), i));

                        worker.updateTrainingExerciseNoteWithNewOrderNumber(accountId, i - 1,
                                new TrainingExerciseInstance(null,
                                        trainingExerciseInstance.getTrainingId(),
                                        trainingExerciseInstance.getDayOfWeek(), i));
                    }
                } else {
                    worker.deleteTrainingExerciseWithOrderNumber(accountId, trainingExerciseInstance);

                    for (int i = trainingExerciseInstance.getOrderNumber() + 1; i <= maxOrderNumber; ++i) {
                        worker.updateTrainingExerciseWithNewOrderNumber(accountId, i - 1,
                                new TrainingExerciseInstance(null,
                                        trainingExerciseInstance.getTrainingId(),
                                        trainingExerciseInstance.getDayOfWeek(), i));
                    }
                }
            } catch (Exception e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with remove exercise:   " + e, e);
                }
```

```
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error remove in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.77. DeleteTrainingExerciseSuccessTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class DeleteTrainingExerciseSuccessTask extends AsyncTask<Void, Void, Integer> {
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public DeleteTrainingExerciseSuccessTask(ExerciseInTrainingEditing exerciseInTrainingEditing) {
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.deleteTrainingExerciseSuccess(exerciseInTrainingEditing);
            } catch (Exception e) {
                if (MainActivity.LOG) {
```

```
                Log.d(MainActivity.TEG, "Problem with removing TrainingExerciseSuccess:   " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error removing in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.78. DeleteTrainingExerciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class DeleteTrainingExerciseTask extends AsyncTask<Void, Void, Integer> {
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public DeleteTrainingExerciseTask(ExerciseInTrainingEditing exerciseInTrainingEditing) {
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.deleteTrainingExercise(exerciseInTrainingEditing);
```

```
            } catch (Exception e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with removing TrainingExerciseSuccess:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error removing in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.79. DeleteTrainingTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class DeleteTrainingTask extends AsyncTask<Void, Void, Integer> {
    private final int trainingId;
    private final boolean needSetNullCurrentTraining;
    private final int accountId;

    public DeleteTrainingTask(int trainingId, boolean needSetNullCurrentTraining, int accountId) {
        this.trainingId = trainingId;
        this.needSetNullCurrentTraining = needSetNullCurrentTraining;
        this.accountId = accountId;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
```

```
                Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
            }
        }
        try {
            if (needSetNullCurrentTraining) {
                worker.setNullCurrentTraining(accountId);
            }

            worker.deleteTrainingFromId(trainingId);
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with removing Training:   " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error removing in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.80. InsertFriendshipTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.Friendship;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.io.IOException;
import java.net.MalformedURLException;

public class InsertFriendshipTask extends AsyncTask<Void, Void, Integer> {
  private final Friendship friendship;

  public InsertFriendshipTask(Friendship friendship) {
    this.friendship = friendship;
  }

  @Override
```

```
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.insertFriendship(friendship);
            } catch (IOException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with inserting Friendship:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error inserting in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.81. InsertNewExerciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.remote_db.ExerciseToTag;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TargetMuscle;

import java.io.IOException;
import java.net.MalformedURLException;
import java.util.List;

public class InsertNewExerciseTask extends AsyncTask<Void, Void, Integer> {
    Exercise exercise;
```

```
    List<ExerciseToTag> exerciseToTagList;
    List<TargetMuscle> targetMuscleList;

    public InsertNewExerciseTask(Exercise exercise, List<ExerciseToTag> exerciseToTagList,
List<TargetMuscle> targetMuscleList) {
        this.exercise = exercise;
        this.exerciseToTagList = exerciseToTagList;
        this.targetMuscleList = targetMuscleList;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.insertExercise(exercise);

                for (ExerciseToTag exerciseToTag : exerciseToTagList) {
                    worker.insertExerciseToTag(exerciseToTag);
                }

                for (TargetMuscle targetMuscle : targetMuscleList) {
                    worker.insertTargetMuscle(targetMuscle);
                }
            } catch (IOException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with inserting:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error inserting in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.82. InsertTrainingExerciseSuccessTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.content.Context;
import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;

import java.io.IOException;
import java.net.MalformedURLException;

public class InsertTrainingExerciseSuccessTask extends AsyncTask<Void, Void, Integer> {
    private final TrainingExerciseSuccess trainingExerciseSuccess;

    public InsertTrainingExerciseSuccessTask(TrainingExerciseSuccess trainingExerciseSuccess) {
        this.trainingExerciseSuccess = trainingExerciseSuccess;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.insertTrainingExerciseSuccess(trainingExerciseSuccess);
            } catch (IOException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with inserting TrainingExerciseSuccess:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error inserting in remote db: " + e, e);
            }
            return 1;
```

```
        }
    }
}
```

1.83. InsertTrainingExerciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;

import java.io.IOException;
import java.net.MalformedURLException;

public class InsertTrainingExerciseTask extends AsyncTask<Void, Void, Integer> {
    private final TrainingExercise trainingExercise;

    public InsertTrainingExerciseTask(TrainingExercise trainingExercise) {
        this.trainingExercise = trainingExercise;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.insertTrainingExercise(trainingExercise);
            } catch (IOException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with inserting TrainingExercise:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
```

```
            Log.d(MainActivity.TEG, "Error inserting in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.84. InsertTrainingTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;
import android.widget.ProgressBar;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;

import java.io.IOException;
import java.net.MalformedURLException;
import java.util.List;

public class InsertTrainingTask extends AsyncTask<Void, Void, Integer> {
  private final Training training;
  private final int type;
  private final List<TrainingExercise> trainingExerciseList;

  public InsertTrainingTask(Training training, int type, List<TrainingExercise> trainingExerciseList) {
    this.training = training;
    this.type = type;
    this.trainingExerciseList = trainingExerciseList;
  }

  @Override
  protected Integer doInBackground(Void... voids) {
    try {
      HttpWork worker = null;
      try {
        worker = new HttpWork();
      } catch (MalformedURLException e) {
        if (MainActivity.LOG) {
          Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
        }
      }
      try {
```

```
            worker.insertTraining(training);
        } catch (IOException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with inserting TrainingExercise:   " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error inserting in remote db: " + e, e);
        }
        return 1;
    }
}

@Override
protected void onPostExecute(Integer integer) {
    if (type == 1) {
        for (TrainingExercise trainingExercise : trainingExerciseList) {
            new InsertTrainingExerciseTask(trainingExercise).execute();
        }
    }
}
}
```

1.85. MoveTrainingExerciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class MoveTrainingExerciseTask extends AsyncTask<Void, Void, Integer> {
    private final int type;
    private final int accountId;
    private final int orderNumber;
    private final TrainingExerciseInstance trainingExerciseInstance;

    public MoveTrainingExerciseTask(int type, int accountId, int orderNumber, TrainingExerciseInstance
trainingExerciseInstance) {
```

```
        this.type = type;
        this.accountId = accountId;
        this.orderNumber = orderNumber;
        this.trainingExerciseInstance = trainingExerciseInstance;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }

            try {
                if (type == 0) {
                    worker.updateTrainingExerciseSuccessWithNewOrderNumber(accountId,                100,
trainingExerciseInstance);
                    worker.updateTrainingExerciseNoteWithNewOrderNumber(accountId,                100,
trainingExerciseInstance);

                    if (orderNumber > trainingExerciseInstance.getOrderNumber()) {
                        for (int i = trainingExerciseInstance.getOrderNumber() + 1; i <= orderNumber; ++i) {
                            worker.updateTrainingExerciseSuccessWithNewOrderNumber(accountId, i - 1,
                                    new TrainingExerciseInstance(null,
                                            trainingExerciseInstance.getTrainingId(),
                                            trainingExerciseInstance.getDayOfWeek(), i));

                            worker.updateTrainingExerciseNoteWithNewOrderNumber(accountId, i - 1,
                                    new TrainingExerciseInstance(null,
                                            trainingExerciseInstance.getTrainingId(),
                                            trainingExerciseInstance.getDayOfWeek(), i));
                        }
                    } else {
                        for (int i = trainingExerciseInstance.getOrderNumber() - 1; i >= orderNumber; --i) {
                            worker.updateTrainingExerciseSuccessWithNewOrderNumber(accountId, i + 1,
                                    new TrainingExerciseInstance(null,
                                            trainingExerciseInstance.getTrainingId(),
                                            trainingExerciseInstance.getDayOfWeek(), i));

                            worker.updateTrainingExerciseNoteWithNewOrderNumber(accountId, i + 1,
                                    new TrainingExerciseInstance(null,
                                            trainingExerciseInstance.getTrainingId(),
```

```
                        trainingExerciseInstance.getDayOfWeek(), i));
                }
            }

            worker.updateTrainingExerciseSuccessWithNewOrderNumber(accountId, orderNumber,
                new TrainingExerciseInstance(trainingExerciseInstance.getExerciseTitle(),
                    trainingExerciseInstance.getTrainingId(),
trainingExerciseInstance.getDayOfWeek(), 100));
            worker.updateTrainingExerciseNoteWithNewOrderNumber(accountId, orderNumber,
                new TrainingExerciseInstance(trainingExerciseInstance.getExerciseTitle(),
                    trainingExerciseInstance.getTrainingId(),
trainingExerciseInstance.getDayOfWeek(), 100));
        } else {
            worker.updateTrainingExerciseWithNewOrderNumber(accountId,                100,
trainingExerciseInstance);

            if (orderNumber > trainingExerciseInstance.getOrderNumber()) {
                for (int i = trainingExerciseInstance.getOrderNumber() + 1; i <= orderNumber; ++i) {
                    worker.updateTrainingExerciseWithNewOrderNumber(accountId, i - 1,
                        new TrainingExerciseInstance(null,
                            trainingExerciseInstance.getTrainingId(),
                            trainingExerciseInstance.getDayOfWeek(), i));
                }
            } else {
                for (int i = trainingExerciseInstance.getOrderNumber() - 1; i >= orderNumber; --i) {
                    worker.updateTrainingExerciseWithNewOrderNumber(accountId, i + 1,
                        new TrainingExerciseInstance(null,
                            trainingExerciseInstance.getTrainingId(),
                            trainingExerciseInstance.getDayOfWeek(), i));
                }
            }

            worker.updateTrainingExerciseWithNewOrderNumber(accountId, orderNumber,
                new TrainingExerciseInstance(trainingExerciseInstance.getExerciseTitle(),
                    trainingExerciseInstance.getTrainingId(),
trainingExerciseInstance.getDayOfWeek(), 100));
        }
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Problem with move exercise:   " + e, e);
        }
    }

    return 0;
} catch (Exception e) {
    if (MainActivity.LOG) {
        Log.d(MainActivity.TEG, "Error moving in remote db: " + e, e);
```

```
            }
            return 1;
        }
    }
}
```

1.86. PullExerciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.app.Activity;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.ProgressBar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.SearchFragment;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

public class PullExerciseTask extends AsyncTask<Void, Void, Integer> {
    private final Context context;
    private final ProgressBar progressBar;

    public PullExerciseTask(Context context, ProgressBar progressBar) {
        this.context = context;
        this.progressBar = progressBar;
    }

    @Override
    protected void onPreExecute() {
        progressBar.setVisibility(ProgressBar.VISIBLE);
        progressBar.setIndeterminate(true);
    }

    @Override
    protected Integer doInBackground(Void... ignore) {
        try {
```

```
        HttpWork worker = new HttpWork();
        Exercise[] exercises = worker.pullAllExercises();

        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.dbName,
Context.MODE_PRIVATE, null);
        for (Exercise exercise : exercises) {
            db.execSQL(exercise.getSQLiteInsertQuery());

            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, exercise.getSQLiteInsertQuery());
            }
        }

        return 0;
      } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error pulling exercises: " + e, e);
        }
        return 1;
      }
    }

    @Override
    protected void onPostExecute(Integer integer) {
      progressBar.setIndeterminate(false);
      progressBar.setVisibility(ProgressBar.GONE);

      Fragment fragment = new SearchFragment(SearchFragment.generateExerciseOverviewList(context),
0, null);
      FragmentTransaction fTrans = ((FragmentActivity)
context).getSupportFragmentManager().beginTransaction();
      fTrans.replace(R.id.fragment_cv, fragment, "searchFragment").commit();
      fTrans.addToBackStack(null);
    }
}
```

1.87. PullRemoteDBTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.AsyncTask;
import android.util.Log;
import android.view.View;
import android.widget.ProgressBar;
```

```
import android.widget.TextView;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentTransaction;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.fragments.LoginFragment;
import com.example.testbottomnavigationbar.fragments.MainFragment;
import com.example.testbottomnavigationbar.fragments.ReconnectFragment;
import com.example.testbottomnavigationbar.fragments.SearchFragment;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

public class PullRemoteDBTask extends AsyncTask<Void, Void, Integer> {
    private final Context context;
    private final ProgressBar progressBar;

    public PullRemoteDBTask(Context context, ProgressBar progressBar) {
        this.context = context;
        this.progressBar = progressBar;
    }

    @Override
    protected void onPreExecute() {
        progressBar.setVisibility(ProgressBar.VISIBLE);
        progressBar.setIndeterminate(true);
    }

    @Override
    protected Integer doInBackground(Void... ignore) {
        try {
            HttpWork worker = new HttpWork();
            SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.dbName,
Context.MODE_PRIVATE, null);

            SQLiteHelper.pullAccount(worker, db);
            SQLiteHelper.pullExercise(worker, db);
            SQLiteHelper.pullExerciseTag(worker, db);
            SQLiteHelper.pullExerciseToTag(worker, db);
            SQLiteHelper.pullTraining(worker, db);
            SQLiteHelper.pullTrainingExercise(worker, db);
            SQLiteHelper.pullTrainingExerciseNote(worker, db);
            SQLiteHelper.pullTrainingExerciseSuccess(worker, db);
            SQLiteHelper.pullTrainingType(worker, db);
```

```
            SQLiteHelper.pullExerciseTrainingType(worker, db);
            SQLiteHelper.pullMuscle(worker, db);
            SQLiteHelper.pullTargetMuscle(worker, db);
            SQLiteHelper.pullBodyCondition(worker, db);
            SQLiteHelper.pullFriendship(worker, db);

//            db.execSQL("INSERT INTO CurrentAccount(AccountId)\n" +
//                    "VALUES ('1')");

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error pulling remote db: " + e, e);
            }
            return -1;
        }
    }

    @Override
    protected void onPostExecute(Integer integer) {
        progressBar.setIndeterminate(false);
        progressBar.setVisibility(ProgressBar.GONE);

        if (integer == 0) {
            if (!hasCurrentAccount(context)) {
                setLoginFragment(context);
            } else {
                setMainFragment(context);
            }
        } else if (integer == -1) {
            Fragment fragment = new ReconnectFragment();
            FragmentTransaction fTrans = ((FragmentActivity)
context).getSupportFragmentManager().beginTransaction();
            fTrans.replace(R.id.activity_main_fragment_cv, fragment, "reconnectFragment").commit();
            fTrans.addToBackStack(null);
        }
    }

    private boolean hasCurrentAccount(Context context) {
        SQLiteDatabase currentAccountDB =
context.openOrCreateDatabase(SQLiteHelper.currentAccountDBName, Context.MODE_PRIVATE,
null);
        SQLiteDatabase db = context.openOrCreateDatabase(SQLiteHelper.dbName,
Context.MODE_PRIVATE, null);

        Cursor cursor = currentAccountDB.rawQuery("SELECT * FROM CurrentAccount;", null);
        if (cursor != null && cursor.moveToFirst()) {
```

```
            int accountId = cursor.getInt(0);

            Cursor cursor1 = db.rawQuery("SELECT * FROM Account WHERE AccountId = " + accountId +
";", null);
            if (cursor1 != null && cursor1.moveToFirst()) {
                return true;
            }
        }

        return false;
    }

    private void setLoginFragment(Context context) {
        Fragment fragment = new LoginFragment();
        FragmentTransaction fTrans = ((FragmentActivity)
context).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.activity_main_fragment_cv, fragment, "loginFragment").commit();
        fTrans.addToBackStack(null);
    }

    private void setMainFragment(Context context) {
        Fragment fragment = new MainFragment();
        FragmentTransaction fTrans = ((FragmentActivity)
context).getSupportFragmentManager().beginTransaction();
        fTrans.replace(R.id.activity_main_fragment_cv, fragment, "mainFragment").commit();
        fTrans.addToBackStack(null);
    }
}
```

1.88. SignUpTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;
import com.example.testbottomnavigationbar.remote_db.ExerciseToTag;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TargetMuscle;

import java.io.IOException;
import java.net.MalformedURLException;

public class SignUpTask extends AsyncTask<Void, Void, Integer> {
```

```
    private final Account newAccount;
    private final BodyCondition newBodyCondition;

    public SignUpTask(Account newAccount, BodyCondition newBodyCondition) {
        this.newAccount = newAccount;
        this.newBodyCondition = newBodyCondition;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.insertAccount(newAccount);
                worker.insertBodyCondition(newBodyCondition);
            } catch (IOException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with sign up:    " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error sign up in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.89. UpdateAccountInfoTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
```

```
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class UpdateAccountInfoTask extends AsyncTask<Void, Void, Integer> {
    private final Account account;
    private final BodyCondition bodyCondition;
    private final Account newAccount;
    private final BodyCondition newBodyCondition;

    public UpdateAccountInfoTask(Account account, BodyCondition bodyCondition, Account
newAccount, BodyCondition newBodyCondition) {
        this.account = account;
        this.bodyCondition = bodyCondition;
        this.newAccount = newAccount;
        this.newBodyCondition = newBodyCondition;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }

            try {
                worker.updateAccountInfo(account.getAccountId(), newAccount);
                worker.updateBodyConditionInfo(account.getAccountId(), newBodyCondition);
            } catch (Exception e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with update account info:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error updating in remote db: " + e, e);
            }
            return 1;
```

```
        }
    }
}
```

1.90. UpdateCurrentTrainingTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseNote;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;

import java.net.MalformedURLException;
import java.util.List;

public class UpdateCurrentTrainingTask extends AsyncTask<Void, Void, Integer> {
    private final int accountId;
    private final int trainingId;
    private final List<TrainingExerciseSuccess> trainingExerciseSuccessList;
    private final List<TrainingExerciseNote> trainingExerciseNoteList;

    public UpdateCurrentTrainingTask(int accountId, int trainingId, List<TrainingExerciseSuccess>
trainingExerciseSuccessList, List<TrainingExerciseNote> trainingExerciseNoteList) {
        this.accountId = accountId;
        this.trainingId = trainingId;
        this.trainingExerciseSuccessList = trainingExerciseSuccessList;
        this.trainingExerciseNoteList = trainingExerciseNoteList;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.updateCurrentTraining(accountId, trainingId);
```

```
            worker.deleteTrainingExerciseSuccessFromTraining(accountId, trainingId);
            worker.deleteTrainingExerciseNote(accountId, trainingId);

            for (TrainingExerciseSuccess trainingExerciseSuccess : trainingExerciseSuccessList) {
                worker.insertTrainingExerciseSuccess(trainingExerciseSuccess);
            }

            for (TrainingExerciseNote trainingExerciseNote : trainingExerciseNoteList) {
                worker.insertTrainingExerciseNote(trainingExerciseNote);
            }
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with updating current training:   " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error updating in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.91. UpdateNoteTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class UpdateNoteTask extends AsyncTask<Void, Void, Integer> {
    private final int accountId;
    private final int trainingId;
    private final int dayOfWeek;
    private final int orderNumber;
    private final String note;

    public UpdateNoteTask(int accountId, int trainingId, int dayOfWeek, int orderNumber, String note) {
        this.accountId = accountId;
```

```
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
        this.orderNumber = orderNumber;
        this.note = note;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.updateNote(accountId, trainingId, dayOfWeek, orderNumber, note);
            } catch (Exception e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with updating note:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error updating in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.92. UpdateTrainingExerciseSuccessTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.InsertSetJSON;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;
```

```
import java.io.IOException;
import java.net.MalformedURLException;

public class UpdateTrainingExerciseSuccessTask extends AsyncTask<Void, Void, Integer> {
    private final InsertSetJSON insertSetJSON;
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public UpdateTrainingExerciseSuccessTask(InsertSetJSON insertSetJSON, ExerciseInTrainingEditing
exerciseInTrainingEditing) {
        this.insertSetJSON = insertSetJSON;
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }


    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.updateTrainingExerciseSuccess(insertSetJSON, exerciseInTrainingEditing);
            } catch (Exception e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with updating TrainingExerciseSuccess:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error updating in remote db: " + e, e);
            }
            return 1;
        }
    }
}
```

1.93. UpdateTrainingExerciseTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;
```

```
import android.os.AsyncTask;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.InsertSetJSON;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class UpdateTrainingExerciseTask extends AsyncTask<Void, Void, Integer> {
    private final InsertSetJSON insertSetJSON;
    private final ExerciseInTrainingEditing exerciseInTrainingEditing;

    public UpdateTrainingExerciseTask(InsertSetJSON insertSetJSON, ExerciseInTrainingEditing
exerciseInTrainingEditing) {
        this.insertSetJSON = insertSetJSON;
        this.exerciseInTrainingEditing = exerciseInTrainingEditing;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.updateTrainingExercise(insertSetJSON, exerciseInTrainingEditing);
            } catch (Exception e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with updating TrainingExercise:   " + e, e);
                }
            }

            return 0;
        } catch (Exception e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Error updating in remote db: " + e, e);
            }
            return 1;
        }
```

```
    }
}
```

1.94. UpdateTrainingTask.java

```
package com.example.testbottomnavigationbar.remote_db.tasks;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.AsyncTask;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

import androidx.fragment.app.FragmentActivity;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.R;
import com.example.testbottomnavigationbar.SQLiteHelper;
import com.example.testbottomnavigationbar.remote_db.HttpWork;

import java.net.MalformedURLException;

public class UpdateTrainingTask extends AsyncTask<Void, Void, Integer> {
    private final String updateTrainingJSON;
    private final int trainingId;

    public UpdateTrainingTask(String updateTrainingJSON, int trainingId) {
        this.updateTrainingJSON = updateTrainingJSON;
        this.trainingId = trainingId;
    }

    @Override
    protected Integer doInBackground(Void... voids) {
        try {
            HttpWork worker = null;
            try {
                worker = new HttpWork();
            } catch (MalformedURLException e) {
                if (MainActivity.LOG) {
                    Log.d(MainActivity.TEG, "Problem with creating httpwork   " + e, e);
                }
            }
            try {
                worker.updateTraining(updateTrainingJSON, trainingId);
            } catch (Exception e) {
                if (MainActivity.LOG) {
```

```
                Log.d(MainActivity.TEG, "Problem with updating Training:    " + e, e);
            }
        }

        return 0;
    } catch (Exception e) {
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Error updating in remote db: " + e, e);
        }
        return 1;
    }
  }
}
```

1.95. Account.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class Account {
    @SerializedName("accountid")
    private int accountId;
    @SerializedName("username")
    private String userName;
    @SerializedName("hashpassword")
    private String hashPassword;
    @SerializedName("birthdate")
    private String birthDate;
    @SerializedName("registrationdate")
    private String registrationDate;
    private String sex;
    @SerializedName("trainingid")
    private Integer trainingId;
    @SerializedName("firstname")
    private String firstName;
    @SerializedName("secondname")
    private String secondName;

    public Account() {

    }

    public Account(int accountId, String userName, String hashPassword, String birthDate, String
registrationDate, String sex, Integer trainingId, String firstName, String secondName) {
        this.accountId = accountId;
        this.userName = userName;
```

```
        this.hashPassword = hashPassword;
        this.birthDate = birthDate;
        this.registrationDate = registrationDate;
        this.sex = sex;
        this.trainingId = trainingId;
        this.firstName = firstName;
        this.secondName = secondName;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO Account(AccountId, Username, HashPassword, BirthDate, RegistrationDate,
Sex, TrainingId, FirstName, SecondName)\n" +
                "VALUES (" +
                "'" +
                accountId +
                "', " +
                "'" +
                userName +
                "', " +
                "'" +
                hashPassword +
                "', " +
                "'" +
                birthDate +
                "', " +
                "'" +
                registrationDate +
                "', " +
                "'" +
                sex +
                "', " +
                "'" +
                trainingId +
                "', " +
                "'" +
                firstName +
                "', " +
                "'" +
                secondName +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"accountid\" : " + accountId + ", " +
                "\"username\" : \"" + userName + "\", " +
                "\"hashpassword\" : \"" + hashPassword + "\", " +
```

```
            "\"birthdate\" : \"" + birthDate + "\", " +
            "\"registrationdate\" : \"" + registrationDate + "\", " +
            "\"sex\" : \"" + sex + "\", " +
            "\"trainingid\" : " + trainingId + ", " +
            "\"firstname\" : \"" + firstName + "\", " +
            "\"secondname\" : \"" + secondName + "\" " +
            "}";
    }

    public int getAccountId() {
        return accountId;
    }

    public String getUserName() {
        return userName;
    }

    public String getHashPassword() {
        return hashPassword;
    }

    public String getBirthDate() {
        return birthDate;
    }

    public String getRegistrationDate() {
        return registrationDate;
    }

    public String getSex() {
        return sex;
    }

    public Integer getTrainingId() {
        return trainingId;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getSecondName() {
        return secondName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
```

```
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public void setSecondName(String secondName) {
        this.secondName = secondName;
    }

    public void setHashPassword(String hashPassword) {
        this.hashPassword = hashPassword;
    }

    public void setAccountId(int accountId) {
        this.accountId = accountId;
    }

    public void setBirthDate(String birthDate) {
        this.birthDate = birthDate;
    }

    public void setRegistrationDate(String registrationDate) {
        this.registrationDate = registrationDate;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }

    public void setTrainingId(Integer trainingId) {
        this.trainingId = trainingId;
    }
}
```

1.96. BodyCondition.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class BodyCondition {
    @SerializedName("accountid")
    private int accountId;
    private float weight;
    private float height;
    private int age;
```

```
@SerializedName("bodyfatshare")
private float bodyFatShare;

public BodyCondition() {

}

public BodyCondition(int accountId, float weight, float height, int age, float bodyFatShare) {
   this.accountId = accountId;
   this.weight = weight;
   this.height = height;
   this.age = age;
   this.bodyFatShare = bodyFatShare;
}

public String getSQLiteInsertQuery() {
   return "INSERT INTO BodyCondition(AccountId, Weight, Height, Age, BodyFatShare)\n" +
         "VALUES (" +
         "'" +
         accountId +
         "', " +
         "'" +
         weight +
         "', " +
         "'" +
         height +
         "', " +
         "'" +
         age +
         "', " +
         "'" +
         bodyFatShare +
         "');";
}

public String getJSON() {
   return "{" +
         "\"accountid\" : " + accountId + ", " +
         "\"weight\" : " + weight + ", " +
         "\"height\" : " + height + ", " +
         "\"age\" : " + age + ", " +
         "\"bodyfatshare\" : " + bodyFatShare + " " +
         "}";
}

public int getAccountId() {
   return accountId;
```

```
    }

    public float getWeight() {
        return weight;
    }

    public float getHeight() {
        return height;
    }

    public int getAge() {
        return age;
    }

    public float getBodyFatShare() {
        return bodyFatShare;
    }

    public void setWeight(float weight) {
        this.weight = weight;
    }

    public void setHeight(float height) {
        this.height = height;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void setBodyFatShare(float bodyFatShare) {
        this.bodyFatShare = bodyFatShare;
    }

    public void setAccountId(int accountId) {
        this.accountId = accountId;
    }
}
```

1.97. Exercise.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class Exercise {
    @SerializedName("exerciseid")
```

```
private int exerciseId;
private String title;
private String description;
@SerializedName("videopath")
private String videoPath;

public Exercise() {

}

public Exercise(int exerciseId, String title, String description, String videoPath) {
   this.exerciseId = exerciseId;
   this.title = title;
   this.description = description;
   this.videoPath = videoPath;
}

public String getSQLiteInsertQuery() {
   return "INSERT INTO Exercise(ExerciseId, Title, Description, VideoPath)\n" +
         "VALUES (" +
         "'" +
         exerciseId +
         "', " +
         "'" +
         title +
         "', " +
         "'" +
         description +
         "', " +
         "'" +
         videoPath +
         "');";
}

public String getJSON() {
   return "{" +
         "\"exerciseid\" : " + exerciseId + ", " +
         "\"title\" : \"" + title + "\", " +
         "\"description\" : \"" + description + "\", " +
         "\"videopath\" : \"" + videoPath + "\" " +
         "}";
}

public int getExerciseId() {
   return exerciseId;
}
```

```
    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public String getVideoPath() {
        return videoPath;
    }
}
```

1.98. ExerciseTag.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class ExerciseTag {
    @SerializedName("tagid")
    private final int tagId;
    private final String title;

    public ExerciseTag(int tagId, String title) {
        this.tagId = tagId;
        this.title = title;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO ExerciseTag(TagId, Title)\n" +
                "VALUES (" +
                "'" +
                tagId +
                "', " +
                "'" +
                title +
                "');";
    }

    public int getTagId() {
        return tagId;
    }

    public String getTitle() {
        return title;
    }
```

```
}
```

1.99. ExerciseToTag.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class ExerciseToTag {
    @SerializedName("exerciseid")
    private final int exerciseId;
    @SerializedName("tagid")
    private final int tagId;

    public ExerciseToTag(int exerciseId, int tagId) {
        this.exerciseId = exerciseId;
        this.tagId = tagId;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
                "VALUES (" +
                "'" +
                exerciseId +
                "', " +
                "'" +
                tagId +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"exerciseid\" : " + exerciseId + ", " +
                "\"tagid\" : " + tagId + " " +
                "}";
    }

    public int getExerciseId() {
        return exerciseId;
    }

    public int getTagId() {
        return tagId;
    }
}
```

1.100. ExerciseTrainingType.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class ExerciseTrainingType {
    @SerializedName("typeid")
    private final int typeId;
    @SerializedName("exerciseid")
    private final int exerciseId;

    public ExerciseTrainingType(int typeId, int exerciseId) {
        this.typeId = typeId;
        this.exerciseId = exerciseId;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO ExerciseTrainingType(TypeId, ExerciseId)\n" +
                "VALUES (" +
                "'" +
                typeId +
                "', " +
                "'" +
                exerciseId +
                "');";
    }

    public int getTypeId() {
        return typeId;
    }

    public int getExerciseId() {
        return exerciseId;
    }
}
```

1.101. Friendship.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class Friendship {
    @SerializedName("accountid")
    private int accountId;
    @SerializedName("friendid")
```

```
    private int friendId;

    public Friendship(int accountId, int friendId) {
        this.accountId = accountId;
        this.friendId = friendId;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO Friendship(AccountId, FriendId)\n" +
                "VALUES (" +
                """ +
                accountId +
                "', " +
                """ +
                friendId +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"accountid\" : " + accountId + ", " +
                "\"friendid\" : " + friendId + " " +
                "}";
    }

    public int getAccountId() {
        return accountId;
    }

    public int getFriendId() {
        return friendId;
    }
}
```

1.102. HttpWork.java

```
    package com.example.testbottomnavigationbar.remote_db;

import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

import com.example.testbottomnavigationbar.MainActivity;
import com.example.testbottomnavigationbar.entities.ExerciseInTrainingEditing;
import com.example.testbottomnavigationbar.entities.InsertSetJSON;
import com.example.testbottomnavigationbar.entities.TrainingExerciseInstance;
import com.example.testbottomnavigationbar.remote_db.tasks.InsertNewExerciseTask;
import com.google.gson.Gson;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.nio.charset.StandardCharsets;

public class HttpWork {
    private final String local_IP = "192.168.0.104";
    private final URL url_get_all_exercises = new URL("http://" + local_IP +
":3000/rpc/get_all_exercises");
    private final URL url_get_all_exercises_tags = new URL("http://" + local_IP +
":3000/rpc/get_all_exercises_tags");
    private final URL url_get_all_exercises_to_tags = new URL("http://" + local_IP +
":3000/rpc/get_all_exercises_to_tags");
    private final URL url_get_all_trainings = new URL("http://" + local_IP + ":3000/rpc/get_all_trainings");
    private final URL url_get_all_accounts = new URL("http://" + local_IP + ":3000/rpc/get_all_accounts");
    private final URL url_get_all_trainings_exercises = new URL("http://" + local_IP +
":3000/rpc/get_all_trainings_exercises");
    private final URL url_get_all_trainings_exercises_success = new URL("http://" + local_IP +
":3000/rpc/get_all_trainings_exercises_success");
    private final URL url_get_all_trainings_exercises_notes = new URL("http://" + local_IP +
":3000/rpc/get_all_trainings_exercises_notes");
    private final URL url_get_all_training_types = new URL("http://" + local_IP +
":3000/rpc/get_all_training_types");
    private final URL url_get_all_exercise_training_types = new URL("http://" + local_IP +
":3000/rpc/get_all_exercise_training_types");
    private final URL url_get_all_muscles = new URL("http://" + local_IP + ":3000/rpc/get_all_muscles");
    private final URL url_get_all_target_muscles = new URL("http://" + local_IP +
":3000/rpc/get_all_target_muscles");
    private final URL url_get_all_friendships = new URL("http://" + local_IP +
":3000/rpc/get_all_friendships");
    private final URL url_insert_training_exercise_success = new URL("http://" + local_IP +
":3000/trainingexercisesuccess");
    private final URL url_insert_exercise = new URL("http://" + local_IP + ":3000/exercise");
    private final URL url_insert_exercise_to_tag = new URL("http://" + local_IP + ":3000/exercisetotag");
    private final URL url_insert_target_muscle = new URL("http://" + local_IP + ":3000/targetmuscle");
    private final URL url_insert_training_exercise = new URL("http://" + local_IP +
":3000/trainingexercise");
    private final URL url_insert_training = new URL("http://" + local_IP + ":3000/training");
    private final URL url_insert_note = new URL("http://" + local_IP + ":3000/trainingexercisenote");
    private final URL url_insert_account = new URL("http://" + local_IP + ":3000/account");
```

```
    private final URL url_insert_body_condition = new URL("http://" + local_IP + ":3000/bodycondition");
    private final URL url_insert_friendship = new URL("http://" + local_IP + ":3000/friendship");
    private final URL url_get_all_body_conditions = new URL("http://" + local_IP +
":3000/rpc/get_all_body_conditions");

    public Exercise[] pullAllExercises() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_exercises.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Exercises   " + response);

            Exercise[] q = gson.fromJson(response, Exercise[].class);
            Log.d(MainActivity.TEG, "videoPath   " + String.valueOf(q[0].getVideoPath()));
            Log.d(MainActivity.TEG, "videoPath   " + String.valueOf(q[1].getVideoPath()));
        }

        return gson.fromJson(response, Exercise[].class);
    }

    public ExerciseTag[] pullAllExerciseTags() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_exercises_tags.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, ExerciseTag[].class);
    }

    public ExerciseToTag[] pullAllExerciseToTag() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_exercises_to_tags.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, response);

            ExerciseToTag[] q = gson.fromJson(response, ExerciseToTag[].class);
            Log.d(MainActivity.TEG, q[0].getExerciseId() + " " + q[0].getTagId());
            Log.d(MainActivity.TEG, q[1].getExerciseId() + " " + q[1].getTagId());
        }
```

```
        return gson.fromJson(response, ExerciseToTag[].class);
    }

    public Training[] pullAllTrainings() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_trainings.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, Training[].class);
    }

    public Account[] pullAllAccounts() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_accounts.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Response  " + response);
        }
        Gson gson = new Gson();

        return gson.fromJson(response, Account[].class);
    }

    public TrainingExercise[] pullAllTrainingExercises() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_trainings_exercises.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, TrainingExercise[].class);
    }

    public TrainingExerciseSuccess[] pullAllTrainingExerciseSuccess() throws IOException {
        HttpURLConnection con = (HttpURLConnection)
url_get_all_trainings_exercises_success.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, TrainingExerciseSuccess[].class);
    }
```

```
    public TrainingExerciseNote[] pullAllTrainingExerciseNote() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_trainings_exercises_notes.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, TrainingExerciseNote[].class);
    }

    public TrainingType[] pullAllTrainingTypes() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_training_types.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, TrainingType[].class);
    }

    public ExerciseTrainingType[] pullAllExerciseTrainingTypes() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_exercise_training_types.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, ExerciseTrainingType[].class);
    }

    public Muscle[] pullAllMuscles() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_muscles.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, Muscle[].class);
    }

    public TargetMuscle[] pullAllTargetMuscles() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_target_muscles.openConnection();
        tunePostConnection(con);
```

```
        String response = getResponseString(con);
        Gson gson = new Gson();

        return gson.fromJson(response, TargetMuscle[].class);
    }

    public BodyCondition[] pullAllBodyConditions() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_body_conditions.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "BodyConditions:  " + response);
        }

        return gson.fromJson(response, BodyCondition[].class);
    }

    public Friendship[] pullAllFriendships() throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_get_all_friendships.openConnection();
        tunePostConnection(con);

        String response = getResponseString(con);
        Gson gson = new Gson();

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Friendships:  " + response);
        }

        return gson.fromJson(response, Friendship[].class);
    }

    public void insertTrainingExerciseSuccess(TrainingExerciseSuccess trainingExerciseSuccess) throws
IOException {
        HttpURLConnection con = (HttpURLConnection)
url_insert_training_exercise_success.openConnection();
        tunePostConnection(con);

        String jsonInputString = trainingExerciseSuccess.getJSON();
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }
```

```
    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

public void insertTrainingExercise(TrainingExercise trainingExercise) throws IOException {
    HttpURLConnection con = (HttpURLConnection) url_insert_training_exercise.openConnection();
    tunePostConnection(con);

    String jsonInputString = trainingExercise.getJSON();
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

public void insertTraining(Training training) throws IOException {
    HttpURLConnection con = (HttpURLConnection) url_insert_training.openConnection();
    tunePostConnection(con);

    String jsonInputString = training.getJSON();
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

public void insertExercise(Exercise exercise) throws IOException {
    HttpURLConnection con = (HttpURLConnection) url_insert_exercise.openConnection();
    tunePostConnection(con);

    String jsonInputString = exercise.getJSON();
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
        os.flush();
    }
```

```
        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void insertExerciseToTag(ExerciseToTag exerciseToTag) throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_insert_exercise_to_tag.openConnection();
        tunePostConnection(con);

        String jsonInputString = exerciseToTag.getJSON();
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void insertTargetMuscle(TargetMuscle targetMuscle) throws IOException {
        HttpURLConnection con = (HttpURLConnection) url_insert_target_muscle.openConnection();
        tunePostConnection(con);

        String jsonInputString = targetMuscle.getJSON();
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void insertTrainingExerciseNote(TrainingExerciseNote trainingExerciseNote) throws
IOException {
        HttpURLConnection con = (HttpURLConnection) url_insert_note.openConnection();
        tunePostConnection(con);

        String jsonInputString = trainingExerciseNote.getJSON();
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
```

```
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

public void insertAccount(Account account) throws IOException {
    HttpURLConnection con = (HttpURLConnection) url_insert_account.openConnection();
    tunePostConnection(con);

    String jsonInputString = account.getJSON();
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

public void insertBodyCondition(BodyCondition bodyCondition) throws IOException {
    HttpURLConnection con = (HttpURLConnection) url_insert_body_condition.openConnection();
    tunePostConnection(con);

    String jsonInputString = bodyCondition.getJSON();
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

public void insertFriendship(Friendship friendship) throws IOException {
    HttpURLConnection con = (HttpURLConnection) url_insert_friendship.openConnection();
    tunePostConnection(con);

    String jsonInputString = friendship.getJSON();
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
```

```
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateTrainingExerciseSuccess(InsertSetJSON insertSetJSON, ExerciseInTrainingEditing
exerciseInTrainingEditing) throws IOException {
        URL url_update_training_exercise_success = new URL("http://" + local_IP +
":3000/trainingexercisesuccess?and=(" +
                "accountid.eq." + exerciseInTrainingEditing.getAccountId() + "," +
                "trainingid.eq." + exerciseInTrainingEditing.getTrainingId() + "," +
                "dayofweek.eq." + exerciseInTrainingEditing.getDayOfWeek() + "," +
                "ordernumber.eq." + exerciseInTrainingEditing.getOrderNumber() + "," +
                "setnumber.eq." + exerciseInTrainingEditing.getSetNumber() + ")");

        HttpURLConnection con = (HttpURLConnection)
url_update_training_exercise_success.openConnection();
        tunePatchConnection(con);

        String jsonInputString = insertSetJSON.getJSON();
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateTrainingExercise(InsertSetJSON insertSetJSON, ExerciseInTrainingEditing
exerciseInTrainingEditing) throws IOException {
        URL url_update_training_exercise = new URL("http://" + local_IP + ":3000/trainingexercise?and=("
+
                "accountid.eq." + exerciseInTrainingEditing.getAccountId() + "," +
                "trainingid.eq." + exerciseInTrainingEditing.getTrainingId() + "," +
                "dayofweek.eq." + exerciseInTrainingEditing.getDayOfWeek() + "," +
                "ordernumber.eq." + exerciseInTrainingEditing.getOrderNumber() + "," +
                "setnumber.eq." + exerciseInTrainingEditing.getSetNumber() + ")");

        HttpURLConnection con = (HttpURLConnection) url_update_training_exercise.openConnection();
        tunePatchConnection(con);
```

```
        String jsonInputString = insertSetJSON.getJSON();
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateTrainingExerciseSuccessWithNewOrderNumber(int accountId, int orderNumber,
TrainingExerciseInstance trainingExerciseInstance) throws IOException {
        URL url_update_training_exercise_success_with_order_number = new URL("http://" + local_IP +
":3000/trainingexercisesuccess?and=(" +
                "accountid.eq." + accountId + "," +
                "trainingid.eq." + trainingExerciseInstance.getTrainingId() + "," +
                "dayofweek.eq." + trainingExerciseInstance.getDayOfWeek() + "," +
                "ordernumber.eq." + trainingExerciseInstance.getOrderNumber() + ")");

        HttpURLConnection con = (HttpURLConnection)
url_update_training_exercise_success_with_order_number.openConnection();
        tunePatchConnection(con);

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, url_update_training_exercise_success_with_order_number.toString());
        }

        String jsonInputString = "{ \"ordernumber\" : " + orderNumber + " }";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateTrainingExerciseWithNewOrderNumber(int accountId, int orderNumber,
TrainingExerciseInstance trainingExerciseInstance) throws IOException {
        URL url_update_training_exercise_with_order_number = new URL("http://" + local_IP +
":3000/trainingexercise?and=(" +
                "accountid.eq." + accountId + "," +
```

```
            "trainingid.eq." + trainingExerciseInstance.getTrainingId() + "," +
            "dayofweek.eq." + trainingExerciseInstance.getDayOfWeek() + "," +
            "ordernumber.eq." + trainingExerciseInstance.getOrderNumber() + ")");

        HttpURLConnection con = (HttpURLConnection) url_update_training_exercise_with_order_number.openConnection();
        tunePatchConnection(con);

        String jsonInputString = "{ \"ordernumber\" : " + orderNumber + " }";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateTrainingExerciseNoteWithNewOrderNumber(int accountId, int orderNumber, TrainingExerciseInstance trainingExerciseInstance) throws IOException {
        URL url_update_training_exercise_note_with_order_number = new URL("http://" + local_IP + ":3000/trainingexercisenote?and=(" +
            "accountid.eq." + accountId + "," +
            "trainingid.eq." + trainingExerciseInstance.getTrainingId() + "," +
            "dayofweek.eq." + trainingExerciseInstance.getDayOfWeek() + "," +
            "ordernumber.eq." + trainingExerciseInstance.getOrderNumber() + ")");

        HttpURLConnection con = (HttpURLConnection) url_update_training_exercise_note_with_order_number.openConnection();
        tunePatchConnection(con);

        String jsonInputString = "{ \"ordernumber\" : " + orderNumber + " }";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateTraining(String updateTrainingJSON, int trainingId) throws IOException {
        URL url_update_training = new URL("http://" + local_IP + ":3000/training?" +
```

```
            "trainingid=eq." + trainingId);

        HttpURLConnection con = (HttpURLConnection) url_update_training.openConnection();
        tunePatchConnection(con);

        String jsonInputString = updateTrainingJSON;
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateCurrentTraining(int accountId, int trainingId) throws IOException {
        URL url_update_current_training = new URL("http://" + local_IP + ":3000/account?" +
                "accountid=eq." + accountId);

        HttpURLConnection con = (HttpURLConnection) url_update_current_training.openConnection();
        tunePatchConnection(con);

        String jsonInputString = "{ \"trainingid\" : " + trainingId + " }";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateNote(int accountId, int trainingId, int dayOfWeek, int orderNumber, String note)
throws IOException {
        URL url_update_note = new URL("http://" + local_IP + ":3000/trainingexercisenote?and=(" +
                "accountid.eq." + accountId + "," +
                "trainingid.eq." + trainingId + "," +
                "dayofweek.eq." + dayOfWeek + "," +
                "ordernumber.eq." + orderNumber + ")");

        HttpURLConnection con = (HttpURLConnection) url_update_note.openConnection();
        tunePatchConnection(con);
```

```
        String jsonInputString = "{ \"note\" : \"" + note + "\" }";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateAccountInfo(int accountId, Account newAccount) throws IOException {
        URL url_update_account_info = new URL("http://" + local_IP + ":3000/account?" +
                "accountid=eq." + accountId);

        HttpURLConnection con = (HttpURLConnection) url_update_account_info.openConnection();
        tunePatchConnection(con);

        String jsonInputString = "{ " +
                "\"firstname\" : \"" + newAccount.getFirstName() + "\", " +
                "\"secondname\" : \"" + newAccount.getSecondName() + "\", " +
                "\"username\" : \"" + newAccount.getUserName() + "\" " +
                " }";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void updateBodyConditionInfo(int accountId, BodyCondition newBodyCondition) throws
IOException {
        URL url_update_body_condition_info = new URL("http://" + local_IP + ":3000/bodycondition?" +
                "accountid=eq." + accountId);

        HttpURLConnection con = (HttpURLConnection)
url_update_body_condition_info.openConnection();
        tunePatchConnection(con);

        String jsonInputString = "{ " +
                "\"age\" : " + newBodyCondition.getAge() + ", " +
                "\"weight\" : " + newBodyCondition.getWeight() + ", " +
```

```
            "\"height\" : " + newBodyCondition.getHeight() + ", " +
            "\"bodyfatshare\" : " + newBodyCondition.getBodyFatShare() + " " +
            "}";
        try (OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
            os.write(input, 0, input.length);
            os.flush();
        }

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        }
    }

    public void deleteTrainingExerciseSuccess(ExerciseInTrainingEditing exerciseInTrainingEditing)
throws IOException {
        URL url_delete_training_exercise_success = new URL("http://" + local_IP +
":3000/trainingexercisesuccess?and=(" +
                "accountid.eq." + exerciseInTrainingEditing.getAccountId() + "," +
                "trainingid.eq." + exerciseInTrainingEditing.getTrainingId() + "," +
                "dayofweek.eq." + exerciseInTrainingEditing.getDayOfWeek() + "," +
                "ordernumber.eq." + exerciseInTrainingEditing.getOrderNumber() + "," +
                "setnumber.eq." + exerciseInTrainingEditing.getSetNumber() + ")");

        HttpURLConnection con = (HttpURLConnection)
url_delete_training_exercise_success.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }

    public void deleteTrainingExercise(ExerciseInTrainingEditing exerciseInTrainingEditing) throws
IOException {
        URL url_delete_training_exercise = new URL("http://" + local_IP + ":3000/trainingexercise?and=("
+
                "accountid.eq." + exerciseInTrainingEditing.getAccountId() + "," +
                "trainingid.eq." + exerciseInTrainingEditing.getTrainingId() + "," +
                "dayofweek.eq." + exerciseInTrainingEditing.getDayOfWeek() + "," +
                "ordernumber.eq." + exerciseInTrainingEditing.getOrderNumber() + "," +
                "setnumber.eq." + exerciseInTrainingEditing.getSetNumber() + ")");

        HttpURLConnection con = (HttpURLConnection) url_delete_training_exercise.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }
```

```
    public void deleteTrainingExerciseSuccessWithOrderNumber(int accountId, TrainingExerciseInstance
trainingExerciseInstance) throws IOException {
        URL url_delete_training_exercise_success_with_order_number = new URL("http://" + local_IP +
":3000/trainingexercisesuccess?and=(" +
                "accountid.eq." + accountId + "," +
                "trainingid.eq." + trainingExerciseInstance.getTrainingId() + "," +
                "dayofweek.eq." + trainingExerciseInstance.getDayOfWeek() + "," +
                "ordernumber.eq." + trainingExerciseInstance.getOrderNumber() + ")");

        HttpURLConnection con = (HttpURLConnection)
url_delete_training_exercise_success_with_order_number.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }

    public void deleteTrainingExerciseNoteWithOrderNumber(int accountId, TrainingExerciseInstance
trainingExerciseInstance) throws IOException {
        URL url_delete_training_exercise_note_with_order_number = new URL("http://" + local_IP +
":3000/trainingexercisenote?and=(" +
                "accountid.eq." + accountId + "," +
                "trainingid.eq." + trainingExerciseInstance.getTrainingId() + "," +
                "dayofweek.eq." + trainingExerciseInstance.getDayOfWeek() + "," +
                "ordernumber.eq." + trainingExerciseInstance.getOrderNumber() + ")");

        HttpURLConnection con = (HttpURLConnection)
url_delete_training_exercise_note_with_order_number.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }

    public void deleteTrainingExerciseWithOrderNumber(int accountId, TrainingExerciseInstance
trainingExerciseInstance) throws IOException {
        URL url_delete_training_exercise_with_order_number = new URL("http://" + local_IP +
":3000/trainingexercise?and=(" +
                "accountid.eq." + accountId + "," +
                "trainingid.eq." + trainingExerciseInstance.getTrainingId() + "," +
                "dayofweek.eq." + trainingExerciseInstance.getDayOfWeek() + "," +
                "ordernumber.eq." + trainingExerciseInstance.getOrderNumber() + ")");

        HttpURLConnection con = (HttpURLConnection)
url_delete_training_exercise_with_order_number.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }
```

```
    public void deleteTrainingFromId(int trainingId) throws IOException {
        URL url_delete_training = new URL("http://" + local_IP + ":3000/training?" +
                "trainingid=eq." + trainingId);

        HttpURLConnection con = (HttpURLConnection) url_delete_training.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }

    public void deleteTrainingExerciseSuccessFromTraining(int accountId, int trainingId) throws
IOException {
        URL url_delete_training_exercise_success_from_training = new URL("http://" + local_IP +
":3000/trainingexercisesuccess?and=(" +
                "trainingid.eq." + trainingId + "," +
                "accountid.eq." + accountId + ")");

        HttpURLConnection con = (HttpURLConnection)
url_delete_training_exercise_success_from_training.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }

    public void deleteTrainingExerciseNote(int accountId, int trainingId) throws IOException {
        URL url_delete_training_exercise_note = new URL("http://" + local_IP +
":3000/trainingexercisenote?and=(" +
                "trainingid.eq." + trainingId + "," +
                "accountid.eq." + accountId + ")");

        HttpURLConnection con = (HttpURLConnection)
url_delete_training_exercise_note.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
    }

    public void deleteFriendship(Friendship friendship) throws IOException {
        URL url_delete_friendship = new URL("http://" + local_IP + ":3000/friendship?and=(" +
                "accountid.eq." + friendship.getAccountId() + "," +
                "friendid.eq." + friendship.getFriendId() + ")");

        HttpURLConnection con = (HttpURLConnection) url_delete_friendship.openConnection();
        tuneDeleteConnection(con);

        getResponseString(con);
```

```
}

public void deleteAccount(int accountId) throws IOException {
    URL url_delete_account = new URL("http://" + local_IP + ":3000/account?" +
            "accountid=eq." + accountId);

    HttpURLConnection con = (HttpURLConnection) url_delete_account.openConnection();
    tuneDeleteConnection(con);

    getResponseString(con);
}

public void setNullCurrentTraining(int accountId) throws IOException {
    URL url_update_current_training = new URL("http://" + local_IP + ":3000/account?" +
            "accountid=eq." + accountId);

    HttpURLConnection con = (HttpURLConnection) url_update_current_training.openConnection();
    tunePatchConnection(con);

    String jsonInputString = "{ \"trainingid\" : null }";
    try (OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes(StandardCharsets.UTF_8);
        os.write(input, 0, input.length);
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
    }
}

private String getResponseString(HttpURLConnection con) throws IOException {
    OutputStreamWriter wr = new OutputStreamWriter(con.getOutputStream());
    wr.flush();

    StringBuilder response = new StringBuilder();
    try (BufferedReader br = new BufferedReader(
            new InputStreamReader(con.getInputStream(), StandardCharsets.UTF_8))) {
        String responseLine = null;
        while ((responseLine = br.readLine()) != null) {
            response.append(responseLine.trim());
        }
    }

    return response.toString();
}
```

```
    private void tunePostConnection(HttpURLConnection con) throws ProtocolException {
        con.setRequestMethod("POST");
        con.setRequestProperty("Content-Type", "application/json; utf-8");
        con.setRequestProperty("Accept", "application/json");
        con.setDoOutput(true);
    }

    private void tunePatchConnection(HttpURLConnection con) throws ProtocolException {
        con.setRequestMethod("PATCH");
        con.setRequestProperty("Content-Type", "application/json; utf-8");
        con.setRequestProperty("Accept", "application/json");
        con.setDoOutput(true);
    }

    private void tuneDeleteConnection(HttpURLConnection con) throws ProtocolException {
        con.setRequestMethod("DELETE");
        con.setRequestProperty("Content-Type", "application/json; utf-8");
        con.setRequestProperty("Accept", "application/json");
        con.setDoOutput(true);
    }

    public HttpWork() throws MalformedURLException {
    }
}
```

1.103. Muscle.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class Muscle {
    @SerializedName("muscleid")
    private final int muscleId;
    private final String title;

    public Muscle(int muscleId, String title) {
        this.muscleId = muscleId;
        this.title = title;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO Muscle(MuscleId, Title)\n" +
                "VALUES (" +
                "'" +
                muscleId +
                "', " +
```

```
                """ +
                title +
                "');";
    }

    public int getMuscleId() {
        return muscleId;
    }

    public String getTitle() {
        return title;
    }
}
```

1.104. TargetMuscle.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class TargetMuscle {
    @SerializedName("exerciseid")
    private final int exrciseId;
    @SerializedName("muscleid")
    private final int muscleId;

    public TargetMuscle(int exrciseId, int muscleId) {
        this.exrciseId = exrciseId;
        this.muscleId = muscleId;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO TargetMuscle(ExerciseId, MuscleId)\n" +
                "VALUES (" +
                """ +
                exrciseId +
                "', " +
                """ +
                muscleId +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"exerciseid\" : " + exrciseId + ", " +
                "\"muscleid\" : " + muscleId + " " +
                "}";
```

```
    }

    public int getExrciseId() {
        return exrciseId;
    }

    public int getMuscleId() {
        return muscleId;
    }
}
```

1.105. Training.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class Training {
    @SerializedName("trainingid")
    private int trainingId;
    private final String title;
    @SerializedName("accountid")
    private int accountId;

    public Training(int trainingId, String title, int accountId) {
        this.trainingId = trainingId;
        this.title = title;
        this.accountId = accountId;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO Training(TrainingId, Title, AccountId)\n" +
                "VALUES (" +
                "'" +
                trainingId +
                "', " +
                "'" +
                title +
                "', " +
                "'" +
                accountId +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"trainingid\" : " + trainingId + ", " +
```

```
            "\"title\" : \"" + title + "\", " +
            "\"accountid\" : " + accountId + " " +
            "}";
    }

    public String getTitleJSON() {
        return "{ " +
            "\"title\" : \"" + title + "\"" +
            " }";
    }

    public int getTrainingId() {
        return trainingId;
    }

    public String getTitle() {
        return title;
    }
}
```

1.106. TrainingExercise.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class TrainingExercise {
    @SerializedName("trainingid")
    private final int trainingId;
    @SerializedName("dayofweek")
    private final int dayOfWeek;
    @SerializedName("ordernumber")
    private final int orderNumber;
    @SerializedName("exerciseid")
    private final int exerciseId;
    @SerializedName("setnumber")
    private final int setNumber;
    @SerializedName("repsnum")
    private final int repsNum;
    private final int timer;
    @SerializedName("accountid")
    private int accountId;

    public TrainingExercise(int trainingId, int dayOfWeek, int orderNumber, int exerciseId, int setNumber,
int repsNum, int timer, int accountId) {
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
```

```
        this.orderNumber = orderNumber;
        this.exerciseId = exerciseId;
        this.setNumber = setNumber;
        this.repsNum = repsNum;
        this.timer = timer;
        this.accountId = accountId;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum, Timer, AccountId)\n" +
                "VALUES (" +
                "'" +
                trainingId +
                "', " +
                "'" +
                dayOfWeek +
                "', " +
                "'" +
                orderNumber +
                "', " +
                "'" +
                exerciseId +
                "', " +
                "'" +
                setNumber +
                "', " +
                "'" +
                repsNum +
                "', " +
                "'" +
                timer +
                "', " +
                "'" +
                accountId +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"trainingid\" : " + trainingId + ", " +
                "\"dayofweek\" : " + dayOfWeek + ", " +
                "\"ordernumber\" : " + orderNumber + ", " +
                "\"exerciseid\" : " + exerciseId + ", " +
                "\"setnumber\" : " + setNumber + ", " +
                "\"repsnum\" : " + repsNum + ", " +
                "\"timer\" : " + timer + ", " +
```

```
            "\"accountid\" : " + accountId + " " +
            "}";
    }

    public int getTrainingId() {
        return trainingId;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }

    public int getOrderNumber() {
        return orderNumber;
    }

    public int getExerciseId() {
        return exerciseId;
    }

    public int getSetNumber() {
        return setNumber;
    }

    public int getRepsNum() {
        return repsNum;
    }

    public int getTimer() {
        return timer;
    }

    public int getAccountId() {
        return accountId;
    }
}
```

1.107. TrainingExerciseNote.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class TrainingExerciseNote {
    @SerializedName("trainingid")
    private final int trainingId;
    @SerializedName("accountid")
```

```
    private final int accountId;
    @SerializedName("dayofweek")
    private final int dayOfWeek;
    @SerializedName("ordernumber")
    private final int orderNumber;
    @SerializedName("exerciseid")
    private final int exerciseId;
    private final String note;

    public TrainingExerciseNote(int trainingId, int accountId, int dayOfWeek, int orderNumber, int exerciseId, String note) {
        this.trainingId = trainingId;
        this.accountId = accountId;
        this.dayOfWeek = dayOfWeek;
        this.orderNumber = orderNumber;
        this.exerciseId = exerciseId;
        this.note = note;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO TrainingExerciseNote(TrainingId, AccountId, DayOfWeek, OrderNumber, ExerciseId, Note)\n" +
                "VALUES (" +
                "'" +
                trainingId +
                "', " +
                "'" +
                accountId +
                "', " +
                "'" +
                dayOfWeek +
                "', " +
                "'" +
                orderNumber +
                "', " +
                "'" +
                exerciseId +
                "', " +
                "'" +
                note +
                "');";
    }

    public String getJSON() {
        return "{" +
                "\"trainingid\" : " + trainingId + ", " +
                "\"accountid\" : " + accountId + ", " +
```

```
            "\"dayofweek\" : " + dayOfWeek + ", " +
            "\"ordernumber\" : " + orderNumber + ", " +
            "\"exerciseid\" : " + exerciseId + ", " +
            "\"note\" : \"" + note + "\"" +
            "}";
    }

    public int getTrainingId() {
        return trainingId;
    }

    public int getAccountId() {
        return accountId;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }

    public int getOrderNumber() {
        return orderNumber;
    }

    public int getExerciseId() {
        return exerciseId;
    }

    public String getNote() {
        return note;
    }
}
```

1.108. TrainingExerciseSuccess.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class TrainingExerciseSuccess {
    @SerializedName("accountid")
    private final int accountId;
    @SerializedName("trainingid")
    private final int trainingId;
    @SerializedName("dayofweek")
    private final int dayOfWeek;
    @SerializedName("ordernumber")
    private final int orderNumber;
```

```
    @SerializedName("exerciseid")
    private final int exerciseId;
    @SerializedName("setnumber")
    private final int setNumber;
    @SerializedName("repsnum")
    private final int repsNum;
    private final int weight;
    private final int timer;


    public TrainingExerciseSuccess(int accountId, int trainingId, int dayOfWeek, int orderNumber, int exerciseId, int setNumber, int repsNum, int weight, int timer) {
        this.accountId = accountId;
        this.trainingId = trainingId;
        this.dayOfWeek = dayOfWeek;
        this.orderNumber = orderNumber;
        this.exerciseId = exerciseId;
        this.setNumber = setNumber;
        this.repsNum = repsNum;
        this.weight = weight;
        this.timer = timer;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO TrainingExerciseSuccess(AccountId, TrainingId, DayOfWeek, OrderNumber, ExerciseId, SetNumber, RepsNum, Weight, Timer)\n" +
                "VALUES (" +
                "'" +
                accountId +
                "', " +
                "'" +
                trainingId +
                "', " +
                "'" +
                dayOfWeek +
                "', " +
                "'" +
                orderNumber +
                "', " +
                "'" +
                exerciseId +
                "', " +
                "'" +
                setNumber +
                "', " +
                "'" +
                repsNum +
```

```
            "', " +
            """ +
            weight +
            "', " +
            """ +
            timer +
            "');";
    }

    public String getJSON() {
        return "{" +
                "\"accountid\" : " + accountId + ", " +
                "\"trainingid\" : " + trainingId + ", " +
                "\"dayofweek\" : " + dayOfWeek + ", " +
                "\"ordernumber\" : " + orderNumber + ", " +
                "\"exerciseid\" : " + exerciseId + ", " +
                "\"setnumber\" : " + setNumber + ", " +
                "\"repsnum\" : " + repsNum + ", " +
                "\"weight\" : " + weight + ", " +
                "\"timer\" : " + timer + " " +
                "}";
    }

    public int getAccountId() {
        return accountId;
    }

    public int getTrainingId() {
        return trainingId;
    }

    public int getDayOfWeek() {
        return dayOfWeek;
    }

    public int getOrderNumber() {
        return orderNumber;
    }

    public int getExerciseId() {
        return exerciseId;
    }

    public int getSetNumber() {
        return setNumber;
    }
```

```
    public int getRepsNum() {
        return repsNum;
    }

    public int getWeight() {
        return weight;
    }

    public int getTimer() {
        return timer;
    }
}
```

1.109. TrainingType.java

```
package com.example.testbottomnavigationbar.remote_db;

import com.google.gson.annotations.SerializedName;

public class TrainingType {
    @SerializedName("typeid")
    private final int typeId;
    private final String title;

    public TrainingType(int typeId, String title) {
        this.typeId = typeId;
        this.title = title;
    }

    public String getSQLiteInsertQuery() {
        return "INSERT INTO TrainingType(TypeId, Title)\n" +
                "VALUES (" +
                "'" +
                typeId +
                "', " +
                "'" +
                title +
                "');";
    }

    public int getTypeId() {
        return typeId;
    }

    public String getTitle() {
        return title;
    }
```

}

1.110. ExerciseSetResult.java

```
package com.example.testbottomnavigationbar;

public class ExerciseSetResult {
    private int weight;
    private int repetitions;
    private int timer;

    public ExerciseSetResult(int weight, int repetitions, int timer) {
        this.weight = weight;
        this.repetitions = repetitions;
        this.timer = timer;
    }

    public int getWeight() {
        return weight;
    }

    public int getRepetitions() {
        return repetitions;
    }

    public int getTimer() {
        return timer;
    }
}
```

1.111. MainActivity.java

```
package com.example.testbottomnavigationbar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.annotation.SuppressLint;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
```

```
import android.widget.ProgressBar;
import android.widget.TextView;

import com.example.testbottomnavigationbar.fragments.AccountFragment;
import com.example.testbottomnavigationbar.fragments.SearchFragment;
import com.example.testbottomnavigationbar.fragments.TimeTableFragment;
import com.example.testbottomnavigationbar.fragments.TrainingsFragment;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.io.IOException;

public class MainActivity extends AppCompatActivity {
    public static final boolean LOG = true;
    public static final String TEG = "dchertanov_";

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

        try {
            SQLiteHelper.handleDBAbsence(this,                                    (ProgressBar)
findViewById(R.id.activity_main_progressbar));
        } catch (IOException e) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "Problem with load DB:  " + e, e);
            }
        }
    }

    public static void hideSoftKeyboard(Context context, View view) {
        InputMethodManager          inputMethodManager          =          (InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(), 0);
    }
}
```

1.112. SQLiteHelper,java

```
package com.example.testbottomnavigationbar;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.widget.ProgressBar;
```

```
import com.example.testbottomnavigationbar.remote_db.Account;
import com.example.testbottomnavigationbar.remote_db.BodyCondition;
import com.example.testbottomnavigationbar.remote_db.Exercise;
import com.example.testbottomnavigationbar.remote_db.ExerciseTag;
import com.example.testbottomnavigationbar.remote_db.ExerciseToTag;
import com.example.testbottomnavigationbar.remote_db.ExerciseTrainingType;
import com.example.testbottomnavigationbar.remote_db.Friendship;
import com.example.testbottomnavigationbar.remote_db.HttpWork;
import com.example.testbottomnavigationbar.remote_db.Muscle;
import com.example.testbottomnavigationbar.remote_db.TargetMuscle;
import com.example.testbottomnavigationbar.remote_db.Training;
import com.example.testbottomnavigationbar.remote_db.TrainingExercise;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseNote;
import com.example.testbottomnavigationbar.remote_db.TrainingExerciseSuccess;
import com.example.testbottomnavigationbar.remote_db.TrainingType;
import com.example.testbottomnavigationbar.remote_db.tasks.PullExerciseTask;
import com.example.testbottomnavigationbar.remote_db.tasks.PullRemoteDBTask;

import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;

public class SQLiteHelper {
    public static final String dbName = "app.db";
    public static final String currentAccountDBName = "currentaccount.db";

    public static String getDbName() {
        return dbName;
    }

    public static String getCurrentAccountDBName() {
        return currentAccountDBName;
    }

    public static boolean isLocalDBExist(Context context, String dbName) {
        File dbFile = context.getDatabasePath(dbName);
        return dbFile.exists();
    }

    public static void handleDBAbsence(Context context, ProgressBar progressBar) throws IOException {
        if (isLocalDBExist(context, currentAccountDBName)) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "currentAccountDb exist");
            }
        }
```

```
        if (isLocalDBExist(context, dbName)) {
            context.deleteDatabase(dbName);
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, "db exist");
            }
//          return;
        }

        SQLiteDatabase db = context.openOrCreateDatabase(dbName, Context.MODE_PRIVATE, null);
        SQLiteDatabase currentAccountDB = context.openOrCreateDatabase(currentAccountDBName,
Context.MODE_PRIVATE, null);
        db.execSQL("PRAGMA foreign_keys=ON;");

        currentAccountDB.execSQL("CREATE TABLE IF NOT EXISTS CurrentAccount (AccountId
INTEGER PRIMARY KEY NOT NULL)");

        db.execSQL("CREATE TABLE IF NOT EXISTS Exercise (ExerciseId INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, Title VARCHAR(256) NOT NULL, Description TEXT, VideoPath
TEXT)");
        db.execSQL("CREATE TABLE IF NOT EXISTS ExerciseTag (TagId INTEGER PRIMARY KEY
NOT NULL, Title VARCHAR(256))");
        db.execSQL("CREATE TABLE IF NOT EXISTS ExerciseToTag (ExerciseId INTEGER NOT
NULL, TagId INTEGER NOT NULL, CONSTRAINT fk_Exercise FOREIGN KEY(ExerciseId)
REFERENCES Exercise(ExerciseId) ON DELETE CASCADE, CONSTRAINT fk_ExerciseTag
FOREIGN KEY(TagId) REFERENCES ExerciseTag(TagId) ON DELETE CASCADE, PRIMARY
KEY(ExerciseId, TagId))");
        db.execSQL("CREATE TABLE IF NOT EXISTS Account (AccountId INTEGER PRIMARY KEY
NOT NULL, Username VARCHAR(256), HashPassword VARCHAR(256), BirthDate DATE NOT
NULL, RegistrationDate DATE NOT NULL, Sex VARCHAR(256), TrainingId INTEGER, FirstName
VARCHAR(256) NOT NULL, SecondName VARCHAR(256) NOT NULL)");
        db.execSQL("CREATE TABLE IF NOT EXISTS Training (TrainingId INTEGER PRIMARY KEY
NOT NULL, Title TEXT, AccountId INTEGER NOT NULL)");
        db.execSQL("CREATE TABLE IF NOT EXISTS TrainingExercise (TrainingId INTEGER NOT
NULL, DayOfWeek INTEGER CHECK (DayOfWeek >= 0 AND DayOfWeek < 7), OrderNumber
INTEGER CHECK (OrderNumber > 0), ExerciseId INTEGER NOT NULL, SetNumber INTEGER
CHECK (SetNumber > 0), RepsNum INTEGER CHECK (RepsNum >= 0), Timer INTEGER CHECK
(Timer >= 0), AccountId INTEGER NOT NULL, CONSTRAINT fk_Exercise FOREIGN
KEY(ExerciseId) REFERENCES Exercise(ExerciseId) ON DELETE CASCADE, CONSTRAINT
fk_Training FOREIGN KEY(TrainingId) REFERENCES Training(TrainingId) ON DELETE CASCADE,
PRIMARY KEY(TrainingId, DayOfWeek, OrderNumber, SetNumber, AccountId))");
        db.execSQL("CREATE TABLE IF NOT EXISTS TrainingExerciseSuccess (AccountId INTEGER
NOT NULL, TrainingId INTEGER NOT NULL, DayOfWeek INTEGER CHECK (DayOfWeek >= 0
AND DayOfWeek < 7), OrderNumber INTEGER CHECK (OrderNumber > 0), ExerciseId INTEGER
NOT NULL, SetNumber INTEGER CHECK (SetNumber > 0), RepsNum INTEGER CHECK (RepsNum
>= 0), Weight INTEGER CHECK (Weight >= 0), Timer INTEGER CHECK (Timer >= 0), CONSTRAINT
fk_Account FOREIGN KEY(AccountId) REFERENCES Account(AccountId) ON DELETE CASCADE,
CONSTRAINT fk_Training FOREIGN KEY(TrainingId) REFERENCES Training(TrainingId) ON
```

```
DELETE CASCADE, PRIMARY KEY(AccountId, TrainingId, DayOfWeek, OrderNumber, SetNumber))");
        db.execSQL("CREATE TABLE IF NOT EXISTS TrainingExerciseNote (TrainingId INETEGER NOT NULL, AccountId INTEGER NOT NULL, DayOfWeek INTEGER CHECK (DayOfWeek >= 0 AND DayOfWeek < 7), OrderNumber INTEGER CHECK (OrderNumber > 0), Note TEXT, ExerciseId INTEGER NOT NULL, CONSTRAINT fk_Exercise FOREIGN KEY(ExerciseId) REFERENCES Exercise(ExerciseId) ON DELETE CASCADE, CONSTRAINT fk_Account FOREIGN KEY(AccountId) REFERENCES Account(AccountId) ON DELETE CASCADE, CONSTRAINT fk_Training FOREIGN KEY(TrainingId) REFERENCES Training(TrainingId) ON DELETE CASCADE, PRIMARY KEY(AccountId, TrainingId, DayOfWeek, OrderNumber))");
        db.execSQL("CREATE TABLE IF NOT EXISTS TrainingType (TypeId INTEGER PRIMARY KEY NOT NULL, Title VARCHAR(256) NOT NULL)");
        db.execSQL("CREATE TABLE IF NOT EXISTS ExerciseTrainingType (TypeId INTEGER NOT NULL, ExerciseId INTEGER NOT NULL, CONSTRAINT fk_TrainingType FOREIGN KEY(TypeId) REFERENCES TrainingType(TypeId) ON DELETE CASCADE, CONSTRAINT fk_Exercise FOREIGN KEY(ExerciseId) REFERENCES Exercise(ExerciseId) ON DELETE CASCADE, PRIMARY KEY(TypeId, ExerciseId))");
        db.execSQL("CREATE TABLE IF NOT EXISTS Muscle (MuscleId INTEGER PRIMARY KEY NOT NULL, Title VARCHAR(256) NOT NULL)");
        db.execSQL("CREATE TABLE IF NOT EXISTS TargetMuscle (ExerciseId INTEGER NOT NULL, MuscleId INTEGER NOT NULL, CONSTRAINT fk_Exercise FOREIGN KEY(ExerciseId) REFERENCES Exercise(ExerciseId) ON DELETE CASCADE, CONSTRAINT fk_Muscle FOREIGN KEY(MuscleId) REFERENCES Muscle(MuscleId) ON DELETE CASCADE, PRIMARY KEY(ExerciseId, MuscleId))");
        db.execSQL("CREATE TABLE IF NOT EXISTS BodyCondition (AccountId INTEGER PRIMARY KEY NOT NULL, Weight REAL NOT NULL, Height REAL NOT NULL, Age INTEGER NOT NULL, BodyFatShare REAL CHECK (BodyFatShare >= 0 AND BodyFatShare <= 100), CONSTRAINT fk_Account FOREIGN KEY(AccountId) REFERENCES Account(AccountId) ON DELETE CASCADE)");
        db.execSQL("CREATE TABLE IF NOT EXISTS Friendship (AccountId INTEGER NOT NULL, FriendId INTEGER NOT NULL, CONSTRAINT fk_Account FOREIGN KEY(AccountId) REFERENCES Account(AccountId) ON DELETE CASCADE, CONSTRAINT fk_Account FOREIGN KEY(FriendId) REFERENCES Account(AccountId) ON DELETE CASCADE, PRIMARY KEY(AccountId, FriendId))");

//      addTestData(context);
        pullRemoteData(db, context, progressBar);
    }

    private static void pullRemoteData(SQLiteDatabase db, Context context, ProgressBar progressBar) throws IOException {
        new PullRemoteDBTask(context, progressBar).execute();

//      pullAccount(worker, db);
//      pullExercise(worker, db);
//      pullExerciseToTag(worker, db);
//      pullExerciseTag(worker, db);
```

```
//        pullTraining(worker, db);
//        pullTrainingExercise(worker, db);
//        pullTrainingExerciseNote(worker, db);
//        pullTrainingExerciseSuccess(worker, db);
    }

    public static Account getAccountFromId(SQLiteDatabase db, int accountId) {
        Cursor cursor = db.rawQuery("SELECT Username, Sex, FirstName, SecondName FROM Account
WHERE AccountId = " + accountId + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            String username = cursor.getString(0);
            String sex = cursor.getString(1);
            String firstName = cursor.getString(2);
            String secondName = cursor.getString(3);

            return new Account(accountId, username, null, null, null, sex, -1, firstName, secondName);
        }

        return null;
    }

    public static BodyCondition getBodyConditionFromAccountId(SQLiteDatabase db, int accountId) {
        Cursor cursor = db.rawQuery("SELECT Weight, Height, Age, BodyFatShare FROM BodyCondition
WHERE AccountId = " + accountId + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            float weight = cursor.getFloat(0);
            float height = cursor.getFloat(1);
            int age = cursor.getInt(2);
            float bodyFatShare = cursor.getFloat(3);

            return new BodyCondition(accountId, weight, height, age, bodyFatShare);
        }

        return null;
    }

    public static int getFriendsCnt(SQLiteDatabase db, int accountId) {
        Cursor cursor = db.rawQuery("SELECT COUNT(*) FROM Friendship WHERE AccountId = " +
accountId + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return 0;
    }

    public static boolean isFriend(SQLiteDatabase db, SQLiteDatabase currentAccountDB, int friendId) {
```

```
        int accountId = getCurrentAccountId(currentAccountDB);

        Cursor cursor = db.rawQuery("SELECT * FROM Friendship WHERE AccountId = " + accountId +
" AND FriendId = " + friendId + ";", null);
        return (cursor != null && cursor.moveToFirst());
    }

    public static int getTrainingCnt(SQLiteDatabase db, int accountId) {
        Cursor cursor = db.rawQuery("SELECT COUNT(*) FROM Training WHERE AccountId = " +
accountId + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return 0;
    }

    public static int getCurrentAccountId(SQLiteDatabase db) {
        Cursor cursor = db.rawQuery("SELECT * FROM CurrentAccount;", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return -1;
    }

    public static int getAccountIdFromUsername(SQLiteDatabase db, String username) {
        Cursor cursor = db.rawQuery("SELECT AccountId FROM Account WHERE Username = '" +
username + "';", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return -1;
    }

    public static int getCurrentTrainingId(SQLiteDatabase db, SQLiteDatabase currentAccountDB) {
        int accountId = getCurrentAccountId(currentAccountDB);

        Cursor cursor = db.rawQuery("SELECT TrainingId FROM Account WHERE AccountId = " +
accountId + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return -1;
    }
```

```
    public static int getMuscleIdFromTitle(SQLiteDatabase db, String title) {
        Cursor cursor = db.rawQuery("SELECT MuscleId FROM Muscle WHERE Title = '" + title + "';",
null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return -1;
    }

    public static int getTagIdFromTitle(SQLiteDatabase db, String title) {
        Cursor cursor = db.rawQuery("SELECT TagId FROM ExerciseTag WHERE Title = '" + title + "';",
null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return -1;
    }

    public static int getTrainingIdFromTitle(SQLiteDatabase db, String title, int accountId) {
        Cursor cursor = db.rawQuery("SELECT TrainingId FROM Training WHERE Title = '" + title + "'
AND AccountId = " + accountId + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return -1;
    }

    public static String getTrainingTitleFromId(SQLiteDatabase db, int trainingId) {
        Cursor cursor = db.rawQuery("SELECT Title FROM Training WHERE TrainingId = " + trainingId +
";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getString(0);
        }

        return "";
    }

    public static int getExerciseIdFromTitle(SQLiteDatabase db, String title) {
        Cursor cursor = db.rawQuery("SELECT ExerciseId FROM Exercise WHERE Title = '" + title + "';",
null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }
```

```
        return -1;
    }

    public static int getMaxSetNumberFromTrainingExerciseSuccess(SQLiteDatabase db, int accountId, int
trainingId, int dayOfWeek, int orderNumber) {
        Cursor cursor = db.rawQuery("SELECT MAX(SetNumber) FROM TrainingExerciseSuccess \n" +
                "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + " AND DayOfWeek
= " + dayOfWeek + " AND OrderNumber = " + orderNumber + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return  -1;
    }

    public static int getMaxSetNumberFromTrainingExercise(SQLiteDatabase db, int accountId, int
trainingId, int dayOfWeek, int orderNumber) {
        Cursor cursor = db.rawQuery("SELECT MAX(SetNumber) FROM TrainingExercise \n" +
                "WHERE AccountId = " + accountId + " AND TrainingId = " + trainingId + " AND DayOfWeek
= " + dayOfWeek + " AND OrderNumber = " + orderNumber + ";", null);
        if (cursor != null && cursor.moveToFirst()) {
            return cursor.getInt(0);
        }

        return  -1;
    }

    public static void pullAccount(HttpWork worker, SQLiteDatabase db) throws IOException {
        Account[] accounts = worker.pullAllAccounts();
        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Account count  " + accounts.length);
        }
        for (Account account : accounts) {
            db.execSQL(account.getSQLiteInsertQuery());
        }
    }

    public static void pullExercise(HttpWork worker, SQLiteDatabase db) throws IOException {
        Exercise[] exercises = worker.pullAllExercises();
        for (Exercise exercise : exercises) {
            db.execSQL(exercise.getSQLiteInsertQuery());
        }

        if (MainActivity.LOG) {
            Log.d(MainActivity.TEG, "Exercise count  " + exercises.length);
        }
```

```
    }

    public static void pullExerciseToTag(HttpWork worker, SQLiteDatabase db) throws IOException {
        ExerciseToTag[] exerciseToTags = worker.pullAllExerciseToTag();
        for (ExerciseToTag exerciseToTag : exerciseToTags) {
            if (MainActivity.LOG) {
                Log.d(MainActivity.TEG, exerciseToTag.getSQLiteInsertQuery());
            }

            db.execSQL(exerciseToTag.getSQLiteInsertQuery());
        }
    }

    public static void pullExerciseTag(HttpWork worker, SQLiteDatabase db) throws IOException {
        ExerciseTag[] exerciseTags = worker.pullAllExerciseTags();
        for (ExerciseTag exerciseTag : exerciseTags) {
            db.execSQL(exerciseTag.getSQLiteInsertQuery());
        }
    }

    public static void pullTraining(HttpWork worker, SQLiteDatabase db) throws IOException {
        Training[] trainings = worker.pullAllTrainings();
        for (Training training : trainings) {
            db.execSQL(training.getSQLiteInsertQuery());
        }
    }

    public static void pullTrainingExercise(HttpWork worker, SQLiteDatabase db) throws IOException {
        TrainingExercise[] trainingExercises = worker.pullAllTrainingExercises();
        for (TrainingExercise trainingExercise : trainingExercises) {
            db.execSQL(trainingExercise.getSQLiteInsertQuery());
        }
    }

    public static void pullTrainingExerciseNote(HttpWork worker, SQLiteDatabase db) throws IOException
{
        TrainingExerciseNote[] trainingExerciseNotes = worker.pullAllTrainingExerciseNote();
        for (TrainingExerciseNote trainingExerciseNote : trainingExerciseNotes) {
            db.execSQL(trainingExerciseNote.getSQLiteInsertQuery());
        }
    }

    public static void pullTrainingExerciseSuccess(HttpWork worker, SQLiteDatabase db) throws
IOException {
        TrainingExerciseSuccess[] trainingExerciseSuccesses = worker.pullAllTrainingExerciseSuccess();
        for (TrainingExerciseSuccess trainingExerciseSuccess : trainingExerciseSuccesses) {
            db.execSQL(trainingExerciseSuccess.getSQLiteInsertQuery());
```

```
        }
    }

    public static void pullTrainingType(HttpWork worker, SQLiteDatabase db) throws IOException {
        TrainingType[] trainingTypes = worker.pullAllTrainingTypes();
        for (TrainingType trainingType : trainingTypes) {
            db.execSQL(trainingType.getSQLiteInsertQuery());
        }
    }

    public static void pullExerciseTrainingType(HttpWork worker, SQLiteDatabase db) throws
IOException {
        ExerciseTrainingType[] exerciseTrainingTypes = worker.pullAllExerciseTrainingTypes();
        for (ExerciseTrainingType exerciseTrainingType : exerciseTrainingTypes) {
            db.execSQL(exerciseTrainingType.getSQLiteInsertQuery());
        }
    }

    public static void pullMuscle(HttpWork worker, SQLiteDatabase db) throws IOException {
        Muscle[] muscles = worker.pullAllMuscles();
        for (Muscle muscle : muscles) {
            db.execSQL(muscle.getSQLiteInsertQuery());
        }
    }

    public static void pullTargetMuscle(HttpWork worker, SQLiteDatabase db) throws IOException {
        TargetMuscle[] targetMuscles = worker.pullAllTargetMuscles();
        for (TargetMuscle targetMuscle : targetMuscles) {
            db.execSQL(targetMuscle.getSQLiteInsertQuery());
        }
    }

    public static void pullBodyCondition(HttpWork worker, SQLiteDatabase db) throws IOException {
        BodyCondition[] bodyConditions = worker.pullAllBodyConditions();
        for (BodyCondition bodyCondition : bodyConditions) {
            db.execSQL(bodyCondition.getSQLiteInsertQuery());
        }
    }

    public static void pullFriendship(HttpWork worker, SQLiteDatabase db) throws IOException {
        Friendship[] friendships = worker.pullAllFriendships();
        for (Friendship friendship : friendships) {
            db.execSQL(friendship.getSQLiteInsertQuery());
        }
    }

//    public static void addTestData(Context context) {
```

// SQLiteDatabase db = context.openOrCreateDatabase(dbName, Context.MODE_PRIVATE, null);
//
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Жим лежа', 'Сделайте глубокий вдох и задержите дыхание. Опустите штангу и коснитесь ею середины груди. В нижней точке ваши предплечья должны находиться в вертикальном положении. Если они находятся под наклоном, измените ширину хвата. Плечи должны располагаться под углом в 75° к телу. Если вы расставляете локти широко, можете травмировать плечи, если прижимаете к телу — снижаете эффективность движения. Отталкивая ногами пол и напрягая ягодицы, выжмите штангу в исходное положение — строго над плечами. Если проследить амплитуду снаряда, получится, что она идёт по небольшой дуге от середины груди до плеч. Если вы будете выжимать штангу строго вверх по вертикали, в верхней точке она окажется не над плечами, а впереди. Это увеличит рычаг до плеч, утомит их и не позволит вам выложиться по полной. Чтобы сохранить правильное положение тела, используйте один совет: представляйте, что вы не жмёте штангу, а отжимаетесь от неё. Толкайте тело в лавку, не забывая при этом сводить лопатки и сохранять прогиб в грудном отделе. Такой мысленный трюк поможет вам удержать плечи от выхода вперёд. Прежде чем увеличивать вес в этом упражнении, попросите кого-нибудь заснять вас на видео с разных ракурсов и оцените положение предплечий и плеч во время жима, размещение грифа на старте и амплитуду его движения. Если вы выполняете упражнение технично и вам легко делать 8–10 раз с грифом, можете увеличивать вес, но делайте это постепенно.',\n" +
// " 'https://www.youtube.com/watch?v=Czl9NcGrIsE')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Становая тяга', 'Плечи и лопатки отведите немного назад. Ноги расставьте уже уровня плеч (примерно на уровне бедер), коснитесь голенями грифа. Положение стоп – параллельно друг другу. Гриф проходит над центром стопы (а не над центром ее передней части). Присядьте – отведите таз назад и выпрямите спину, но не выводите колени за линию носков. Возьмитесь руками за гриф хватом на ширине плеч (расстояние между кистями – 40 см, колени внутри рук). Руки – прямые, локти зафиксированы. При классическом хвате ладони повернуты вниз. Выставьте грудь и плечи немного вперед за линию штанги. Взгляд направлен строго вниз. Не раскачиваясь и не перенося вес тела на носки, сорвите штангу движением мышц ног – квадрицепсов и ягодиц. После того как штанга прошла 20-30% амплитуды, выпрямите поясницу, выталкивая таз вперед, и зафиксируйте позицию. В точке подъема веса можно выставить грудь вперед (но не сводить лопатки). Перед тем как опустить штангу, отведите бедра назад так, чтобы не задеть грифом колени. При движении вниз держите позвоночник в нейтральном положении – не выгибайте и не прогибайте. Только после того, как гриф достигнет уровня колен, их можно согнуть.',\n" +
// " 'https://www.youtube.com/watch?v=KddlE57Yjt0')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Подтягивания обратным хватом', 'При подтягивании обратным хватом можно использовать один из вариантов: средний либо узкий. Среднее расположение ладоней при подтягивании обратным хватом считается классическим. Ладони располагаются на ширине плеч захватом на себя, большие пальцы образуют «замок». Движение вверх следует начинать не спеша и следить за тем, чтобы не поднимать по инерции плечи: их нужно отводить вниз-назад, сводя лопатки. Правильное выполнение упражнения прокачивает бицепс и включает в работу группу спинных мышц. Обратный хват с узким расположением рук позволяет отработать не только бицепсы, но и нижнюю часть широчайших мышц спины. Перед выполнением упражнения следует положить ладони на перекладину на наименьшем расстоянии друг от друга, развернув их на себя и закрыв «замком» больших пальцев. Поднимая туловище вверх, нужно сводить лопатки вместе, отводя назад плечи. В наивысшей точке нижняя часть груди должна прикоснуться к перекладине.

Во время выполнения упражнения на плечевые суставы приходится довольно большая нагрузка, поэтому необходимо двигаться плавно, без резких движений и раскачиваний. Выдох следует делать при движении вниз, следя за тем, чтобы локти не расходились в стороны и были параллельны друг другу. Двигаясь вверх, прижмите подбородок и слегка скрутите корпус, чтобы выделить бицепсы и снизить нагрузку на спинные мышцы. Максимальное напряжение бицепса должно приходиться на верхнюю точку амплитуды. Подтянувшись, зафиксируйте положение тела, обеспечив бицепсу максимальную статическую нагрузку. Опускаться вниз нужно медленно, без рывков — по времени эта фаза должна занимать около 3 секунд.',\n" +
// " 'https://www.youtube.com/watch?v=bF4T6-Mh0MU')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Гиперэкстензии', 'Делается упор бедрами в мягкую подушку чуть ниже линии сгиба поясничного отдела. Живот не должен мешать. Далее делается упор прямых стоп параллельно друг другу в платформу, закрепляется фиксирующими валиками. Для правильного выполнения гиперэкстензии важно проследить за положением коленных суставов. Ноги постоянно должны быть слегка согнуты, чтобы нагрузка равномерно распределялась. Периодическое разгибание спины происходит только до той линии, на которой находятся ноги. Чрезмерный прогиб может привести к травме позвонков. Угол сгиба не должен превышать 90 градусов. От положения рук зависит нагрузка на шейный отдел позвоночника. Если сложить их крестом на груди, то акцент перейдет на ягодицы. Если за шею - на разгибатели. Все движения должны выполняться медленно, без резких рывков. Плавно вниз, чуть быстрее вверх. Дыхание должно быть свободным, на каждом подъеме - выдох. Возможна задержка вверху - 1-2 секунды. Чтобы дать максимальную нагрузку на ягодицы, необходимо поддерживать спину скругленной и акцентировать внимание на ягодичном отделе. Напрягать попу и бицепсы бедер нужно в самом начале упражнения, и держать напряжение до самого конца. В нижней точке гиперэкстензии следует делать паузу, чтобы лишний раз убедиться в растяжении ягодиц. Аналогично в верхней точке. Руки и подбородок в данном варианте упражнения тесно прижимаются к груди, а взгляд направляется вниз. Нервно-мышечная связь имеет ключевое значение: сокращения ягодичных мышц нужно делать осознанно. Усилить нагрузку на заднюю часть ног можно более высоким положением тела относительно мягкой подушки (когда таз зависает в воздухе, а упор производится на середину бедра). При акценте же на мышечные разгибатели спину нужно выпрямлять, а взгляд направлять вперед. Руки можно закладывать за голову, чтобы акцентировать внимание на ровной осанке. ',\n" +
// " 'https://www.youtube.com/watch?v=k76TkJCe8Xs')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Мертвая тяга', 'Ноги на ширине плеч, руки расположены чуть шире плеч. Чтобы не ошибиться с шириной рук, ориентируйтесь по насечкам на грифе. Гриф штанги должен почти касаться голеней. В таком случае стопы будут располагаться под грифом, где-то треть стопы будет за грифом. Вы немного сгибаете ноги в коленях, отводите таз назад и делаете наклон вперёд. Спина прямая, от копчика до шеи — одна линия. Взгляд направлен вперёд. На начальном этапе движения, когда вы с прямой спиной наклоняетесь к штанге, растягиваются ягодичные мышцы и бицепс бедра — основные рабочие мышцы в этом упражнении. Если у вас короткий бицепс бедра, произойдёт следующее: при наклоне бицепс потянет за собой поясницу, так что вы не сможете держать спину прямо. Во время подъёма штанга расположена очень близко к телу: гриф штанги практически скользит по голеням (касание не обязательно, хотя возможно, особенно на первых этапах, чтобы привыкнуть к правильной технике), а затем поднимается выше по бёдрам. Когда вы отрываете штангу от земли, ваш центр тяжести совмещается с центром тяжести штанги. Когда вы наклоняетесь, центр тяжести смещается с крестца вперёд. Если вы держите штангу близко к голеням, центр тяжести штанги совпадает с вашим смещённым центром тяжести и вы удерживаете

равновесие. Если же вы встали далеко от штанги, центры тяжести не совпадут и штанга потянет вас вперёд, увеличивая нагрузку на поясницу. Гриф ведём по ногам. В момент отрыва штанги от земли (или платформы) необходимо напрячь ягодицы и мышцы бёдер. Сделать это нужно сознательно, не ожидая, когда напряжение возникнет само по себе. Напряжение ягодичных мышц необходимо, чтобы стабилизировать тазобедренный сустав. Напряжение мышц заставляет головку бедренной кости вращаться наружу, где она занимает максимально выгодную позицию для передачи усилия. Таким образом, вы стабилизируете сустав и обеспечиваете нейтральное положение позвоночника, за счёт чего нагрузка передаётся на ягодицы и заднюю часть бедра. Из этого положения вы полностью выпрямляетесь, а затем начинаете движение вниз, к исходному положению. Важно выполнять опускание штанги так же плавно, как и подъём, и вести гриф очень близко к бёдрам и голеням.',\n" +
// " 'https://www.youtube.com/watch?v=oqh7hHMlNyU')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Сгибание ног лежа', 'Рассмотрим технику выполнения упражнения лежа. Делая сгибания сидя или стоя, соблюдайте те же самые принципы. Прежде всего отрегулируйте тренажер под свой рост и длину ног. Коленные суставы должны выходить за край скамьи, а валик, в который вы будете упираться нижней частью голени, должен располагаться на несколько сантиметров выше пятки. Тут все просто – чем ближе валик к пятке, тем больше рычаг и тем эффективнее упражнение. На место изгиба скамьи вы ложитесь животом так, чтобы вам было комфортно и в пояснице не было напряжения. Лягте на тренажер и заведите ноги под валик. Руками возьмитесь за специальные ручки или за края скамьи. На выдохе согните голени, стараясь максимально приблизить валик к ягодицам. Передняя поверхность бедер при этом прижата к скамье. На вдохе разгибайте голени, аккуратно опуская вес вниз. Полностью выпрямлять колени и расслаблять бицепс бедра в нижней точке не нужно. Повторите нужное количество раз. Вы можете варьировать нагрузку в этом упражнении, меняя положение носков стоп. Разведя носки наружу, вы сместите акцент в сторону внешней поверхности бедер. Сведя их внутрь – в сторону внутренней. Можно также чередовать сгибания на станке в сидячем или лежачем положении. При наличии соответствующего тренажера, стоит попробовать делать упражнение стоя сначала одной ногой, потом другой. Чем более разносторонней будет проработка мышц, тем качественнее вы получите результат. После тренировки сделайте небольшую растяжку. Это поможет расслабить бицепс бедра и улучшит кровообращение.',\n" +
// " 'https://www.youtube.com/watch?v=Hy3WFtcY_o8')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Отжимания на брусьях', 'В исходном положении (верхней точке) ваш корпус расположен вертикально, вы держитесь на прямых руках, локти развернуты назад. Сделав вдох, опускайтесь вниз настолько, насколько позволяет гибкость ваших плечевых суставов. Ориентируйтесь на угол 90 градусов в локтях. Во время движения локти развернуты назад и прижаты к корпусу. На выдохе, за счет разгибания рук поднимитесь вверх. Если упражнение дается вам с большим трудом, в верхней точке разогните локти. Это даст трицепсам короткую передышку. Если же вы опытный спортсмен, оставляйте небольшой угол в локтях, чтобы не снимать нагрузку с целевых мышц. Работая на массу трицепса, стремитесь выполнить 10–15 повторов в 3–4 подходах. Опускайтесь вниз медленно, а поднимайтесь быстро. Как только все это будет удаваться вам достаточно просто, начинайте осваивать отжимания на брусьях с отягощением. Торопиться в этом вопросе не следует, так как, переусердствовав с нагрузкой, можно травмироваться и, вообще, лишить себя тренировок на долгое время. Перед тренировкой не забываем разминать всё тело(!), а в особенности мышцы, которые собираетесь сегодня проработать.',\n" +
// " 'https://www.youtube.com/watch?v=kvwNKPrRBtA')");

// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Разгибание ног в тренажере', 'Настроить спинку так, чтобы она была опорой. У большинства тренажеров регулируется наклон спинки и есть возможность придвинуть и отодвинуть ее. Правильное исходное положение – спина опирается на спинку, бедро лежит на сиденье, голень зафиксирована подушкой тренажера, а угол между голенью и бедром составляет не более 90 градусов. В реабилитации допускается неполная амплитуда, когда валик крепится выше, чем 90 градусов. Носки должны быть направлены чуть на себя, выполнять упражнение лучше начинать с сокращения квадрицепсов. Движение вниз не должно быть форсированным, лучше плавно опускаться и плавно разгибать голень. Спина должна быть прижатой к спинке полностью, поясничный отдел тоже. Разгибание нужно делать на выдохе, сгибание – на вдохе.',\n" +
// " 'https://www.youtube.com/watch?v=eT4thXaPR8w')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Приседания со штангой', 'Перед подходом подготовьтесь психологически, сосредоточьтесь и сконцентрируйтесь на поднимаемом весе. Перед приседаниями разогрейте тело с помощью кардиоупражнений. Первый сет начинают с нескольких подходов с минимальными отягощениями. Подойдите к штанге, расположенной на уровне ключиц. Возьмитесь за гриф прямым закрытым хватом. Выберите комфортную ширину постановки ладоней, как правило, это немного шире плеч: таким образом лопатки останутся сведенными, а спина ровной в течение всего подхода. Если плечевые суставы зажаты, расставьте руки шире, хотя при таком хвате увеличивается вероятность потери равновесия. Плотно сожмите гриф ладонями, слегка согните ноги в коленях и шагните вперед. Пройдите под штангой, подставьте вторую стопу и приподнимитесь так, чтобы лопатки сошлись, а гриф уперся в верхнюю часть спины. При правильной технике выполнения движений вес ляжет на трапециевидную мышцу и будет поддерживаться задними дельтами. В случае сильного давления на позвонки вернитесь в исходное положение и повторите подход к штанге. Напрягите мышцы ног, разогните колени и снимите вес с упоров. Сделайте небольшой шаг назад и подставьте вторую стопу. Примите нейтральное положение головы – так легче удерживать поясницу прогнутой, меньше риск потери равновесия. Расставьте ноги шире плеч и слегка разведите носки в стороны. Зафиксируйте положение и максимально сконцентрируйтесь перед приседом. На глубоком вдохе отведите таз назад и разведите колени в стороны. Не «заваливайтесь» вперед, достаточно естественного наклона корпуса. Правильный присед напоминает опускание ягодиц на воображаемый стул. Опустившись в нижнюю точку, сразу же поднимайтесь и выдыхайте воздух из легких. Техника подъема – за счет распрямления ног, а не за счет разгибания спины.',\n" +
// " 'https://www.youtube.com/watch?v=GotRb2Ob7MA')");
// db.execSQL("INSERT INTO Exercise(Title, Description, VideoPath)\n" +
// "VALUES ('Подъем штанги на бицепс', 'Встаньте прямо и установите ноги на ширине плеч. Ступни расположены почти параллельно, носки направлены немного в стороны. Возьмите штангу хватом снизу (ладони смотрят вверх) на ширине плеч. Держите спину прямой, не сутультесь. Взгляд направлен вперед. Сделайте глубокий вдох, задержите дыхание и, сгибая руки в локтях, поднимите штангу до уровня верха груди. Во время подъема штанги не двигайте локтями, держите их по бокам туловища и не сгибайте руки в запястьях. Как только кисти окажутся на уровне верха груди, сделайте паузу, выдохните и еще сильнее напрягите бицепсы. Плавно опустите штангу вниз, но не разгибайте руки полностью (не блокируйте локтевой сустав). Во время всего движения не наклоняйте торс ни вперед, ни назад. Держите правильную осанку.',\n" +
// " 'https://www.youtube.com/watch?v=FqF11YdwxrQ')");
//
//
// db.execSQL("INSERT INTO ExerciseTag(Title)\n" +

```
//            "VALUES ('Грудь')");
//      db.execSQL("INSERT INTO ExerciseTag(Title)\n" +
//            "VALUES ('Трицепс')");
//      db.execSQL("INSERT INTO ExerciseTag(Title)\n" +
//            "VALUES ('Ноги')");
//      db.execSQL("INSERT INTO ExerciseTag(Title)\n" +
//            "VALUES ('Спина')");
//      db.execSQL("INSERT INTO ExerciseTag(Title)\n" +
//            "VALUES ('Ягодицы')");
//      db.execSQL("INSERT INTO ExerciseTag(Title)\n" +
//            "VALUES ('Бицепс')");
//
//
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('1', '1')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('1', '2')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('2', '3')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('2', '4')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('2', '5')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('3', '4')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('3', '6')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('4', '4')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('4', '5')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('5', '3')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('5', '5')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('6', '3')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('7', '1')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('7', '2')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('8', '3')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('9', '3')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//            "VALUES ('9', '4')");
```

```
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//              "VALUES ('9', '5')");
//      db.execSQL("INSERT INTO ExerciseToTag(ExerciseId, TagId)\n" +
//              "VALUES ('10', '6')");
//
//
//
//      db.execSQL("INSERT INTO Training(Title)\n" +
//              "VALUES ('Тренировка #1');");
//      db.execSQL("INSERT INTO Training(Title)\n" +
//              "VALUES ('Тренировка #2');");
//      db.execSQL("INSERT INTO Training(Title)\n" +
//              "VALUES ('Тренировка #3');");
//      db.execSQL("INSERT INTO Training(Title)\n" +
//              "VALUES ('Упрощенная тренировка #1');");
//
//
//      db.execSQL("INSERT INTO Account(Username, HashPassword, BirthDate, RegistrationDate, Sex,
TrainingId)\n" +
//              "VALUES ('dchertanov', 'HashPassword', '2000-08-21', '2021-03-31', 'Male', 1);");
//
//
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '0', '1', '1', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '0', '2', '2', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '0', '3', '3', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '1', '1', '4', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '1', '2', '5', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '1', '3', '6', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '2', '1', '7', '3', '10');");
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '2', '2', '8', '3', '10');");
```

```
//      db.execSQL("INSERT INTO TrainingExercise(TrainingId, DayOfWeek, OrderNumber, ExerciseId,
SetNumber, RepsNum)\n" +
//              "VALUES ('1', '2', '3', '9', '3', '10');");
//   }
}
```

1.113. TrainingExerciseSetsObj.java

```
package com.example.testbottomnavigationbar;

import java.util.ArrayList;
import java.util.List;

public class TrainingExerciseSetsObj {
   private String exerciseTitle;
   private List<ExerciseSetResult> setsInfo;

   public TrainingExerciseSetsObj(String exerciseTitle) {
      this.exerciseTitle = exerciseTitle;
      setsInfo = new ArrayList<>();
   }

   public String getExerciseTitle() {
      return exerciseTitle;
   }

   public List<ExerciseSetResult> getSetsInfo() {
      return setsInfo;
   }

   public void addExerciseSetResult(ExerciseSetResult exerciseSetResult) {
      setsInfo.add(exerciseSetResult);
   }
}
```

1.114. TrainingExerciseSuccessObj.java

```
package com.example.testbottomnavigationbar;

import java.util.ArrayList;
import java.util.List;

public class TrainingExerciseSuccessObj {
   private String exerciseTitle;
   private List<ExerciseSetResult> setsInfo;
   private String note;
```

```
    public TrainingExerciseSuccessObj(String exerciseTitle) {
        this.exerciseTitle = exerciseTitle;
        setsInfo = new ArrayList<>();
        note = "";
    }

    public String getExerciseTitle() {
        return exerciseTitle;
    }

    public List<ExerciseSetResult> getSetsInfo() {
        return setsInfo;
    }

    public String getNote() {
        return note;
    }

    public void setNote(String note) {
        this.note = note;
    }

    public void addExerciseSetResult(ExerciseSetResult exerciseSetResult) {
        setsInfo.add(exerciseSetResult);
    }
}
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата