

Programming for mobile devices

TrackIT App

Universitatea de Vest din Timisoara

Facultatea de Matematica si Informatica

Lupulescu Alexandru

Coordinate teacher: Liviu-Octavian Mafteiu-Scai

15.05.2021

# Abstract

The principle of progressive overload suggests that the continual increase in the total workload during training sessions will stimulate muscle growth and strength gain. The tracker provides tools from which the users can visualize their weaknesses and improve their overall performance. It is crucial in fitness and strength training to track progress because it provides a way to visualize progressive overload. This report will provide an overview of the app's functionalities, the design and implementation choices and a comparison between my application and other applications.

## Goal and userbase

The goal of the application is to serve the users with a simple way to track their progress towards their fitness/strength goals. The tracker provides tools from which the users can visualize their weaknesses and improve their overall performance.

The userbase consists of anyone who has an Android phone with a minimum version of 4.1 and wishes to track its strength workouts. The simplicity of the UI allows anyone with a minimum knowledge of using mobile devices to use it.

# My contribution

The problem which the application is trying to solve is to encourage the users to track their training so they can focus on their workouts and progress at an optimal rate.

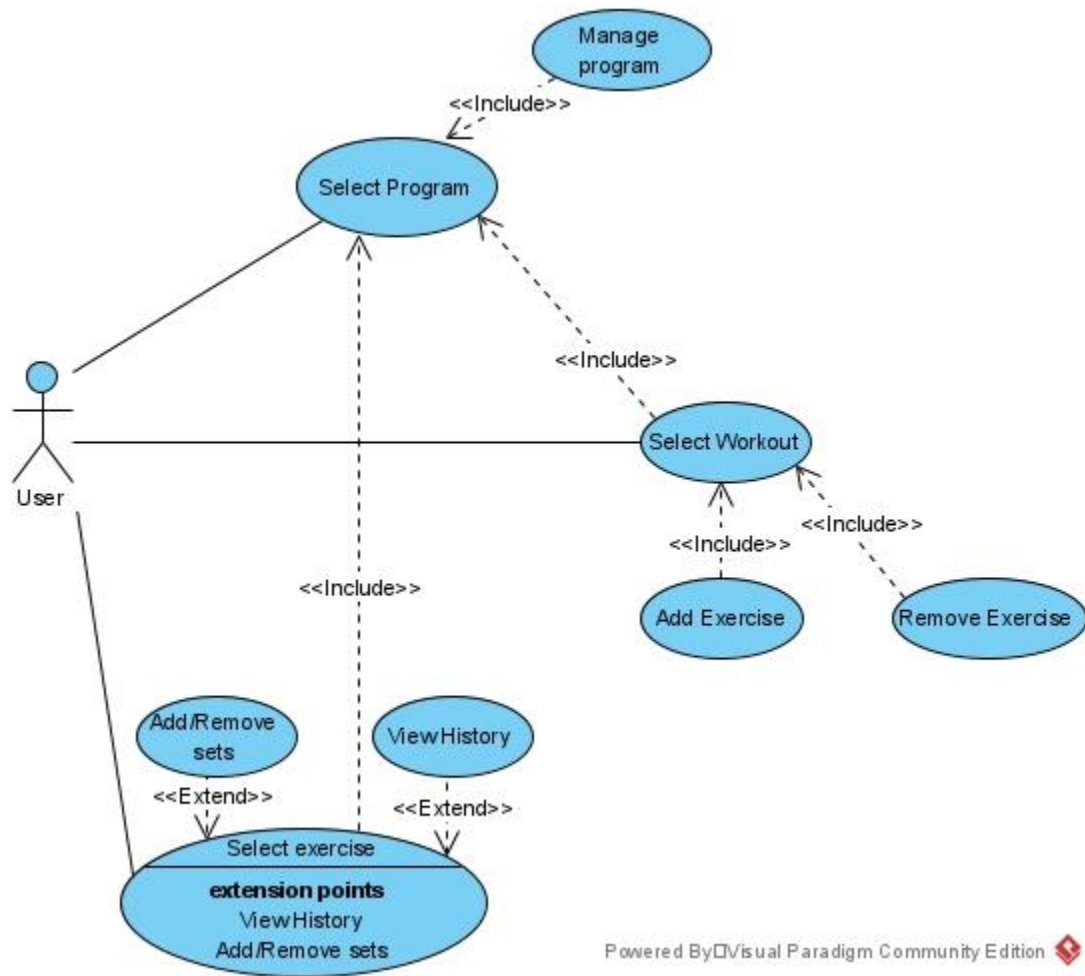
The app is not the first of its kind, there are many options of fitness trackers including:

- [Google Fit](#)
- [RepCount](#)
- [Gym Workout Tracker](#)

The difference between my application and the mentioned apps are:

- my app focuses more on the users' workouts
- it offers illustrations of over 300 exercises
- it focuses on one niche: strength training, which in return offers an unbloated experience

# Functionality

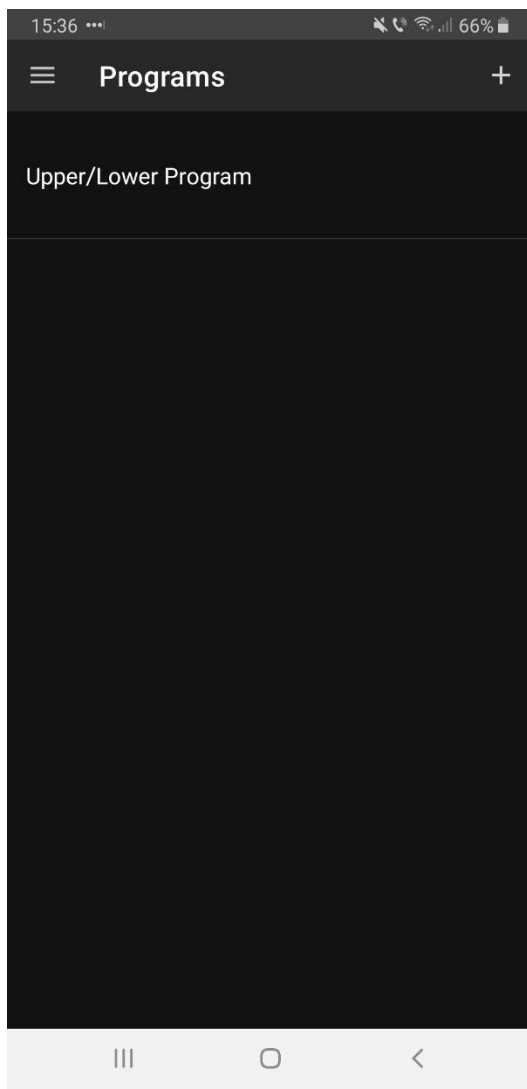


# User guide

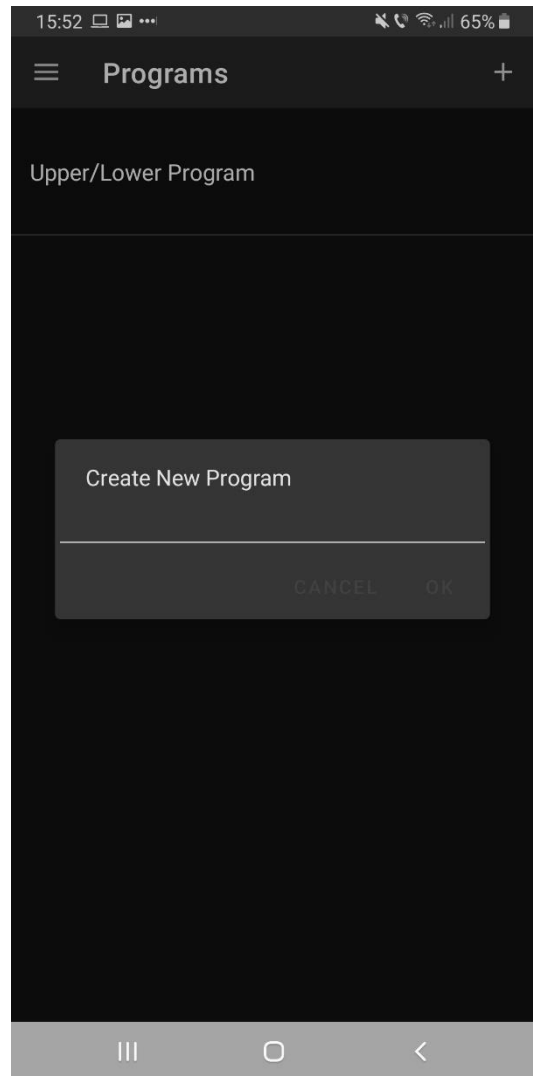
The app provides a prepopulated database with over 300 exercises from which you can choose from and incorporate in your workouts. As specified in the fragments description, you can add your own exercises, programs and workouts.

## Program fragment

All programs will be shown here. A new program can be added from the + button on the top right corner. Accessing the workout programs can be done by tapping on the program.



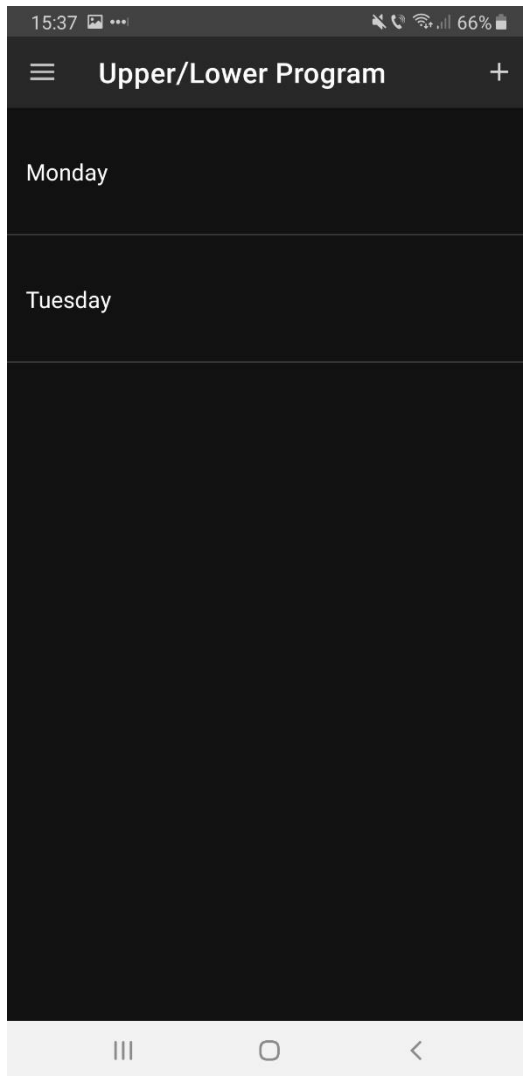
(main program fragment)



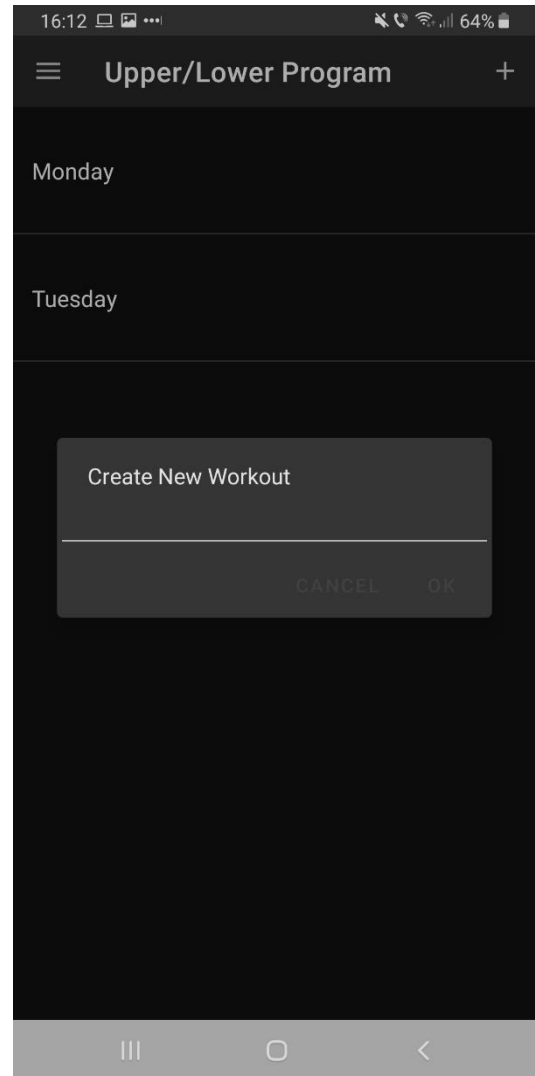
(after pressing + button)

## Workout programs fragment

All workout programs from the selected program will be shown here. A new workout program can be added by tapping on the + from the top right corner. To access the exercises from the workout, tap on the desired workout.



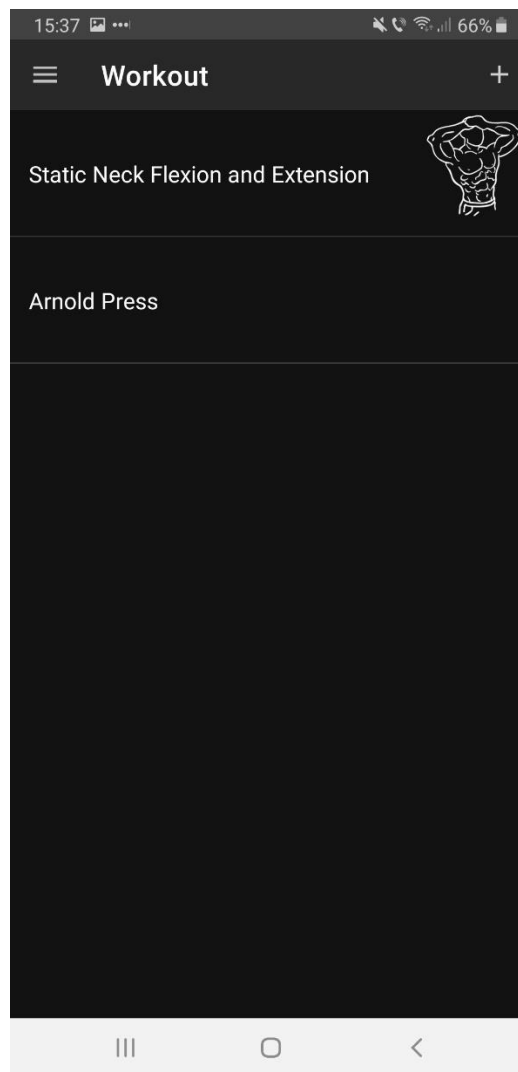
(main fragment)



(after pressing on the +)

## Workout exercise fragment

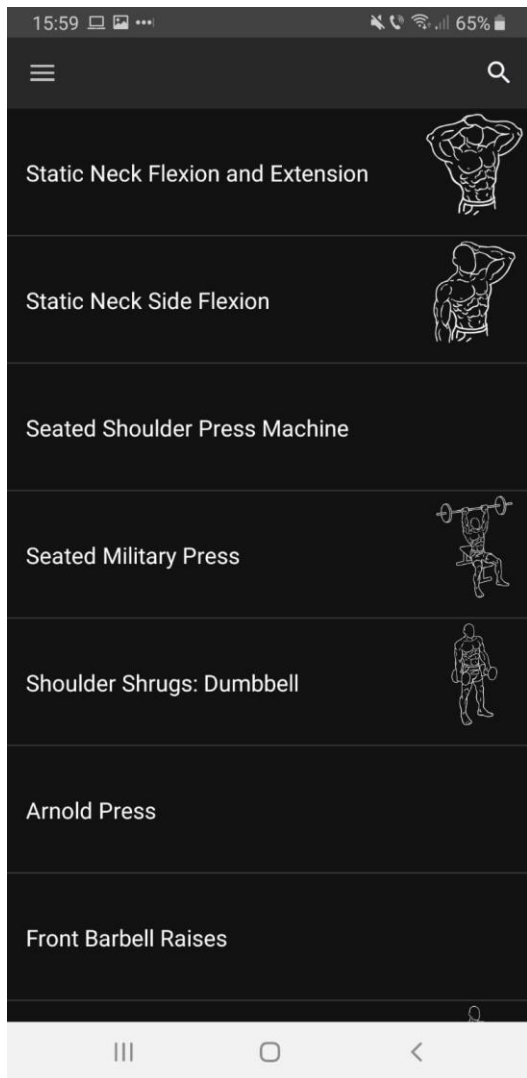
All exercises from the clicked workout will be shown here and new exercises can be added to it from the + button.



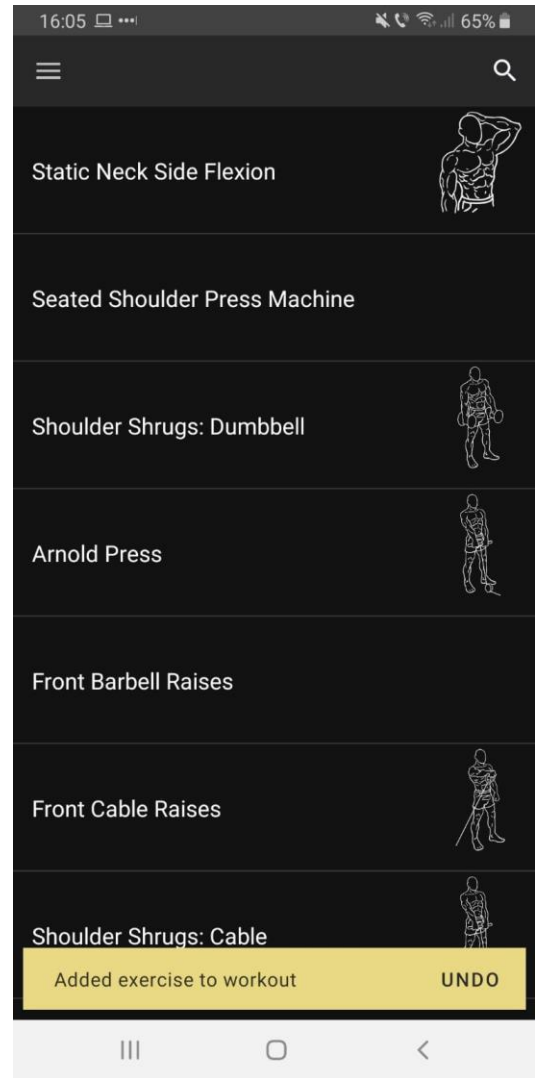
### Add exercise to workout program fragment

This is what is shown after clicking on the + button from the Workout exercise fragment. You can add an exercise to the workout by sliding left on it. The list can be filtered by name by clicking on the magnifying glass from the top right corner.





(main fragment)

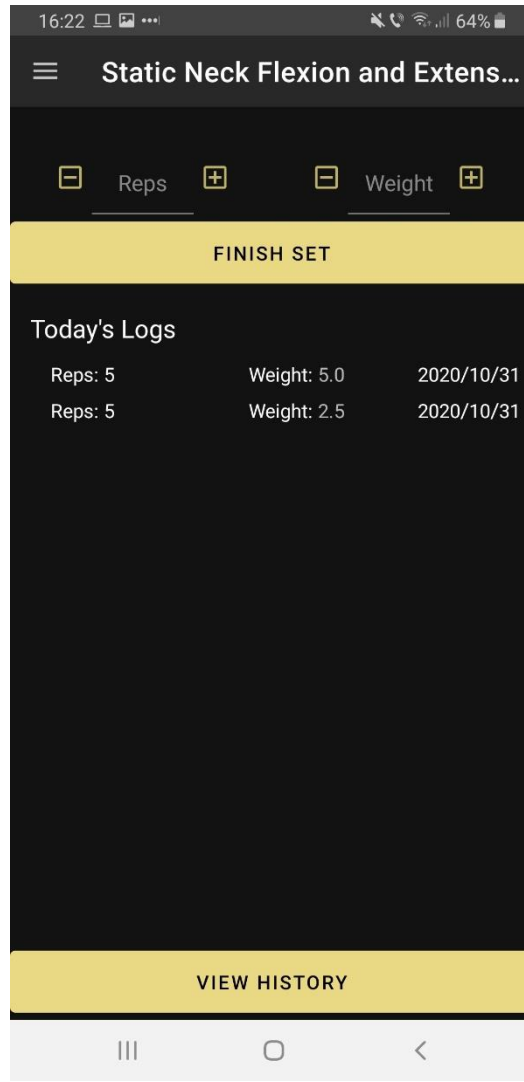


(what is shown after adding  
and exercise, you can undo  
the action by tapping on  
undo)

## Exercise track fragment

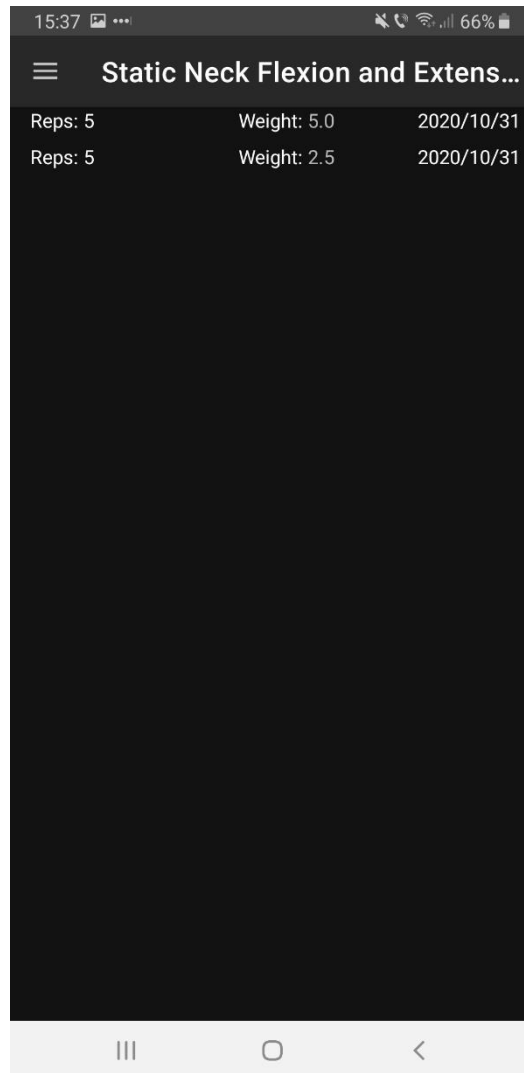
After tapping on an exercise from the workout exercise fragment, you will be provided with a view in which you can do the tracking on the selected exercise. Specify the reps and weight by tapping on the reps and weight dialogs and input the value. The minus and plus buttons can be used to input the reps/weight more easily. Press on “finish set” after you finished typing in the reps and weight.

Press on the “view history” button to access the sets from previous sessions.



## View exercise history fragment

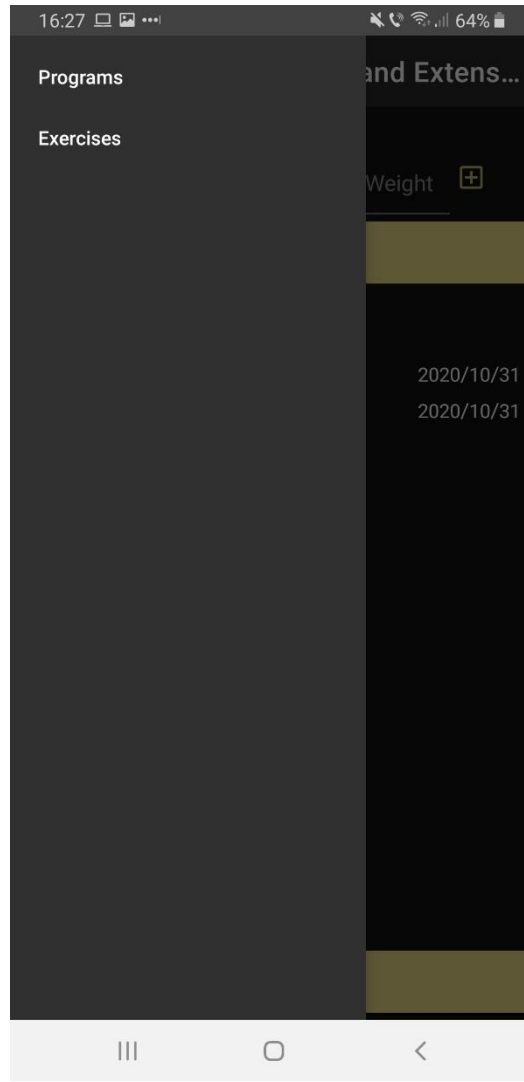
All sets from previous sessions will be shown here.



Static Neck Flexion and Extens...		
Reps: 5	Weight: 5.0	2020/10/31
Reps: 5	Weight: 2.5	2020/10/31

## Sidebar menu

You can also access the sidebar menu by sliding right on any fragment.



## App's structure (technical manual)

This section proposes to explain the design of the activities/fragments and the structure of the database.

The application uses [Room](#) to save data to the database and follows the MVVM (Model–view–viewmodel) paradigm.

The RoomMyDatabase file initializes the database and on first startup it populates it with the exercises and programs from a JSON file.

```
public static RoomMyDatabase getDatabase(final Context context) {
    if (INSTANCE == null) {
        synchronized (RoomMyDatabase.class) {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                    RoomMyDatabase.class, name: "room_database")
                    .addCallback(onOpen(db) -> {
                        super.onOpen(db);
                        SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);
                        boolean firstStart = preferences.getBoolean( key: "firstStart", defValue: true);
                        databaseWriteExecutor.execute() -> {
                            if (firstStart) {
                                populateDB(context);
                                SharedPreferences.Editor editor = preferences.edit();
                                editor.putBoolean("firstStart", false);
                                editor.apply();
                            }
                        });
            }
        }
    }
    return INSTANCE;
}
```

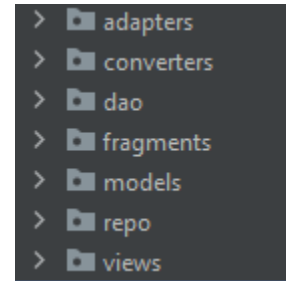
The app consists of 6 models:

- Exercise -> represents the exercises table
- ExerciseProgramsMany -> many-to-many table for exercises and programs
- ExerciseReps -> table for rep counter for each exercise
- Programs -> the programs table

- ProgramsWithExercises -> table with exercises inside programs
- SubPrograms -> the workouts table

Each model has its own:

- DAO -> all queries are defined here
- Repository -> from here we basically call the DAO and prepare the data for the viewmodel
- adapter -> bridge between the data and the view
- view -> bridge between the UI and repository



There are 6 fragments:

- ExercisesFragment -> a RelativeLayout with a RecyclerView displaying the exercises
- ProgramsFragment -> reuses the ExerciseFragment layout, displays the programs
- ProgramsExercisesFragment -> reuses the ExerciseFragment layout, displays the workout
- AddExercisesToSubprogramFragment -> reuses the ExerciseFragment layout, displays a list with all the exercises

which need to be slid in order to add the exercise to the workout

```
ItemTouchHelper.SimpleCallback<ItemTouchHelperCallback> = new ItemTouchHelper.SimpleCallback<>(dragDirs: 0, swipeDirs: ItemTouchHelper.RIGHT | ItemTouchHelper.LEFT) {  
    @Override  
    public void onChildDraw(Canvas c, RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder, float dx, float dy, int actionState, boolean isCurrentlyActive)  
    {  
        if (actionState == ItemTouchHelper.ACTION_STATE_SWIPE) {  
            // Get RecyclerView item from the ViewHolder  
            View itemView = viewHolder.itemView;  
  
            Paint p = new Paint();  
            Paint p2 = new Paint();  
            Drawable icon;  
            p.setColor(Color.parseColor("colorString: "#E9D985"));  
            p2.setColor(Color.parseColor("colorString: "#FFFFFF"));  
            if (dx > 0) {  
                /* Set your color for positive displacement */  
                icon = ContextCompat.getDrawable(getContext(), R.drawable.add);  
                // Draw Rect with varying right side, equal to displacement dx  
                c.drawRect((float) itemView.getLeft(), (float) itemView.getTop(), dx,  
                    (float) itemView.getBottom(), p);  
                icon.setBounds((float) itemView.getRight() - convertDpToPx(32) - icon.getIntrinsicWidth(), (float) itemView.getTop() + ((itemView.getBottom() - itemView  
                icon.draw(c);  
            } else {  
                /* Set your color for negative displacement */  
                icon = ContextCompat.getDrawable(getContext(), R.drawable.add);  
  
                // Draw Rect with varying left side, equal to the item's right side plus negative displacement dx  
                c.drawRect((float) itemView.getRight() + dx, (float) itemView.getTop(),  
                    (float) itemView.getRight(), (float) itemView.getBottom(), p);  
                icon.setBounds((float) itemView.getRight() - convertDpToPx(32) - icon.getIntrinsicWidth(), (float) itemView.getTop() + ((itemView.getBottom() - itemView  
                icon.draw(c);  
            }  
            super.onChildDraw(c, recyclerView, viewHolder, dx, dy, actionState, isCurrentlyActive);  
        }  
    }  
}
```

- SetsRepsFragment -> uses in\_exercise\_fragment layout, displays the fragment containing the sets/ reps of a particular exercise;
- SubProgramsFragment -> displays all the exercises of a specific workout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/backgroundListColor"
    app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli...">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fastScrollEnabled="true"
        android:layout_alignParentTop="true"
        android:persistentDrawingCache="scrolling"
        android:minHeight="100dp"
        android:scrollbarSize="4dp"
    />

</RelativeLayout>
```

(the exercise\_fragment layout)



## Conclusions and future work

I enjoyed learning about the Room ORM and how it binds together with Android. In the future, I'd like to add a countdown timer and a strength meter.

## References

<https://developer.android.com/training/data-storage/room>

[https://en.wikipedia.org/wiki/Progressive\\_overload](https://en.wikipedia.org/wiki/Progressive_overload)