System design document for SvettIT

Adam Törnkvist, Isak Magnusson, Maria Fornmark, Mathias Lammers, Viktor Fredholm.

25-10-2019

1 Introduction

The purpose of this System Design Document is to explain the design of the SvettIT training application for Android and the design choices made for development. SvettIT is intended to replace traditional training applications by having similar statistics to more advanced training applications, and at the same time allow the user to plan and structure their months and workouts on a more detailed level than a normal calendar would do. The System Design Document was made to ensure that all the developers have the same vision and thoughts about the design and implementation of SvettIT.

1.1 Definitions, acronyms, and abbreviations

- Session
  A session is a period devoted to a specific activity. In this document a session is seen as a training session, where a period of time is devoted to training.
- Exercise
  A session is built up of exercises, and an exercise consists of intervals or sessions and repetitions depending on if it is a strength or cardio exercise.
- StrengthExercise (StrEx):
  For example biceps, legs or other workouts to gain strength.
- CardioExercise(CarEx):
  For example running, walking or intervals.
- UpcomingView
  The view where the user can see upcoming workout sessions.
- CalendarView
  The view were the calendar is seen.
- ProgressView
  The view where goals and progress is shown.
- AddGoal
  The view where a goal is added
- CreateSession
  The view where a session is created
- CurrentSession
  The view shown when a session is selected, showing the details of that session.

2 System architecture

2.1 Top level description

This Android application consists of two major parts, one being the planning and the other being the statistics. The planning of the application consists of the construction of new activities and that they will show up in the calendarview and the UpcomingSessionsView. The calendar is just like a normal calendar and it shows the planned workouts. The upcoming sessions view is where the user can see all the next set of planned activities in order, based on what date they are set on, newest first.

The statistics part of the application shows the progress towards goals and statistics of earlier workouts. The goals are self selected and there will appear a progress bar on this page where the user can see where they were when the goal started, what the total goal is and how close they are to reaching their goal, both in percentage in a percentage-bar and also in numerals. The statistics are in the form of graphs, in these graph the user sees the accomplishments from previous workouts, and it can show generic things like ‘exercise time’ which could be a graph, only going up with each session. The statistics can also show progression of maximum strength in a specific exercise, for example. ‘squats’. There will also be percentages like ‘total distance run’ which sums up all total runs for the last year and month.

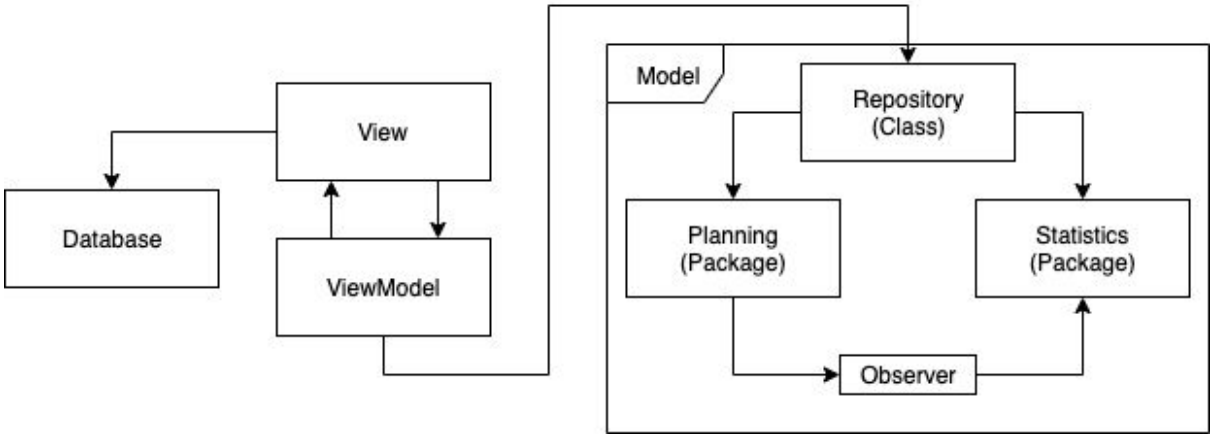

Figure 1: Overview of the packages in the application and how they communicate

Figure 1 shows the connection between the packages, it is not an exact representation like a UML with arrows like ‘Implements’ and ‘has a’ and similar. Here we can see that the Repository handles the Planning and Statistics separately. Since the Statistics needs data to show statistics from there is an observer that ‘transports’ the data from Planning when a session is done.

The data from the user activity is stored in a database, which allows the data to be accessed even after the application has been shut down. Session related data is inserted, edited or removed when creating, editing or deleting sessions. The database also stores all the goals created by the user. When the application starts it fetches all the information and recreates the objects.

2.2 Application flow

When the application launches, the user is presented with a view of the upcoming sessions for the following days, these are sorted by date with the soonest at the top. From this view, it is possible to create a new session or navigate to either the calendar view or to the results page. When the user chooses to create a session, a new view is opened where the user can create a new session and add all the exercises they want to. If a user selects an already existing Session, either from the calendar or the UpcomingView, it takes them to CurrentSession, where they can edit their session however they want.

In the calendar page the user sees a month view of a calendar, if the user selects a date in the calendar which contains one or more activities, they will appear underneath. If a date is selected when the user presses the create session button they will get to the CreateSessionView with the date already selected.

In the results page the user will see graphs and progressions towards goals, as long as the user have got any goals and/or finished sessions.

3 System design

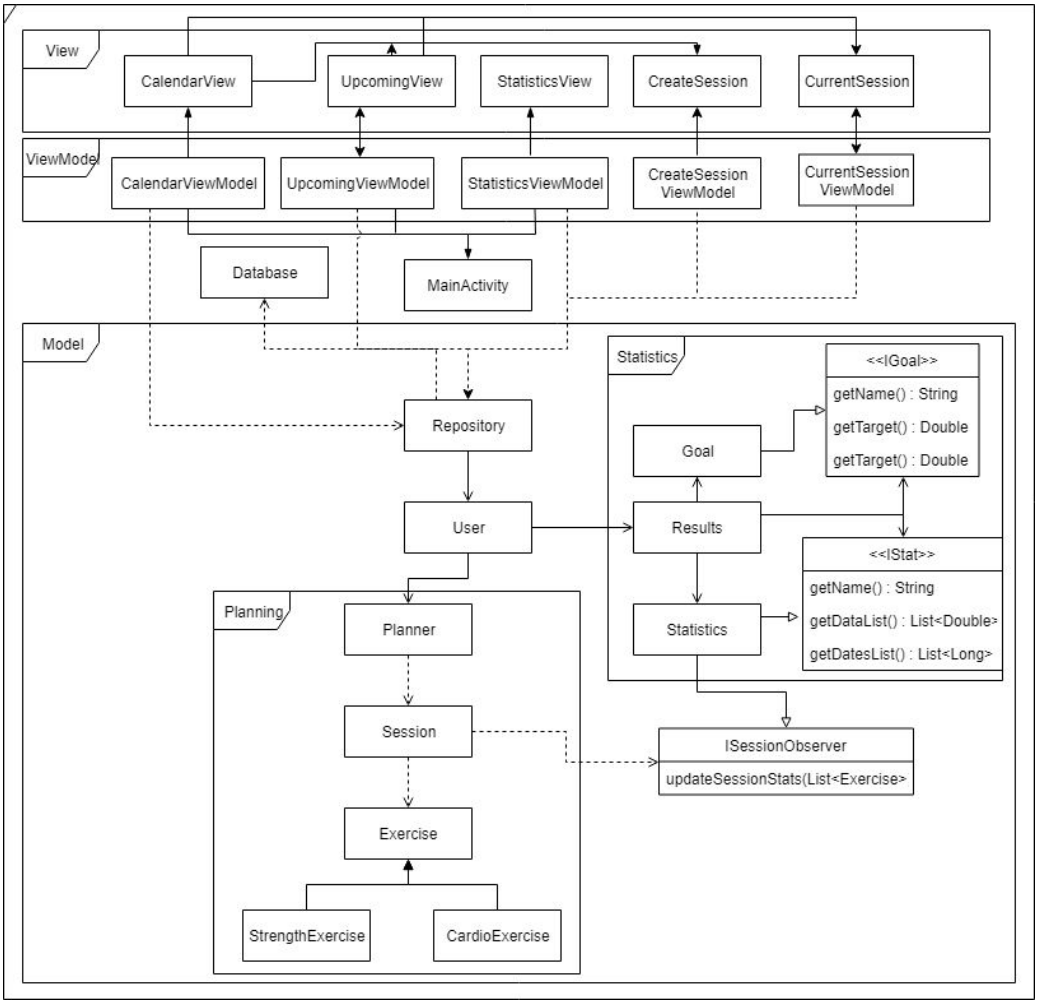

Figure 2: Design model

3.1 Design model

Figure 2 shows the design model of the application. The model package is directly related to the domain model. The user has results, a planner and a set of routines. The routines consist of collections of exercises that the user has saved for reuse later. The planner has a list of sessions and the sessions have the exercises. There are two types of exercises, cardio and strength. The information from the sessions is collected by statistics when the user marks a session as finished.

3.2 Design patterns

The architectural design pattern chosen for this application is the Model-View-ViewModel (MVVM) pattern. This is one of the most widely used architectural patterns for developing in Android, because of its development flexibility, logic separation and ease of testing, which were the main reasons for us choosing it over other popular architectural patterns such as

the Model-View-Presenter (MVP) pattern or the Model-View-Controller (MVC) pattern. By using this pattern, we can separate the logic of the application within the Model, whilst allowing the user to interact with the View, displaying all the information. The ViewModel is then used for providing streams of data relevant to the View.

As opposed to conventional Java development, in Android it is more difficult to pass objects between different parts of the application. As such, in order to access the model from different activities, we decided to use a Singleton pattern by letting a repository hold the model. With the help of a static class RepositoryHolder, the model can be accessed by the ViewModel classes. The Singleton pattern is normally recommended to be used with caution, but after thorough research and consulting with our supervisors, we found this pattern to be the best solution for our application.

To transfer the data from a session to statistics an Observer was used that is notified when the user marks a session as done. This helps to decrease the dependency of the statistics to the current active sessions but still lets the data go from a session when it’s done and finished to the corresponding statistic.

4 Persistent data management

The application uses a simple database to store all sessions, exercises and goals. It has the following tables: A SessionTable, a CardioExerciseTable, a StrengthExerciseTable and a GoalTable. The SessionTable has the following columns: an ID, a text name, a date, an image and an integer used as boolean for checking if a session is finished. Both ExerciseTables have the following columns: an ID (which is automatically incrementing and primary key), a SessionID which is used for the exercise to find its Session, and a name. The differences are that the CardioExercise also stores a distance and time, while StrengthExercise stores sets, reps and weight. The GoalTable has the following columns: an ID, a name and a target.

Each time the application is started, it recreates all objects with the data from the database. When you save a Session, the Session and its exercises are stored. When you edit, delete or mark a session as finished all sessions and exercises are removed in the database and then read with the new information from the stored objects. The goals have no relation to the other tables.

Despite Exercises not being allowed to exist without belonging to a Session the ExerciseTables’ SessionID does not reference the Sessions’ ID. Values that are not allowed to be NULL in the application can be stored as such in the Database. For example, the user will be prompted to enter an ExerciseName if he leaves the field empty when creating an exercise (and nothing gets saved), while it could be stored as NULL in the database if no such prevention was enforced.

5 Quality

5.1 Testing and continuous integration

There are JUnit tests in a separate folder in the project. Code coverage analysis has been used to check what parts of the code is tested. It showed that all of the classes in the model are tested and all relevant methods covered. The focus has been on testing the components of the model.

Travis has been used for continuous integration. It has been used to automatically build the project and run the tests before accepting a pull request. This is done automatically in github. The link to the repository in Travis is: (https://travis-ci.org/DecibelM/OOP_TDA367)

5.2 Known issues

The SvettIT project have several constraints that limits the design of the application. These are mostly beyond the project which is SvettIT but still have to be accounted for.

- The time available for developing the application has been restricted, which has made the design implementation for some features more basic than we would have wished for.

- Limited knowledge within the development team will make more advanced functionality harder to implement. This means that we will not be able to have a database to backup multiple users history, which in turn makes it so SvettIT will be designed in a way that every instance of SvettIT corresponds to a single user. A redeeming fact is that most users normally only need one profile per device.

- We want the navigation and user experience to be easy to understand and fast to do things. This forces our menus to cut down on functionality and customisation in order to keep them smaller and less complex, which will aid in faster and simpler navigation.

- At the moment, cardio exercises doesn't have a way of specifying the unit for distance. For some exercises, for example sprinting, the user may be more likely to measure the distance in hundreds of meters, whilst other exercises would be more likely to be measured in kilometres.

- For the convenience of the user, originally the plan was for a Session to be created with the use of one or several saved Routines, which in turn would consist of one or several Exercises. As such, the user wouldn't have to fill in every Exercise for each Session. After facing some difficulties with implementation, resulting in development delays, it was decided for Sessions to simply consist of Exercises, to prioritize the implementation of other functionality within the application instead.

- When changing or deleting a Session or Exercise, the Database does not know which object has been affected, or which of its rows that needs to be changed. To solve this problem the database wipes out all exercise and session rows and recreates them with the new info from the stored objects (after they have been changed). Would something go wrong such as the application shutting down while this is happening, a lot of data would be lost. It could also lead to the application becoming slow if there is a lot of data. This could be avoided by each Session object and Exercise object having a unique ID which corresponds to the ID in the database, and that way you could find the relevant row without affecting the others.

- The implementation of removing an exercise when creating or editing a session is poorly implemented. As is right now, it just sets the exercise name to “a” and sets all the variables to -1 when destroying the fragment displaying the exercise. It then checks if the exercise’s time or sets is less than 0 (which is not valid user input). If so it renames the exercise to “REMOVE ME”. It is then added to the list of exercises to be added to the Session. But right after each exercise is added, it checks if the ExerciseName is “REMOVE ME”, and filters those out. The proper solution would be that a destroyed fragment is gone and disregarded right away, never becoming an exercise, let alone put in the ExerciseList.

- The separation between View and ViewModel classes is not perfect at all times. This is due to lack of knowledge in Android development. One example is in the CreateSessionActivity where a lot of the logistics and dependencies of the model could be moved to the CreateSessionViewModel to decrease the size and complexity of the CreateSessionActivity.

- When doing a completely new exercise, goals can not be made until the user has completed the exercise at least once. This means that the first time a user saves an exercise it will not be able to count towards a goal progress.This is because the name of the exercises are set by user input and there is no real way of knowing what the user is going to create. A solution could be done with more time by simply adding the name of the exercise as a text field and suggesting it when the user creates an exercise.

- The application crashes when time, distance or weight is ended with a dot. This is bad incase the user would forget to add something after a dot or accidentally add the dot itself. The fix could be done by simply checking if there is a dot at the end of one of those fields and if so adding a zero in the end.

- If you add a new goal for an exercise that already has a goal, the old goal is not updated anymore. With more time this could be fixed so that the goal creation checks if there already is a goal for the same exercise.

- You get statistics for all exercises, it is not possible to choose. A more optimal way to handle the statistics would be by sorting the Exercises or adding a way to

ignore/generate exercise specific statistics.

- You need to update the goals view for a new goal to be shown. To fix this we would need to update the fragment which shows the goals when a goal is created.

- Goals can not be deleted or edited, neither as objects or from the database. This is something that could be implemented with more time and something that would be beneficial for an application like this.

6 References

N/A