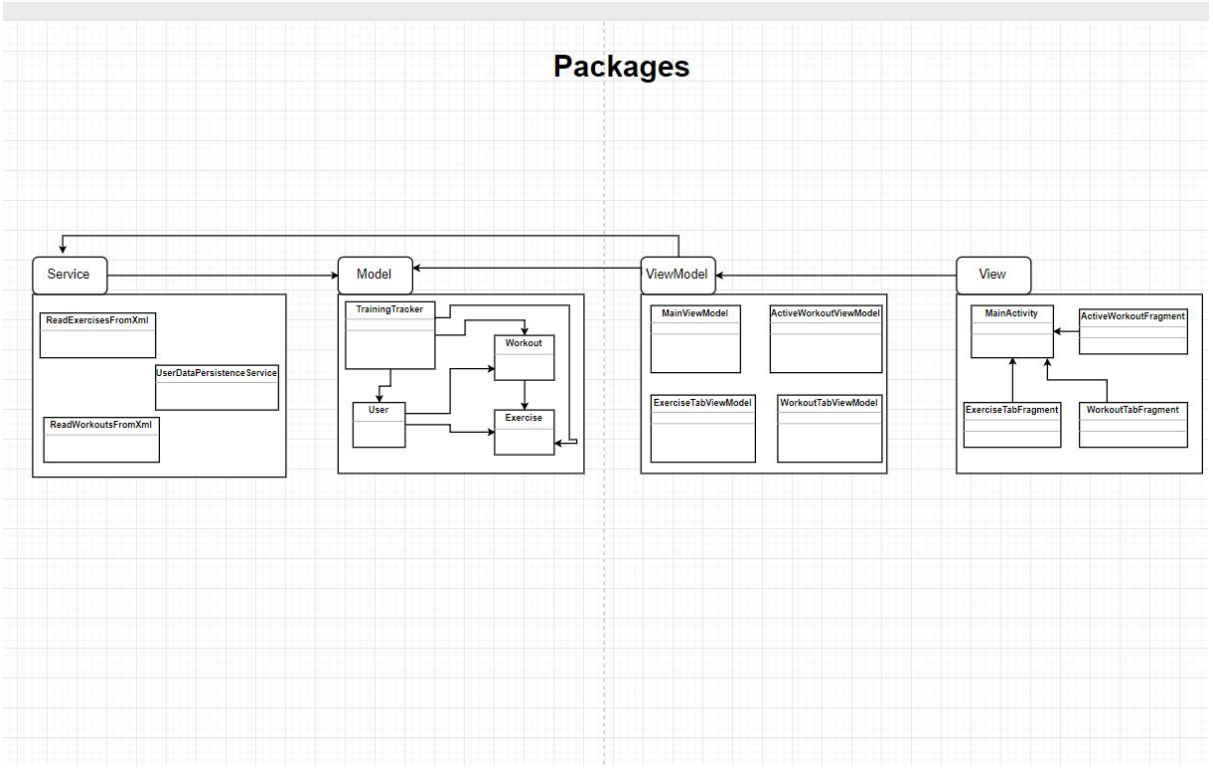
System Design Document for TrainingTracker

1. Introduction

This document will provide documentation on the System Design for the application “TrainingTracker”, an application aimed at enhancing and simplifying a user’s experience at the gym. The Application is an android app and has been developed using Android Studio IDE.

2. System architecture



The above UML diagram shows the different packages within the application and the dependencies between them. The application has been built with the design pattern MVVM (model, view, viewmodel) in mind.

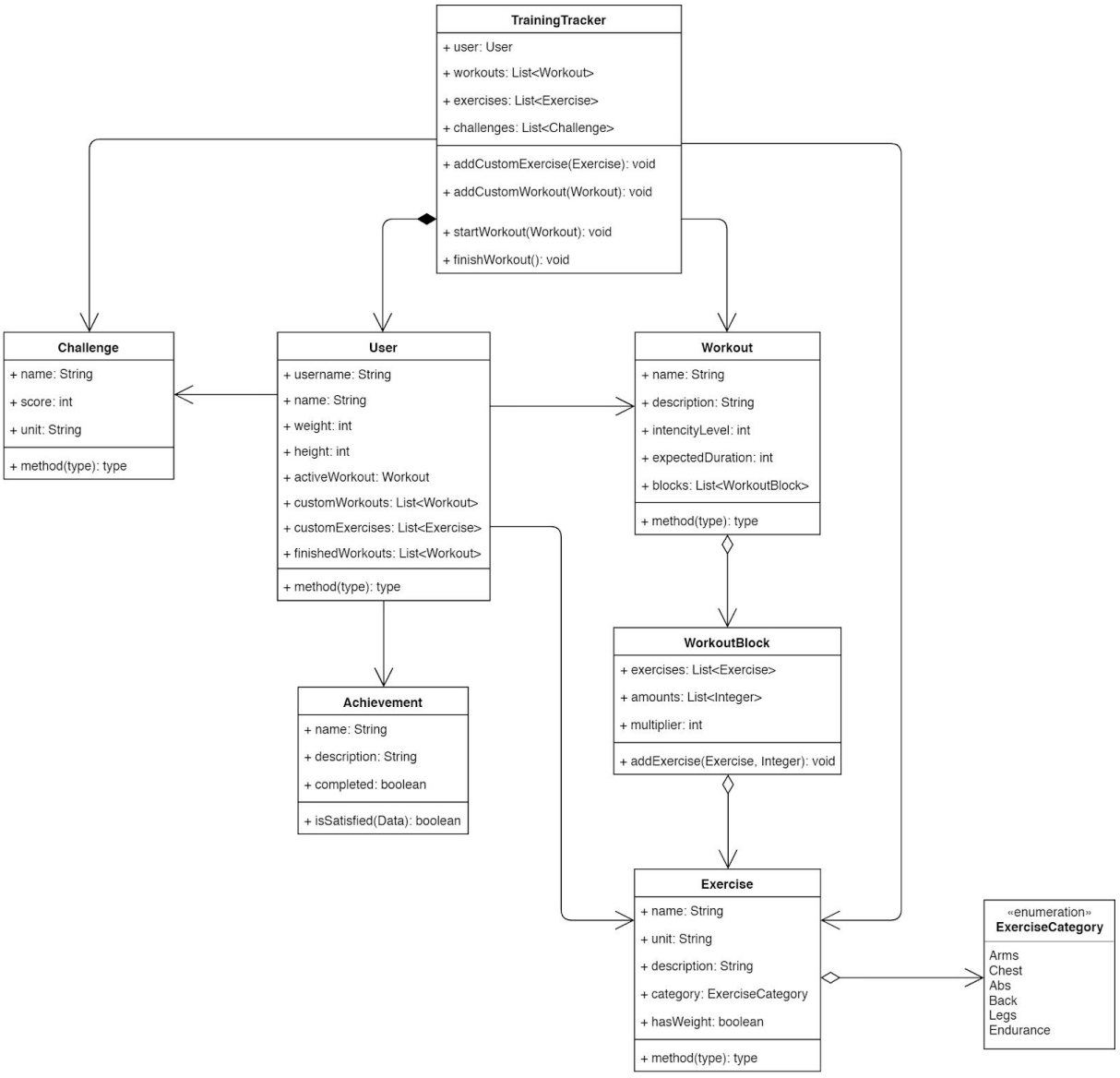
The View-package is the GUI of the application and is what the user will see and interact with. It makes a representation of the data in the model and allows the user to modify and interact with the data. The MainActivity class is the starting point of the application. It instantiates the View and the MainViewModel, and works as a Navigation Manager for the fragments. Each fragment in the view has a reference to an object implementing the

Navigation Manager interface, which in this case is the MainActivity. Navigating from one fragment to another is done through the Navigation Manager, which is responsible for creating a new instance of chosen fragment, assigning it the correct viewmodel and setting it to be displayed.

The MainViewModel loads saved data into the model upon creation, using services, and holds a reference to the current TrainingTracker instance. There are other ViewModels in the package, providing methods for accessing or modifying different parts of the model. All viewmodels are given the reference to TrainingTracker, when initialized. Whenever a fragment needs to access the model it does so via its viewmodel. The ViewModel-package acts as a bridge between the View and the Model, providing controller functions and reducing the direct dependency on the model from the view, making it more modular.

In order to further reduce dependencies on the model, interfaces are being used. Classes in the View-package and the ViewModel-package depend on interfaces of the model, meaning that the classes of the model can be independently changed as long as they fulfil their respective interfaces.

2.1 ‘Model package’



The above is the domain model, showing the classes within the model package(not including interfaces. TrainingTracker is the main class of the model, containing the data used during runtime. It has a reference to the current User and its data, and holds lists of the Workouts, Exercises and Challenges in the app. Through the reference to TrainingTracker, the rest of the model can be accessed and modified.

2.2 ‘View package’

The “View package” contains the visual part of the application, the GUI. It is responsible for displaying the data in the model to the user, while also letting the user modify the data through interaction with the interface.

The main class of this package is the MainActivity, which is responsible for all navigation between fragments. When a fragments needs to switch over to another fragment it does so via the MainActivity.

The view in this application is designed in such a way that each separate screen of the interface is represented by a Fragment. A Fragment can be displayed as a part of the screen or cover the entire screen. It holds the information of how it is to be displayed, and what components it is made of. Each fragment also has a corresponding .xml file which defines the visual elements of the fragment, these elements are then used and manipulated in the fragments class. Each fragment also has a reference to a ViewModel, which it can use to populate its components with the data from the model. When the app switches from one fragment to another it does so either through the bottom navigation bar (and its listener in the MainActivity class), or through the reference to the MainActivity in the current Fragment.

2.3 ‘ViewModel package’

The “ViewModel Package” is responsible for handling the communication between the View and the Model. Whenever the MainActivity or any of the fragments in the View package needs to access or modify data in the Model, it uses the reference to its respective ViewModel object. The ViewModel, in turn, holds an instantiation of the TrainingTracker from the Model package in order to access data and logic in the Model, and contains methods to access and modify the model.

2.4 ‘Service package’

The “Service Package” contains various functionality for importing pre-created exercises and workouts, and functionality for serializing and de-serializing application data, in order to save information between app-sessions.

3. Persistent data management

The premade exercises and workouts are stored in xml format in the res\raw directory, and are loaded during startup by their respective service classes. The user specific data (user variables, custom exercises/workouts, challenges and achievements) are stored using Json, utilizing Google’s implementation of Json, gson. Icons used in the application are stored in Vector format in the res\drawable directory, native to the android application.

4. Access control and security

The application currently does not include different roles for usage. The only available role is the user which has access to all the functionality that the user interface provides. Each user

has the same options as every other user. The app does not support multiple uses on a single instance of the app but the thing that may differ between two users running the app on two different units are the individual exercises and workouts that the users themselves create as well as the individual user data.

5 References