Developer Documentation for Location Simulator

Lars Fockenga

March 14, 2025

Contents

1 Installation

To work with this project, clone the repository. It is recommended to do this manually and not with a tool like Android Studio, as the root of the gradle project is not the very first folder, but instead the android-app folder contained in it.

To now work with the project, open the android-app folder as a new project in Android Studio. Android Studio should now automatically guide you through the rest of the setup and install all dependencies.

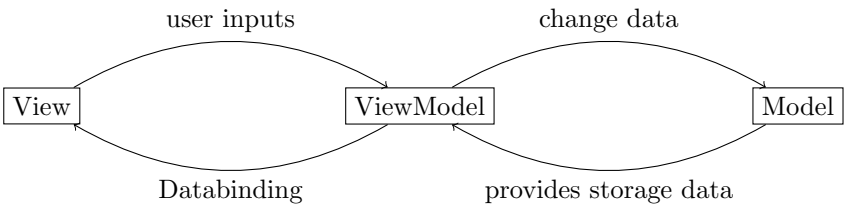

Figure 1: Overview of the MVVM architecture

2 General Information

2.1 Architektur

This app uses the Model View ViewModel (MVVM) architecture, which is a common architecture, splitting the software into three layers:

- **The Model** contains the data, which can be modified by the ViewModel. The data is independent of the other layers, meaning multiple views implementations can use the same model. It might also contain some business logic, like accessing the devices main memory.
- **The ViewModel** takes user inputs from the View and modifies the models data accordingly. It also iffer the view access to the state of the model, so any changes can be reflected in the UI.
- **The View** reacts to changes in the ViewModel and reflects those changes in the UI. It should contain no business logic.

This architecture was used as the apps UI used Jetpack Compose, which is made to work with a ViewModel. The separation of the layers also allows for independent changes between UI and business logic, as well as bester testability, as no complex UI tests are needed to test the business logic.

2.1.1 Data Bindings

For databindings, Jetpack Compose already provides states that take care of updating and recomposing the UI.

2.1.2 Further Resources

This YouTube video further explains the MVVM architecture and was used by the developers of this app to learn it.

3 Project Structure

The following section refer to all the packages found in android-app/app/src/main/java/com/ispgr5/locationsimulator.

3.1 core.util

- **TestTags** Contains all the test tags which can be used to test the app with it's UI. The most common use cases would be to check if a UI element exists, to get it's value or to click on it.

3.2 data

3.2.1 repository

- **ConfigurationRepositoryImpl** Implements all the function that can be called to read configurations from the database, or write them to it.

3.2.2 source

- **ConfigurationDao** Data Access Object for the Database where SQL queries are defined.
- **ConfigurationDatabase** Defines the database which stores the configurations.

3.2.3 storageManager

- **ConfigurationStorageManager** This class is responsible for importing and exporting configurations. It turns sound files into a base64 string, creates a json for the configuration, and compresses them using gzip.
- **SoundStorageManager** This class allows us to use the sound files, which are stored on the devices main storage.

3.3 di

- **AppModule** This data injection module loads the database when starting the app and provides it's interface to the view models.

5 App Workflow