Weather App

Architecture Documentation

SoSe 2024, Mobile Computing

Lunes Hellmich, Matrikelnummer 5201109

Julia Pohl, Matrikelnummer 5194520

Introduction

The WeatherApp informs the user about the current weather and weather forecast. The information on the weather encompasses the current weather conditions as well as hourly forecast data and daily forecast data for the next week. The user can get the weather of their current location or choose a place from a list. They can also add places to that list. The app follows the MVVM (Model-View-ViewModel) architecture pattern.

UI-Layer

Built using Jetpack Compose, the UI-Layer is responsible for rendering views and handling user interactions. The AddPlaceScreen is the starting screen of the app. Here, the user can choose a place from a list of places. They can search for places they want to add to the list. They can also just click on their current location. If the user clicks on their current location or on a place from the list, they are navigated to the PlaceWeatherScreen. This screen displays the weather information corresponding to the place the user chose. Each screen is composed of multiple composable functions, with state managed via ViewModel. ViewModels hold the state, which is observed by composables to update the UI reactively. There is a ViewModel for the weather data and a ViewModel for the current location. The PlaceViewModel is used for the place that the user selected and the autocompletion of places in the search bar.

Data Layer

The Data Layer manages data from various sources, including the OpenWeatherMap API and a local Room database. All of the weather data is received from the OpenWeatherMap API. Since we have to use different URLs for requests for different kinds of data, we created different interfaces and classes for the daily weather data, the hourly weather data, and the current weather data. The OpenWeatherMap API also delivers the geo-coordinates of a place and the place for specific geo-coordinates as well. For the autocompletion of places in the search bar, we call the Here API. For all of these API calls, we created different models. Models are data classes that match the structure of the data of the response from the APIs. We used Gson annotations in the model classes so that the JSON data from the APIs is deserialized into an object of that data class. We used Retrofit and OkHttp for sending our requests to the URLs of the APIs. The Room database, which is responsible for storing places and its interface, can also be found in the data layer.

Key Components and Libraries

Jetpack Libraries we use are Compose, Jetpack Navigation Compose for the navigation between UI screens, and LiveData. We also use the following third-party libraries: Retrofit and OkHttp for sending our requests to the APIs, Room as a database for storing the places the user added, Gson for deserializing the JSON weather data, and Googles Play Services location for getting the current location.