Mobile Sports Applications and Data Mining

Lab 1.2 | Hl1033/CM2001 HT23

Programming & Networking

Aim

During this lab, you will delve into the world of REST APIs and will learn how to seamlessly integrate them into your application. REST APIs are one of the few ways that you can "legally" integrate someone else's work into your work ;)

General

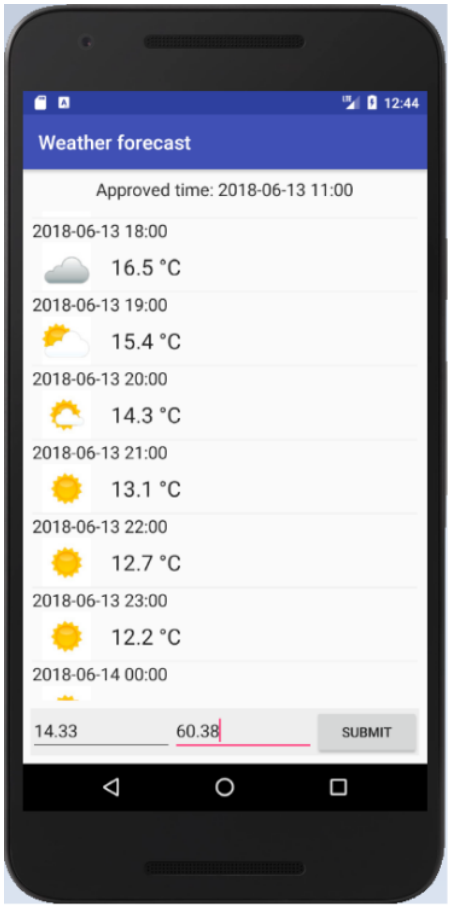
This assignment will be completed in groups of two students who have successfully passed Lab 1.1. To ensure the quality of your work, please adhere to the following guidelines:

- The solution should be organized in a set of appropriate classes.
- Names of classes, methods and fields should describe their usage.
- Structure and document your code effectively. It is recommended to follow the MVVM (Model-View-ViewModel) architecture. While MVC (Model-View-Controller) is also acceptable, keep in mind that applications with "god" classes will be considered insufficient.
- **NB! It is mandatory to submit your source code at least a day prior to your presentation.**

The app

Most of us use some sort of weather forecast almost every day. In this lab, you will have the opportunity to create your own bespoke weather application.
Numerous institutes dedicate their time to forecast weather accurately, and some of these institutes make this data publicly available to external users through APIs (application programming interfaces). By sending a GET request to the appropriate endpoint, we can retrieve this data and use it in our own application.
For this lab, you can use the following APIs to do this:

1. International data: https://open-meteo.com/en/docs
2. Swedish data: https://opendata-download-metfcst.smhi.se/



Both APIs provides you with data in the form of JSONObjects, it is your task to write a parser that can unpack this data, and then display this data to the screen.

Basic requirements (1 point)

Create a simple yet effective application that displays the weather forecast for at least 7 days ahead. The forecast should include the temperature and some visual representation of cloud coverage, such as image, colour, or code (like the image). To generate the forecast, the user will need to input the longitude and latitude of the desired location. More detailed requirements for this version of the application are found in “Detailed requirements”.

Higher level requirements (2 points)

To receive two points for this application, you are required to extend its functionality. Instead of inputting longitude and latitude manually, you should use a second API to convert placenames to coordinates. For this you can use the following APIs:

1. Swedish places: https://www.smhi.se/wpta/backend_solr/autocomplete/search/{place}
2. International places: https://geocode.maps.co/search?q={place}

Furthermore, you should implement at least two of the following points:

- Use the Retrofit library to get and process your API response **[Android only]**.
- Group the received data so that detailed (hourly) data is shown for today, and less detailed (maybe a daily overview) is shown for the other days.
- The ability to mark places as favourite and have a different quick view for these favourites.
- Adding a settings screen in which the user is allowed to define the refresh rate and other preferences regarding the UI.

Important while testing

During the development process, you will probably test your application a lot. This can result in many API calls, which can get you blacklisted. To avoid this, you can test using a private server that we have set up. This server returns a similar JSON response as the SMHI API.

Forecast data: https://maceo.sth.kth.se/weather/forecast?lonLat=lon/14.333/lat/60.383
Coordinates data: https://maceo.sth.kth.se/weather/search?location=Sigfridstorp

NB! Your final solution is not allowed to use the test APIs.

Detailed requirements

The below are mandatory to pass the assignment.

- A separate class or function for parsing the JSON-data.
- Store the parsed data, to create data persistence (through e.g., serialization, Shared Preferences, or a database)
- The forecast data should be displayed in a scrollable list.
- The longitude and latitude inputs should only be allowed to take floats as inputs.
- The user should be able to use the application in both portrait and landscape layout.
- Accessing the data should happen in the background (e.g., custom thread or existing framework).
- You should check if the user has internet, if not, use old data but show the user that there is no internet connection.
- Handle all errors or exceptions appropriately.

Hints

- Use Postman (online) before creating your application to analyse the return of your URL endpoint. This will help you understand what you are parsing.

Android

- When written in Java; Volley is a nice framework to use for the handling of HTTP requests.
- When written in Kotlin; you can use Coroutines or Retrofit for the handling of HTTP requests.
- You can use org.json to parse the JSON. (Alternatively, GSON also works well)
- You will need the android.permission.INTERNET permission in your manifest.
- Use the ConnectivityManager to check internet access (this will require an extra permission)

iOS

- Handle parsing and downloading from network using “threaded” _tasks as described in the lectures.
- Use the built-in functionality to stack your items on the screen to handle rotation.
- Use AFNetworking to check your network status.