COMP.SE.110 Group project

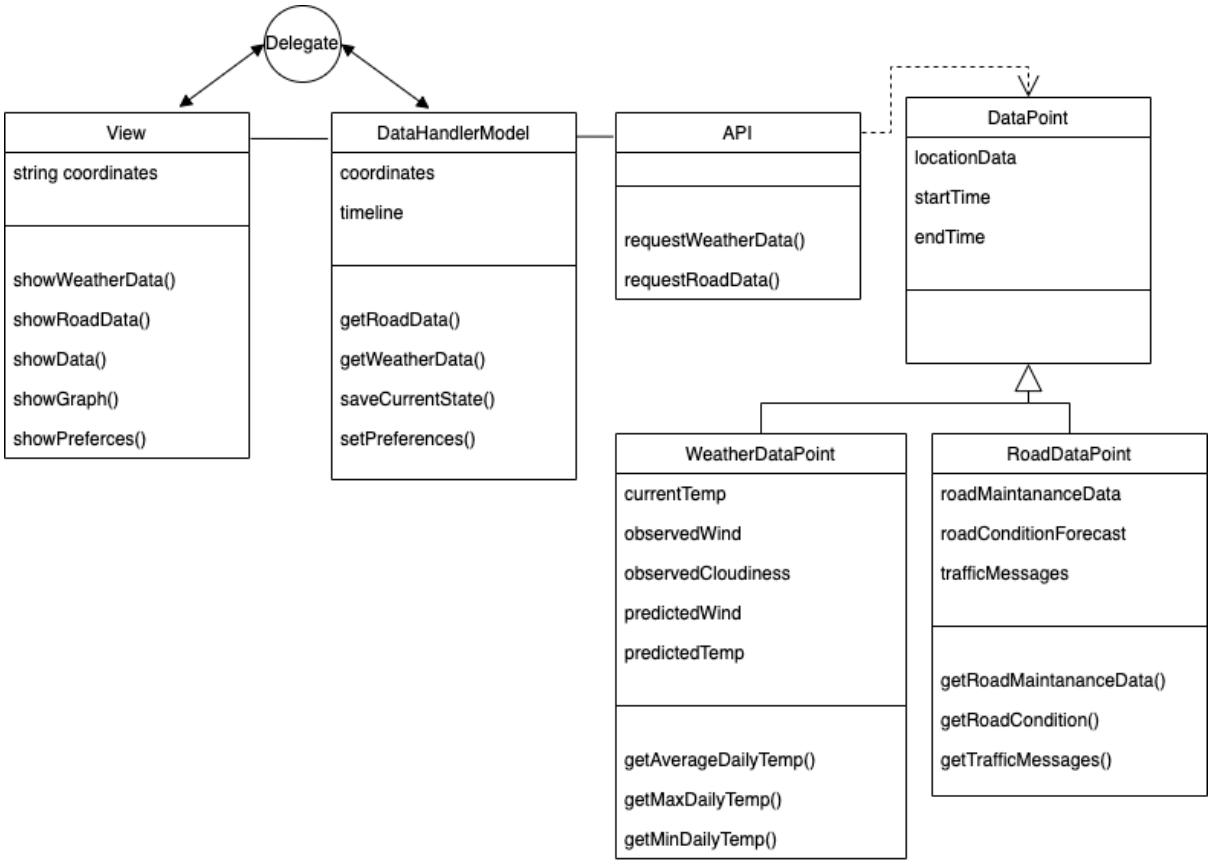
Ville Venetjoki, Ilari Miettunen, Viivi Mustonen ja Kasperi Kouri

The purpose of this program is to present its user with data from the Finnish meteorological institute as well as DigiTraffic in an intuitive and accessible manner. The user must be able to customize what and how the data is shown. The user must also be able to save some preferred views that can be accessed later.

We chose to implement the program using the Python programming language as all the group members had previously used it and because it felt best among the accepted languages. Python has many useful built-in libraries as well as an easy syntax which allows us to spend more time planning than struggling with coding. The drawback with Python for this course is that it doesn't have a direct way to implement interfaces.

The user interface will be implemented using QML. QML follows the Qt approach on event-based programming, which means that control is integrated into the view. This means that as the view gets inputs from the user, it calls the model for required data which gets updated when it is received.

Class diagram:



Due to Python not having an easy way of creating interfaces, our program is going have classes directly interacting between each other. The three main components are going to be the view, the model, and the API. During development some components may be split into smaller ones to avoid having “god classes”. They will still serve the purpose of the same component, but just be split up into smaller files.

The main idea is that the model will contain the actual logic and saved information. When the model needs some data, it doesn’t already have, it’s going to call the API program, which will make a request to either FMI or DigiTraffic according to given parameters. The API will store retrieved information into objects derived from the base class DataPoint and pass them onto the model.

The view will consist of methods that show some visualization of data received from the model. When a button is pressed and a method is called for, it’s going to make a function call towards the model and make changes to the window according to data received from the model.

The model is going to act as the main logic and storage component in the project. It’s going to ask the API for information in the form of DataPoint objects, store and pass necessary data from said objects to the view. It’s going to have get-methods, which the view will use to retrieve certain information. Additionally, the model is going to have some way of saving information in a way that can be accessed later.

The API’s purpose is to make http requests according to parameters received from the model. Received data is parsed and passed on to the model for further processing. The implementation of the API must be multi-purpose since there are many types of requests (forecast, observations, etc.) from both data sources, which return different types of data (XML, JSON).

- Possible third-party libraries
    - python “requests”-library
        - Used because it is much more intuitive and easier to use compared to pythons built in urllib-library
        - “pip install requests” in the python terminal to install
    - Pandas –library
        - Data analysis library will be used to parse XML into easy-to-use Pandas dataframes
    - PyQt5/QML
        - Used to build the user interface
    - Folium or Google maps (Optional)
        - We might try integrating an interactive map into the project to visualize traffic messages and road maintenance messages