Design document

Kalle Gustafsson

Tuomas Mäenpää

Aleksi Suonsivu

COMP.SE.110 Software Design

Table of Contents

1. Introduction

The purpose of this document is to offer a high abstraction view into the group project program. In this document we describe the rough functionality and implementation of our project. We chose to implement the program with Python and the PyQT-framework.

2. Program functionality

2.1 Requirements

Use case 1):

- User can select which monitoring station's (one or more) data they are interested in. - User can select which greenhouse gas / data variable they are interested in (minimum options: CO2, SO2, NOx)
- User can set a time period from which data is shown - In addition to viewing raw data, user can ask for a minimum, maximum and average values of the selected data
  - For a given time period for a given station
  - For several stations for a given time

    - For several stations for a defined time period
- Not all measuring stations provide exact same data. However, the user interface should be user friendly in such a way that the user does not need to do “test-and-try" to see where the data exists. In other words: the application should only allow selecting data that is actually available.
- Error handling: you must efficiently prepare for errors in the data and handle null values.

Use case 2):

- The user can select a time range for viewing data from STATFI
- User selects one or more available datasets: CO2 (in tonnes), CO2 intensity, CO2 indexed, or CO2 intensity indexed (one or more).
- User is shown a visualization of the data

Use case 3):

- User is given a visualization of the CO2, SO2 and NOx values from SMEAR for a time period specified by the user, for the station(s) selected by the user.
- The user can select data available from STATFI on historical averages
- The user can specify the time range (e.g. year 2010, or years 2000-2009) they are interested in.
- The historical values are shown alongside real data in a comparable way.

Use case 4):

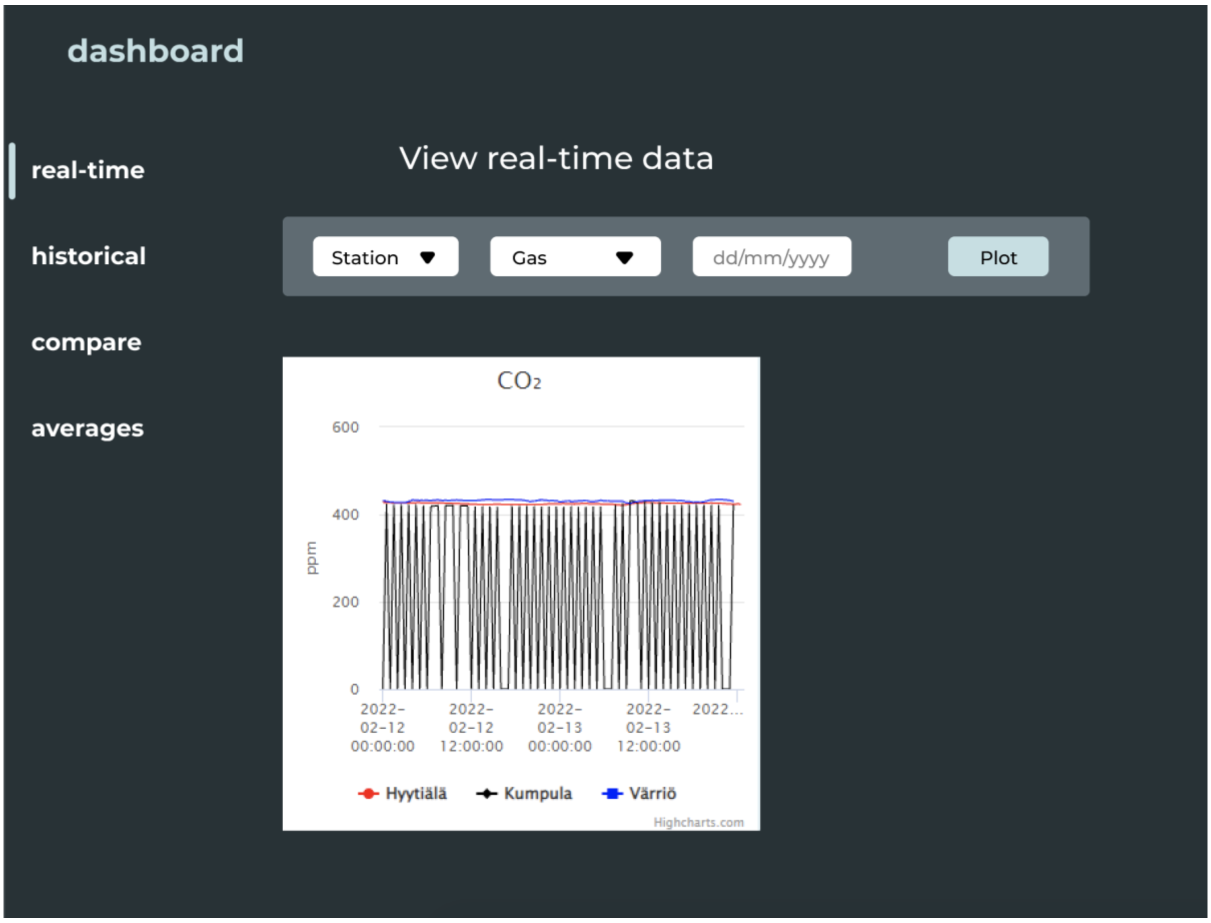
- The user selects a time range for viewing data from STATFI - User selects data from CO2 (in tonnes), CO2 intensity, CO2 indexed, or CO2 intensity indexed.
- User is shown a visualization of the data
- The user is able to choose real time data from SMEAR. NOTE: SMEAR data does not handle the same timespan as STATFI. You will need to check how long back data is retrievable from SMEAR and guide the user accordingly.
- Breakdown of the historical average to the selected year based on SMEAR data is visualized.

General:

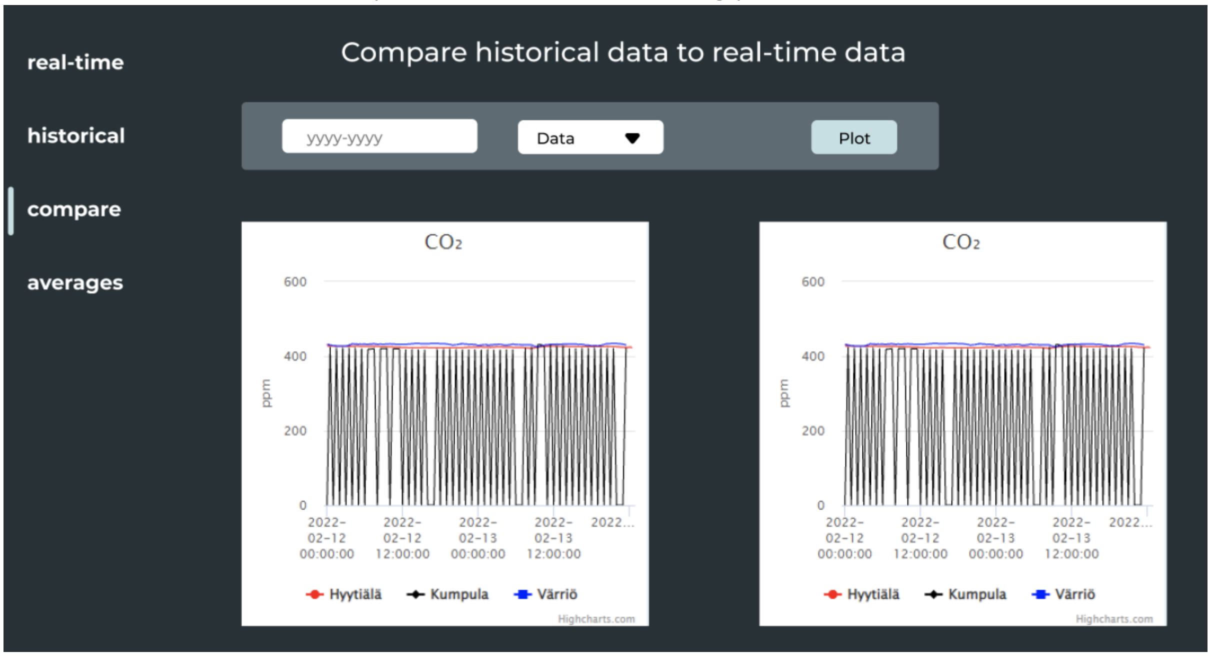
- The user can save preferences (e.g. certain stations, certain time period in history, which greenhouse gas(es) from SMEAR, which statistics from STAT FI) and apply them when using the software later
- Design must be such that additional data sources could easily be added

2.2 UI-prototype

The program opens to a view resembling the following figure. The plot button has been pressed already in this figure but there won’t be a plot shown at start in the final program.



The “historical” view is the same appearance-wise as “real-time” view. In “compare” view there are two charts to be compared to each other followingly.



The “averages” view also resembles “historical” view in terms of appearance.

3. Program structure

3.1 High abstraction description

3.1.1 External Interface

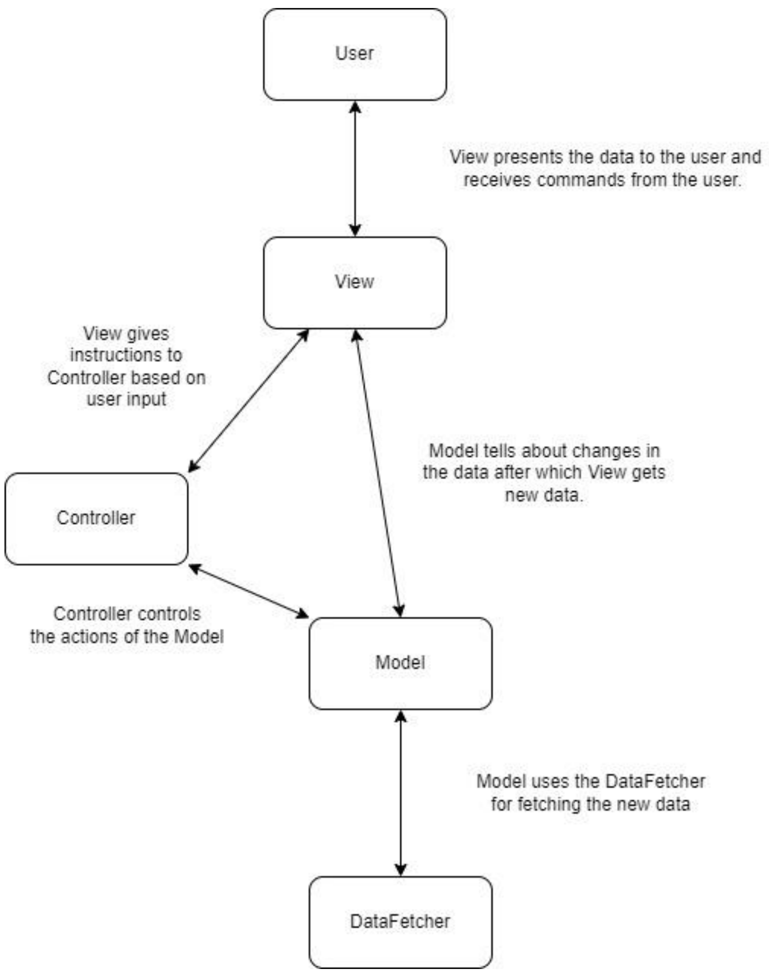
The program reads the data from the database using an external interface. This interface will be implemented inside the DataFetcher class. This class is used to fetch data and forward it to the user in correct form.

3.1.2 Internal Interfaces

The internal interfaces of the program will be implemented using MVC architecture.. (Model, View and Controller)

- Model works as a middleman between the external interface and the internal interfaces. Model handles the data received from the DataFetcher class and forwards it to the other classes.
- View fetches the data from the Model. Model will also contain functions that will be used based on the user input.
- Controller interface will include methods that communicate with View and Model.

4. MVC


User
View presents the data to the user and receives commands from the user.
View
View gives instructions to Controller based on user input
Model tells about changes in the data after which View gets new data.
Controller
Controller controls the actions of the Model
Model
Model uses the DataFetcher for fetching the new data
DataFetcher

4.1 Model

- Receives the data from the DataFetcher and processes the data to correct form.
- Contains the backend logic of the program.
- Sends information about changes in the data to the View class.

Attributes: CO2 flux, CO2, SO2, NOx

Methods: realtime(), historical(), compare(), averages()

4.2 View

- Plots the data.
- Updates the UI based on the user input.

Attributes: gas, station, time

4.3 Controller

- Listens to View's instructions.

- Tells the Model to behave based on user input.

- Implements the logic that updates the Model and View in response to the user input

Methods: handleRealtime(), handleHistorical(), handleCompare(), handleAverages()

4.4 DataFetcher

- Fetches data from STATFI and SMEAR API's

Public methods:

- get_realtime(self, start_date, end_date, table_variables, interval, aggregation)
- get_historical(self, years = [], categories = ["Khk_yht", "Khk_yht_index", "Khk_yht_las", "Khk_yht_las_index"])
- get_historical_options(self)
- get_realtime_options(self)

Private methods:

- _init_hist_options(self)
- _init_realtime_options(self)
- _get_stations(self)
- _get_smear_station_tables(self, station_id)
- _get_smear_station_table_variables(self, station_id, table_id)
- _get_historical_limits(self)

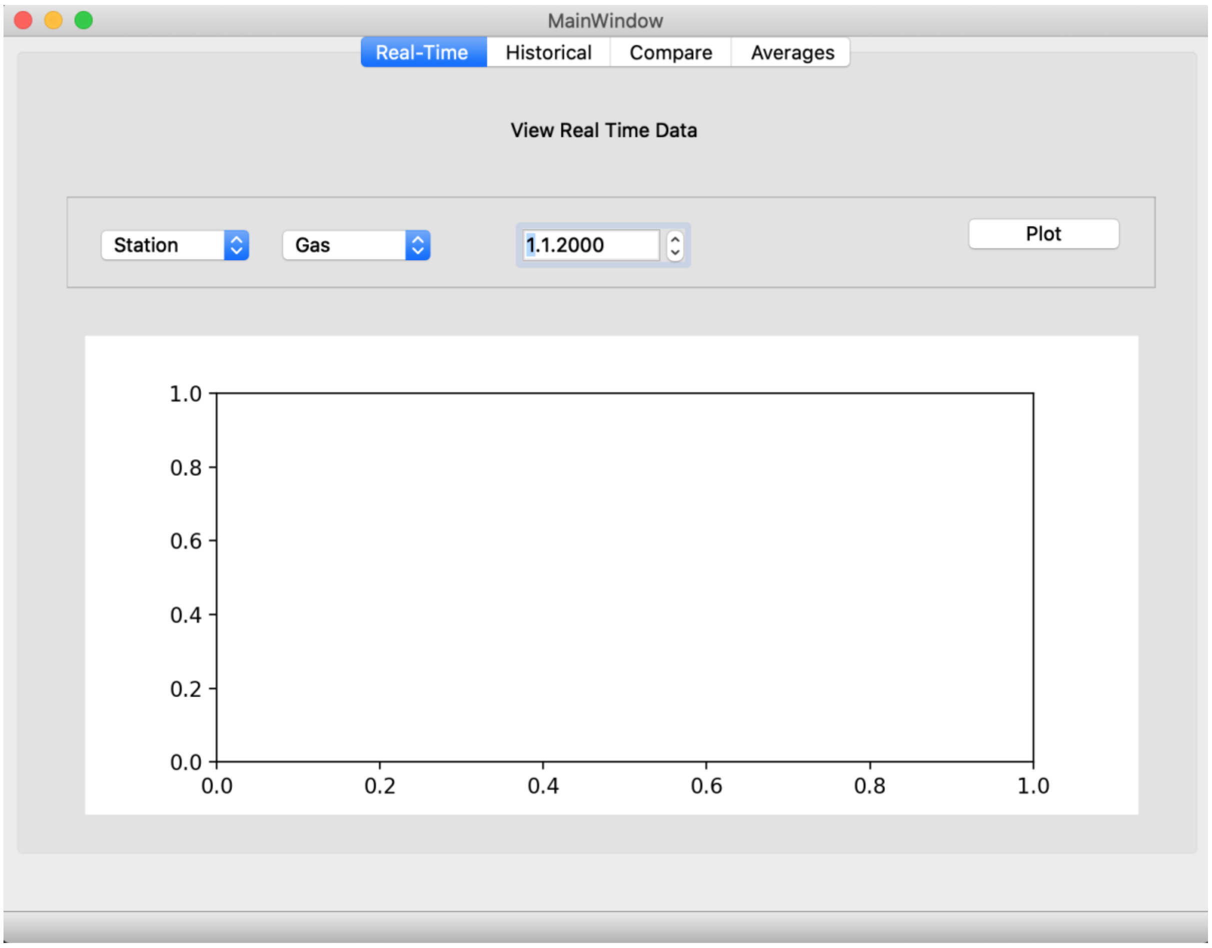
Attributes:

- hist_options
- realtime_options

5. SELF EVALUATION

5.1 UI

The UI design has stayed pretty much the same as in the first draft of the design document. Our UI is divided into a four-tab-window, which can be used to navigate between different views. This tab view makes it easy to implement different features while keeping the UI simple and easy to use. Below is an early draft of the mainwindow.



This version of our mainwindow is a really plain draft of the final UI and the style and layout will change later. At the moment the UI does not plot anything. For plotting we will use either Matplotlib or PyGraph.