21.2.2021

Weather_power_app

COMP.SE.110 Project work

Teemu Mökkönen, Jaakko Raina, Eero Eriksson,
Miska Romppainen

Table of contents

1. Introduction

This document is created in order to give the user specific instructions of how to use the program properly and present methods used to create the software – both the front- and the backend. General idea of the program is to monitor electricity consumption and compare it to the given weather forecast. This is done by fetching power and weather data from an API, presenting it to the user in an interface. User can choose to compare different data from multiple to a specific date – this is completely up-to the end-user. User can create graphs from chosen data. At this point of the development, we are not sincerely certain what we want to include to the program and what we want to present the user, but we are certain we want to present graphs for end-user from data given.

1.2 High level description of the application

Our application will monitor electricity consumption and compare it to weather forecast. Power data is fetched from Fingrid data API and weather data is fetched from FMI open data interfaces. This is done by creating the program a *Logic*-class for the application in which all the calculations and data fetching is done. By using &QNetworkAccessManager we allow the application to send requests and receive replies. This data is then presented to the user in the UI (user-interface) main window. At this point of the development, we are still working on creating the user a simple and "easy-to-use"-interface, which we can then update and expand if necessary - without compromising the user experience.
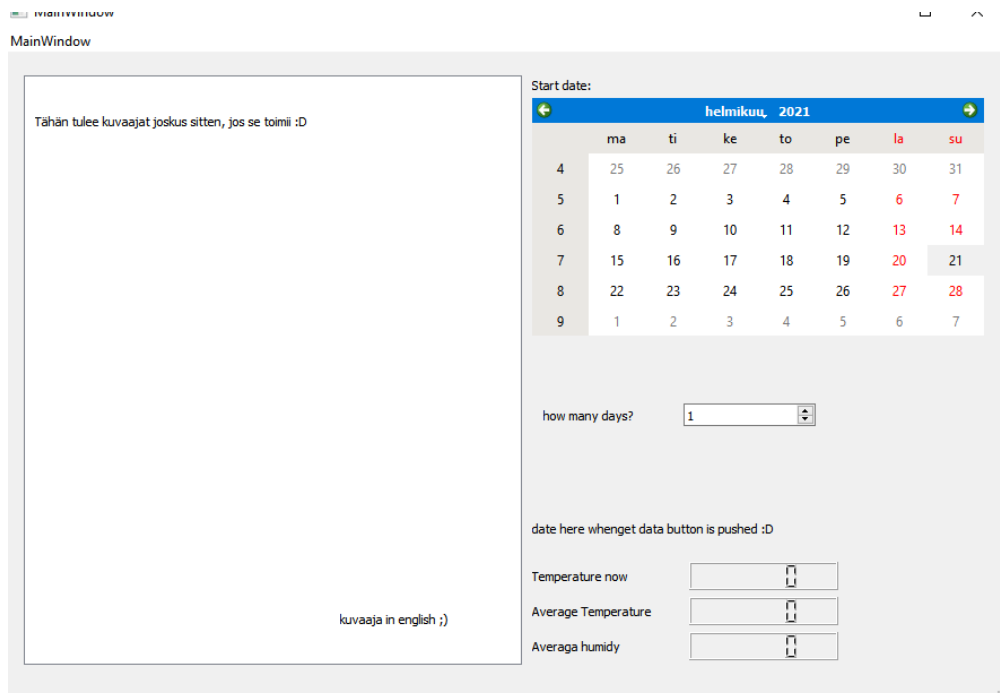

Picture 1: User interface 0.1

Our UI is demonstrated in picture 1. Left side of the interface is dedicated for the presented graph. Right side includes calendar from which the user can choose the date to fetch data from. Underneath the calendar user can choose how many days of data they need – this is still work in progress. Then when user hits a button all the data is shown as a graph and in from of LCD. We took this approach to the user-interface, so it is very self-explanatory, easy to access and easy to learn. All

the widget naming will be redone in the final product to better fit its functionality – this makes the product even more simple.

2. Components

We have split the program into smaller, "easier-to-handle" components. The logical function in the program is the logic presenter itself. We use the Logic-file to create the graph from the fetched data points and to fetch the data. At this point of development, we are certain we don't need another component to handle the logic. The UI is handled by the MainWindow component, all the visual and user-accessible data is handled and presented in the MainWindow.
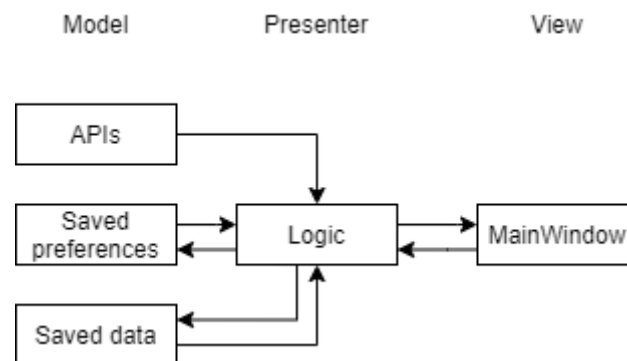
Picture 2: Class diagram 0.1

The class diagram in the picture 2 presents how the data is handled and how the data flows in the application. The Logic is the presenter itself. All the data flows through logic. Logic then converts the data in the desired form of output and passes the data onwards to the proper component. All visual - "viewed" - data is presented in the MainWindow.

3. Modules and classes

The modules we currently use in the project are Qt Core, Qt Charts, Qt Network and Qt Widget.

- widgets
    - QFileDialog
    - QCalenderWidget
    - QMainWindow
    - QListWidget
- core
    - QTemporaryFile
    - QXmlStreamReader
    - QFile
- network
    - QNetworkReply
    - QNetworkAccessManager
- QtCharts
    - QLineSeries

Picture 3: Modules and classes 0.1

Used modules and classes are presented in the picture 3. Currently QNetworkReply and QNetworkAccessManager are some of the classes used from Qt Network – Qt Network module is implemented so we can access the API. We use Qt Widget in order to create the main window of the UI and all its widgets - including the Calendar widget. The fetched data from the API is handled by the QXmlStreamReader from Qt Core. We use Qt Core also in order to write an output file for the user, this is done with QFile class.