Project Osaka University

Group: ITV2A-1

Students:
Arjan Bakema – 384728
Anton Bonder – 409075
Tim Bunk – 409678
Bram Rustenhoven – 360270
Ruben Eekhof - 405033
Stijn Wolthuis – 399653

Table of content

1. Preface

1.1 About the project

We have been commissioned by Osaka University to expand their existing research to Europa. Osaka university is researching if there is any correlation between happiness and weather. Our assignment is to create an application that collects and displays weather data.

1.2 Task division

We have dived our project group in two teams: Web development and backend. Web development is responsible for the website with dashboard and the representation of the data. The backend is responsible for generating the weather data and storing it the correct way. They also need to "prepare" the data, so it is ready to be shown on the website.

Name	**Tasks**
Anton Bonder	Web development, receive data from the server to the website. Make the charts, maps and table in the dashboard.
Ruben Eekhof	Web development. Responsible for the login system, and making the data downloadable on the website.
Arjan Bakema	Web development, Responsible for styling/coding the dashboard and login page.
TIm Bunk	Backend, designing and implementing the two java applications, the two python scripts that handle the temperature and humidity, writing all the documentation that belong to these tasks, helping Bram out with the VM's.
Bram Rustenhoven	Backend, configuring and troubleshooting the connection and communication between the virtual machines, making sure everything works smoothly on the VM's.
Stijn Wolthuis	Backend, writing documentation and PowerPoint.

1.3 Deadline of the project

The deadline of this project is on the 4th of February, 24hours beforehand we must send all the documentation and program code to the research at Osaka University. The presentation is on the 4th of February, the presentation will be given online on Blackboard Collaborate to the researchers of Osaka university.

2. Requirements

2.1 Target audience

The researchers of Osaka University need an application that can compare the weather of areas. They want to download that data so that they can use it in their already existing applications.

2.2 Website

The researchers of Osaka University want to access this application from anywhere so that is why the application is going to be a website with a dashboard.

- Website must support both Chrome and Firefox.
- Website must be in English.
- Website must have a login screen.
    - The website only has one user.
- Website must have the color scheme and logo of Osaka University.

Also, the website must display the stored data with some filters, graphs, and a table.

- There is one table that shows the top 10 humidity peak values in percentages from the last 4 weeks of data was collected.
    - The data inside of this top 10 table only comes from countries inside Europe.
- There are 2 line-graphs for displaying the rainfall in millimeters.
    - You can select a weather station that you want to see the rainfall of.
        - You only see weather stations that have an average temperature below 13.9 degrees Celsius over the last 2 days.
        - The only weather station you can select are from Europe or Japan.
    - The graph can go back up to 2 days.
- There is a logout button where the user is able to logout, he will be redirected to the login screen.
- There also needs to be a download button that downloads all the data that is on the screen to a .xml or .csv file.

2.3 Security

Osaka University also wants to be protected from eavesdropping. So, whenever any data goes over the internet it must be encrypted.

3. System configurations

3.1 IP address configuration

	Management VM	VM A (Generator)	VM B (low spec)	VM C (data)
IP-address	10.0.1.127	10.0.1.128	10.0.1.130	10.0.1.129
Subnet mask	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0
Default gateway	-	10.0.1.127	10.0.1.127	10.0.1.127

3.2 Additional configurations

We had to install a couple of additional software packages in order to properly use the virtual machines. Only the management VM had access to the internet however, so we had to find a workaround. We found a way to give the VM's access to the ubuntu package repository via a PowerShell script, which is pictured below.

```
1   param($vm, $user="leenfiets")
2   ssh -R 1080:us.archive.ubuntu.com:80 $user@$vm
```

It takes the IP-address of a virtual machine and optionally the username as command line arguments and forwards the ubuntu archive connection towards that address.

We installed the following additional software on the virtual machines using this PowerShell script:

- OpenJDK Java runtime environment 14, headless
- Apache2
- PHP 7.4
- Net-utils
- Python 2 and 3
- OpenSSH server

4. Virtual machine A (Generator)

4.1 Purpose

Virtual machine A uses Windows 10 as an operating system. The purpose of this virtual machine is to run a generator that produces weather data from different weather stations.

It can generate data for 8000 weather stations a second. All that data can be retrieved by listening to port 7790.

5. Virtual machine B

5.1 OS

For this machine we have chosen to use Ubuntu server 20.04.1 LTS. Our reasons for choosing a headless Ubuntu server were the low impact it has on performance, and our familiarity with Ubuntu.

5.2 Java application

5.2.1 Purpose

This machine runs a java application that collects data from the generator every second. For each cluster (a cluster is one socket connection that sends data from 10 weather stations) a thread is created which receives the data. Eventually the collected data will be sent to virtual machine C. This machine will save all the data to .txt files.

5.2.1 Dealing with generator data

The generator produces a lot of weather data that is nested inside of .xml format, but we only need:

- Weather station ID
- Temperature (degrees Celsius)
- Dew point (degrees Celsius)
- Rainfall (centimeters)

Once we have this data, we save it as a weather station inside of a HashMap. Obviously, it is not possible to save all the incoming data so that is why we have chosen to create averages for the incoming data. We calculate the average using this formula:

$$average_{new} = average_{old} + \frac{value_{new} - average_{old}}{size_{new}}$$

This calculation is used to calculate the average of an hour of data, which will then be sent to the data storage machine.

5.2.2 Dealing with missing data.

If the received data is missing something, then that entire piece of data will be ignored. This is feasible since we calculate the average of an hour of data. Because we receive data every second missing a second of data does not really matter as we will eventually end up with enough valid data for a proper average.

5.2.3 Sending the data over the network.

Every full hour of the day it will send all the calculated average to port 7790 on the data storage machine.

The data is sent and stored in the following format:

What	Example data	Written as
Year	2021	21
Month	1	01
Day	25	25
Hour	13	13
Station ID	5602	005602
Temperature (degrees Celsius)	12.1	+12.1
Humidity (Percentage)	78.2	78.20
Rainfall (Centimeters)	0.14	00.14

This data will be written as on String like this:
21012513005602+12.178.2000.14

Every piece of data has a standard length, making it easy to interpret the received string.

5.2.4 From temperature and dewpoint to humidity

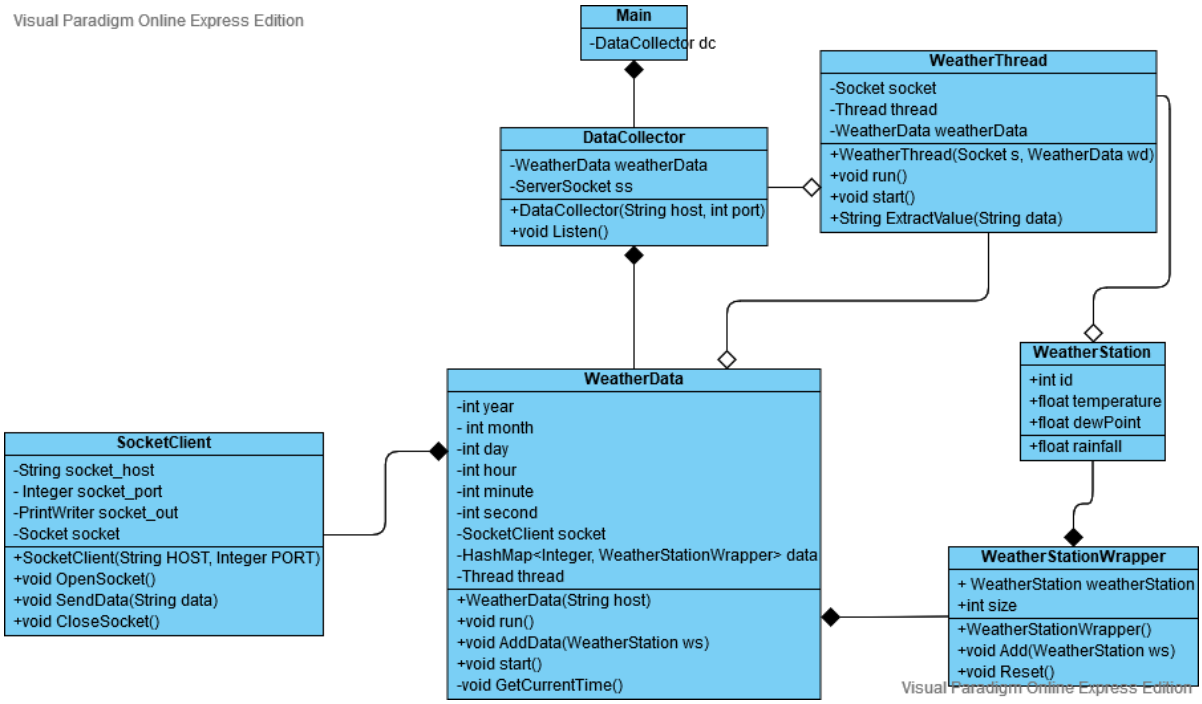
Since the weather stations don't produce data for the humidity, we must calculate that ourselves. We can do this because we do have the temperature and dewpoint in degrees Celsius. The formula used to calculate the relative humidity is as follows:

```
// calculate the humidity with the temperature and dewpoint
// first calculate value A
// A = (17.625 * temperature) / (243.03 + temperature)
Double A = (17.625 * temperature) / (243.03 + temperature);

// calculate value B
// uses the same formula, but with the dewpoint instead
// E = (17.625 * dewpoint) / (243.03 + dewpoint)
Double B = (17.625 * dewpoint) / (243.03 + dewpoint);

// calculate the relative humidity with both values
// Rh = (e^B) / (e^A) * 100
Double humidity = (Math.exp(B) / Math.exp(A)) * 100;
```

5.2.5 Encryption

In order to protect the data being sent to and from the client, we use SSL encryption on the website. We self-signed a certificate and imported it into Firefox manually. This is sub-optimal since Firefox does not fully trust self-signed certificates, but this is easy to solve in a production environment since we can get a certificate signed by a trusted certificate authority.

5.2.6 Class diagram



This is the class diagram for the java application. The DataCollector is the main class. This class creates a WeatherThread for every incoming connection to deal with the incoming .xml data. It will put all of the data inside a WeatherStation class and send it to the WeatherData class. The WeatherData class will then put the WeatherStation inside of a WeatherStationWrapper that will calculate the average values.

Once every whole hour the WeatherData class will send all the data to virtual machine C but before doing so it will calculate the humidity first.

6. Virtual machine C

6.1 Purpose

This virtual machine receives all the data from virtual machine B. Then that data will be saved systematically to .txt files. It also hosts the website.

6.2 OS

For this machine we have chosen to use Ubuntu server 20.04.1 LTS. The main reason for our choice was for performance but also because we have some experience using Ubuntu.

6.3 Java application

6.3.1 Purpose

This application waits for incoming data from virtual machine B. When it receives the data, it will write it to the correct .txt files.

6.3.2 Temperature.txt

The following data will be written to temperatures.txt:

What	Example data	Written as
Hour	13	13
Day	25	25
Month	1	01
Year	2021	21
Station ID	5602	005602
Temperature (degrees Celsius)	12.1	+12.1

This data will be written as on String like this inside the .txt file:
13250121005602+12.1

6.3.3 Humidity.txt

The following data will be written to humidity.txt:

What	Example data	Written as
Hour	13	13
Day	25	25
Month	1	01
Year	2021	21
Station ID	5602	005602
Humidity (percentage)	78.1	78.10

This data will be written as on String like this inside the .txt file:
1325012100560278.10

6.3.4 Rainfall data

The data of the rainfall will be stored somewhat differently compared to the temperature and humidity. This is because the website can ask for very specific sets of data.

On the website you can select a weather station that you want to display in the graph.

To quickly find the correct data we structured the data like this:

For every day of data, we create a new folder with the date as name like this:
23-1-21 (day-month-year)

Inside this folder there will be loads of .txt files. Every .txt file represents a weather station. For example, a weather station with id 2000 will be named 2000.txt.

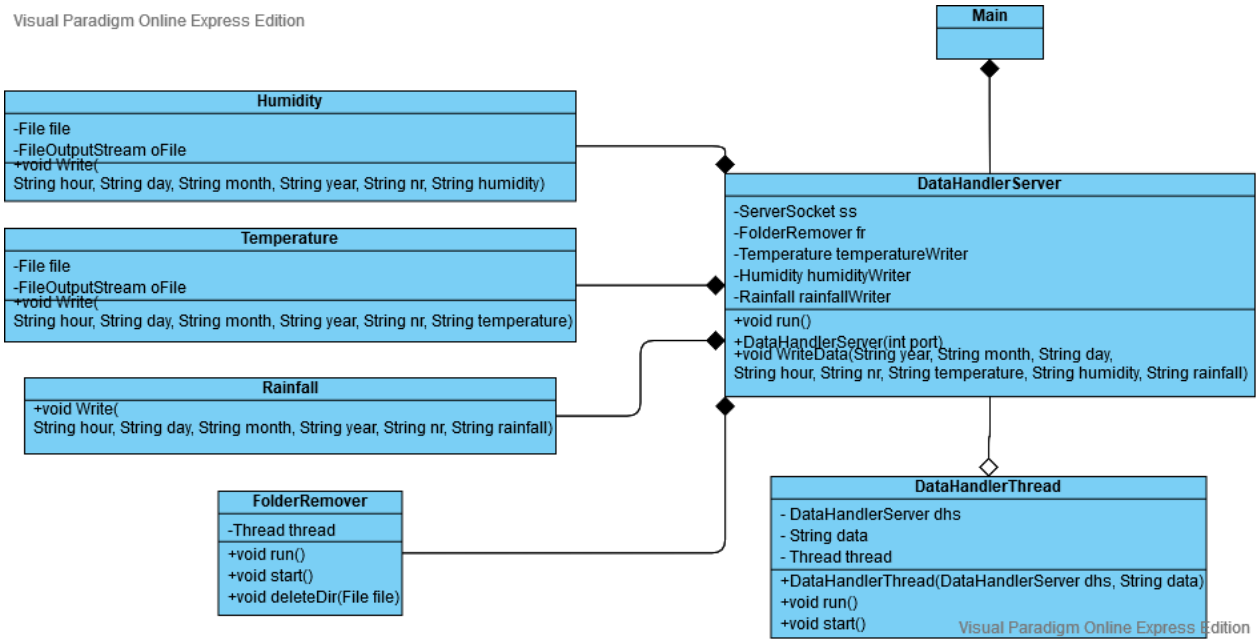
The data inside the file will look like this:

What	Example data	Written as
Hour	13	13
Rainfall (Centimeters)	0.81	00.81

This data will be written as on String like this inside the correct .txt file:
1300.81

The java application also searches for folders that are 3 days old or older and deletes them all because we don't need that data anymore. Only the data of yesterday and the day before yesterday will be used.

6.3.5 Class diagram



This is the class diagram of the application.

The DataHandlerServer is the main class. This class waits for an incoming connection and when it connects to something it will start reading the incoming data line by line. For every line of data, a separate thread will be created that handles that line of data.

The thread that will be created is the DataHandlerThread. It will split up all the data and sends it back to the DataHandlerServer. The DataHandlerServer will then send the data to the appropriate classes. Meaning that humidity will be send to the humidity class, temperature will be sent to the temperature class and the rainfall will be send to the rainfall class.

All these classes write the data to the correct .txt files.

There also is a FolderRemover class which will be triggered once every 24 hours to check if there are any old folders that need to be removed.

6.4 Python scripts

6.4.1 Purpose

The Python scripts are going to be used to summarize the data that was collected for the temperature and humidity. At the start of each day these scripts will run to start summarizing all the data that was collected the day before.

6.4.2 Script temperature

The only reason we need the temperature is to see if the average temperature of a station is under 13.9 degrees Celsius for the last two days. If that is the case, then we add that station number to a .txt file called valid_stations.txt.

This file can be used by the website to see what weather station have an average temperature beneath 13.9 degrees Celsius. Only those weather stations can be displayed in the rainfall graph.

To calculate the average temperature for a day, we use the already existing temperatures.txt that was created by the Java application. Then we write those averages to a file called average_temperatures.txt. Also, the temperatures.txt file will be cleared because we no longer need that data anymore.

Then the average_temperatures.txt will be read to calculate the average temperature over 2 days. If the temperature is below 13.9 degrees Celsius then it will be added to the valid_stations.txt. If a temperature inside average_temperatures.txt is older than 2 days, it will be deleted from the file.

When writing to the valid_stations.txt only European countries and Japan will be added. It also includes the name, country, latitude and longitude of that weather station. This data from the weather station can be received from a .txt file that contains all the data about the weather stations.

Average_temperatures.txt:

What	Example data	Written as
Day	1	01
Month	25	25
Year	13	13
Station ID	5602	005602
Temperature (degrees Celsius)	12.1	+12.1

This data will be written as on String like this inside the .txt file:
012513005602+12.1

Valid_stations.txt:

What	Example data	Written as
Station ID	5602	005602
Name	IJMUIDEN	IJMUIDEN
Country	NETHERLANDS	NETHERLANDS
Latitude	70.933	70.933
Longitude	-8.667	-8.667

This data will be written as on String like this inside the .txt file:
005602,IJMUIDEN,NETHERLANDS,70.933,-8.667

6.4.3 Script Humidity

Every day this script will search for the peak value of humidity of the day before for every weather station. All those peak values will be saved to a file called humidity_peak_values.txt for up to 4 weeks.

It can find those peak values inside the humidity.txt created by the Java application. Once it read all the data from the humidity.txt it can be cleared because we no longer need that data.

Once it finished with all the above it will read through the humidity_peak_values.txt and search for the top 10 highest peak values. Those peak values will be written to humidity_top_10.txt. The website uses this file to create a table with the top10 highest humidity's from the last 4 week.

When writing to the humidity_top10.txt only European countries will be added. It also includes the name, country, latitude and longitude of that weather station. This data from the weather station can be received from a .txt file that contains all the data about the weather stations.

Humidity_peak_values.txt:

What	Example data	Written as
Hour	13	13
Day	25	25
Month	1	01
Year	2021	21
Station ID	5602	005602
Humidity (percentage)	78.1	78.10

This data will be written as on String like this inside the .txt file:
1325012100560278.10

Humidity_top10.txt:

What	Example data	Written as
Hour	13	13
Day	25	25
Month	1	01
Year	2021	21
Station ID	5602	005602
Humidity (percentage)	78.1	78.10
Name	IJMUIDEN	IJMUIDEN
Country	NETHERLANDS	NETHERLANDS

This data will be written as on String like this inside the .txt file:
1325012100560278.10,IJMUIDEN,NETHERLANDS

6.5 Website

6.5.1 Dashboard

We wanted the website to be organized and clean, so we used a dashboard to display the graphs and top 10 humidity on a single page.

6.5.2 CSS

The researchers of Osaka University wanted the website to have the color scheme of the university. The most dominant colors that are used in the CSS are:

- White
- Blue (r:61, g:57, b:153)
- Grey (r:237, g:237, b:237)

6.5.3 Login

The dashboard is only accessible through a login, the researchers are given a single account that they can use to visit the dashboard.

Username: Researcher
Password: Osakaresearch321!

6.5.4 Receive data from the server.

For receiving data from the server to the webpages, we use three PHP files to make JSON's. By using JSON we can easily show the data on the website and export it to XML. We use three different PHP files.

Chart.php
The chart.php search for the data for the charts on the website. The JSON contains the rainfall and the time of the rainfall. The chart.php needs two parameters: days back and the weather station number.

Top10.php
The top10.php convert the file humidity_top10.txt to a JSON for the website. We use this JSON for the table on the website. The JSON contains time, date, stn, humidity, name of the station, country of the station, latitude of the station and the longitude of the station. The top10.php does not need a parameter. This json file is also converted to an xml file that is then downloadable on the website.

Valid_stations.php
The valid_stations.php convert the file valid_stations.txt to a JSON for the website, we use this JSON for the map on the website. The JSON contains: stn, name of the station, country of the station, latitude of the station and the longitude of the station. This json file is also converted to an xml file that is then downloadable on the website.

6.5.5 Map

For the map with the weather stations, we use the Google Maps API. Because of this we can use a map for selecting a weather station for the rainfall.

6.5.6 Chart

For the charts on the website, we use the Google Chart API. We use this API so we can easily show the rainfall.

6.5.7 Top 10 humidity table

The top 10 humidity table is made with HTML, CSS and JavaScript.