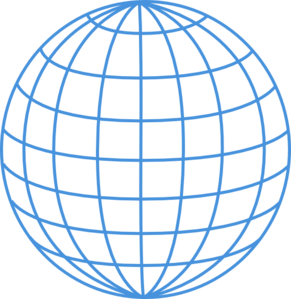
s



UNWDMI  
Team Schnitzel

Project report

Theme 2.2  
February 2017  
332975 - Bart Bakker  
346877 - Danny D. Jager  
347419 - Eran B. Machiels  
274633 - Joost P. de Vreede  
343567 - Jouke Y. Rienstra  
351509 - Matthijs S. Bonnema



# Title page

**UNWDMI – Team Schnitzel***Project report*

This document encompasses the information used for the production of, the design choices about and the instructions for the use of the *Team Schnitzel Weather Service.*

This document is part of project for Theme 2.2 of the HBO-ICT education of the Hanzehogeschool Groningen.

Team members:

332975 Bart Bakker

346877 Danny D. Jager

347419 Eran B. Machiels

274633 Joost P. de Vreede

343567 Jouke Y. Rienstra

351509 Matthijs S. Bonnema

Tutors:

General tutor A.E.A.G. Thüss

Technical tutor J.H.F. Bredek

*February 2017*

# Abstract

# Table of contents

Contents

[Title page 1](#_Toc474074884)

[Abstract 3](#_Toc474074885)

[Table of contents 5](#_Toc474074886)

[Introduction 1](#_Toc474074887)

[I. About the United Nations Weather Data Management Institute 3](#_Toc474074888)

[II. Customer Requirements 5](#_Toc474074889)

[III. Analysis of data 7](#_Toc474074890)

[IV. Data processing 9](#_Toc474074891)

[V. Data representation 11](#_Toc474074892)

[VI. Conclusion 13](#_Toc474074893)

[VII. References 15](#_Toc474074894)

# Introduction

Team Schnitzel (TS) is a part of the United Nations Weather Data Management Institute (UNWDMI). The team produces applications for clients of the UNWDMI. In the beginning of January 2017, the team was contacted by a client from Serbia, the University of Belgrade. The University was interested in the opportunities the UNWDMI could offer them. At a meeting in mid-January, it was decided that TS would produce an application to help the University with their research.

During the following two months, TS worked out their application. It is called the ‘*Team Schnitzel Weather Service*’ (TSWS). The TSWS receives localized weather data from weather stations from all over the world. The data is received via a collection server. Currently, this server is set up to receive up to a maximum of 800 weather stations at the same time. For the further use of this data, a distribution server is used. This server provides the clients with the requested data for their research.

Both servers are lacking resource overhead with the use of these applications. The resource allocation provided quite a few difficult design choices. But the provided TSWS still provided the data as requested by the client.

This report provides the background information of the whole production process of the TSWS. In the first chapter, the general information about the UNWDMI will be discussed. What is the UNWDMI, what is the data that is available to clients? The second chapter will provide information about the case of the University of Belgrade. What are the wishes of the clients and how do they want to access this data? Following in the third chapter, there will be an explanation about how the data that is available to the UNWDMI will be received from the first moment it enters the systems, to the moment that it is stored for the use in further applications. The actual processing of this data for the case of the client, begins in chapter four. This is where the calculations for the for their use cases are enlightened and where all other data modifications otherwise invisible to the client are discussed. In the fifth chapter, the part of the application with witch the client interacts will be explained. This also gives explanation about the use of the final product and how it connects to their provided points of research. Finally, in chapter six, the conclusion of the TSWS will follow.

# About the United Nations Weather Data Management Institute

The United Nations Weather Data Management Institute (UNWDMI) is an organization that provides global weather information for the use of local governments, regional weather forecasting bureaus and other organizations that have an interest in the provided data. It has its headquarters in Groningen, the Netherlands. It is a small organization of 90 people that receives weather data from all local weather institutes from countries connected to the United Nations.

|  |
| --- |
| C:\Users\Joost\AppData\Local\Microsoft\Windows\INetCacheContent.Word\UNWDMI_Global.png  Figure 1 – A rendering of the connected UNWDMI weather stations, across the globe. |

These local institutes provide weather data from their own countries to the UNWDMI in a standardized manner. In all, there are 8000 stations, all over the world. Figure 1 shows a representation of the locations over the globe.

The bulk of these stations lays within Europe, Northern America and Asia. All data stations provide an update every second. The UNWDMI collects this data in a centralized system. After the collection, the data will be used for applications built to the specific needs of its customers.

The UNWDMI currently has two systems to its disposal for these two tasks. Both systems are chosen to make optimal use of the available hardware, regarding the amount of data that needs to be processed. This means that every bit of processing power available in the system will be used to the fullest. The system will always produce the most accurate result possible.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Team member*** | ***Department*** | | *Matthijs S. Bonnema* | Management | | *Joost P. de Vreede* | Service Management | | *Bart Bakker* | ICT Services | | *Jouke Y. Rienstra* | Application Development | | *Eran B. Machiels* | General Management | | *Danny D. Jager* | Data Acquisition |   Table 1 – The members of Team Schnitzel and their departments within the UNWDMI. |

Team Schnitzel (TS) is one of several production teams within the UNWDMI. The production team is a team of people from several people from departments within the UNDWMI. The current consistency of TS is displayed in table 1. Since all teams within the UNWDMI consist of several departments, expertise from all their working fields will contribute to a more streamlined product.

# Functional design

## Customer requirements

Team Schnitzel (TS) was contacted by a delegation from the University of Belgrade. The University expressed the need for weather data for the from the regions in which they perform their primary research. In the start of January, the delegation visited the headquarters of the UNWDMI for a meeting with TS, to work out the fine details of their expected uses.

The main points of their research encompass climate changes in short to medium timeframes. It is the goal to be able to monitor trends within all the different datasets the UNWDMI has to offer. The University also asked for the calculation of two specific datasets:

* A calculation of the humidity values for all the weather stations within Serbia. The data should be available at intervals of 10 seconds per station. The data should be represented in a graph for the past 60 minutes. It would be valuable if it is possible to zoom in within the graph, so that even subtle changes are easy to recognize. The primary goal for these values is to help academics in the research for fungi and other trails involving humidity dependent variables.
* A representation of the 5 weather stations, within the Balkan area, that have the highest visibility distance. The visibility ranking should be calculated per day, from midnight onwards. A permanent history of these values should be available at all times. If the data for the current day is requested, the values should be calculated at once for all the measurements until that point of the day.

An interface needs to be created for all of the users within the University to interact with the data. A website would suffice, but there are certain precautions to take into account for the design. Most of the researchers in the field will use their mobile devices to access the data. This could be tablets, as well as ‘smartphones’. The website should look the same on all devices, regardless of the physical size or screen resolution. Also, in some areas where the academics operate, mobile coverage is poor or non-existent. It should be possible to access the most recent data beforehand. Most of the devices that will be used, are able to work with spreadsheets.

The whole system should be protected against unauthorized access. The University expressed concern that their data should not be available to users that have not been granted permission from them. It was decided to protect the interface with a login system. TS will manage the user accounts and passwords. All updates within the users will be submitted to TS by the university on a regular basis.

All returned results in the website, together with the saved data in the underlying systems should be stored in the local time formatting of Europe/Belgrade.

## Defenition of the Balkan

During the development of the application, it became apparent that there are several possibilities to define the area that is colloquially called ‘the Balkan’. The three most logical uses were conveyed to the University. These versions were as follows:

1. The Balkan peninsula, containing parts of (but not limited to) Croatia, Greece, Serbia, Slovenia and the whole of Albania, Bulgaria, Bosnia and Herzegovina, Kosovo, Macedonia and Montenegro. This set contains 138 weather stations.
2. The states which have their geographical borders beyond the Balkan peninsula. These states are: Albania, Bosnia and Herzegovina, Bulgaria, Croatia, Greece, Kosovo, Macedonia, Montenegro, Romania, Serbia, Slovenia and Turkey. This set contains 317 weather stations.
3. The same countries as in set 2, except Greece and Turkey. This set contains 196 weather stations.

The reply from the University was to use the first set of weather stations for the visibility calculations. Figure 2, 3 and 4 represent the coverage of these possibilities.

|  |
| --- |
| Figure 2 – A rendering of the all weather stations within the Balkan peninsula. |

|  |
| --- |
| Figure 4 – A rendering of all weather stations within the Balkan area, according to state borders, excluding Turkey and Greece. |

|  |
| --- |
| Figure 3 – A rendering of all weather stations within the Balkan area, according to state borders. |

## Analysis of data

|  |
| --- |
| Figure 5 – A selection of an incoming data stream. |

The data of all weather stations is provided to the UNWDMI in a standardized manner. It is transmitted in a byte stream, containing an XML structure. A single set of data contains information about 10 stations at the same time. This data needs to be translated to a format that can be used by all the other systems of Team Schnitzel Weather Service.

The original data string is always compiled in the same manner. Figure 5 shows an example of the received data. It is set up in an XML format. This is a way of dividing large amounts of data into logical segments. It is a structure that can use multiple *layers* of data. All layers and segments start and end with a corresponding *tag*. A tag is a string that is encompassed by the < and > signs. Furthermore, the last of the two tags always has a forward slash ( / ) in front of it. This is called the *closing tag*.

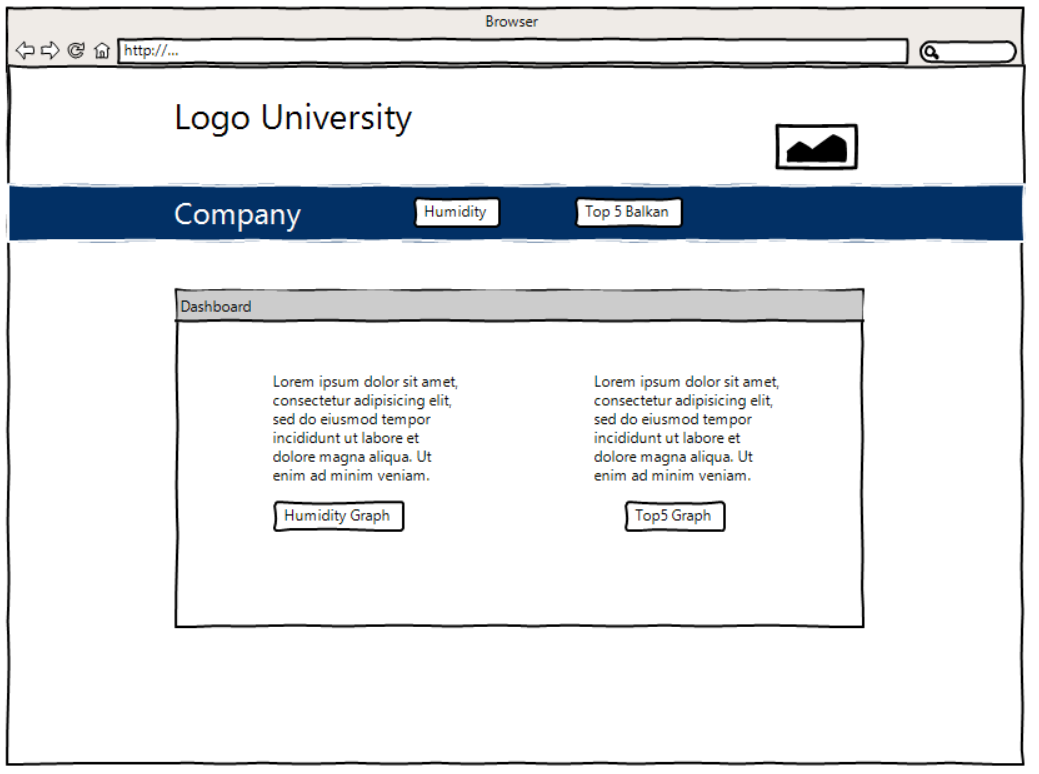
All data streams start and end with the *WEATHERDATA* tag, as shown on line 1 and 19 in figure 5. In between these tags are the data sets from the 10 stations. Figure 5 only shows the data from one station, between the *MEASUREMENT* tags on lines 2 and 17. These lines are repeated another 9 times, each time for a different station. The data provided by the stations are as follows (the number represents the line in figure 5):

1. The station number. All weather stations have a corresponding number that identifies the location and other information that is specific to the location.
2. The date on which the data was sent.
3. The time on which the data was sent.
4. The temperature in °C.
5. The dew point in °C.
6. The air pressure at the height of the weather station, in millibar.
7. The air pressure at sea level, in millibar.
8. The visibility in kilometers.
9. The wind speed in kilometers per hour.
10. The precipitation in centimeters.
11. The amount of fallen snow, in centimeters.
12. A string that represents events that happened since the last data stream. It represents the following six possible events: freezing, downfall as rain, snow and hail, lightning, tornado’s and whirlwinds.
13. The cloud coverage as a percentage.
14. The wind direction in degrees.

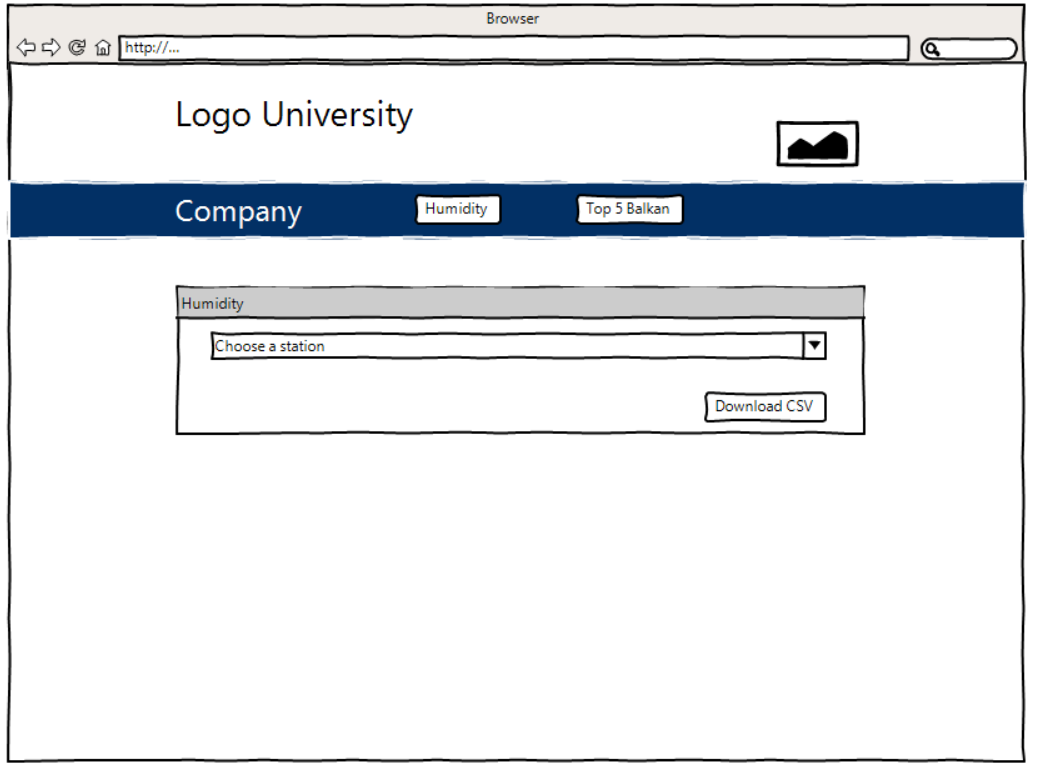
## Design Mockups

We were told by the customer, Belgrade University to implement the logo and the colors of their university. We came up with the following design(mockup).

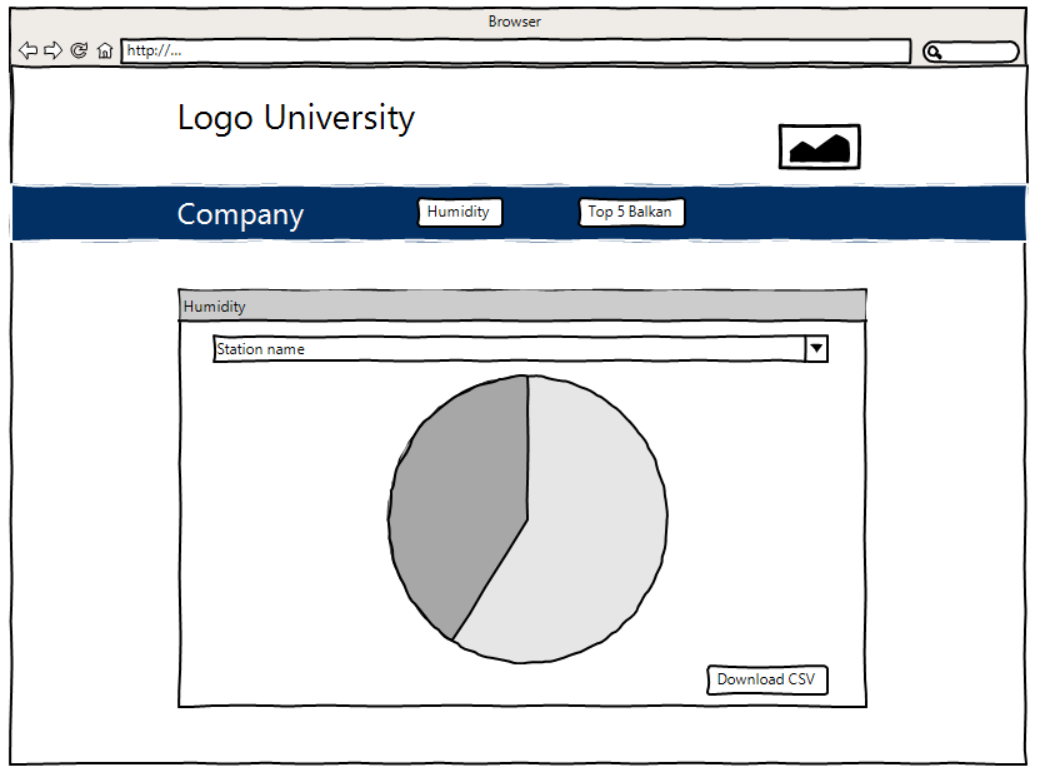
**Web page: Home screen**

****

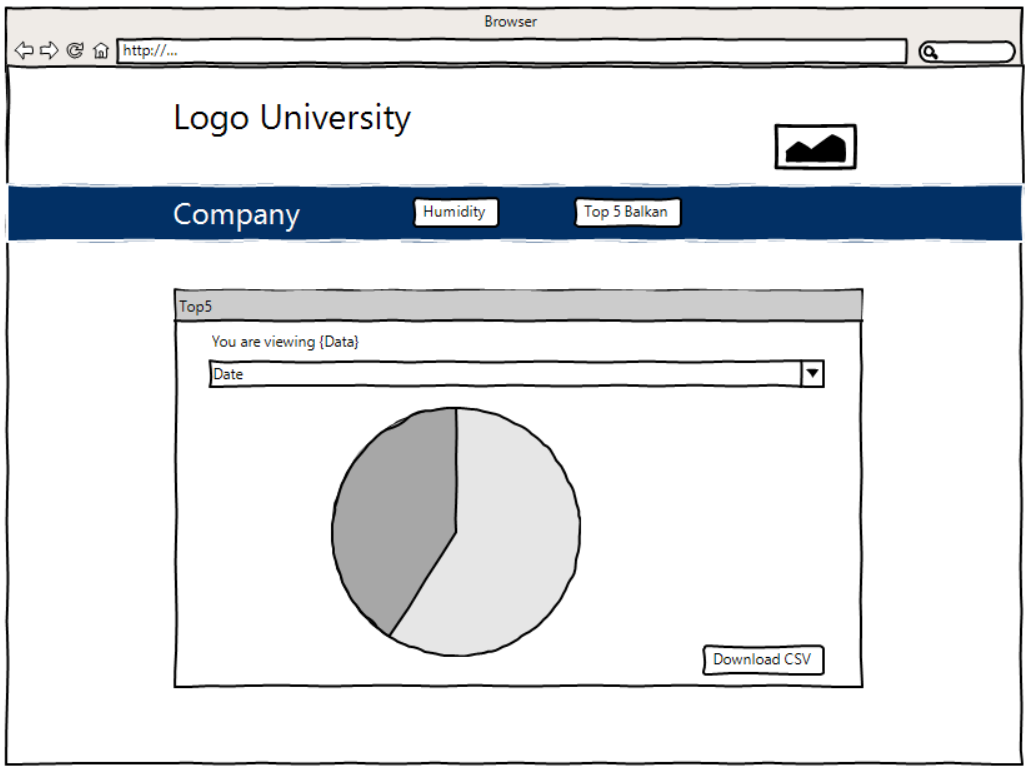
**Web page: Humidity empty**

****

**Web page: Humidity filled with data.   
Pi chart is pure for design purposes. Actual representation will be a graph.**

****

**Web page: Top 5 balcan visibility.  
 Pi chart is pure for design purposes. Actual representation will be a barchart.**



## 

# Technical design

This chapter describes the how the project is planned to be executed. The technical specifications of the project and our ideas on how to implement this is detailed in this chapter.

## Project management

To improve the collaboration between our team. We are executing the project with the scrum(agile) method. The software developers use this the most. This makes our team more effective.

We work in ‘sprints’ of one week. This means that every week we want certain points to be finished.

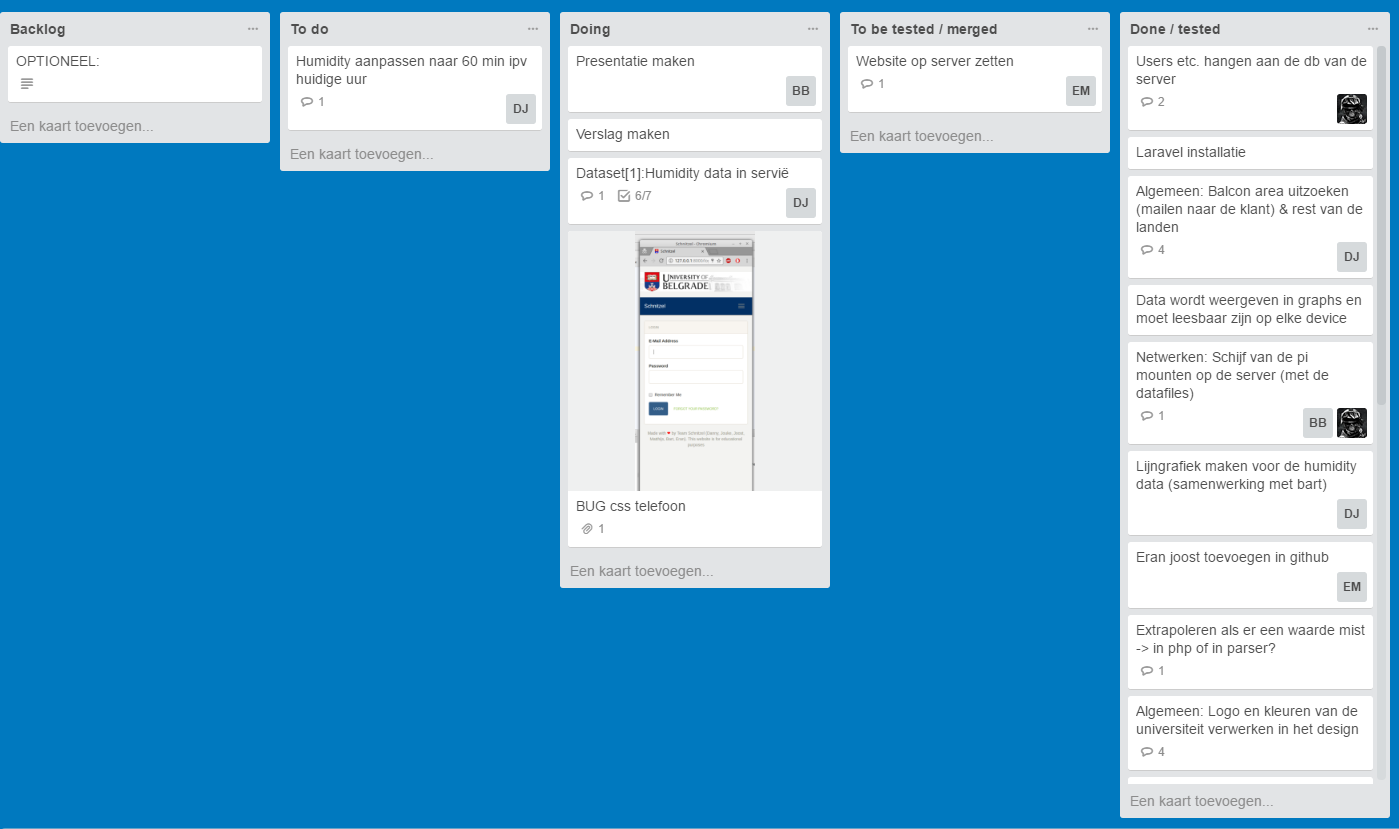
We are separating the project in several points that can be assigned to different team members. We use Trello.com for a visible presentation of the points that have to be executed.

We are using the following categories:

* Backlog: Idea’s fort he project
* Todo: Parts of the project that have to be executed by a team member.
* Doing; When someone is working on a certain part of the project.
* To be tested / merged: When code has to be merged between different team members or when code has to be tested.
* Done / Tested: Parts that are finished.

De volgende ScrumBoard leden zijn aanwezig:

* Jouke Rienstra = JR
* Eran Machiels = EM
* Bart Bakker = BB
* Matthijs Bonnema
* Joost de Vreede

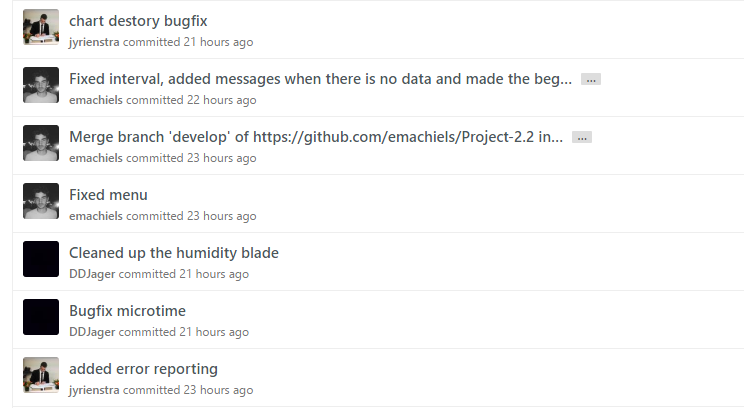
An actual representation of our scrum board: 

## Version control

Our team has a good experience with Git as version control for software development. Git shows historical changes by different team members in the code and makes it possible to merge files more easily together.

We use GitHub which is a visual presentation of Git. We are using a private repository for this project so that the code is not shared with outsiders.

An actual visual representation of GitHub.



## Communication tools

For communication we use WhatsApp for quick interactions. We use meetings and e-mail for formal interactions. Trello is used to communicate for what has to be done.

## Programming languages

For the realization of this project several languages are used.

For the user interface we will use a website which will be programmed in PHP. We use a MVC framework called Laravel. Laravel is object oriented and has several handy functions. MVC stands for Model View Controller. This means that programming logic is divided.

The model contains interactions with the database. The view is a visual representation of the data and the controller contains all the logic.

Larvel uses a file for routes(url). A route could be /humidty/stations. Which could provide all the stations from Serbia from example.

For the parsing of the data to excel files we use Java and Python. After it’s programmed we will test which one is running faster and processing the data the best.

## Data processing

@TODO MATTHIJS

De mappen structuur het idee erachter yy-mm-dd

Wanneer worden ze weggegooid?

Allen het idee hierachter

Uitvoering -> execution of the project

Deeltje wel database -> top5 van voorgaande dagen

## Os keuze en waarom

Matthijs

# Execution of the project

## NSE certification http

## NSE Filesharing

## NSE Firewall

## NSE Fail to ban

## SE Implementation of design

@bart

Misschien nog deeltje eran

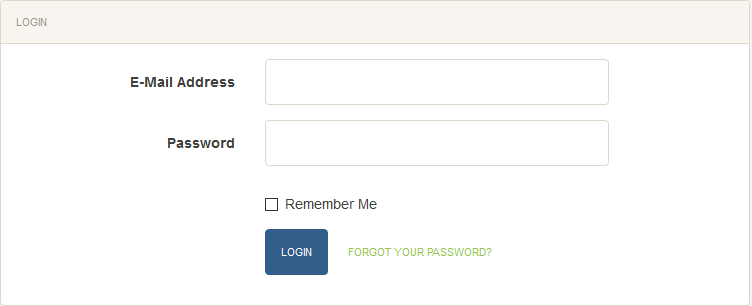
## Using data offline

@jouke

## SE Visibiliy logic

@eran

# Authentication Laravel



Authentication process (Login) of the website

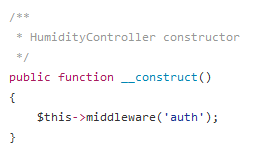
To make the realisation of the authentication process in Laravel really easy, Team Schnitzel used the ‘built-in authentication’ functionality of Laravel. The reason Team Schnitzel chose to use the pre-built function of Laravel is because user authentication is already a solved problem. The built-in function has been tested and used by many developers and it saves a lot of time which can be used to realise the other requirements of the customer.

Laravel has an easy scaffolding technique which generates a pre-built controller, middleware, views and even a basic layout. This is done by typing

*‘php artisan make:auth’* in the command line.

One of the requirements was that users should not be able to register an account by itself, so the registration form is hidden and only accessible by administrators.

There are multiple ways to prevent other pages from being accessed by guest visitors. One would be to group the routing in an ‘auth middleware’ (created by php artisan make:auth) and the other way would be to put the ‘auth middleware’ in the constructor of a controller where the current routing refers to. Team Schnitzel went with the latter option.



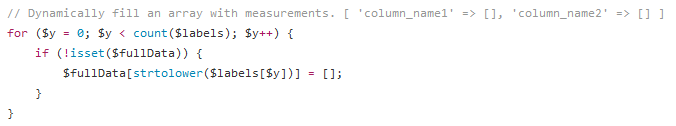
# CSV to Array

Because the CSV to array conversion plays a big part in this project, it’s important to explain this process very globally with a little bit of detail. The CSV to array conversion can be found in HumidityController.php and Top5visibilityController.php. The code is documented very well, so feel free to look in the source code when looking for more detail.

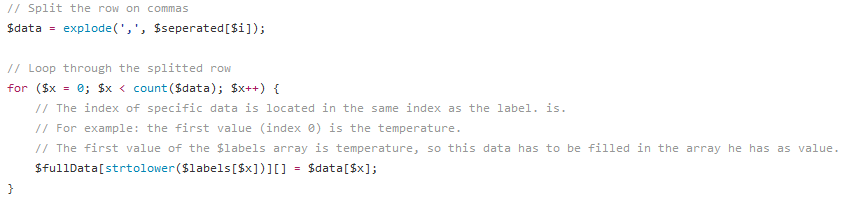
The code starts of with an if statement, where the code checks if a certain folder and file exists. This folder has an Y-m-d format. If the .csv file with the given id exists, it will execute the code within. The id of the file is the id of the weather station. For the humidity data, this gets handled by the HTML select box at /humidity where the name of a weather station is linked to their weather station id. The id then gets passed to javascript and an AJAX script calls the humidity/live/data/{id} page and expects a JSON response.

When the if statement passes, the code will then get the .csv file. Depending on the operating system, the data of the csv gets split up into an array. This is done by the php explode() function.

The measurement types (date, time, humidity, temperature, etc) then gets split up as well. The measurement types then becomes keys with an array as a value.



The array is now a multidimensional array. The code now loops over the entire dataset, skips the first line, skips empty rows and when the data is valid, it gets split by a comma again. The data then gets assigned their specific measurement type.



$data = [

‘temperature’ => [15.2, 16.3],

‘humidity’ => [4.8, 7.2]

];

This is an example of the data, where $data[‘temperature’][0] is linked to $data[‘humidity’][0] (15.2 and 4.8) and $data[‘temperature’][1] links to $data[‘humidity’][1] (16.3 and 7.2)

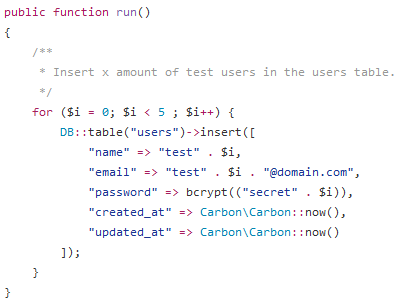
Source:

<https://github.com/emachiels/Project-2.2/blob/develop/website/app/Http/Controllers/HumidityController.php>

<https://github.com/emachiels/Project-2.2/blob/develop/website/app/Http/Controllers/Top5visibilityController.php>

## Website authentication, creating a database seeder

Because we have an authentication process in our application, our project group needed to have some login information. The easiest way is to let everyone in the project group have an easy way to work with test accounts without the having to create one manually. Luckily, with Laravel, the PHP MVC framework we’re using, the proces of creating test accounts is simplified with the concept known as database seeders. With the PHP Artisan CLI we could easily create a Seeder class. The Seeder class must have a run() method in order to work correctly.



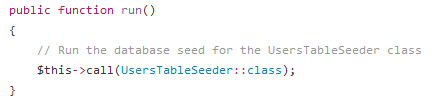
In the code above we inserted 5 rows in the users table. The details of the account are always called test and secret with the current loop count as a suffix. The passwords are hashed with the bcrypt (blowfish) algorithm.

Example:

Username: test2

Password: secret2

Before we can run the “php artisan db:seed” command however, we must first specify the class that we recently made (UsersTableSeeder) in the DatabaseSeeder class by using the call() method.



Everyone can now run a simple “php artisan db:seed” command in the command line and the test accounts are generated.

Source:

<https://github.com/emachiels/Project-2.2/blob/develop/website/database/seeds/DatabaseSeeder.php>

<https://github.com/emachiels/Project-2.2/blob/develop/website/database/seeds/UsersTableSeeder.php>

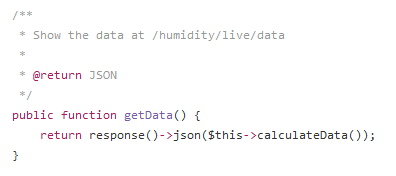
## Receiving the humidity data

The humidity data is received in .csv files from the Raspberry Pi. Because the humidity data, which is calculated on the Raspberry Pi must be received every 10 seconds, we first thought it was a good idea to run a cron job on the server. A crob job is a scheduled task. This is the reason why we first developed a custom php artisan command which was supposed to be called every 10 seconds on the server using a Laravel scheduler (Laravel’s API for cron jobs).

We later noticed that this was not a very smart decision because real-time data is always client-sided, so we needed Javascript to call HTTP GET requests. Fortunately we could reuse most of our logic, so not much time was wasted by changing our infrastructure.

The calculateData() function is where we receive and filter the data. For the more technical readers of this document: to view the calculateData() function in more detail, please refer to the file itself (can be found below). The code of our project is documented very well.

When we go to /humidity/live/data we call the calculateData() function and it decodes to JSON. Now we can use AJAX to render de data every 10 seconds and display it in the graphs:



Source:

<https://github.com/emachiels/Project-2.2/blob/develop/website/app/Http/Controllers/HumidityController.php>

<https://github.com/emachiels/Project-2.2/blob/develop/website/app/Console/Commands/calculateHumidity.php> (old: custom command)

# Displaying the humidity data

@bart moet het vanaf hier overnemen

# Conclusion

# References

Picture of globe on cover: clipartfest.com

All chart renders use part of the OpenStreetMap and are rendered in QGIS, as seen on pages 3 and 6, © OpenStreetMap