

POLITECNICO DI MILANO – COMO CAMPUS



ADVANCED WEB TECHNOLOGIES PROJECT 2015-2016

Prof. Piero Fraternali

**Project Title**

**<<Water Consumption Portal>>**

**Student**

<b>ID</b>	<b>Surname</b>	<b>Name</b>
<b>10513874</b>	<b>Tuncel</b>	<b>Mustafa Anil</b>

## Table of Contents

Introduction .....	5
Objective .....	5
Textual Description of the Application .....	6
UML Use Case Diagram .....	13
UML Class Diagram of the Application .....	19
Sequence Diagrams of the Application .....	20
Data Model of the Application .....	24
Deployment Diagram of the Application .....	25
Motivation of the Web Service APIs chosen .....	25
Google Chart API .....	25
Interaction with the Google Charts .....	25
Google Maps API .....	28
Interactions with the Google Maps .....	28
Google Maps Geocode API .....	28
Interactions with the Geocode API .....	29
Running JEE Code .....	29
Motivation of the JavaScript Libraries .....	30
Bootstrap .....	30
jQuery .....	31
jQuery-UI .....	31
Usage Tests .....	32
Login .....	32
Home Page .....	32
Initial Requirement .....	32
View Granularities .....	33
Initial Requirement .....	33
Hourly View .....	35
Initial Requirement .....	35
Average Values .....	35
Initial Requirement .....	35
Map View .....	36
Initial Requirement .....	36

Map View Details .....	37
Initial Requirement .....	37
Evaluation of the Experience .....	38
Post-project Remarks.....	39
Glossary.....	40
Table of Acronyms .....	40
References .....	41

## Table of Figures

Figure 1. Login Page of the application.....	6
Figure 2. The error message .....	7
Figure 3. Home page of the application.....	7
Figure 4. Date selection .....	8
Figure 5. Neighborhood and user averages.....	9
Figure 6. Pop-up showing the hourly water consumption histogram .....	9
Figure 7. Spinner visualized during the processing.....	10
Figure 8. The map view .....	11
Figure 9. Different icons to indicate different consumption types.....	11
Figure 10. The Map pop-up window.....	12
Figure 11. Use Case Diagram .....	13
Figure 12. Activity diagram for the login use-case.....	14
Figure 13. Activity diagram for the monitor the consumption use-case .....	16
Figure 14. UML Class Diagram .....	19
Figure 15. UML sequence diagram for login use case .....	20
Figure 16. UML sequence diagram for access map view use case .....	21
Figure 17. UML sequence diagram for show consumption values use case .....	21
Figure 18. UML sequence diagram for monitor the consumption use case.....	22
Figure 19. UML sequence diagram for access hourly view use case .....	23
Figure 20. UML sequence diagram for access the average consumptions use case .....	23
Figure 21. The Data Model.....	24
Figure 22. Deployment Diagram of Water Consumption Portal.....	25
Figure 23. Code snippet that asynchronously updates the chart .....	26
Figure 24. Code snippet that asynchronously renders the chart on a pop-up window .....	26
Figure 25. JavaScript function that draws the Google Chart in the hourly-view .....	27
Figure 26. Example of the key specification when loading the API .....	28
Figure 27. Interactions with the Geocode API .....	29
Figure 28. A sample JavaDoc for the IUserRepository Interface .....	29
Figure 29. Application scales for a low resolution .....	30
Figure 30. A usage example of JQuery-UI .....	31
Figure 31. Login Page .....	32
Figure 32. Home Page .....	33
Figure 33. Weekly View.....	34
Figure 34. Monthly View .....	34
Figure 35. Hourly View .....	35
Figure 36. Average values .....	36
Figure 37. Map View .....	37
Figure 38. Map View Pop-Up Window.....	38

## Introduction

The purpose of this application is to get a practical experience regarding the various steps of the development of an enterprise web project. The project is built using Spring Framework which provides a comprehensive programming and configuration model for modern Java-based enterprise applications.

The application uses Hibernate as an ORM (Object Relational Mapping) Framework which is an implementation of the JPA (Java Persistence API) specification. For the configuration of the database connection a property file is used in order not to “hard code” the JDBC connection strings.

Maven build automation tool is used in the project to manage the project’s build. The front-end of the Water Consumption Portal is made responsive to various screen sizes using Bootstrap libraries. Besides Bootstrap, the application is enriched with several front-end widgets. Google Charts API, Google Maps API and the Google Maps Geocode API are used in the Water Consumption Portal.

## Objective

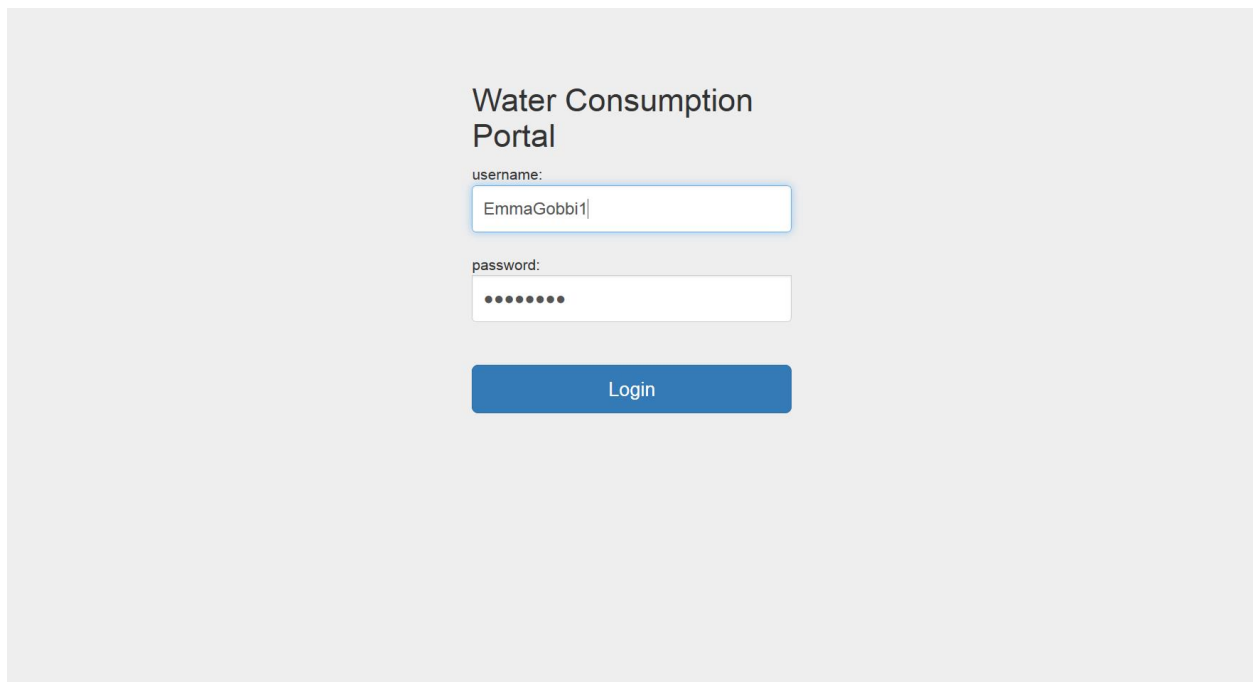
The objective of the Water Consumption portal is to provide users a web application in which they can monitor their water consumption data in various granularity settings. Furthermore, it allow users to compare their consumption with the consumption made in the neighborhood. Water Consumption Portal is also providing a map view to visualize the registered users on the map.

## Assumptions

The only assumption I made in this project is the limitation of the number of the bars to be shown on the histogram. Having many bars on the histogram is making the histogram very difficult to be read by a user. The assumption simply limits the number of bars to be visualized in the daily view lest the histogram be unreadable.

## Textual Description of the Application

Water Consumption Portal is a web application that allows users to monitor their water consumption data measured from their smart meter. The application starts with the login page, in which the user has to provide her username and her password (*Figure 1*).

The image shows a web application login page titled "Water Consumption Portal". It features two input fields: "username:" with the text "EmmaGobbi1" and "password:" with masked characters "••••••••". Below the fields is a blue "Login" button. The entire form is centered on a light gray background.

*Figure 1. Login Page of the application*

If the login succeeds she accesses the Home Page that visualizes the water consumption data (*Figure 2*). Otherwise an error message is shown (*Figure 3*).

# Water Consumption Portal

username:

password:

Login

Wrong username password combination

Figure 2. The error message

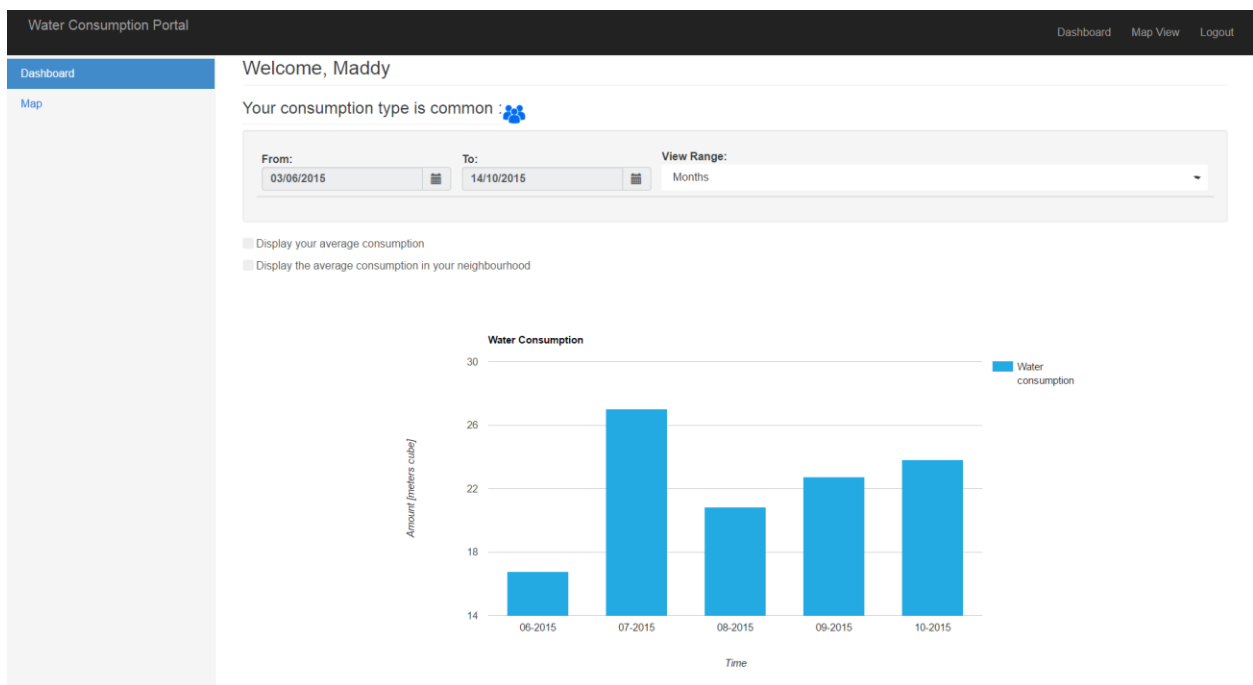


Figure 3. Home page of the application

In the Home Page of the application, an icon is denoting whether the consumption data of the user is individual or common. A menu allows user to change the granularity of the consumption data in monthly, weekly and daily views. Using the two date pickers, user can specify the view range she wants to visualize (Figure 4). The second date should be coming after the first one, otherwise a message will be shown. Also if the selected view is daily view, the day range to be visualized must be less than sixty. Using two checkboxes the user enriches the histogram with the average consumption of the user and the average consumption in the neighborhood of the user (Figure 5). In the day view, clicking on a bar opens a pop-up window to display the hourly consumption data of the user, if the smart meter has hourly data to be displayed. Otherwise a message is displayed stating there is no hourly data available for your selection (Figure 6). While the chart is loading, a spinner appears depicting the processing state of the chart (Figure 7).

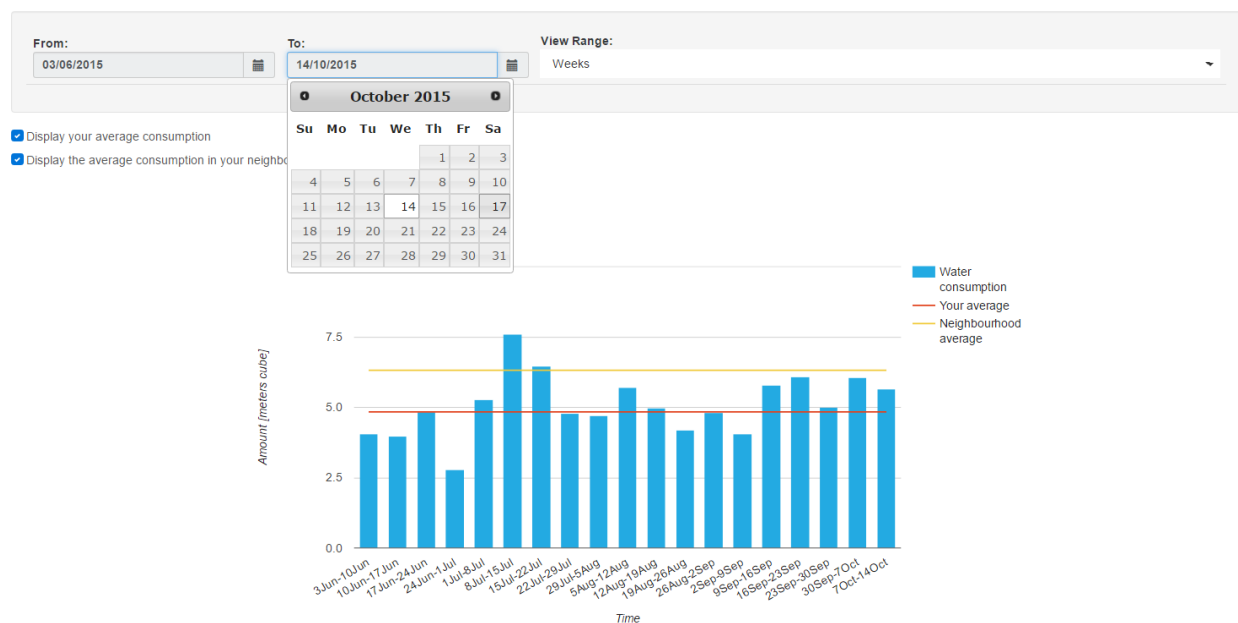


Figure 4. Date selection



- ☒ Display your average consumption
- ☒ Display the average consumption in your neighbourhood

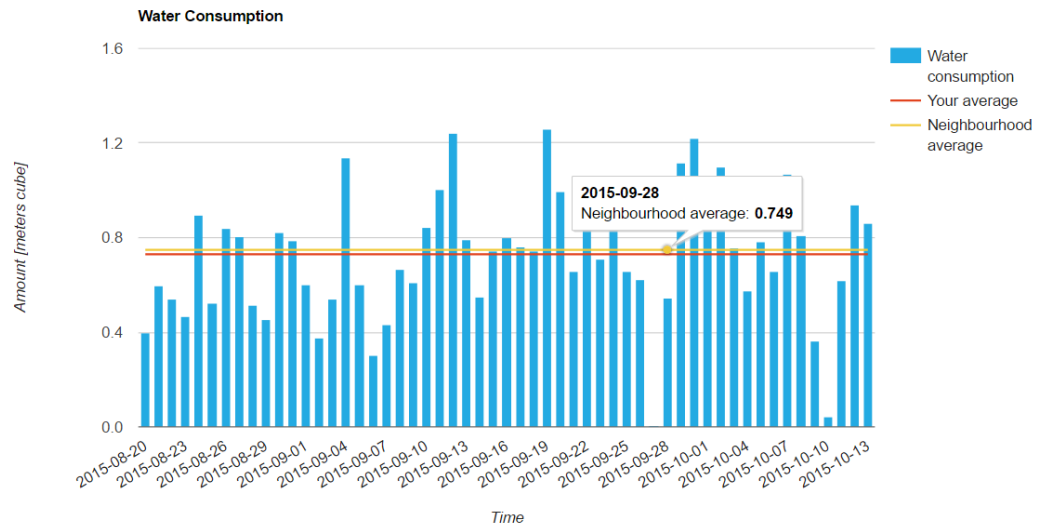


Figure 5. Neighborhood and user averages

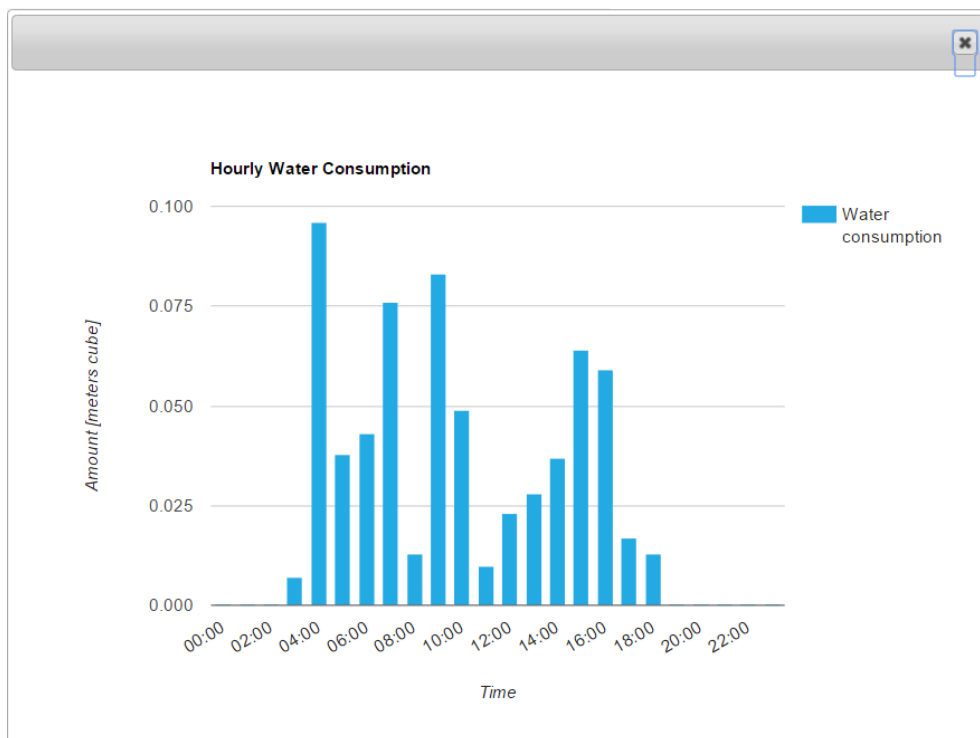


Figure 6. Pop-up showing the hourly water consumption histogram

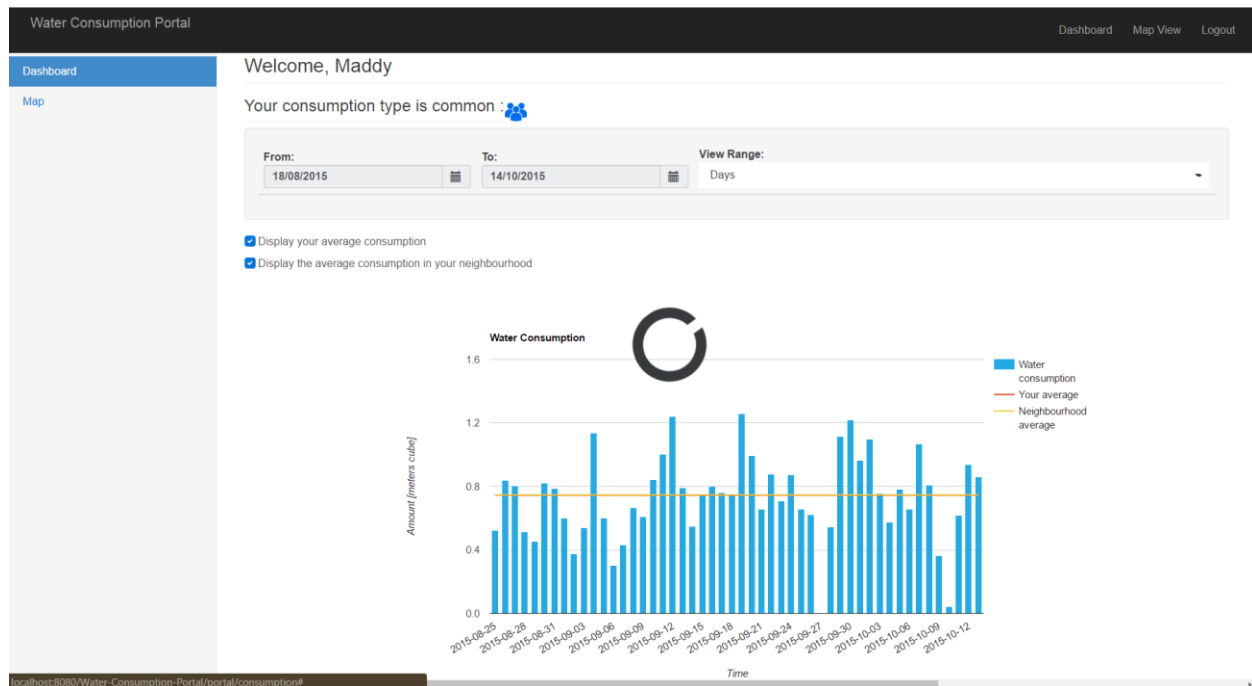


Figure 7. Spinner visualized during the processing

From the home page, user can access a map view that displays the addresses of the registered users on the map (Figure 8). The consumption type of the registered users (common or individual) is denoted on the map using different icons (Figure 9). 'C' icon is referring to the common consumption type and 'I' icon is referring to the individual consumption type. Clicking on an icon displays a pop-up window with the current total, daily, weekly and monthly average consumptions of the user (Figure 10).

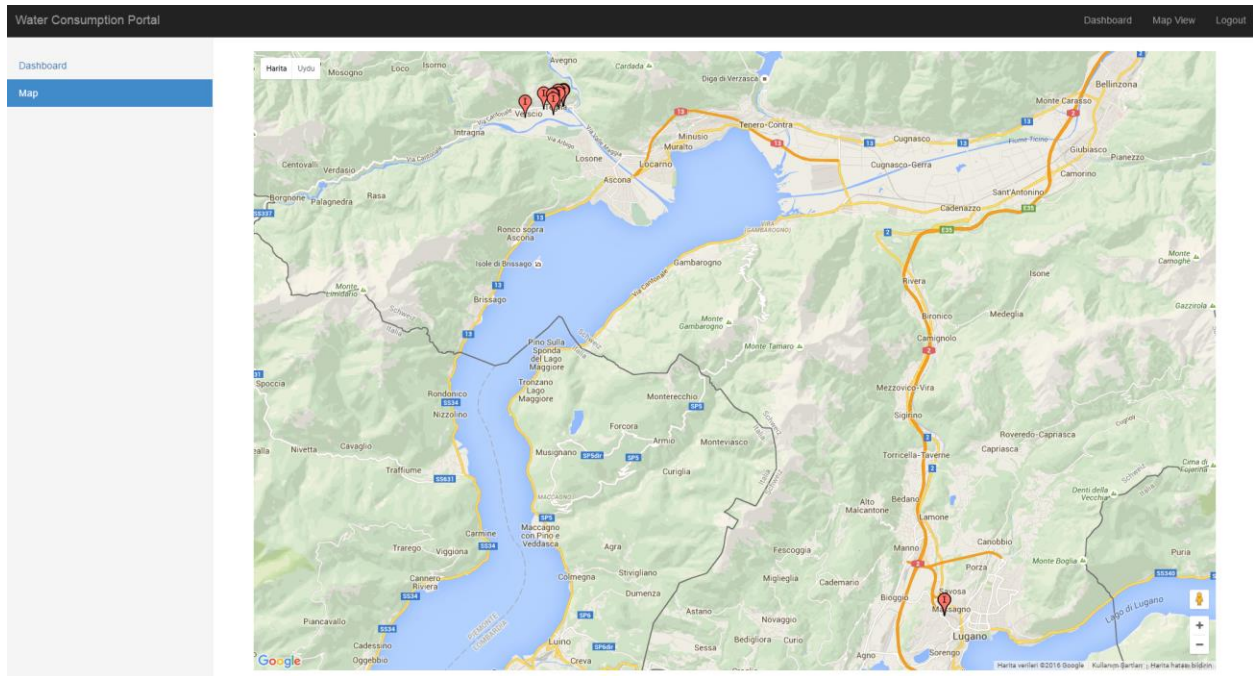


Figure 8. The map view



Figure 9. Different icons to indicate different consumption types

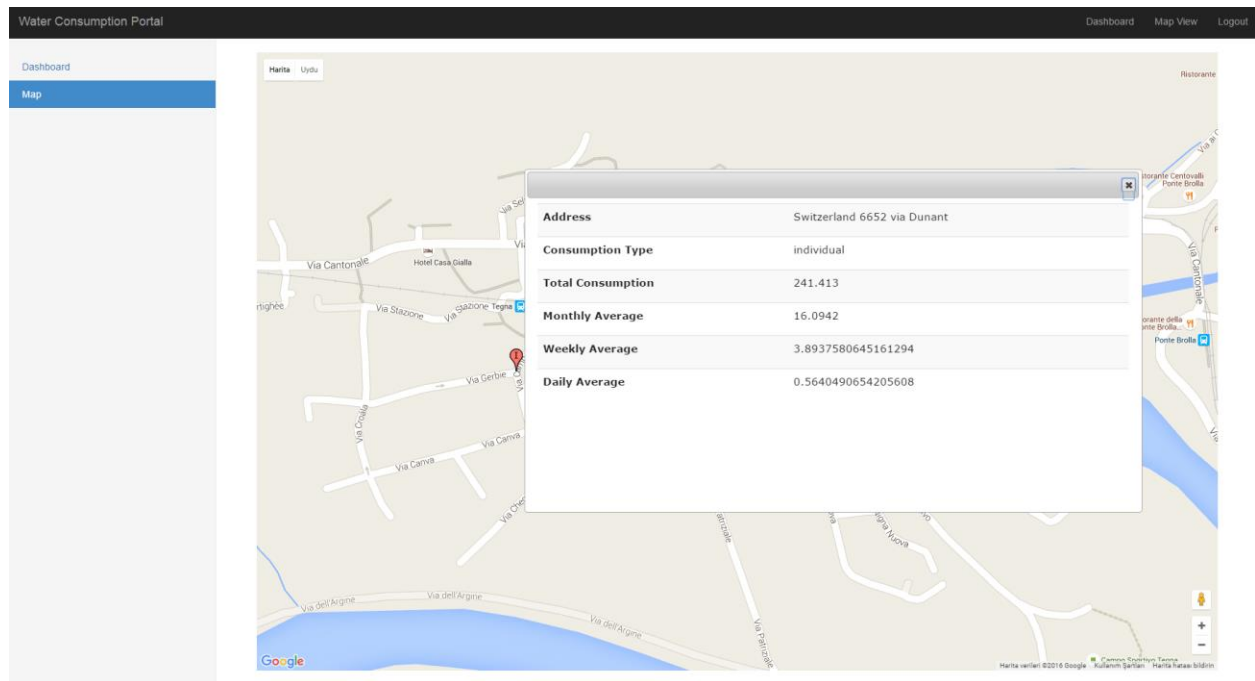


Figure 10. The Map pop-up window

## UML Use Case Diagram

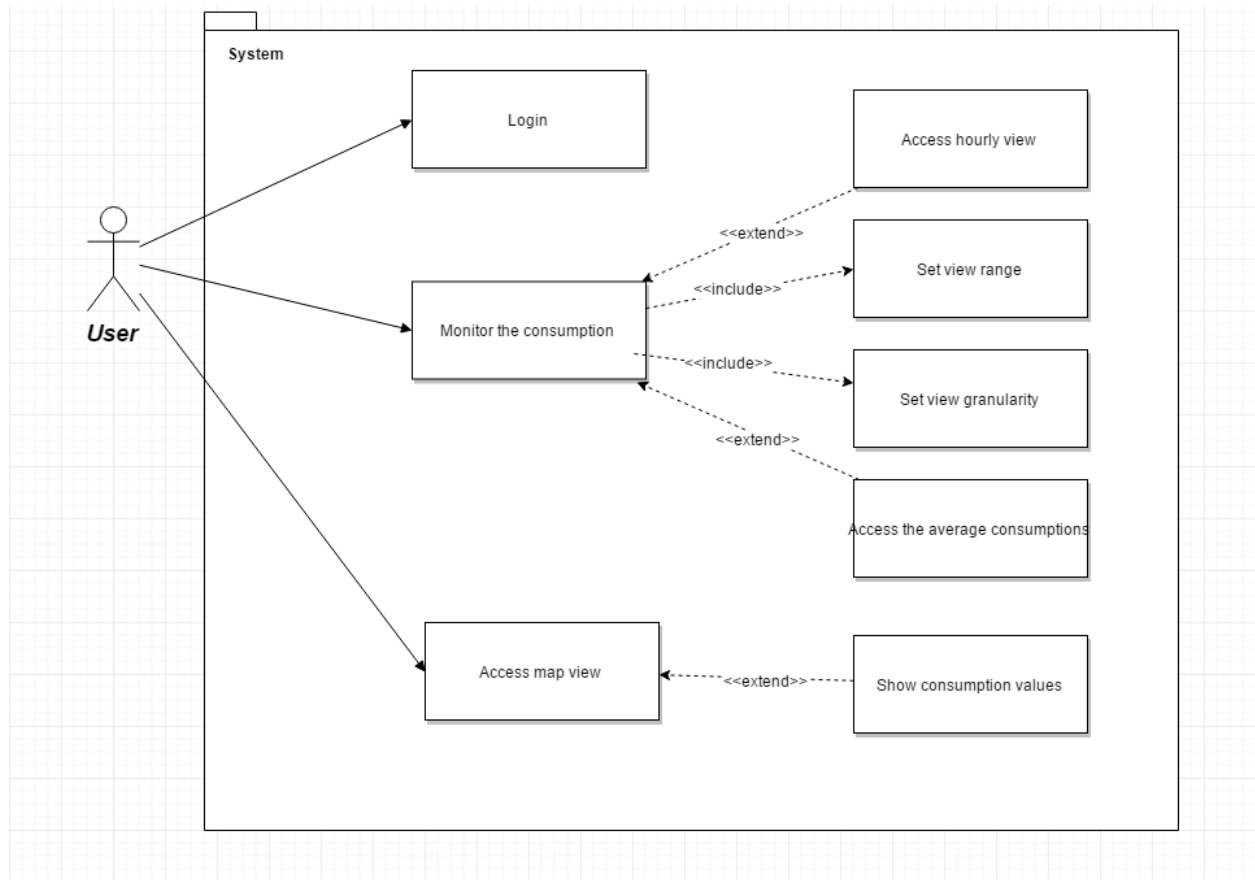
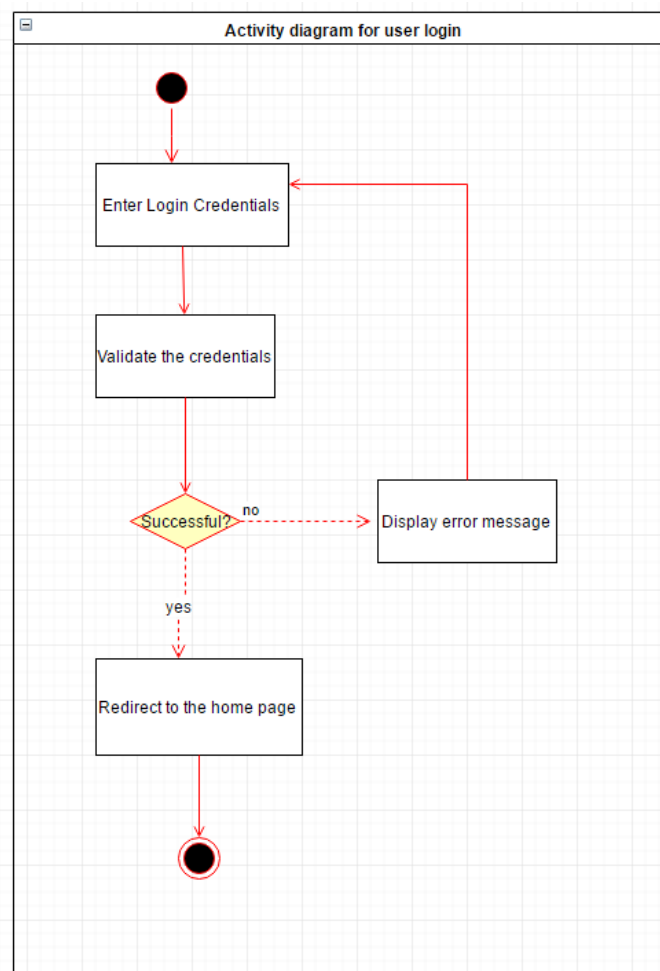


Figure 11. Use Case Diagram

**Table 1** “Login” specification sheet.

Title	Login
Purpose	To express how Water Consumption Portal users log-in to access the portal.
Pre-condition	The user must hold a valid username and password.
Post-condition	The user successfully logs into the application and accesses the home page view corresponding to his/her.
Workflow	<ol style="list-style-type: none"><li>1. The user receives an input form asking for username and password.</li><li>2. The user inputs his/her credentials.</li><li>3. If the credentials are correct, the user is authenticated to access the home page view.</li></ol>



*Figure 12. Activity diagram for the login use-case*

**Table 2** “Monitor the consumption” specification sheet.

<b>Title</b>	<b>Monitor the consumption</b>
Purpose	To express how the users monitor the past consumption data.
Pre-condition	Data must be present.
Post-condition	None.
Workflow	<ol style="list-style-type: none"><li>1. The user selects a date range to be visualized.</li><li>2. The user selects the granularity of the view type.</li><li>3. The consumption data of the user based on the selected date range and granularity is displayed.</li></ol>

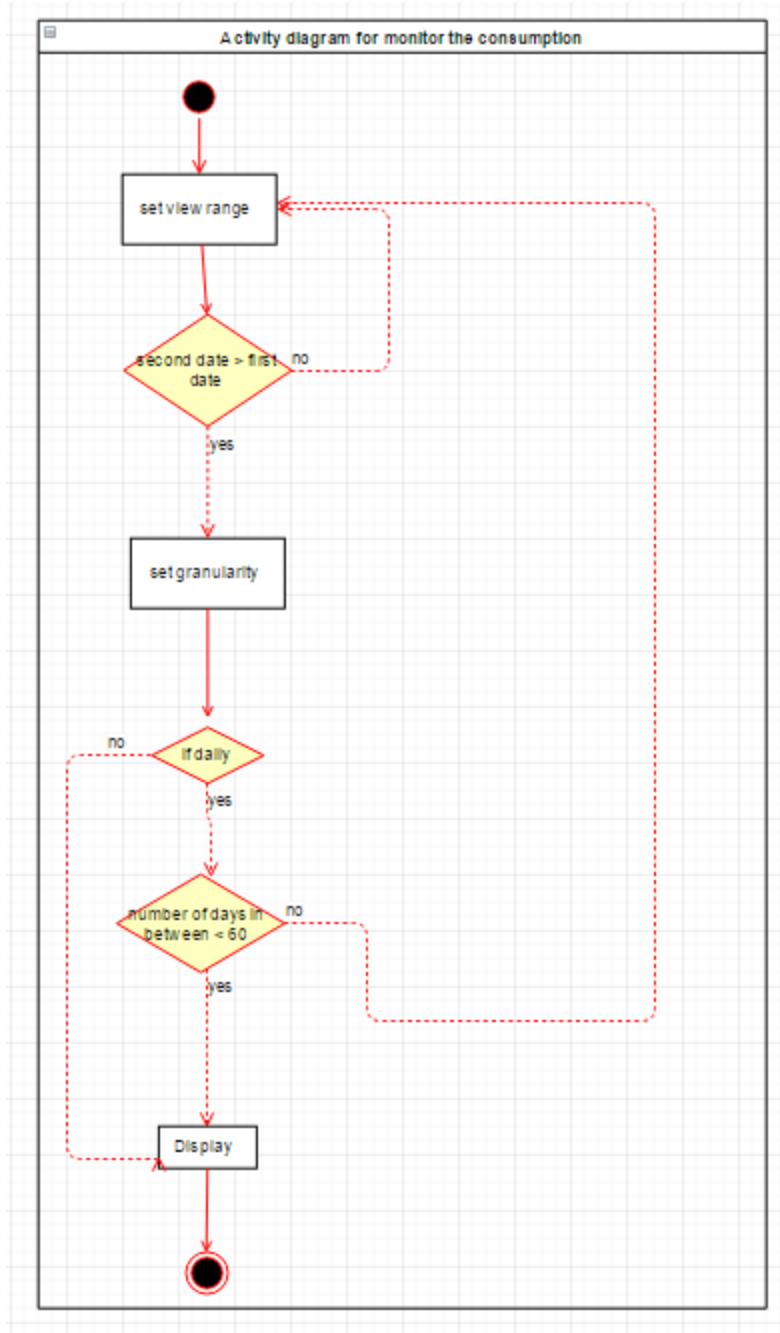


Figure 13. Activity diagram for the monitor the consumption use-case



**Table 3** “Access hourly view” specification sheet.

<b>Title</b>	<b>Access hourly view</b>
Purpose	To express how the users access the hourly view of the consumption data.
Pre-condition	1. Hourly consumption data must be present.  2. The daily view range must be selected.
Post-condition	The user can access the hourly-view of the consumption related to a selected day.
Workflow	1. The user clicks on a bar of the histogram.  2. The hourly view is displayed if it is supported otherwise an error message is shown.

**Table 4** “Access the average consumptions” specification sheet.

<b>Title</b>	<b>Login</b>
Purpose	To express how the user accesses the average consumption of the neighborhood or the average consumption hers/his.
Pre-condition	The user has accessed the average values.
Post-condition	The average lines are displayed.
Workflow	1. The user selects the corresponding check boxes to the average lines he/she wants to display  2. The average lines are drawn on the chart.

**Table 5** “Access the map view” specification sheet.

<b>Title</b>	<b>Login</b>
Purpose	To express how the user can have access to the map view.
Pre-condition	None.
Post-condition	None.
Workflow	1. The user clicks on the link to see the map view. 2. The map view is displayed.

**Table 6** “Show consumption values” specification sheet.

<b>Title</b>	<b>Login</b>
Purpose	To express how the user can view the consumption values of the addresses denoted by markers on the map.
Pre-condition	Markers must be present on the map.
Post-condition	The total, daily, weekly and monthly average consumption data are shown.
Workflow	1. The user clicks on the marker. 2. A pop-up displaying the current total, daily, monthly and weekly averages appears.



## Sequence Diagrams of the Application

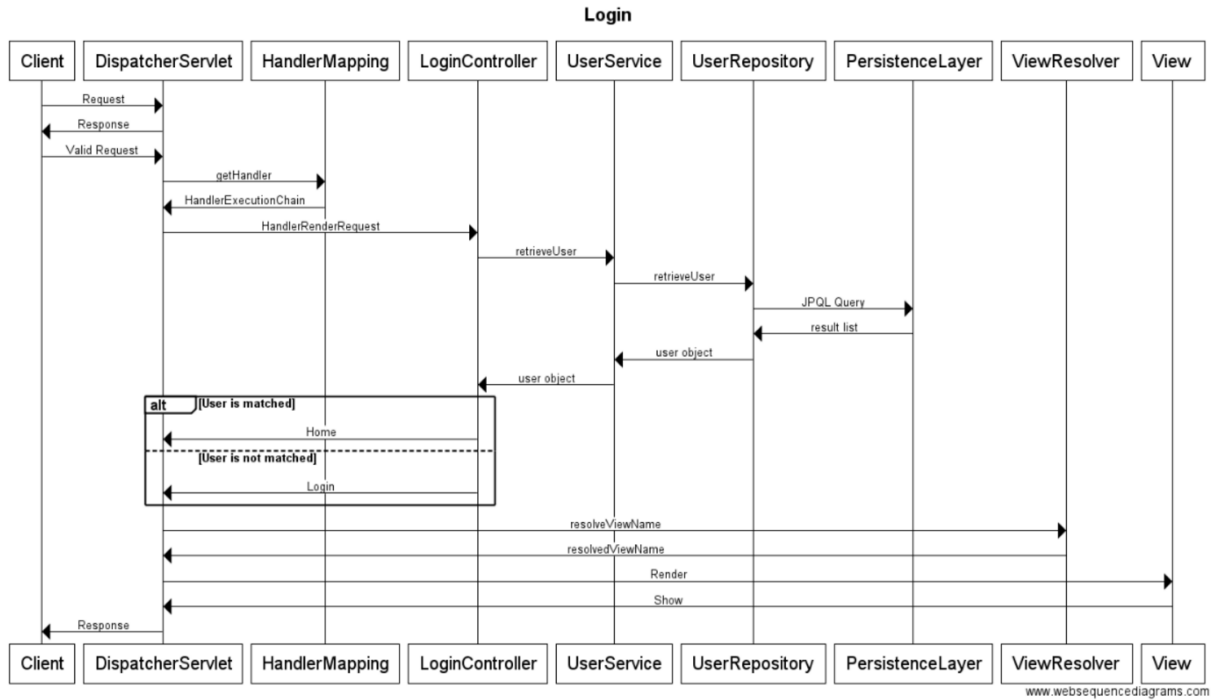


Figure 15. UML sequence diagram for login use case

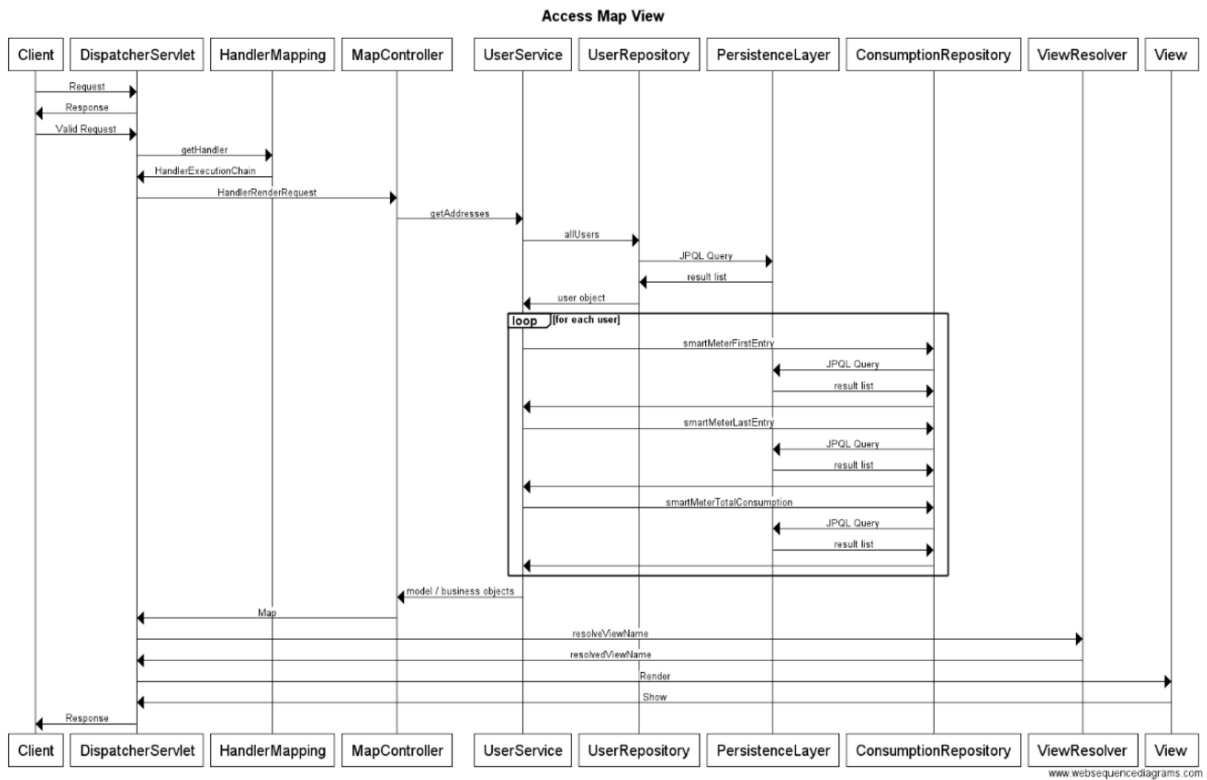


Figure 16. UML sequence diagram for access map view use case

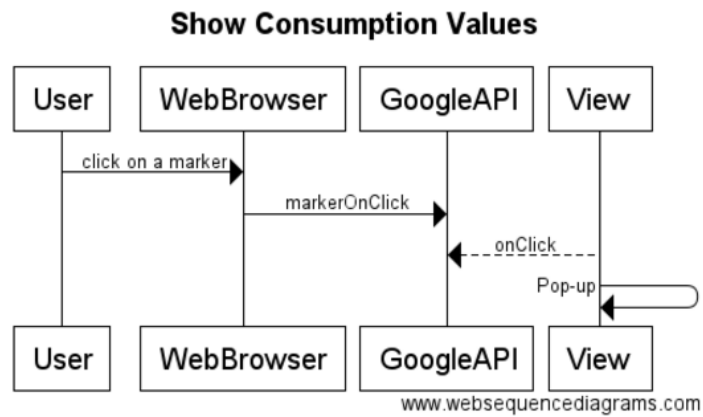


Figure 17. UML sequence diagram for show consumption values use case

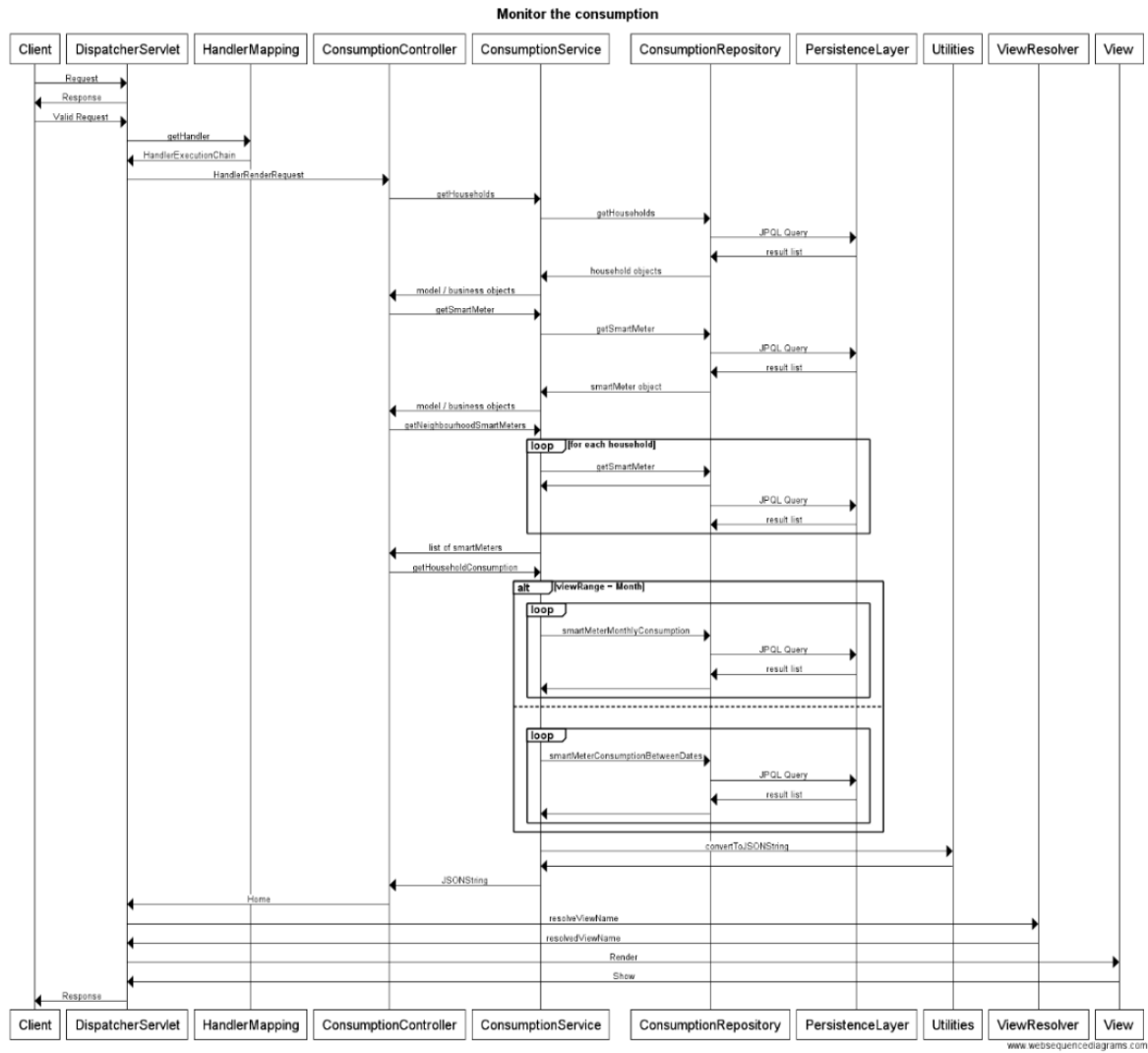


Figure 18. UML sequence diagram for monitor the consumption use case

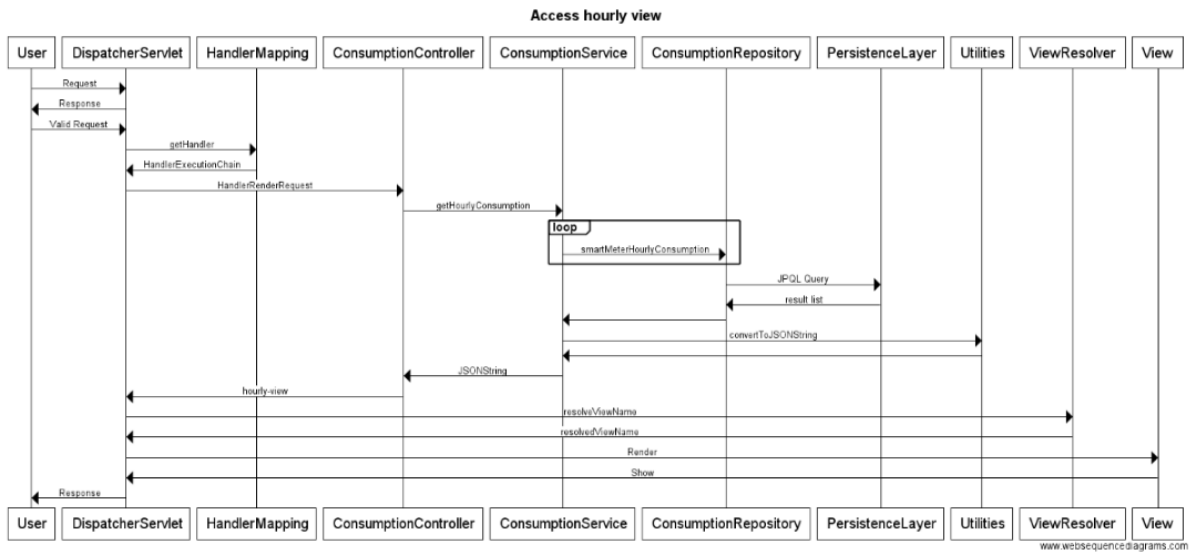


Figure 19. UML sequence diagram for access hourly view use case

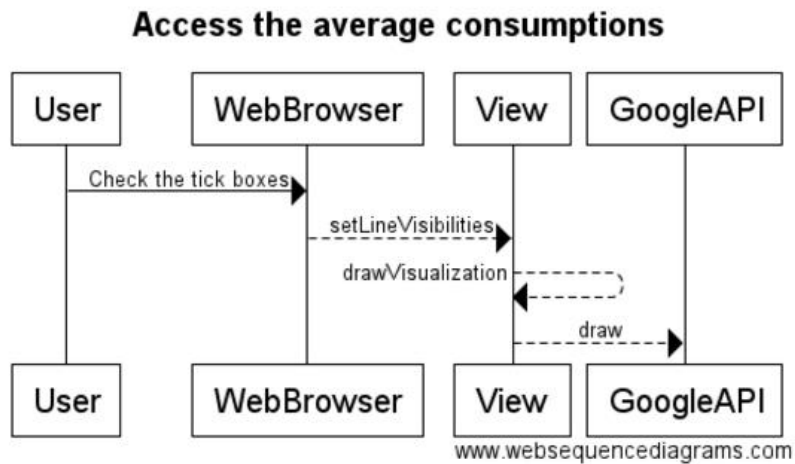


Figure 20. UML sequence diagram for access the average consumptions use case

## Data Model of the Application

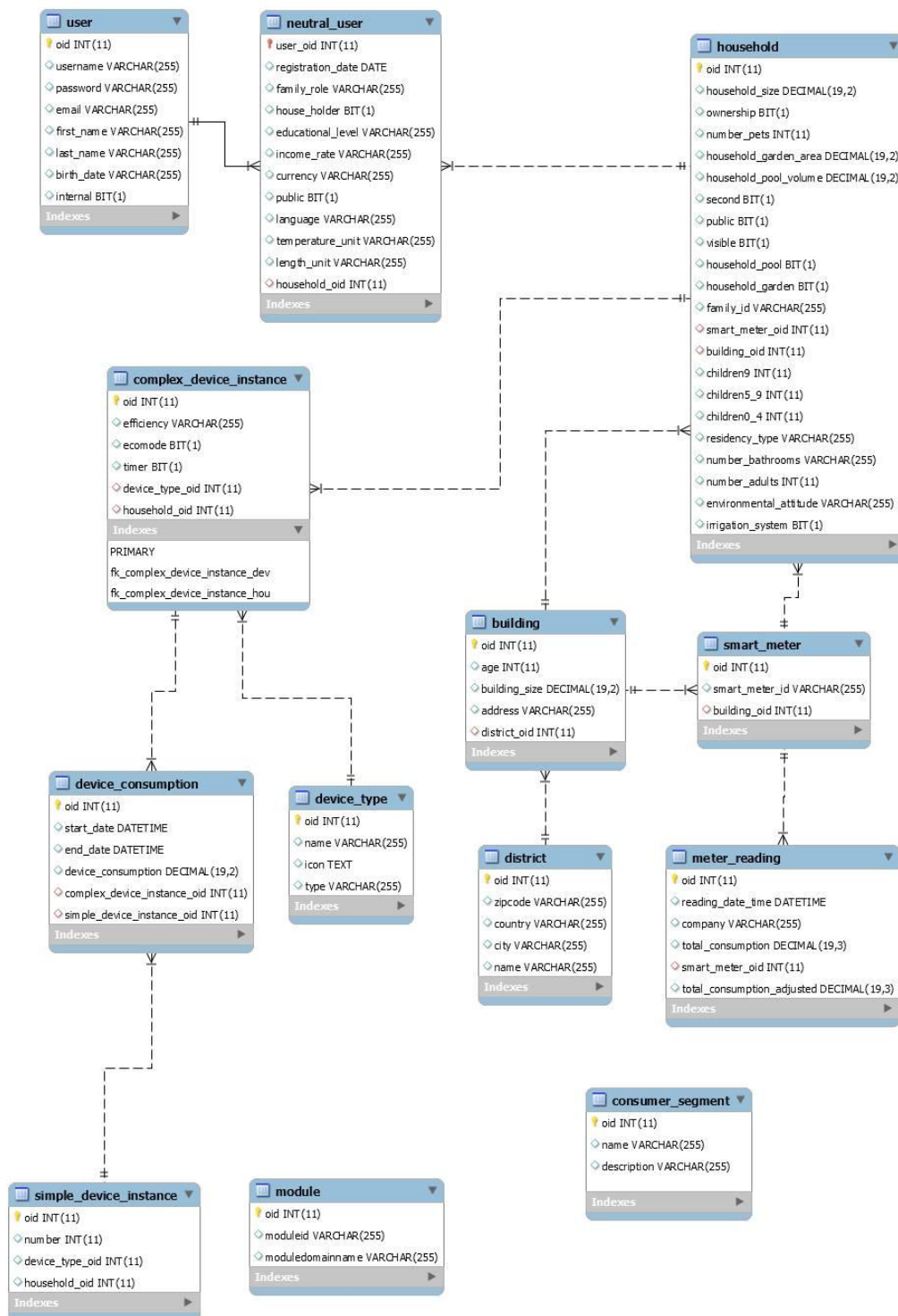


Figure 21. The Data Model



## Deployment Diagram of the Application

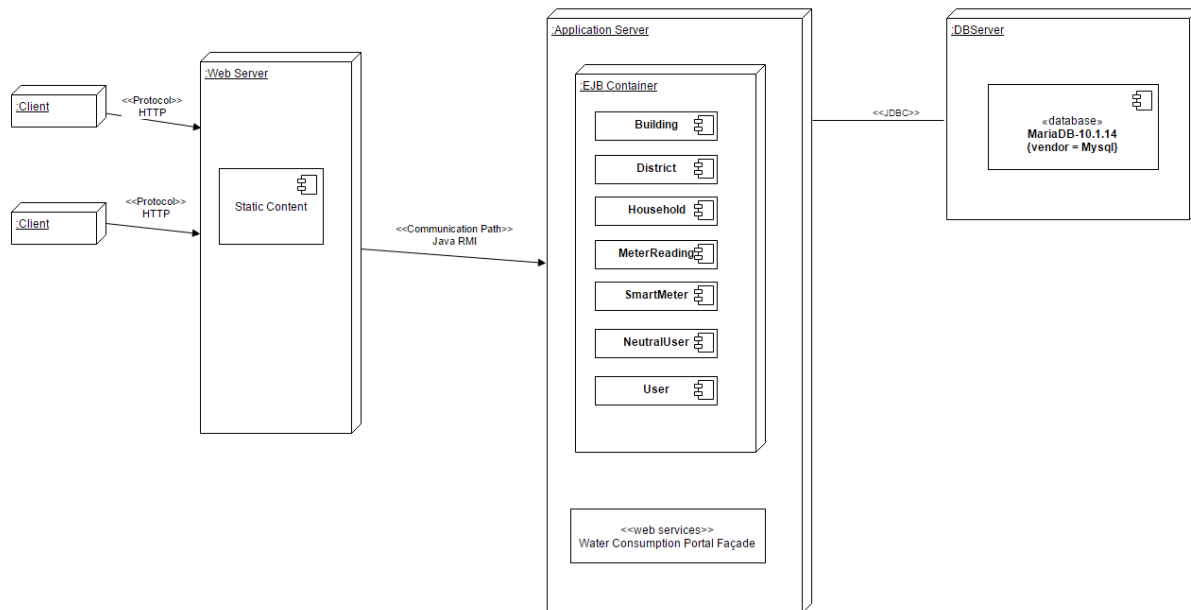


Figure 22. Deployment Diagram of Water Consumption Portal

## Motivation of the Web Service APIs chosen

### Google Chart API

The main reason for choosing Google Charts was to use the same chart tools Google uses, completely free. Besides, Google chart tools are powerful, simple to use and customizable. Google charts also do provide a rich documentation [2].

### Interaction with the Google Charts

The data to be visualized is retrieved and stored as java objects. Various classes are used in order to represent the input structure that the Google Chart is expecting. Then the java objects are converted into JSON using Jackson data binder library. Afterwards, the JSON object is added to the model as an attribute. During the loading of the home page, the JSON object is sent to the function in charge of the drawing the Google Chart. After user changes the dates or the view granularity, an asynchronous call is made to alter the chart.

```

$.get("consumptions?firstDate="
      + $("#firstDate").val() + "&lastDate="
      + $("#lastDate").val() + "&viewRange="
      + $("#viewRange").val(), function(data,
      status) {
    dataInJson = data.householdConsumption;
    $("#spinner_loader").removeClass();
    drawVisualization();
  })

```

Figure 23. Code snippet that asynchronously updates the chart

If the view granularity is set to the daily view and the user clicks on one of the bars. Then another asynchronous call is made to draw the chart on a pop-up window.

```

$.get("hourly-view?date=" + dateValue, function(data,
      status) {
    if (status == "success")
    {
      $("#hourly_chart_div").dialog({width: 800,height:600});
      drawHourlyVisualization(data.hourlyConsumption);
    }
    else
    {
      alert("non successful call");
    }
  })

```

Figure 24. Code snippet that asynchronously renders the chart on a pop-up window

```

function drawHourlyVisualization(hourlyDataInJson) {

    var data = new google.visualization.DataTable(hourlyDataInJson);

    var options = {
        title : 'Hourly Water Consumption',
        vAxis : {
            title : 'Amount [meters cube]'
        },
        hAxis : {
            title : 'Time'
        },
        seriesType : 'bars',
        series : {
            1 : {
                type : 'line'
            },
            2 : {
                type : 'line'
            }
        },
        colors : [ '#23AAE2', '#e2431e', '#f1ca3a' ],
        interpolateNulls : false
    };

    var chart = new google.visualization.ComboChart(document
        .getElementById('hourly_chart_div'));

    chart.draw(data, options);

}

```

Figure 25. JavaScript function that draws the Google Chart in the hourly-view

When the user checks the average checkboxes, the visibility states of the series that are sent to the chart are altered correspondingly.

## Google Maps API

The motivation of using Google Maps was due to their popularity and again the rich documentation. By using the Google Maps API, it is possible to embed Google Maps site into an external website, on to which site specific data can be overlaid [3].

### Interactions with the Google Maps

In order to use the Google Maps API, a standard API Key is generated. The key is included when loading the API. Using the key allows me to monitor my application's API usages in the Google API Console, it also enables access to the free daily quota [4].

```
<script async defer src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap" type="text/javascript"></script>
```

*Figure 26. Example of the key specification when loading the API*

The data to be sent to Google Maps API is retrieved as Java objects then - using Jackson data binder - the data is converted to JSON and later attached to the model as an attribute, as it is done in the Google Chart API.

## Google Maps Geocode API

The Google Maps Geocoding API is a service that provides geocoding and reverse geocoding of addresses. Geocoding is the process of converting addresses (like a street address) into geographic coordinates (like latitude and longitude), which you can use to place markers on a map, or position the map [5].

## Interactions with the Geocode API

For each marker to be drawn on the map, the Geocode API is asynchronously called to retrieve the coordinates from the addresses represented as string. After having drawn the coordinates on the map, the 'fitbounds' function of Google Maps API is called to fit all of the markers within the same boundary, lest the user zoom and move herself to find a good view.

```
for (var x = 0; x < addresses.length; x++) {  
    $.getJSON('http://maps.googleapis.com/maps/api/geocode/json?address='+addresses[x].address+'&sensor=false', null, function (data) {  
        var p = data.results[0].geometry.location;  
        var latlng = new google.maps.LatLng(p.lat, p.lng);  
        latlngbounds.extend(latlng); //for fitbounds  
        coordinateData = latlng;  
        drawMarkers(latlng, i++, map);  
        map.fitBounds(latlngbounds); //fitting the marker boundaries  
    });  
}
```

Figure 27. Interactions with the Geocode API

## Running JEE Code

The code is to be shown during the presentation. The code is complete of comments and a JavaDoc documentation for every java class of the project is generated.

```
public interface IUserRepository
```

```
Author:
```

```
anil
```

### Method Summary

All Methods	Instance Methods	Abstract Methods
Modifier and Type		Method and Description
java.util.List<User>		<b>allUsers ()</b> Retrieves all of the users
java.lang.Integer		<b>numberOfNeighbours (Household household)</b> Retrieves the numberOfNeighbours belonging to a Household
<b>User</b>		<b>retrieveUser (java.lang.String username, java.lang.String password)</b> Retrieves the user matched with the username and password

Figure 28. A sample JavaDoc for the IUserRepository Interface

## Motivation of the JavaScript Libraries

### Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web [6]. Bootstrap is used in all of the pages of the Water Consumption Portal Application to make the application scalable to various resolutions.

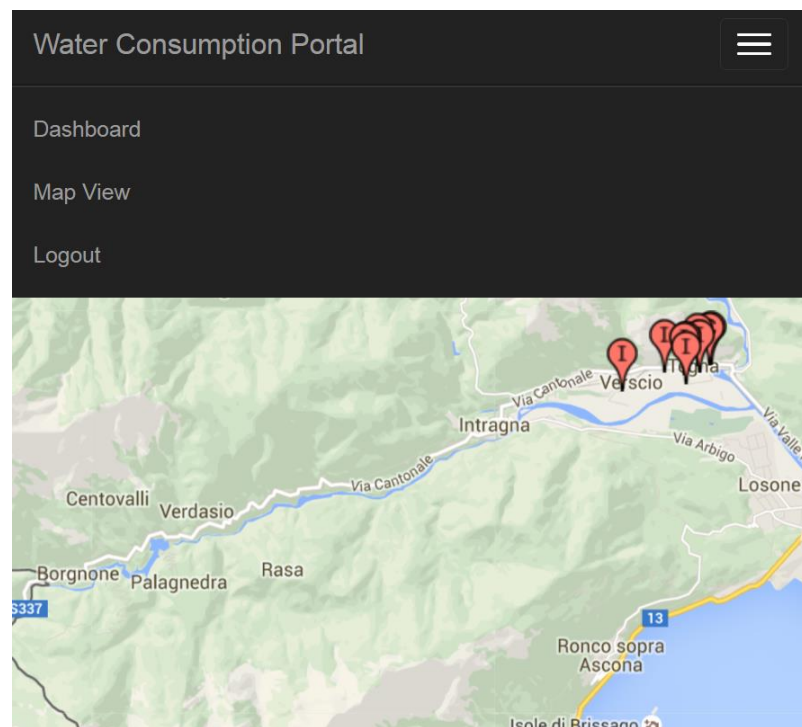


Figure 29. Application scales for a low resolution

## JQuery

JQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers [7]. In the project, JQuery is used almost in every JavaScript function of the project.

## JQuery-UI

JQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library [8]. Besides Bootstrap, JQuery-UI is used in order to enrich the user experience.

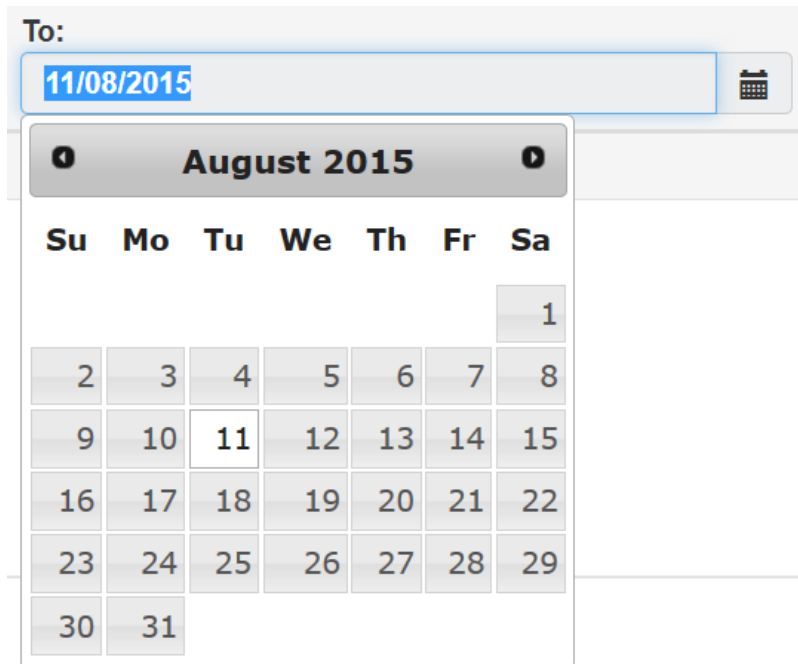
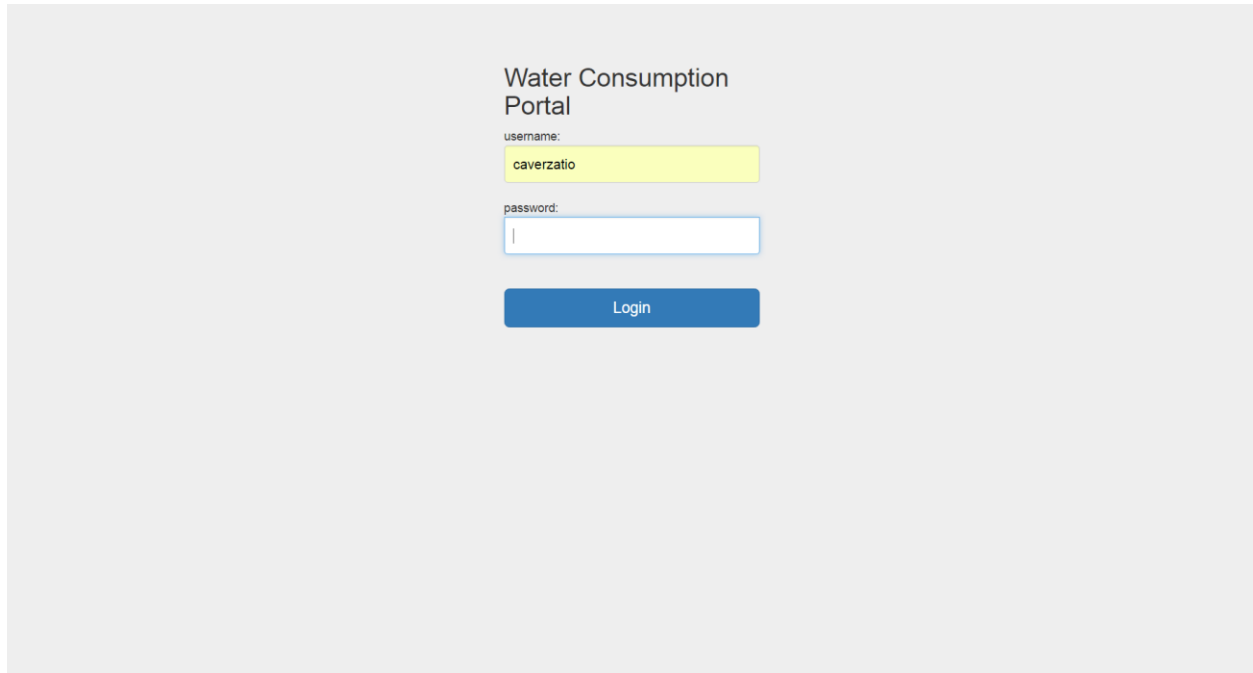


Figure 30. A usage example of JQuery-UI

## Usage Tests

### Login



The screenshot shows a login interface for the 'Water Consumption Portal'. The title 'Water Consumption Portal' is centered at the top. Below it, the label 'username:' is followed by a text input field containing the text 'caverzatio'. Below this, the label 'password:' is followed by an empty text input field. At the bottom of the form is a blue button with the text 'Login'.

*Figure 31. Login Page*

### Home Page

#### Initial Requirement

After the successful login, the user accesses the Home Page. The Home Page publishes the registration details of the user and a different icon denoting whether the consumption data are individual or common.



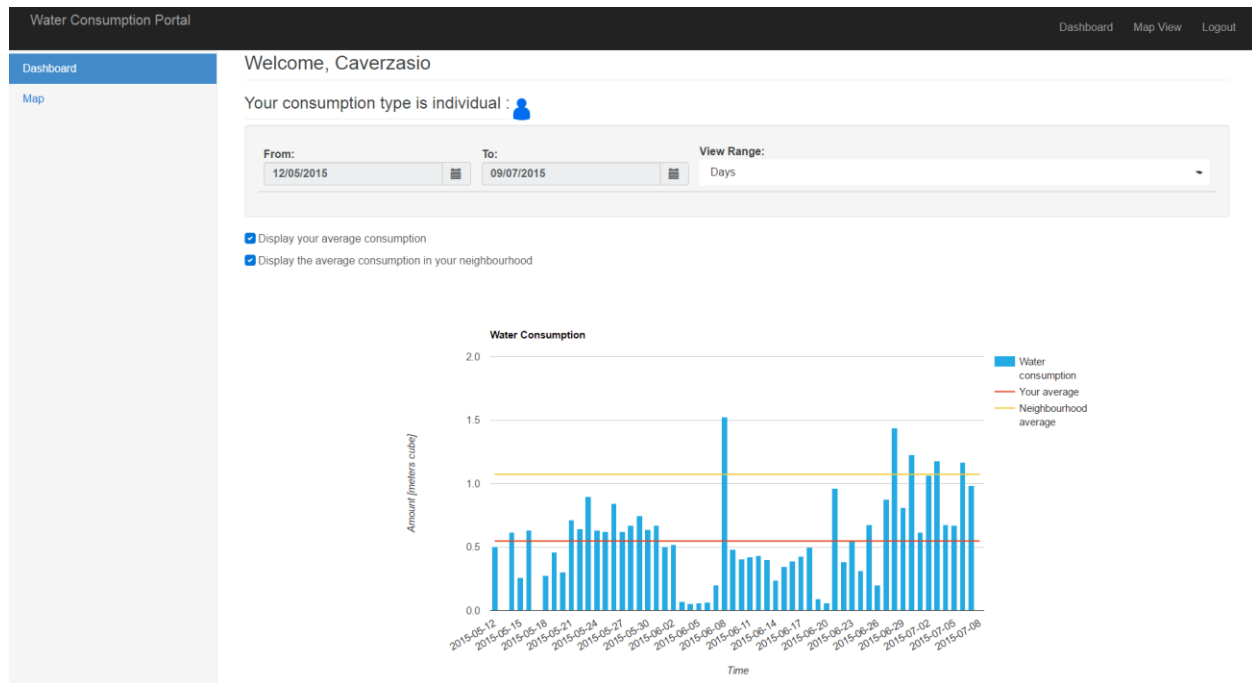


Figure 32. Home Page

## View Granularities

### Initial Requirement

By default, the histogram shows the consumption data at the granularity of the day. A menu allows the user to change the granularity at the week and at the month level.

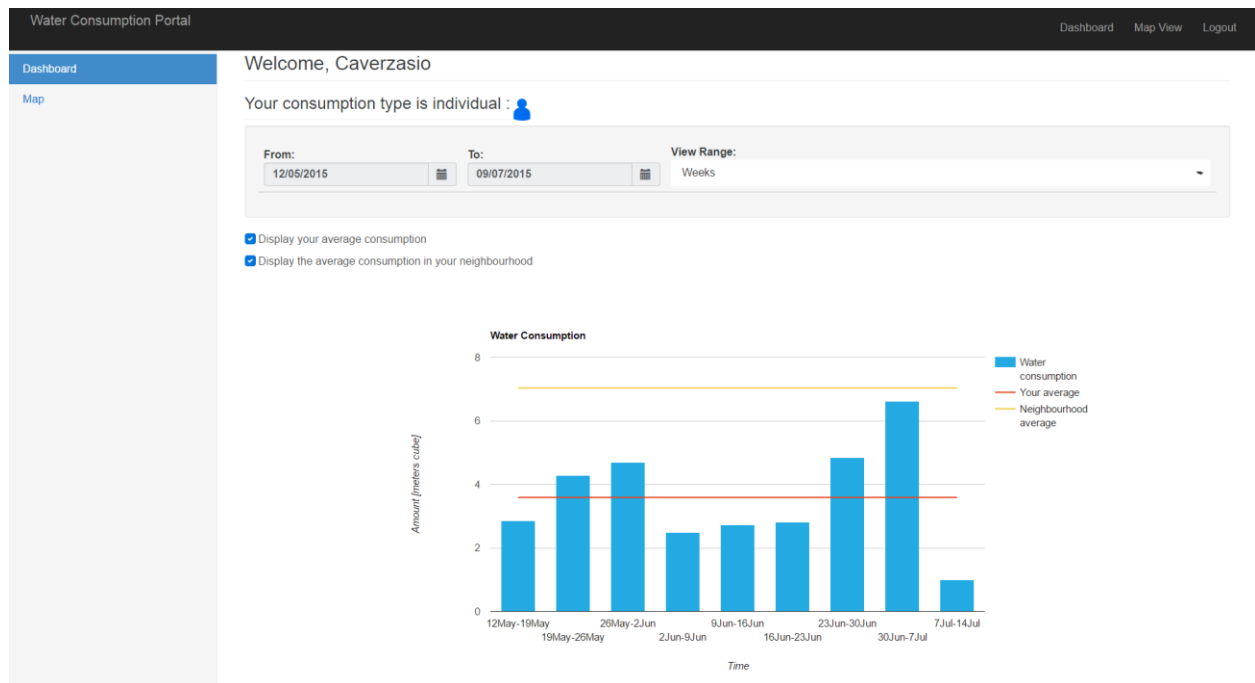


Figure 33. Weekly View

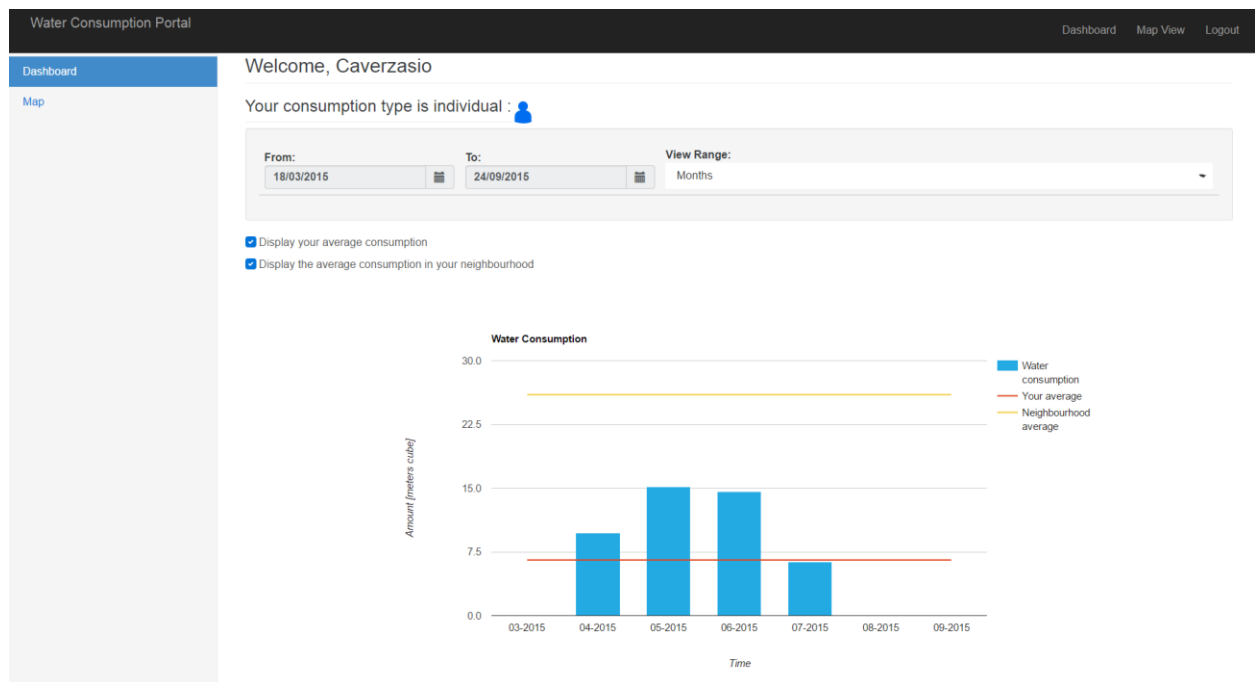


Figure 34. Monthly View

## Hourly View

### Initial Requirement

In the day view, clicking on a daily bar in the histogram opens a pop-up window with the breakdown of the consumption into the 24 hours, if the smart meter supports such a resolution.

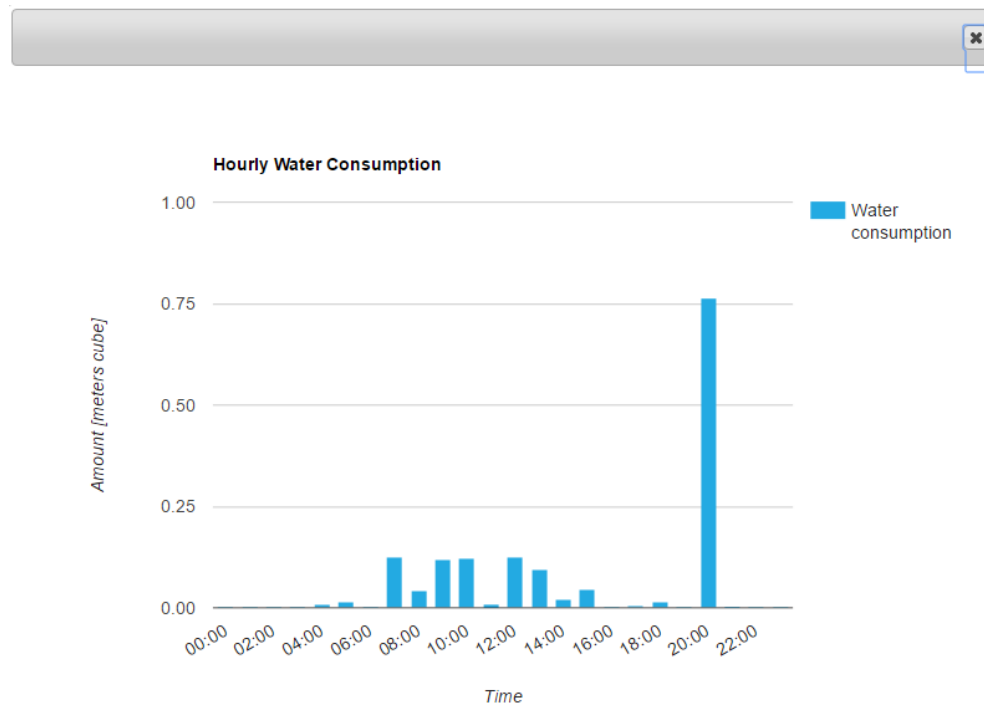


Figure 35. Hourly View

## Average Values

### Initial Requirement

Two tick boxes allow the user to enrich the histogram with two additional pieces of information: the average consumption of the user and the average consumption of the neighbourhood of the user. These are displayed as two horizontal bars crossing the histogram. The average consumption of the user is the average value of all the consumption data recorded in the database for the householdID associated with her. The average consumption of the neighbourhood is the average value of all the consumption data recorded in the database for the all the householdIDs associated with users having the same ZIP code as the current user.

- ☒ Display your average consumption
- ☒ Display the average consumption in your neighbourhood

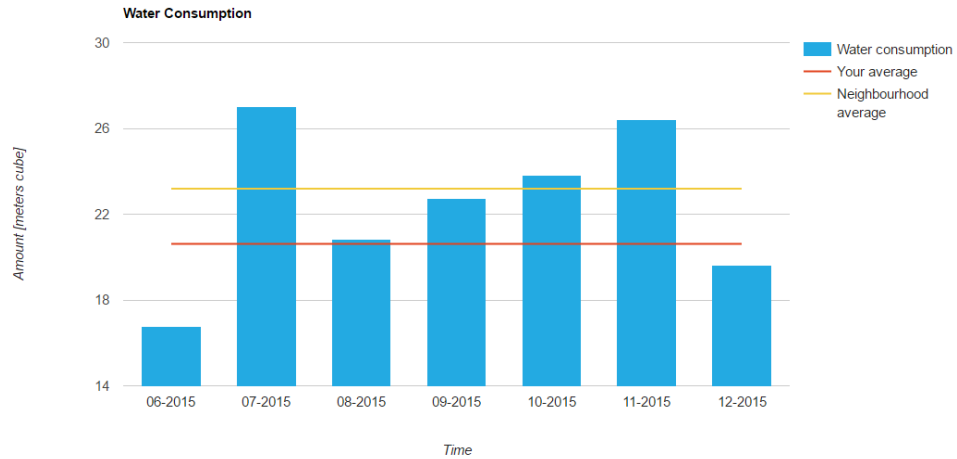


Figure 36. Average values

## Map View

### Initial Requirement

From the home page a link permits the user to access a map view. The map view shows the placement on the map of the registered users, based on their address. On the map page, a form allows the user to opt-in to be visualized in the map; to this end, she must insert a valid address matching the ZIP code inserted at registration time. Each opted-in user is displayed on the map with an icon that denotes if he has individual or common smart meter.

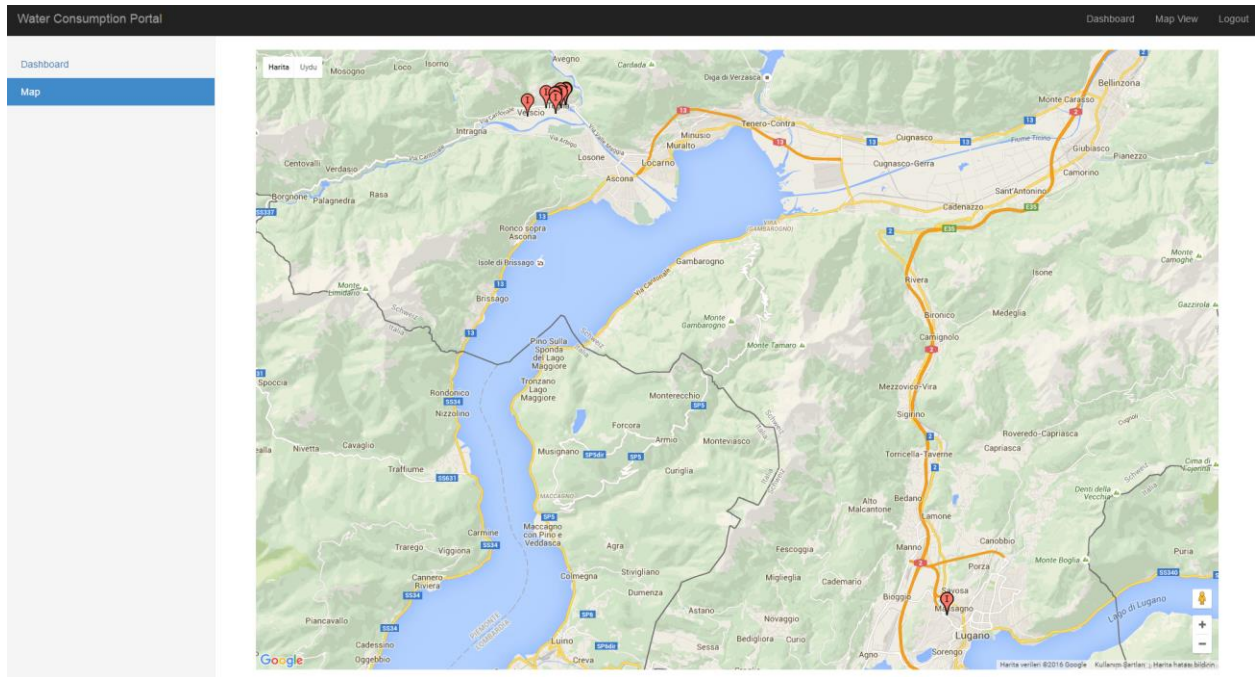


Figure 37. Map View

## Map View Details

### Initial Requirement

Clicking on the icon displays a pop-up window with the value of the current total, daily, weekly and monthly average consumption of the user.

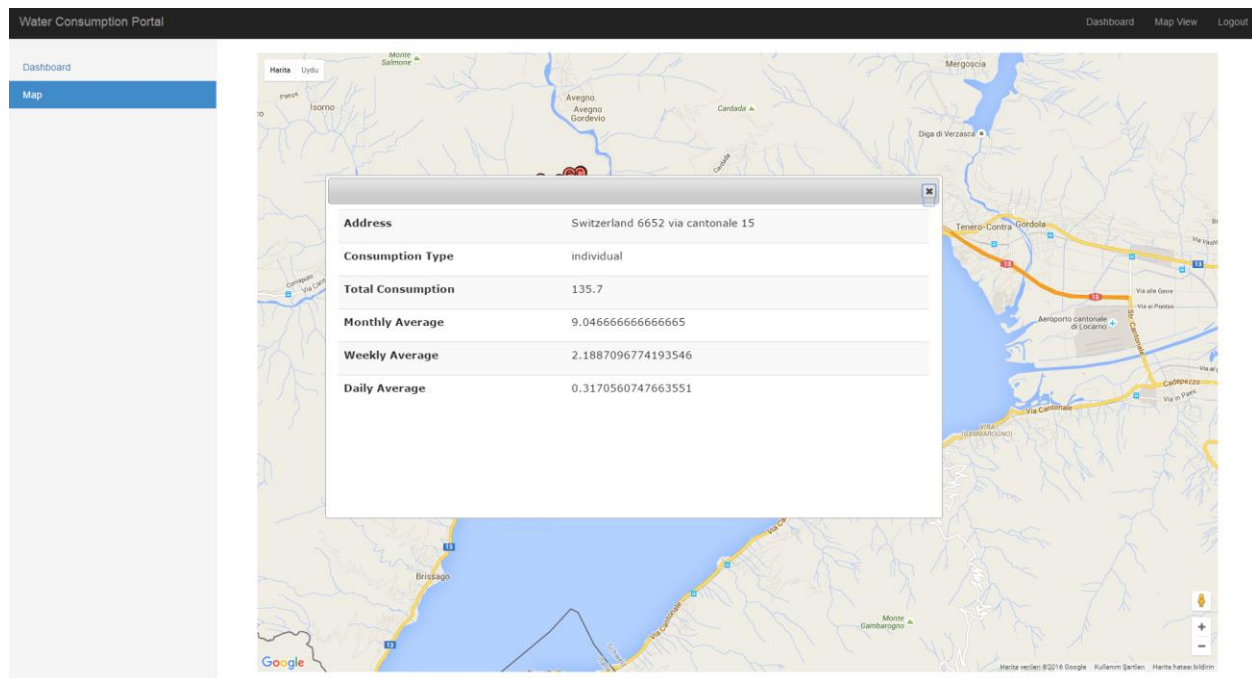


Figure 38. Map View Pop-Up Window

## Evaluation of the Experience

The most difficult part of the project was the learning part since this is my first web project. Before starting the project, I was not familiar with any web technologies. Besides Java I did not know any other technology or framework I used in this project (not even JavaScript). This is the main reason it took me so long to complete the project. I started the project in the late April. I spent weeks trying to grasp the ideas behind. I followed the material on the BeeP belonging to the bachelor course of Web Technologies. Then I started learning Spring MVC, Maven, JPA and Hibernate. After having grasped the idea, I started modifying the simple projects we did in the exercise sessions with Dr. Ilio Catallo. Then I started coding. I created a private git repository on BitBucket and used git version control system. I attended almost all of the tutoring sessions to take advice about the problems I was facing during the project. Even though I spent so much time to do this project, it helped me learn plenty of technologies and methodologies. At the end of day, I am very satisfied with the technologies I learnt and the experience I earned.

## Post-project Remarks

Unlike the consumption data, the map data is not stored in a session. Therefore, any time user refreshes the map page, new data are retrieved from Google Map Servers. Solving this issue is considered to be a post-implementation. Another post-implementation is to modify the data to be sent to the Google Chart on the client side. Google Charts does not perform any validation on datatables. (If it did, charts would be slower to render.) Therefore whenever user changes the view granularity or the view range, Google Chart is expecting a new JSON data. To provide that JSON data on the client side rather than the server side is considered to be a performance improvement. Another post-implementation is addressing the scalability of the application. With a high number of users, the application sends bigger data to the Google Maps API and Google Maps Geocode API. Since the application is using the free quote of the Google Maps API, this may turn into a problem. One possible solution is to upgrade the Google Maps API plan to the premium plan and pay for the extra network usages. Another possible solution could be showing a small portion of the registered users on the map. Such as the users within the same district or the users within the same street.

## Glossary

### Table of Acronyms

Serial	Acronym	Meaning
1	UML	Unified Modeling Language
2	AJAX	Asynchronous JavaScript and XML
3	MVC	Model View Controller
4	REST	Representational State Transfer
5	HTML	Hyper Text Markup Language
6	JSTL	Java Standard Tag Library
7	JSON	JavaScript Object notation
8	API	Application Programmers Interface
9	JPA	Java Persistence API
10	XML	Extensible Markup Language
11	JDBC	Java Data Base Connectivity

*Table 1. Table of Acronyms*



## References

- [1] The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.
- [2] <https://developers.google.com/chart/glossary>
- [3] Ian Rose. *"PHP and MySQL: Working with Google Maps"*. Syntaxxx. Retrieved October 13, 2014.
- [4] <https://developers.google.com/maps/documentation/javascript/get-api-key>
- [5] <https://developers.google.com/maps/documentation/geocoding/start>
- [6] <http://getbootstrap.com/>
- [7] <http://api.jquery.com/>
- [8] <https://jqueryui.com/>
- [9] Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera, Designing Data-Intensive Web Applications (The Morgan Kaufmann Series in Data Management System
- [10] Spring Framework Documentation, <http://docs.spring.io/spring-framework/docs/2.0.x/reference/mvc.html>
- [11] <http://agilemodeling.com/artifacts/deploymentDiagram.htm>