OULU UNIVERSITY OF APPLIED SCIENCES

Jeveen Shrestha

WEB APPLICATION DEVELOPMENT FOR BUILDING AUTOMATION DEVICE (HEATING SYSTEM) IN LOCAL NETWORK

WEB APPLICATION DEVELOPMENT FOR BUILDING AUTOMATION DEVICE (HEATING SYSTEM) IN LOCAL NETWORK

Jeveen Shrestha
Bachelor's Thesis
Spring 2016
Degree Programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Jeveen Shrestha
Title of the bachelor's thesis: Web Application Development for Building Automation Device (Heating System) in Local Network
Supervisor: Pekka Alaluukas
Term and year of completion: Spring 2016 Pages: 37

After doing a practical training in Ouman Oy in summer of 2015, I was provided a project to develop a web application that would communicate to their heating regulator. The application was called Chart-Viewer. The Chart-viewer was developed in order to make their customers able to monitor their heating system through their mobile device sharing the same network with their heating regulator.

The Chart-viewer was called so because the customers would view the overall heating architecture of their building, which is called Chart, through it. The chart is created using another web application called Chart-editor. A chart is created using one of the most popular technologies of HTML5, Canvas. Using the tools of canvas, an overall architecture of the heating system of a building or a house is created in the chart-editor. When the chart is created, it is saved in the JSON format and sent to a web-server called Oulink through the FTP protocol.

The Chart-viewer, which is created in the web-server, can communicate with the device using Ouman API using the JSONRPC protocol. One of the Ouman API methods, called 'getChart' method is used to get a chart from the web-server. On a successful call and rendering, the chart is displayed in the chart-viewer.

Displaying a chart is one of the many requirements of the application. The application can be used to change and monitor different system settings of the device, too. The application can show active alarms and different measurements like outdoor temperature, humidity, CO2 and many more. The application is full responsive and can be operated in mobile devices of every operating system and screen size.

Though the project is successfully completed for now, it is still a prototype, as it cannot be sent for production yet. Once the chart-rendering engine called JEngine is refactored and some changes are made in Ouman API, the application can be made ready for production.

PREFACE

It was almost the end of August 2015 and also of my practical training in Ouman Oy where I had been refactoring and modifying code in Chart-editor. After the long and frustrating 3 months I had been able to understand how the chart-editor worked and I had refactored and modified codes as required. The chart-editor now could be run locally as a desktop application.

I was working in my room when Mr. Kari Vengasaho, the R&D Manager of Ouman Kempele came to me and said that I had been given a project for my thesis. He said I had to develop a web application, which could display a chart and do other monitoring tasks for an Ouflex device. I do not remember being that happy before. I was excited as well as worried because I knew it would not be an easy task. Then I immediately emailed Mr. Pekka Alaluukas, who had been my teacher at Oulu University of Applied Sciences, and told him that I had been provided the thesis. He too was very happy and immediately agreed to be my tutoring teacher for the thesis. Then Mr. Juha Kylmänen was appointed my supervisor.

Now when the project has been done and I look back, I realize it would not have been possible without the help of these people. Therefore, I would like to thank my teacher and tutor teacher for this project, Mr. Pekka Alaluukas for providing the necessary instruction and guidance. I am very much grateful to Mr. Kari Vengasaho for all the support and guidance. He has always believed in my ability and me. Then, I also wish to thank Mr. Juha Kylmänen without whom the project would not have been possible. He helped me to understand how Ouman heating systems worked and how a communication system functioned. I also would like to thank my colleagues in R&D Kempele who had been my great inspiration. Last but not the least I would like to thank my friend Miss Sanisha Maharjan for being my great motivation all the time and for her never ending encouragement.

May 2016, Oulu

Jeveen Shrestha

TABLE OF CONTENTS

VOCABULARY

Building Automation: When a centralized Building Automation System controls the heating, ventilation, air conditioning, lighting and other systems of a building, it is called Building Automation. [1]

Chart: The overall architecture of a heating system of a building or a house is called a chart. It is drawn in a chart-editor.

Chart-editor: The web application, where heating system components and points can be dragged and dropped into HTML5 canvas to draw an overall architecture of a heating system, is called Chart-editor.

Ouflex-Device: It is a DIN-rail attached, freely programmable monitoring, and control and adjustment device. The device has 34 I/O points and diverse telecom and bus connections. In addition, the device offers 24 Vac and 15 Vdc voltage outputs. The display module of the device can be detached and moved. External I/O modules via bus connections can extend the number of the devices I/O points. [2]

Ouflex-Tool: It is a flexible programming tool. The programming of an Ouflex device takes place with a user-friendly Ouflex Tool. It includes a comprehensive process library, with which the errors taking place in programming can be decreased and creating the new equipment is clearly faster and easier than with traditional programming tools. [3]

Oulink-Device: It is a network adapter, which is providing a Modbus TCP/IP interface to an Ouflex Device. It also has a Web-server. [4]

Ounet: It is a web service of a centralized remote control for housing services. With it, the property automation can be monitored and controlled without visiting the site – in real time and easily. The use of the service requires that the property has Ouman automation equipment. In addition, a suitable Ouman remote connection is needed (Ouman 3G, Ouman Access, SMS). [5]

1 INTRODUCTION

Ouman is a building automation development, testing and production company. It develops automation devices and manufactures them in their own factories. Building automation is a necessity today. Basically, no building operates without it. The majority of Finnish housing companies have Ouman's automation. Adjusting the heating is one of the most important services provided by Ouman. Ouman has a long experience of the heating network operation of housing companies in Nordic circumstances. It also knows the operation of ventilation in housing companies and the building engineering of the property in one way or another. Its strength can be found in unit regulators, which operate independently and which can be connected to become a part of the total system when necessary. In addition to unit regulators, it manufactures system products, which are possible for managing even larger entities.

The chart-viewer is a small part of a bigger system, which consists of an Ouflex tool, an Ouflex device, a chart editor and an Oulink device. The Ouflex device is a heating regulator, which needs to be configured and programmed for customers. The necessary configurations and programming is done by the Ouflex tool, which is a desktop application. Once the configuration is done, the tool generates a JSON file. The file is a description file, which consists of all the necessary device points and information about the device. This file can be imported to chart-editor, which links the device points with the components of the heating system in the chart. The chart is saved and a JSON file is created. This chart is sent to Oulink through an FTP protocol. Oulink is a web-server. Now the chart-viewer fetches the chart, renders and displays it. Therefore, the system is completed with the development of the chart-viewer.

The main objective of this project was to make customers able to monitor their heating system in their mobile devices locally. The application can be accessed if the mobile device and the Oulink device share the same network. Users can read different values as well as write them, too. System settings can also be done.

As a web-server is involved in the system and the application has to run in all the operating system, the application was developed using a web technology. HTML5, CSS and JavaScript were used. The communication was done using Ouman API. The whole application frame was created using AngularJS. The Application was made responsive using Bootstrap. With the use of these technologies, a pretty decent application was developed.

2 WEB DEVELOPMENT TECHNOLOGIES

Gmail, Facebook, Google Maps, Google Translate and YouTube are some of many web applications, which are used by everybody every day. These applications have changed the face of the Internet in past few years. These applications have revolutionized the web development industries. They have made our life easier, the whole world narrower and all our friends and relatives closer. This chapter will explain what these web applications are.

Web applications run on web browsers like Google Chrome, Mozilla Firefox, Internet Explorer, Safari and Opera. These web browsers are called clients. They make request for resources like pages and images to web servers, where web applications exist. They present the applications and traverse them in the World Wide Web. There are two parties involved in communication, a client and a server. Therefore, web applications are also called a client-server software application.

One of the most frequently asked questions is, “What is difference between a website and a web application?” Though both have similarities, they have some distinctions. Websites are like brochures, which give information about a certain thing but a reader cannot interact with it. It is like sitting in a conference where an audience just listens to the presenter but is not allowed to interact with them. But web applications are interactive and lively. Users can click buttons, fill in forms and post things. Web applications expect users to interact and their behavior and appearance change according to users’ interaction. Facebook and YouTube are very good examples of web applications. Users can view post, like and comment the post or create their own post, upload images and videos in Facebook. YouTube plays users’ desired music and videos.

The following chapters will explain how these web applications are developed. A Web application development involves content, a presentation and design and interaction. HTML creates a web application and makes the content readable by browsers. CSS applies styles and design to the content and JavaScript makes the web application interactive.

2.1 HTML5

HTML or Hyper Text Mark-up Language is a computer language, which creates websites. Text or contents are written within the tags. Different tags do different things to the text or content within them.

Example 1

```
<b>This is a test</b>
```

In Example 1, a pair of tag wraps a sentence. This tag makes the text bold. There are tags for styling a text, listing a list, displaying images, separating a block of text into paragraphs, aligning a text, drawing using canvas, creating tables and many more. When it is done writing the HTML code, the file is saved in the HTML format, which means that the file will be given an .html or .htm extension. Now this file can be opened in web browsers. In the browser, the effect of different tags can be seen in the text but the tags themselves cannot be seen. Therefore, a web browser renders html files. Any kind of rudimentary text editor or advanced IDE can be used to create HTML pages.

With the advancement of web technology, web developers' needs and desires started increasing. In order to meet their needs and desires, HTML has been going through many improvements and revisions. HTML5 is an improved version of HTML 4.0. HTML5 introduced some new tags but it still uses many old tags, too. <header>, <section>, <article> and <footer> are some of the new tags. These tags clearly separate web contents into a header, sections and a footer. HTML5 now supports videos, music and geo-location and can even draw using canvas tools. HTML5 is an attempt of W3C to make features like geo-location, videos and canvas an integral part of web browsers. Before it, different kinds of plugins had to be used for these features. Now the development of a desktop application-like web application is made possible by the use of HTML5.

Only the use of HTML does not make a website beautiful. The website is still ugly and it is just a block of text. To make it look beautiful and attractive, styling is needed it is done by the use of CSS.

2.2 CSS

CSS or Cascading Style Sheets adds styles to html pages. It styles a text, fills colors, adds paddings and margins between and around a text, increases and decreases font sizes, enables to use desired font-families, aligns a text or a block of text, modifies a size of images, adds and removes borders. If HTML is plain and tasteless, CSS adds spice to it.

CSS styles are defined in ‘property: value’ format where property can be applied to html tags, classes and ids.

Example 2

```
p{align: center;}
```

In Example 2, p is called a selector, which is an html <p> tag, align is a CSS property and center is one of the values of align. On execution of the code, text, which is wrapped inside the <p> tag, will be center aligned.

CSS can be applied to an HTML page in 3 ways. They are In-line style, Internal or embedded styles and external styles. In-line style is written inside an html tag as an attribute.

Example 3

```
<p style= “color: black;”></p>
```

Internal or embedded style is written within a <style> tag in an html page. The <style> tag is a child element of a <head> element.

Example 4

```
<!DOCTYPE html>
<head>
<title>
<style>
        p{color: black;}
</style>
…
```

External style is written in a separate file with a .css extension. Afterwards, the file has to be linked with the html page using-

Example 5

```
<link rel= “stylesheet” href= “style.css”>
```

2.3 Bootstrap

CSS frameworks are the software frameworks developed using CSS to make web designing easier and more standard. These frameworks have grid systems, button styles and colors, tables, a set of icons, graphical interface components like accordions, tabs, a slideshow, a modal etc. and a web topography. Some CSS frameworks are more functional frameworks. They use JavaScript and have more features but they are design oriented. Web designing from the scratch can be tedious and time consuming. CSS frameworks make it easier and quicker. Bootstrap, Baseguide, Cascade framework, Chopstick, foundation, Material Design Lite Schema UI are some of the CSS Frameworks used by web designers.

“Bootstrap is the most popular HTML, CSS and JS framework for developing responsive, mobile first projects on the web” [6].

Bootstrap makes a front-end web development faster and easier. HTML and CSS based typography, buttons, tables, forms; navigation, image carousels and modal are included in it. It has an amazing grid system, which helps in responsive web design.

“Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops” [7].

To get started with Bootstrap, one needs to download a bootstrap CSS and JavaScript files from the official website or include a CDN within the <script> tags inside the <head> element.

Example 6

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href=
"http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" />

<!-- jQuery library -->
<script src=
"https://ajax.googleAPIs.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src=
"http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
```

Then to ensure a correct rendering of web pages in mobile devices, a <meta> tag is added inside the <head> element.

Example 7

```
<meta name= "viewport" content= "width=device-
width, initial-scale=1">
```

Now it is possible to start using bootstrap. First, the whole page is wrapped inside a container by adding a container class to the HTML element, which is required to be at the center.

Example 8

```
<div class= "container"></div>
```

Similarly different classes can be added to different HTML elements for different design and styles.

Example 9

```
<table class= "table"></table>
<button class= "btn"></button>
<img src= "…" class= "img-responsive">
```

Bootstrap is very popular for its gridding system, which makes responsive design possible. Creating page layouts through series of rows and columns uses Grid.

Example 10

```
<div class= “container”>
<div class= “row”>
<div class= “col-xs- 6 col-sm- 6 col-md- 6 col-
lg- 6”></div>
<div class= “col-xs- 6 col-sm- 6 col-md- 6 col-
lg- 6”></div>
</div>
```

In Example 10, the way the grid system works is shown. First, the content is wrapped inside a container. The content, which is required in columns, is given col classes. Columns should be placed as immediate child elements of the element with the class row. The maximum number of 12 columns can be used in a row. Bootstrap also has classes for colors, topography, text modifications, font styling, alignments, listing items and many more.

2.4 JavaScript

JavaScript is a scripting language used to make web pages more interactive.

“JavaScript is the programming language of the Web. The overwhelming majority of modern websites use JavaScript, and all modern web browsers – on desktops, game consoles, tablets, and smart phones – include JavaScript interpreters, making JavaScript the most ubiquitous programming language in history. JavaScript is part of the triad of technologies that all Web developers must learn: HTML to specify the content of Web pages, CSS to specify the presentation of web pages, and JavaScript to specify the behavior of web pages ” [8].

As mentioned above, JavaScript is a part of 3 technologies used to develop a web-application. HTML adds content to the web page, CSS designs and styles the page whereas JavaScript gives life to the web page. The web page become interactive- buttons start clicking, pages start behaving differently with each click of a button, images move, pages change and color change. It can dynamically change the content and style of HTML elements.

JavaScript can be included in a web page in two different ways - internally and externally as in CSS. Once the script is added, it runs in the browser line by line starting from the top and finishing to the bottom.

Example 11

```
<script>
        alert(“Hello, world. ”);
</script>
```

Example 11 shows how Internal JavaScript is written in a web page.

Example 12

```
<script src=”script.js”></script>
```

Example 12 is an example of how an external script is added to a web page. An external JavaScript is a text file with a .js extension, just like an external CSS resource with a .css extension. JavaScript file is added to a page by including it within the <script> tag.

2.5 AngularJS

Writing an application from scratch can cause difficulties in maintaining in a long run, therefore using a framework is a good idea. AngularJS is a Single-Page Application Framework.

“AngularJS, commonly referred to as angular, is an open-source web application framework. Google and the community that assists with creating Single-Page Application maintain it. SPA consists of one HTML page with CSS and JavaScript on the client side. The goal of angular is to simplify both development and testing of web applications by providing client-side model-view-controller (MVC) capability as well as providing structure from the entire development process, from design through testing. ” [9].

To start using Angular, a reference to an angular script is done in an HTML page as shown in Example 13, an ng-app directive is added to the <html> tag or the part of application, which is going to be an angular application as shown in

Example 14. Then data binding is done using directives. In Example 15, an ng-model is an angular directive and the name wrapped within curly brackets is a data binding expression.

Example 13

```
<script src= “js/angular.js”></script>
//OR
<script
src=“https://ajax.googleAPIs.com/ajax/libs/angularjs/[version]/angular.min.js”></script>
```

Example 14

```
<!DOCTYPE html>
<html ng-app>
….
</html>
```

Example 15

```
Name : <input type= “text” ng-model= “text”/> {{
name }}
```

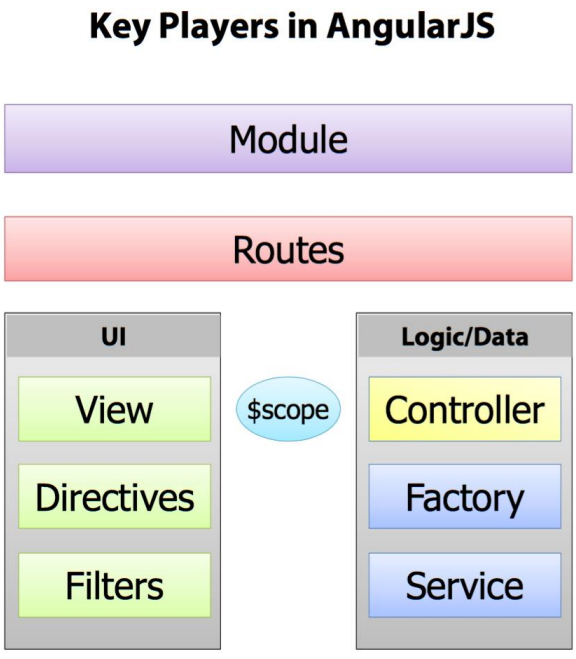

FIGURE 1. *Building Blocks of AngularJS*

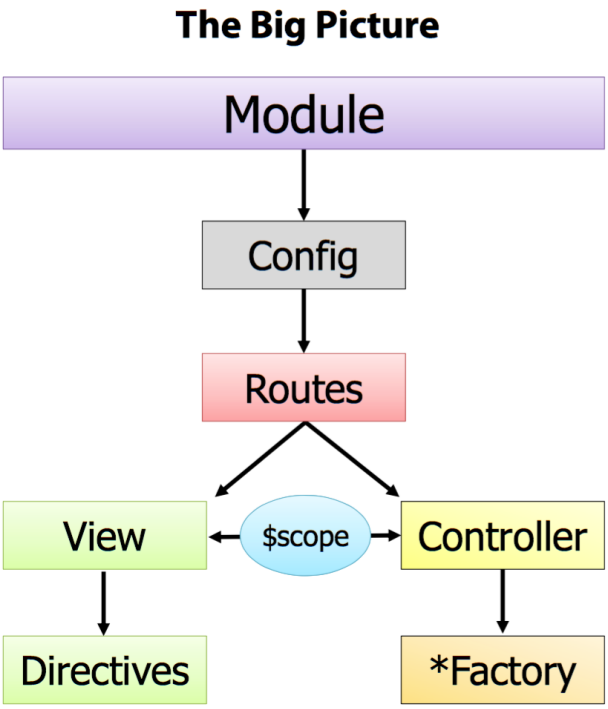

FIGURE 2. *The Big Picture*

FIGURE 1 and 2 show the building blocks of AngularJS. **Modules** are the containers for all other blocks – controllers, routes, factories / services, directives and filters. In Example 16, an angular module is created and it is named myApp. The variable, myApp can create controllers, services, directives and filters as shown in Example 16.

Example 16

```
var myApp = angular.module("myApp", []);

myApp.service("myService", [function(){
        //code
}]); // service, myService is created

myApp.controller("myController", ['$scope', func-
tion($scope) {
        // code
}]); // controller, myController is created

myApp.directive("myDirective", [function() {
        //code
}]); directive, myDirective is created
```

```
myApp.directive(“myFilter”, [function() {
        //code
}]); filter, myFilter is created
```

Factories and Services are used to make the RESTful calls and share data between controllers. Factories are singletons and used to handle custom logic. In programming, singletons are those classes, from which only one object or instance can be created. In AngularJS, a factory or a service returns only one object therefore, it is a singleton. In Example 17, the factory variable is an object, which is returned by myService service.

Example 17

```
myApp.service(“myService”, [function(){
        var factory = {};
        factory.name = “John Smith”;
        return factory;
}]);
```

Controllers are the “brains” for a view. They retrieve data from a factory or a service and store it, handle events triggered by the view, know how to handle custom logic and rely on the $scope object to interact with the view. $scope is the “glue” (viewModel) between a controller and a view. In Example 18, myService is passed to myController as an argument. The name object in myService is now assigned to $scope.name. As $scope is viewModel between controllers and views, the name variable can be binded to template in view as shown in Example 19

Example 18

```
myApp.controller(“myController”, [‘$scope’,
‘myService’, function($scope, myService) {
        $scope.name = myService.name;
}]);
```

Views render the user interface. They content HTML and CSS. Views bind data provided by a controller via the $scope object and use directives to enhance HTML and render data.

Example 19

```
<h2>{{name}}</h2> // result -> John Smith
```

Routes have a unique path each. Routes reference a controller and views. They can include route parameters, for example /customers/:customerId.

3 TYPES OF WEB APPLICATION

According to the article published by W3C on 26th May 2010, there are 4 types of web application architectures- Client-side Static Mash up, Server-side Static Mash up, Client-side Dynamic Mash up and General Web Application Architecture [10].

3.1 Client-side Static Mash up

In this type of architecture, the web-application is also called widgets. A client to run them locally downloads these widgets. Widgets are customized according to the data sent by a client to the widget source, which is then downloaded. Google Maps is an example of this kind of web application.

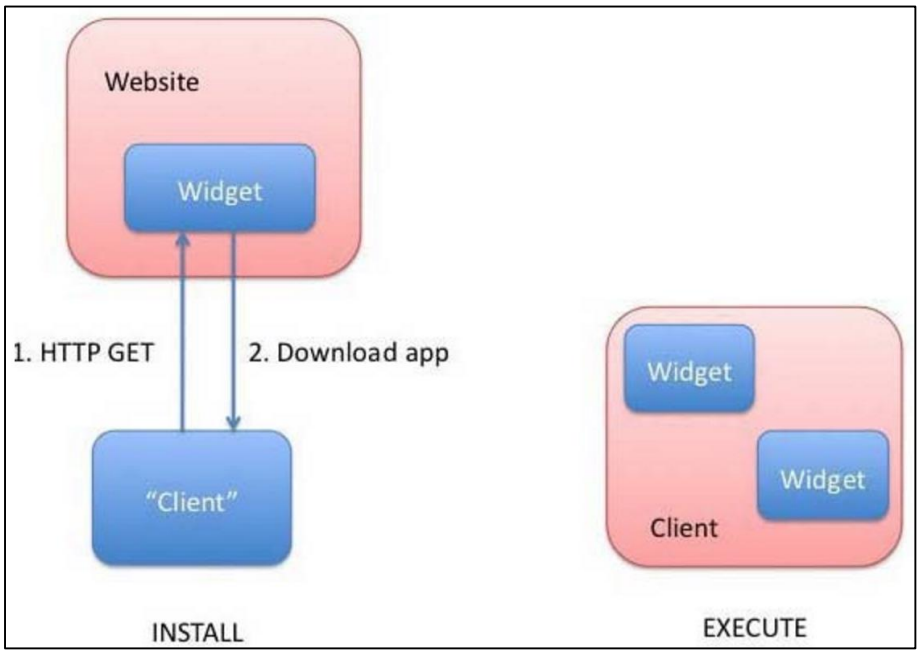

FIGURE 3. *Client-side Static Mash up* (W3C, 2010) [10]

3.2 Server-side Static Mash up

This type of architecture does not necessarily need a separate downloading process as type 1, but the widgets need to be installed to a container. These widgets are on the server side. By the means of iframes, contents of many websites are combined in a website, after the static validation. iGoogle is an example of this kind of web application.

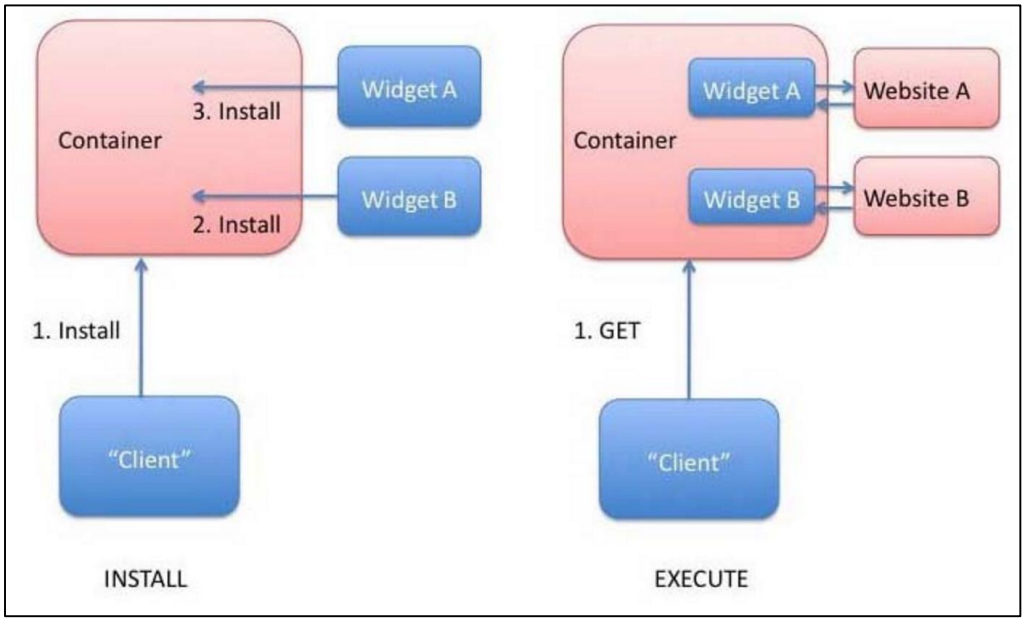

FIGURE 4. *Server-side Static Mash up* (W3C, 2010) [10]

3.3 Client-side Dynamic Mash up

In this type of architecture, a client creates content, which then requires content from other websites. These contents from other websites are dynamically assembled on the client side.

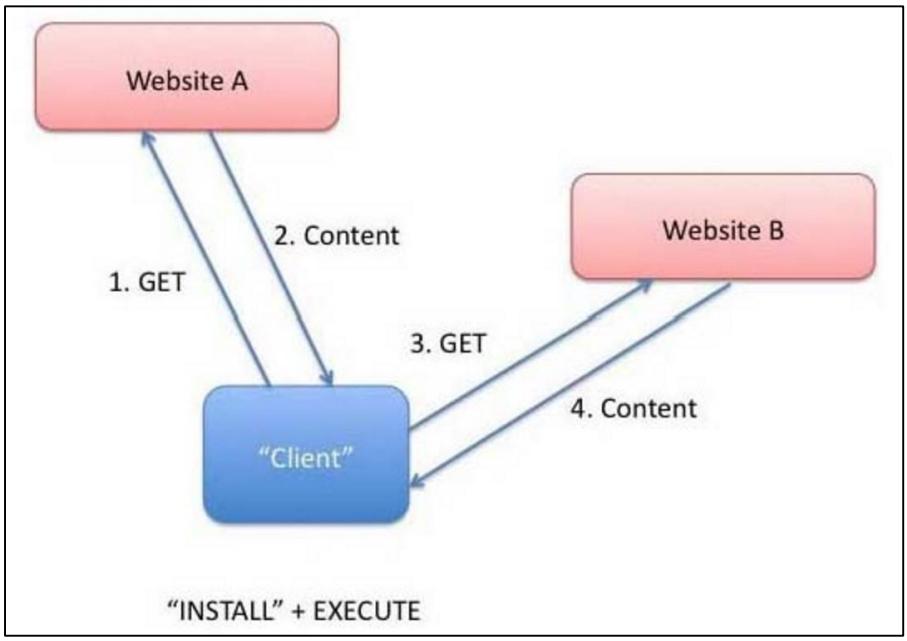

FIGURE 5. *Client-side Dynamic Mash up* (W3C, 2010) [10]

3.4 General Web Application Architecture

In this type of architecture, a user starts an application on a browser. This application may take help from other websites to show the data required by the user. The websites exchange data between each other. For example, when a user tries to book an airplane ticket on an airline's website, they can be shown tickets of other airlines, too. Here the ticket information is shared between websites.

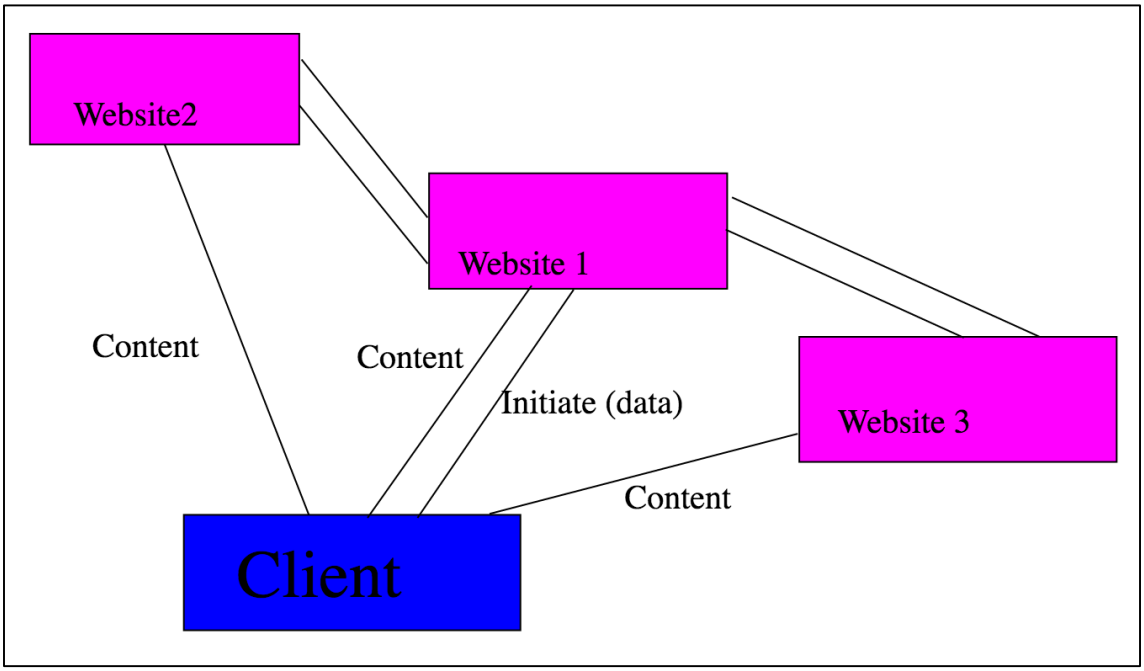

FIGURE 6. *General Web Application Architecture* (W3C, 2010) [10]

4 LOCAL CHART-VIEWER

Local Chart-viewer is a web-application developed by using HTML, CSS and JavaScript. After a 4-month-long reading and debugging the local chart-editor, they came up with an idea of developing also a local Chart-viewer, which would display and monitor the chart created on the local chart-editor. They came up with the name Local chart-viewer because they already have a chart-viewer in the Ounet application running remotely, which is doing the same work as it would. The Ounet application is a huge project of Ouman. The Chart-viewer is just a small part of this application. This application runs on a remote server so that the users could manage their systems remotely. The local chart-viewer project started with an objective to make it run locally. It should operate in the same network to which the Ouflex device is connected. As the Ouflex tool, which is a device programming application, is a desktop application and a local chart-editor could run as a desktop application too, the chart-viewer also required to run locally too in order to complete the system. Hence, the project was initiated.

During the last four months while reading and debugging codes of the chart-editor, I had become well familiar with how the chart-viewer should work. I had also become accustomed to Ouman-API. Therefore, I was given the opportunity to develop it as my thesis project.

4.1 Objectives

As mentioned before, the main objective of this project was to develop a locally running web application, which could display and monitor a chart. The user should be able to check the status of their heating system in the network sharing with the device. The application should have responsive design so that users could operate it in any mobile device using any platform. Another important objective of this project was code refactoring. As the application would be using already existing logics and codes, this was the best time to refactor the codes. The code should be clean, readable, maintainable and expandable.

According to Mr. Juha Kylmänen, the supervisor of my thesis, there has always been a need for a graphical touch interface. Customers wanted a shiny GUI for the devices but were not prepared to pay manufacturing costs of such kind of screen. Just knowing that customers wanted this kind of GUI was not enough to start the project, it would require a real order from someone.

Mr. Kylmänen said that it was not possible or practical to create a hard coded touch interface to some specific hardware. It would be difficult to maintain. As the Oulink Server has a Web Server and it can communicate with API and there has been a quite versatile HTML graphical editor and viewer, it would make sense if these things were used to create a GUI editor.

Once a graphical interface had been created with a chart-editor, it could be transferred to a website and run in any personal mobile device. Therefore there would be no worries for any hardware. The end user could be able to use the UI with a device they would prefer Mr. Kylmänen said.

4.2 Web Interface Design

The Web Interface Design was required to be simple, usable and understandable by users. The design was needed to be responsive, so that it could be run in any device, or on any platform. This was the reason why Bootstrap was chosen to design the application. AngularJS was then chosen as a JavaScript Framework. As I have more experience with this framework and I was quite comfortable working with it, it was my first choose. With the use of Bootstrap and AngularJS the development was faster and easier.

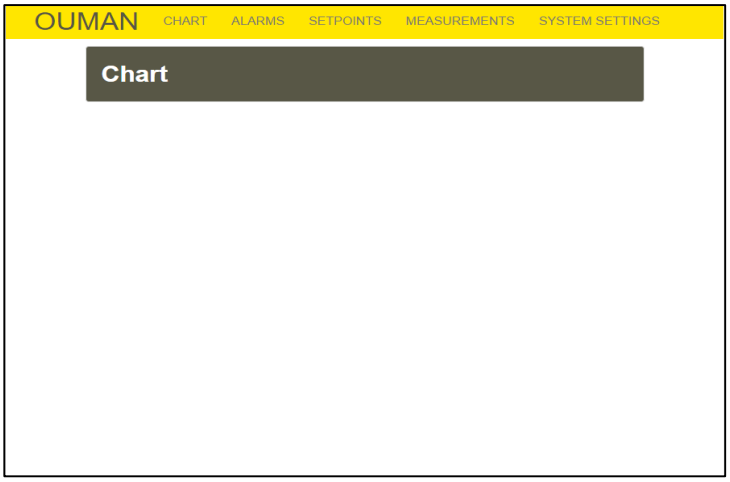

FIGURE 7. *Windows view of the Chart-View*

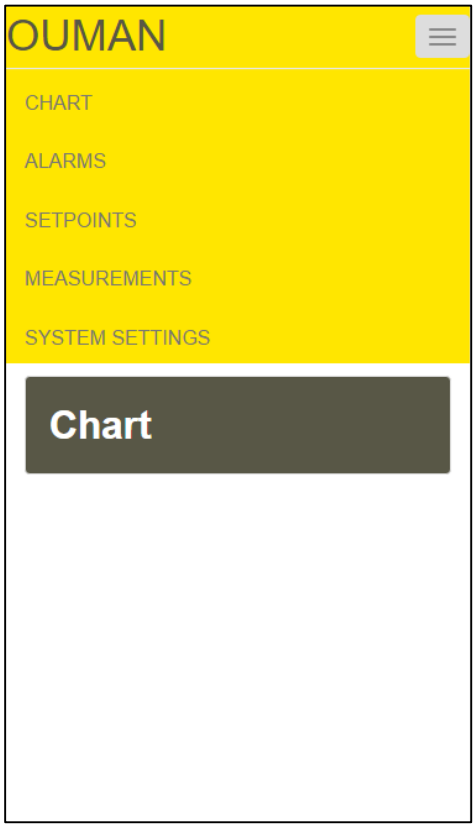

FIGURE 8. *Mobile view of the chart viewer*

After the selection of right frameworks and necessary technologies, the development started. According to the specification given to me, the application should have 4 tabs initially, Chart, Alarms, Set points and Measurements. Thus, using a menu bar and navigation components provided by Bootstrap, a page was created with a header and navigation. The navigation component of Bootstrap could collapse into a hamburger menu on smaller screens; the requirement of responsive design was accomplished. The first design of the chart-viewer is shown in FIGURE 7 and FIGURE 8. As Bootstrap is a responsive CSS framework, one design can be viewed in different screen sizes. FIGURE 8 shows a chart-viewer in a mobile view with the hamburger menu.

AngularJS was downloaded and included in the index page. When Angular was ready to be used, an overall framework of the application was created using ngRoute, which is an Angular module used for routing pages. The navigation designed earlier came to work using ngRoute.

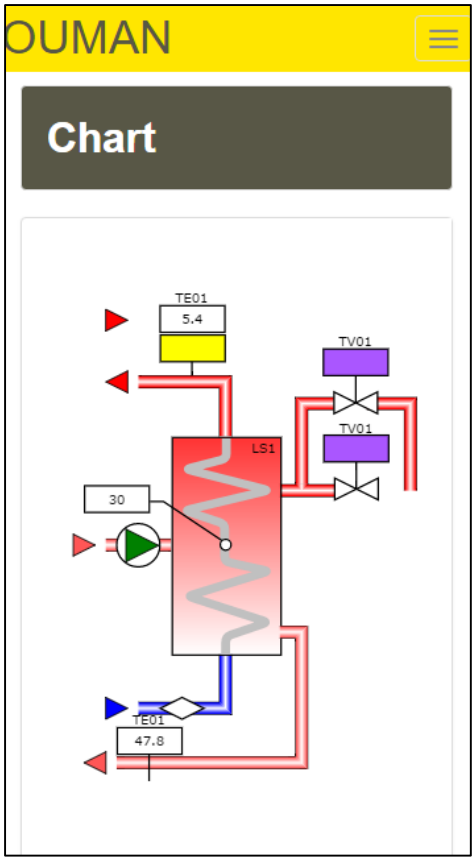

FIGURE 9. *Chart View*

When talking about the content, each tab would display. The first tab, Chart, would display the chart, second tab, Alarm, would display active alarms and their status, while Measurement and Set points would display measurement values and set points values respectively. Measurement values are Read-only values whereas Set points are Read and Writeable values.

The chart, which is created in a local chart-editor, is transferred to a web server in the JSON format. Using Ouman-API this chart is again transferred to the application. Then the chart-rendering engine called JEngine renders the chart into a canvas element of the application. Ouman API is used to get active alarms, measurements and set points and displayed in their respective tabs. The Chart-view with a sample chart is shown in FIGURE 9. The chart shown here is the architecture of the heating system. This chart is received in the web application using Ouman API from the web server. In the chart, different measurements can be displayed and some measurement can be modified, too.

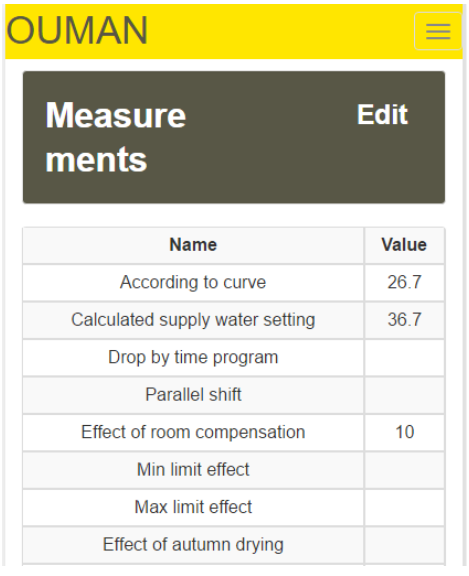

FIGURE 10 a. *Measurement*

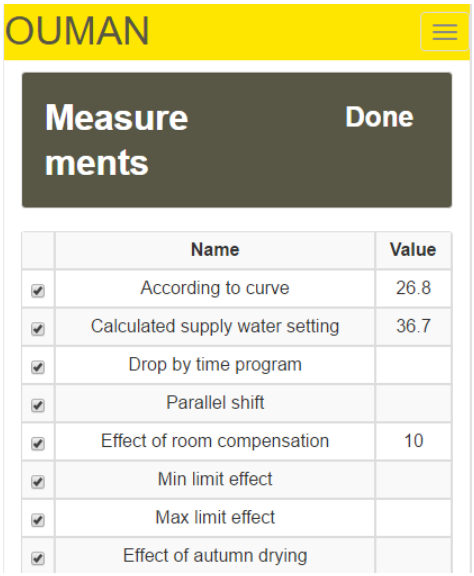

FIGURE 10 b. *Edit Mode*

In FIGURE 10 a., the view of the Measurement tab is shown. Each row of the table shows the name of a measurement and its value, for example outdoor temperature. In FIGURE 10 b., the edit mode in the Measurement tab is shown. In a system there can be hundreds of measurements, which might be unnecessary all the time. Therefore in the edit mode unnecessary measurements can be unchecked. Since these are read-only points, values cannot be changed. The tables are created using a Bootstrap table class.

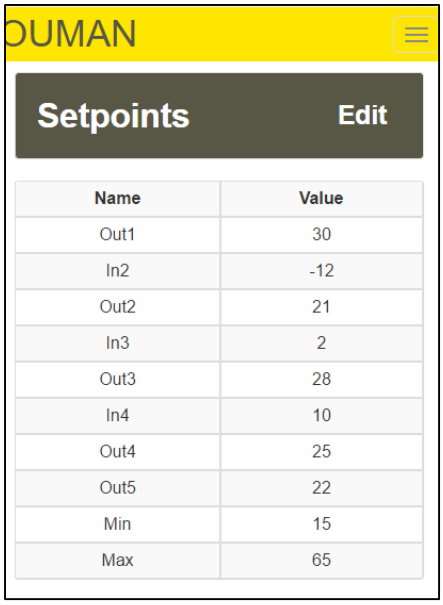

FIGURE 11 a. *Set Points*

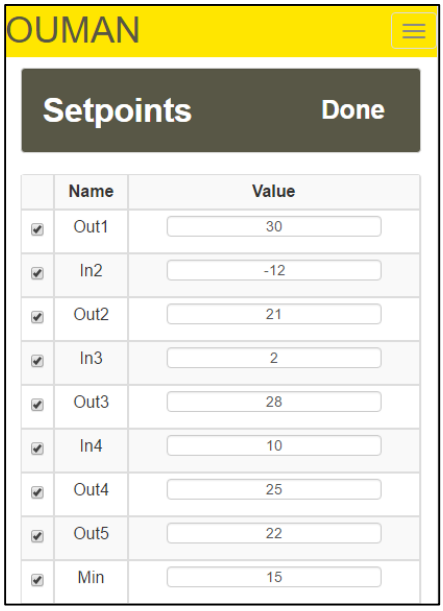

FIGURE 11 b. *Edit Mode*

In FIGURE 11 a., a view of the Set points tab is shown. Set points are Writeable measurements, which means that their values can be changed. In the first one, a table of set points with their names and values is shown and in the second figure, the edit mode of this view is shown. In this view, points can be checked or unchecked as in the measurement and the values can be changed, too. The values can be changed at once or one by one. Once the value or values are changed, the Done button is clicked. Then the values will be written to the device using Ouman API.

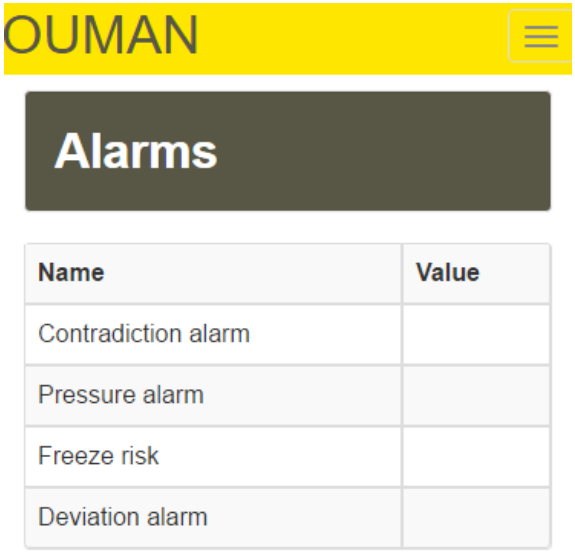

FIGURE 12. *Alarms View*

In FIGURE 12, the contents of the Alarms Tab are shown. This tab is supposed to show all the active alarms in the system. This view should be able to acknowledge alarms, too. Due to some problems, this view is not functional at the moment.

When the demo version of the application was almost ready, a change in the specification was made. The application required having one more tab called System setting. This tab should display all the necessary settings of the device like ip addresses, access addresses, time and date, as shown in FIGURE 13. After a few days, another additional feature was required according to which the application should have a dynamic navigation. By the end of December, a demo version of the application was completed.

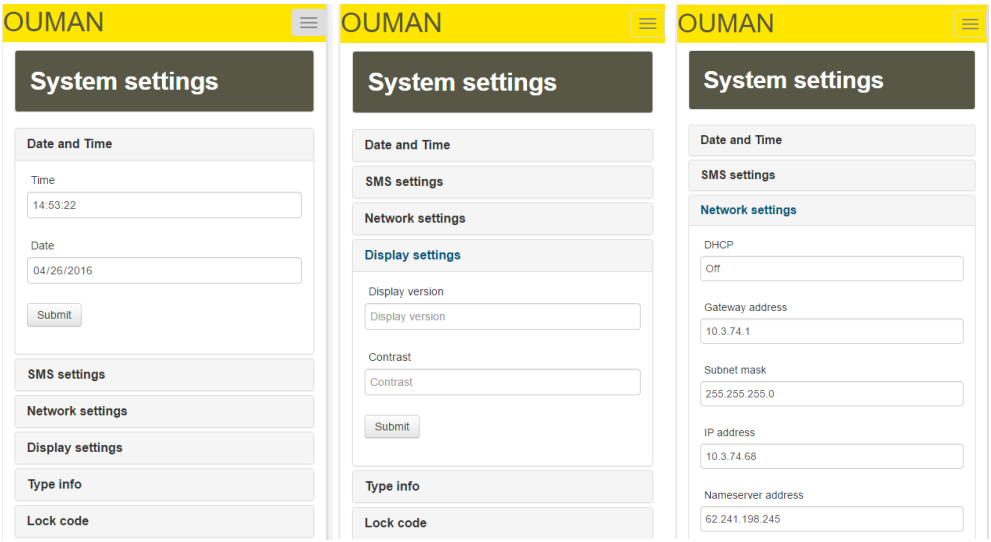

FIGURE 13. *System settings view*

4.3 Application Structure

The Local Chart-viewer is a Single Page Application (SPA). The whole application was an Angular module. Then 3 more modules where created and included in the main module; controllersModule, servicesModule and directivesModule. The ControllerModule was used to create necessary controllers similarly other Modules were used, too. According to the number of tabs, views were created. The first version of specification required 4 tabs, therefore 4 views were created at first. Then 1 more view was added in the later time of development. The views were just HTML templates without data, so controllers were required. At first 4 controllers were created respective to 4 views. Controllers make the connection between views and models. The necessary data was obtained using Ouman-API and it was processed in Services. The folder structure of the application is shown below.

```
app /
    --components /
            ----alarms /
            ------controllers /
            ------services /

            ----chartViewer /
            ------chart /
            --------controllers /
            --------services /

            ----measurement /
            ------controllers /
            ------services /

            ----setpoints /
            ------controllers /
            ------services /

            ----systemSettings /
            ------controllers /
            ------services /
    shared /
            ----controllers /
            ----services /
            ----directives /
            ----filters /
    app.js

assets /
    css /
    images /
js /
libs /
index.html
```

5 THINGS THAT NEEDED TO BE COMPLETED

The JEngine, which was used in this project, and which renders the chart in both the chart-editor and the chart-viewer, is outdated. We have the newest version of the JEngine in Ounet. We need to fetch the engine from Ounet's version control. If there are (and there are) differences between Ounet and the local editor, we need to modularize JEngine and implement local features in own modules.

Also all the other dependencies need to be fetched from Ounet's version control. The features that need to be fetched from Ounet need to be figured out. It is possible that we implement some of those features from scratch. This is by far the biggest task.

Code has to be refactored. Code should work well with Ounet. Code should follow some kind of convention, which will make maintainability and expendability in future. We need to do some refactoring to Ounet so that we can get some of the codes from it.

Ouman API is outdated, too. Refactoring and bug fixing is necessary. Ouman API performance needs improvement. The parsing needs some performance testing to validate how heavy operation it really is. Also, I think we need to do a bit performance testing with Ouman API.

Some parts of the Ouman API method are not working in some devices. A Right configuration and implementation is required.

We also need read history alarms and system settings from Ouman API. Generally speaking we need to think what needs to be done to Ouman API. What features it needs and how we use it. Do we e.g. read from a device or Oulink and if there are performance issues.

Once these issues are solved, the development of the chart-editor and the chart-viewer can be taken for production.

6 CONCLUSION

Any learning process is incomplete unless applied to some practical project. My web development learning process was not complete before starting this project. I had done few projects before at school but they were unpractical and unrealistic projects. I was struggling to understand the technologies of web. I did not have a sure and certain understanding of JavaScript and its frameworks. Though I had used JavaScript and JQuery before, I was not quite happy with the knowledge I had. I was having a great difficulty in associating these technologies together or I was not being able to put them together. Then came AngularJS. I was very excited to learn it. It took me 2 months to learn and understand the concept and building blocks of AngularJS. But I could make only small applications. At that time I used to wonder if I would ever be able to understand them well. I knew I just needed an opportunity to implement my knowledge in real world. After making 10 small Angular applications, I got the opportunity, the opportunity to work for Ouman Oy as an intern.

I started my internship in Ouman Oy in March 2015. I was given a project, which required the knowledge of AngularJS and AJAX. I was very excited but my excitement did not last long. The project was not as simple as I expected. My task was to create an application, which could communicate with their system that controls the heating system. I just knew how a web page communicates with a web server but I never knew how a web page communicates with home automation devices. The first 3 weeks went learning about the system and API, which is used to communicate with the device. I started developing the application. There were many challenges in the process but there were many new things to learn, too. I completed a demo version of the application by the end of May.

As everyone was satisfied with my work, the company offered me a practical training place that summer. That summer I spent for 4 months in reading and debugging the code of their already existing application. I had to understand the codes, how applications worked and how to modify the codes. By the end of summer I had a plenty of knowledge in debugging the application and reading

codes. There is a saying “If you cannot read a code, you cannot write it”. Then I felt, I could write code.

The task was not completed due to lack of some requirements but everyone was satisfied with my work. In the end of summer 2015, I was given the thesis work. My task was to create a web UI, which could display the chart drawn on the local chart-editor. The local chart-editor was the application I was working on that summer.

The project started from the 1st of September that year. Developing a chart – viewer was very challenging. I had to read through hundreds of lines of code, understand the logic, modify, refactor and use it again in a chart – viewer. I learnt how a web application could be made to communicate with devices, which have embedded web servers in them. I learnt how an API could be used to communicate with a device. I learnt better ways to develop applications with Angular. Earlier I had not been developing the application the right way. The development of the applications is needed to be modular, manageable, maintainable, understandable and clean. I would not say that now I can develop the application the right way but I know I am on the right track and I am one step closer to that right way. Before, the concept of responsive design seemed to be blurred and unclear, though I worked in a couple of projects earlier, but now the knowledge has enhanced with an incredible amount. Now I feel I am confident and comfortable enough to develop applications using Bootstrap and AngularJS.

Besides from learning web development technologies, during this project I got an opportunity to work in the real working environment, I got an opportunity to experience the real working life, meet and interact with real professionals. I learnt how hardworking and dedicated one needs to be in order to meet the goals. Though we were given many deadlines at school by teachers, I had to experience how scary the real deadlines feel. I had to feel and learn many things, which are very important for a professional carrier.

Therefore, the whole thesis project has become a good experience for me. As I mentioned before, I learnt many things, which had not been possible without this kind of projects earlier.

REFERENCES

1. KMC Controls, 2013. Understanding Building Automation and Control System, Date of retrieval 2.05.2016
2. http://ouman.fi/en/document-bank/, Ouflex Device, Brochure, Date of retrieval 5.05.2016, Ouman Oy
3. http://ouman.fi/en/document-bank/, Ouflex Tool, Brochure, Date of retrieval 6.05.2016, Ouman Oy
4. http://ouman.fi/en/document-bank/, Oulink Device, Brochure, Date of retrieval 6.05.2016, Ouman Oy
5. http://ouman.fi/en/document-bank/, Ounet, Brochure, Date of retrieval 7.05.2016, Ouman Oy,
6. Twitter Bootstrap, Getting started with Bootstrap, Date of retrieval 10.05.2016, http://www.getbootstrap.com/
7. Bootstrap Get Started, Date of retrieval 10.05.2016, http://www.w3schools.com/
8. Flanagan, D, 2011, JavaScript: The Definitive Guide, 6th Edition, The United States of America, O' Reilly Media, Inc.
9. AngularJS, Date of retrieval 20.05.2016, Wikipedia, http://www.wikipedia.com/
10. Web Application Architecture, W3C, 26.06.2010, https://www.w3.org/2001/tag/2010/05/WebApps.html/