VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Riina Pussinen

INSTALLATION LAYOUT CONFIGU-RATION

Wärtsilä

Tekniikka
2017

ACKNOWLEDGEMENT

I would like to express my gratitude to Wärtsilä for giving me this opportunity to do my thesis in inspiring industrial context.

I am also grateful for Dr Ghodrat Moghadampour for supervising this process and his support and guidance through the process of making this thesis.

Final thanks belong to my friends, especially roommates, and family for mental support and cheering me through this process.

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Riina Pussinen
Opinnäytetyön nimi	Installation layout configuration
Vuosi	2017
Kieli	englanti
Sivumäärä	54
Ohjaaja	Ghodrat Moghadampour

Opinnäytetyö toteutettiin projektina Wärtsilälle. Tarkoituksena oli kehittää Wärtsilän sisäisessä käytössä olevaa Genius Dashboardia, jossa voidaan suunnitella moottorin ulkoasua.

Työssä paranneltiin jo valmiiksi olevaa Genius Dashboardia, johon käytettiin JavaScript-ohjelmointikieltä. Lisäksi työssä käytettiin Angular.js sovelluskehystä. Toteutusta varten käytettiin WebStormia editorina.

Opinnäytetyön tuloksena saatiin parannettua Genius Dashboardia käyttäjäystävällisemmäksi. Tämä tapahtui lisäämällä objekteille erilaisia ominaisuuksia, joiden avulla käyttäjän on helpompi suunnitella pohjapiirustusta. Lisättyjä ominaisuuksia olivat objektin kloonaus, tyypin vaihto, skaalaus, rivittäytyminen, sekä objektien käsittely ryhmänä.

Avainsanat Front-end, JavaScript, Genius Dashboard, WebStorm

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Riina Pussinen
Title	Installation Layout Configuration
Year	2017
Language	English
Pages	54
Name of Supervisor	Ghodrat Moghadampour

This thesis was done as a project for Wärtsilä. The main purpose of the thesis was to improve and implement Genius Dashboard, where one can configure the layout of engines.

The already existing Genius Dashboard was improved using JavaScript as programming language. Angular.js framework was also used. For implementation WebStorm was used as an editor.

The result of the thesis was an improved Genius Dashboard, which was implemented to be more user friendly. This was implemented by adding different features to the objects, which will make it easier for the user to configure the layout of an engine. Added features were object’s cloning, change of type, scaling, snapping and spreading as well as grouping of objects.

Keywords Front-end, JavaScript, Genius Dashboard, WebStorm

TABLE OF CONTENTS

LIST OF FIGURES, TABLES AND CODE SNIPPETS

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheets
GIT	Version Control System
HTML	Hypertext Markup Language
IDE	Integrated development environment
JSON	JavaScript Object Notation
SVG	Scalable Vector Graphics
UI	User Interface
VPN	Virtual Private Network

1 INTRODUCTION

This thesis has been made in Vaasa University of Applied Sciences' Information technology department. The field of the specialisation of the student is software engineering. The client of the thesis is Wärtsilä's Digitalisation department. The main purpose of the project was to develop already existing Genius Dashboard, to make it more efficient and user friendly. The layout of the engines, thrusters and other equipment's can be configured in Genius Dashboard. The goal was to have a working view for the user to be able to effortlessly execute the wanted configuration.

Genius Dashboard should work flawlessly in all the browsers. In the end of the thesis there will be the status of the follow-up plan of the subject and whether the subject will be developed further.

2 WÄRTSILÄ

Wärtsilä is a Finnish corporation which manufactures and services power sources and other equipment in the energy and marine markets. It was founded in 1834 and its headquarters are in Helsinki, Finland.

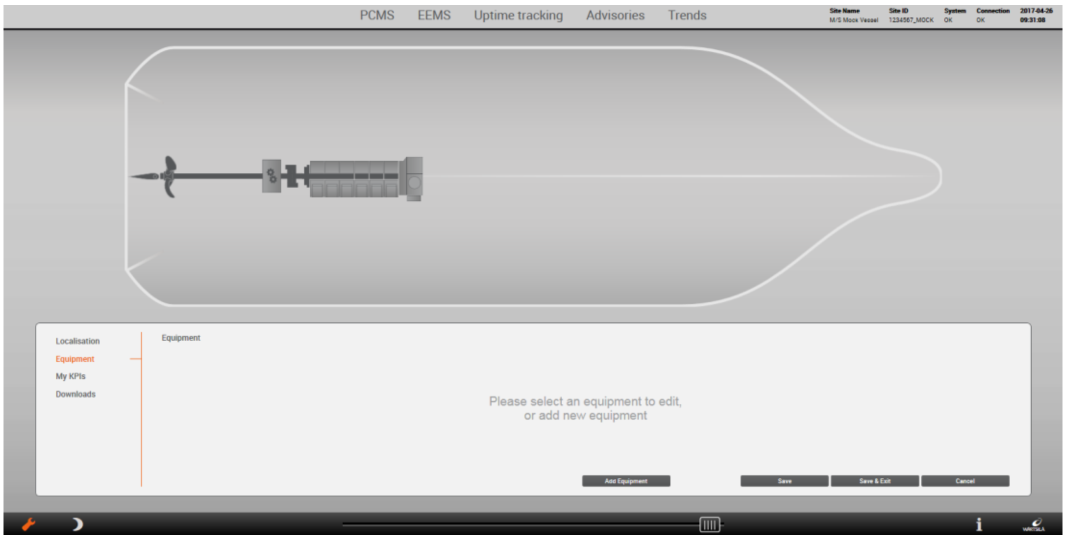
Wärtsilä has three main businesses, which are Energy Solutions, Marine Solutions and Services. Energy Solutions is focusing on the energy market, Marine Solutions is focusing on marine market and Services combine the two, marketing both of the above. /1/

2.1 Genius Dashboard

Genius Dashboard is Wärtsilä’s own tool, used only inside the company. In the layout one can configure the layout of engines, thrusters and other equipment fairly easily. The user has to drag-and-drop the engine’s different parts into the dashboard to make the prototype of the engine’s layout. Genius Dashboard does not currently support many different functionalities, but it has already implemented redo, undo, rotating and moving object. In addition to this, the user can save work space with the components configured.

2.2 Current state

Before my thesis, Genius Dashboard was a functional software. It only needs a few additions to make it more useful and agile. Genius Dashboard had few functionalities implemented. They included rotation, movement, redo and undo, and removing objects. The user can also save the layouts.



Please select an equipment to edit,
or add new equipment

Figure 1. Start –view

In figure 1 is shown the starting point of configuring the layout. The user can select to add engines via pressing the "Add engines" –button. Also "Save", "Save and exit" and "Cancel" –buttons are available at this point.

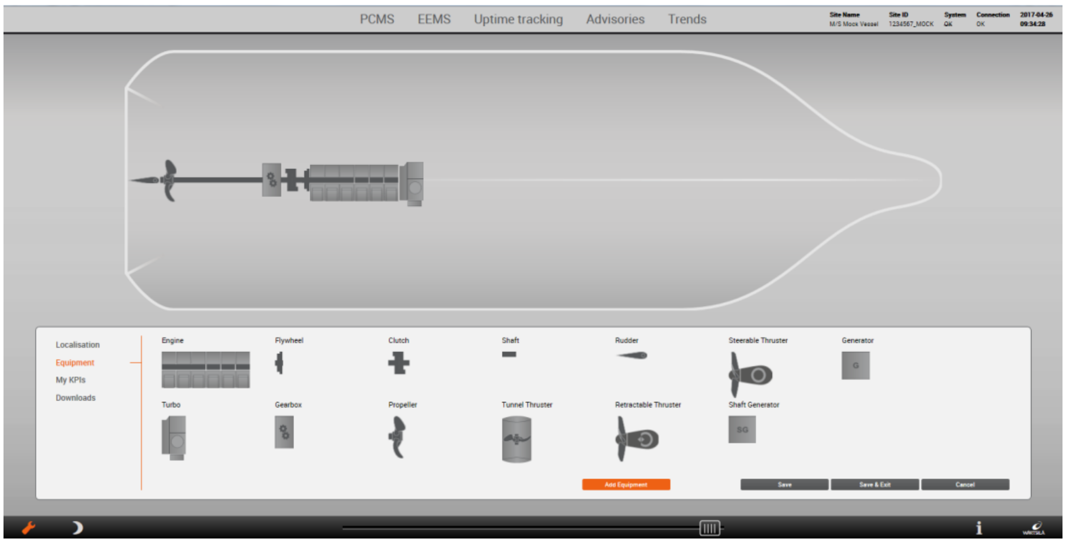


Figure 2. Add engine –view

In figure 2 is displayed what it looks like when a user has pressed the “Add engine” –button. All the available objects are shown and from here the user can drag the selected object to the grid.

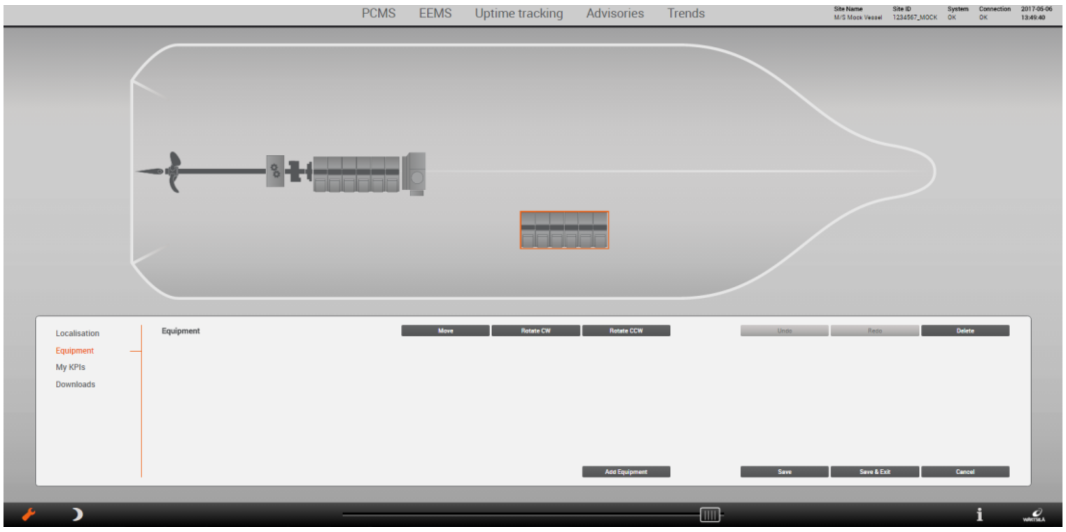


Figure 3. Selected object’s features

Possible features for the object are displayed in Figure 3. The user can move or rotate the selected object. Also when the user has moved/ rotated the selected object, the layout can be saved. The selected object has orange frames around it, so the user knows which object is selected.

2.3 Purpose of the thesis

The purpose of the thesis was to get better functionalities for Genius Dashboard. As well as making it more useful and user friendly. This included adding some functionalities to the dashboard. The current version does not have that many options on saving and copying objects, so it is fairly slow to use. The required functionalities were cloning, grouping, spreading and changing the type of objects. In addition to this, the user must have an option to do customer specific configurations. These will be necessary additions to make Genius Dashboard more agile and user-friendly.

3 USED TECHNOLOGIES

This chapter presents used technologies and programming languages in the thesis.

3.1 Used programming languages

The programming languages that are used in the thesis will be presented below. JavaScript was the main programming language, but AngularJS was a supporting framework. In the project balancing the combined technologies was a challenge.

3.1.1 JavaScript

JavaScript is a programming language which is used for developing web-pages. It was developed by Brendan Eich for Netscape Communications Corporate and it was published in 1995. JavaScript was originally called Mocha and after that LiveScript. Name was changed to JavaScript for marketing reasons. JavaScript's most important feature is to add dynamic functionality to web-pages. In JavaScript one can combine CSS, HTML and JavaScript. CSS to create the layout of the program, HTML to define the content and JavaScript to program the functionality of the program. JavaScript is usually used for developing web-pages. It is not however tight up to web applications, it can be used also for game development. Only requirement for JavaScript is that browser supports it. It works most commons browsers, including Chrome, Firefox and Internet Explorer. /3/, /4/

JavaScript is very similar to many popular programming languages, such as Java, C# and C++. They are similar so that developers with different backgrounds can easily learn to write JavaScript code. That said, JavaScript must not be mixed with Java. JavaScript is actually closer to C than Java by its syntax. /3/, /4/

```
<html>

        <head>

                <title>JavaScript example</title>

        </head>

        <body>

                <script>

                        Document.write(“Hello world!”);

                </script>

        </body>

</html>
```

Code Snippet 1. Example JavaScript

In Code Snippet 1 is shown a simple JavaScript code. JavaScript is recessed in an HTML-document. The JavaScript part says “Hello world” so in the HTML-document will be printed “Hello world!”. This is a basic JavaScript example.

```
<html>

        <head>

                <title>JavaScript example 2</title>

        </head>

        <body>
```

```
                <script>
                        Function sayHello(){
                                Alert(“Hello!”)
                        }
                        </script>
                        <button          onclick=”setTimeout(function(){sayHello()},
1000)”> Say Hello</button
        </body>
</html>
```

Code Snippet 2. Example 2 for JavaScript

The JavaScript in Code Snippet 2 displays a repetitive code recessed in HTML-document. When the button is clicked it will alert a function “sayHello” 1000 – times.

3.1.2 AngularJS

AngularJS is a structural framework for dynamic web applications. When using AngularJS, one can use HTML as a template language and extend HTML’s syntax to express an application’s components distinctly and concisely. AngularJS is ideal to use with any server technology, because of its data binging. /2/

AngularJS is designed for applications, this could have been HTML if it was designed for applications. HTML does not contain much for creating applications, but for static documents it is an excellent declarative language. Static documents and dynamic applications impedance mismatch has been solved with library and frameworks. AngularJS does it another way, it tries to minimize the impedance mismatch between applications' need for creating HTML constructs and document centric HTML. AngularJS uses directives, which are a new syntaxes and are used in browsers through construct. For example data binging, Grouping of HTML into reusable components, supporting forms and form validation and DOM control structures for repeating, showing and hiding DOM fragments. /2/

When using AngularJS you do not have to worry about registering callbacks, manipulating HTML DOM programmatically, marshaling data to and from the UI or writing much initialization code just to get started. When it comes to building UIs and wiring software components together, AngularJS is built to use declarative code. Imperative code is beneficial for expressing business logic. /18/

AngularJS implements MVC by dividing app into MVC components. AngularJS manages all the components for the user and serves the connections between them as well. For the user's interface definition AngularJS uses HTML, it is also used to determine the execution of the app. For each element, HTML decides which controllers use them. The user just defines what they want and AngularJS will take care of all the dependencies. /18/

AngularJS uses data models that are old JavaScript objects. They do not require getter and setter functions and the user can simply add and change properties and loop objects and arrays as they want. The code will look clear and intuitive. AngularJS's data models are objects and they behave like a temporary storage area for the user to put and retrieve data. AngularJS calls data models "scopes" and the scope does not have data to begin with, it fetches the data from the controller based on the logics in the application. Changes are updated automatically to the view by AngularJS. /18/

3.1.3 HTML

HTML is abbreviation from Hypertext Markup Language. HTML is an openly standardized description language. It uses markup to describe the structure of Web pages. When creating HTML pages, HTML elements are the building blocks and they are presented by tags. HTML tags are labelled pieces of content, “heading”, “paragraph” and “table” are examples of HTML tags. Tags’ contents are surrounded by angle brackets, they typically come in pairs like <tb> as opening tag and </tb> as closing tag. Web browsers read and display HTML documents. HTML can also mark the text structures, such as what part of the text is in the title and what piece of text. /14/

The HTML code is structured into text that consist of nested and successive elements. The elements are represented by angles marked with an angular interlinkage, the tags that name the element type. Browsers do not display the tags as they are, but handle them as technical guides for the actual content of the page to be structured. Usually, the elements have an initial identifier and the ending tag marked with a slash, and the content between them containing text and other HTML elements : <Tag> Element content </tag>. The start tag may also include the attributes that specify element attributes more precisely than the type of element named by the tag.

```
<tag attribute=”value”> element content</tag>
```

Code Snippet 3. Example 2 for HTML

Code Snippet 3 displays how attribute “value” is added for the HTML tag. Some HTML elements do not have any content. A non-contained element is represented by, for example, a <br/> tag that is a mandatory line break. Unsubstantiated elements do not need a termination tag except for XHTML code. Even then, a separate ending tag can be replaced so that the slash acting as the stop sign is immediately marked at the end of the start identifier: <tag/>. /14/

Structural marking determines the purpose of the element. For example <h2>Test</h2> means that the word “Test” is a second-degree title. Structural markup does not determine what the text looks like in the browser, but usually the browsers follow some of the default ways of presenting different types of elements. The layout of the texts can be further regulated with CSS style instructions. /14/

3.2 Used software

Below all software that were used in the thesis will be presented. The software include editor, git client and cisco connection. The thesis was implemented with only free software.

3.2.1 WebStorm

WebStorm is a lightweight JavaScript IDE, which is used for client-side development as well as server side. It comes with build-in coding assistance, which means simply that the editor suggests fitting functions. It also has on-the-fly error detection, which comes in handy from time to time.

WebStrom also supports latest technologies, as well as AngularJS, Node,js and more. It has a unified UI for working with Git, SVN and more. This became handy when I started working with the thesis and had to download previous codes for the Genius Dashboard. /5/

3.2.2 Git

Git is a version control software, which is designed to work distributed and as efficiently as possible. Git is designed for POSIX-compatible operating systems, including Linux and OS X. It is open source and free for the users. Git is designed to work with small and large projects. It has many useful features such as branching and merging. Git allows user to have multiple branches that are independent and it is fast in creation, merging and deleting development lines. Using git the user can switch between branches to for example experiment with some codes. After committing changes, the user can go back to the previous branch. The user should have a master –branch (this branch would have only the codes that goes to production), a branch for testing and also several smaller branches to work with issues. /16/

3.2.3 SourceTree

SourceTree is a Git client that provides a graphical interface for repositories. It provides the user with local and remote repositories in the interface.

The user can easily check out where the project is going. The user can also easily create new repository via button. SourceTree has a sidebar when the user opens a repository. The sidebar provides options for the current workspace, shows available branches in the project, remote branches as well as tags. SourceTree also provides the user with the repository’s history. There is also an optional filter included. SourceTree has an option of viewing diffs between commits. /6/

3.2.4 Cisco VPN Client

Cisco VPN Client provides secure remote access in organizations. To work from home, a Cisco VPN client is required to get into Wärtsilä's sites on their domain. Without the connection, one cannot get into the sites.

Cisco VPN solutions provide security using encryption and authentic authentication technologies. They protect the data in transit from unauthorized access and attacks. A Cisco VPN client is highly secure and flexible. /15/

3.3 MVC Architecture

MVC (Model-View-Controller) architecture means basically that code is divided into three parts; view, controller and model. Models use the database and abstract it as programming interface that the other code uses. Views handle all code related to HTML and controllers receive user's given requests, search and update information using models and forward the data to the view that will display the page. /17/

3.3.1 View

The view determines the interface layout and data display presentation in the user interface. In the case of a database application, so-called template files are used, which define the HTML code for the page and the variables that come with it. Views will never process the database directly and very rarely they have the actual code over three lines a row. In order for the data to be retrieved directly from the view, the matched variable names whose content is displayed are used. The view does not need to know anything about where the content of the variables comes. It is enough for the views to know that the information is where it should be. It is the duty of the controller to assign information to the right variables. /17/

3.3.2 Controller

The controller which is also known as the handler binds the template and view and transmits information between them. The controller receives user requests from the user, uses the template to retrieve and modify information and ultimately generally transfers the performance to the view. Usually, the controller transmits information to the view, for example, items returned by a list template or an error message. Instead, the controller will never process the database directly or send HTML or other content to the browser. These are roles for models and views.

3.3.3 Model

Models contain all the information and processing functions in the application database. All the database related code is included in the models. Models are implemented by collecting activities related to different things and data objects into classes and/or functions. /17/

Model class can in practice be a mere abstract level that hides the gritty details of the database code behind user interface. Often, it may be wise to make models for abstract things like sessions and the rights that a user has signed up for. They do not have to use the database (at least not directly), but they are still models. /17/

4 DESIGNING AND DEFINING THESIS

This chapter presents the project’s definitions. The project consists of web-view, where layouts of the engines can be configured. The engines can be placed on a vessel layout by drag-and-drop technology. The user selects components from a selection of engine parts and then drags them to the layout. In the layout there are multiple choices for functionalities for the objects in the grid.

4.1 Starting point

The aim of this projects was to improve Wärtsilä’s functional layout editor called Genius Dashboard, where the user can create a layout of engines conveniently and effortlessly. Objects must have different functionalities implemented and must work in different browsers, for example in Chrome, Mozilla and Internet Explorer.

4.2 Defining project

The definition of the projects came from the client. The list of required features was fully written by the client. The aim of the project was to improve the already existing Genius Dashboard and add some functionalities to objects. Genius Dashboard is a web-view, where the user can drag-and-drop objects to the grid. When the objects are dragged to the grid, they can be modified by different functionalities. Client of the project wanted to add some more functionalities to the objects to make Genius Dashboard more agile, user friendly and efficient to use.

4.2.1 Requirement specification

The requirement specification creates a basis for the acquisition, why and what needs the acquisition will fulfil. Requirement specification focuses on what is needed from the project, not how it works technically. In the requirement specification all the required functions are specified. /7/

Table 1. Requirement specification

Required features	**Importance**
The layout editor must use a JSON API and SVG graphics	1
Standard configurations must be available that can be used as starting point	1
Snapping of objects to grid must be possible for easy alignment	1
Spread objects evenly must be possible for easy alignment	1
Grouping of objects together must be possible	2
Possibility to select between options of the same type	1
Possibility to rotate component or object	1
Possibility to scale component or object	1
Cloning of component or object	1
Configuration for overlays of component or object	2
Possibility to do customer specific configurations	2
1 = must have, 2 = nice to have	

4.2.2 Use Case Diagram

Use case diagram is used to capture the dynamic aspect of a system. It shows the relationship between user and use cases. There are few purposes of use case diagrams. They are the following:

- Gather requirements of a system
- Get an outside view of a system
- Identify external and internal factors influencing the system

The project has one user. The user can select the layout of the engine (ship or power plant). Then the user can configure the layout of the engine, by dragging objects. The user can modify the layout with different functionalities. /8/

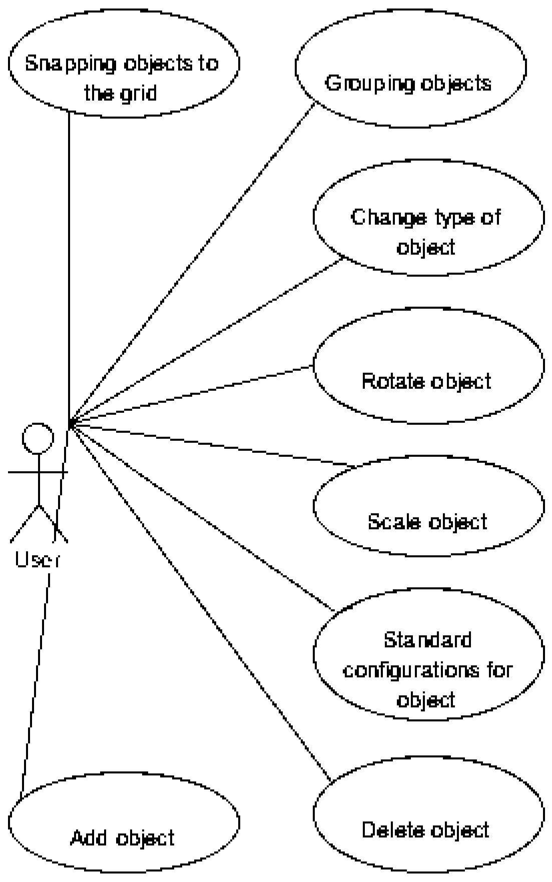


Figure 4. Use Case Diagram

All of the use cases are shown in figure 4. Snapping objects to the grid means that the objects have to align to the same level so they are easy to configure correctly.

There will also be functionalities that the user can add to the objects. The objects have to be able to clone, rotate as well as scale. Also the user must be able to add and delete objects from the grid. Some of the objects will have the option to change the type of it and it is also a requirement that the object's type must be changeable.

For the Genius Dashboard, the standard configurations must be available at the starting point as well, this means that the functionalities must be available when starting configuring the layout.

4.2.3 Sequence Diagram

A sequence diagram is an interaction diagram. It shows how objects engage with one another and what order. A sequence diagram describes as adjacent vertical lines at the same time living creatures' life lines and horizontal arrows between them pass messages in chronological order. Time passes from top to bottom in the diagram. /10/

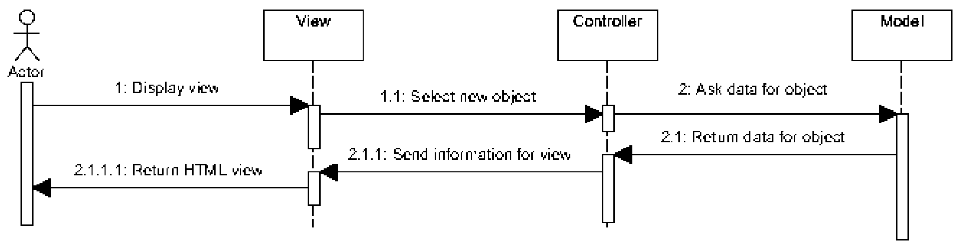

Actor
View
Controller
Model
1: Display view
1.1: Select new object
2: Ask data for object
2.1: Return data for object
2.1.1: Send information for view
2.1.1.1: Return HTML view

Figure 5. Sequence diagram for adding new object

Figure 5 is a sequence diagram which displays what happens when user adds new object. First user selects wanted object. View tells the controller that objects is selected and controller asks for the object's data from the model. Then the model returns the correct data-values such as data part and type to the controller. The controller sends the information to the view, which will allow the HTML view to display a new part in the browser.

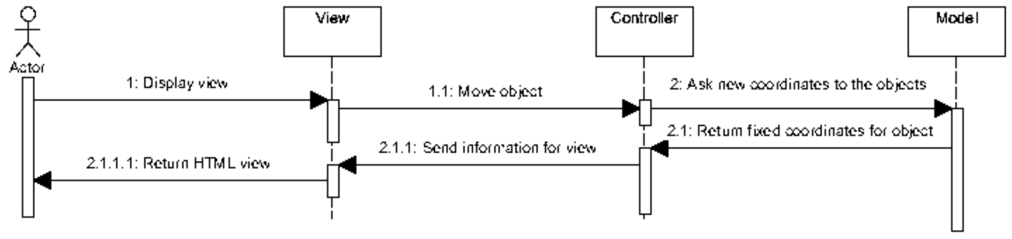

Actor
View
Controller
Model
1: Display view
1.1: Move object
2: Ask new coordinates to the objects
2.1: Return fixed coordinates for object
2.1.1: Send information for view
2.1.1.1: Return HTML view

Figure 6. Sequence diagram for snapping the objects to the grid

Figure 6 is a sequence diagram which displays when an object is snapped to the grid. First the user chooses the wanted object to drag to the grid. View tells the controller that the wanted feature is moving the object. The controller then asks new coordinates from the model, when the object is let go. The ´model will identify the objects and sends the controller new coordinates. In the controller is a function, which will snap the objects to the grid. Controller then sends the fixed information to the view and a HTML view is returned to the user.

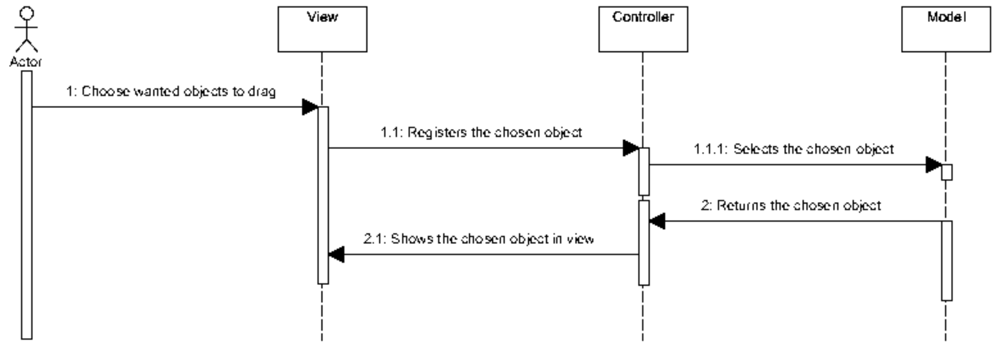

Actor
View
Controller
Model
1: Choose wanted objects to drag
1.1: Registers the chosen object
1.1.1: Selects the chosen object
2: Returns the chosen object
2.1: Shows the chosen object in view

Figure 7. Sequence diagram for dragging object to the grid

Figure 7 is a sequence diagram which displays what happens when the user drags the object to the grid. First the user chooses wanted object to drag to the grid. View tells the controller that the objects is dragged, the model will identify this object. It returns the correct data-values such as data part and type to the controller. The controller then displays the chosen object in the grid.

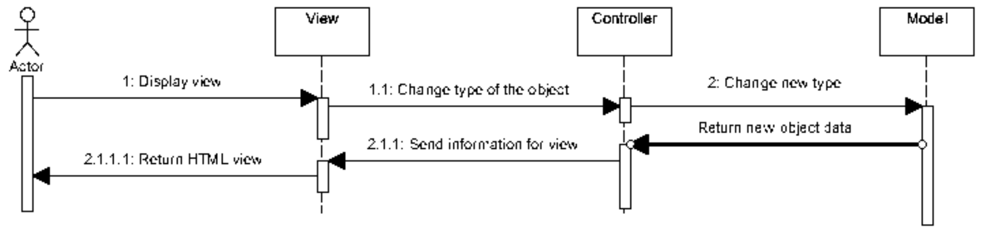

Actor
View
Controller
Model
1: Display view
1.1: Change type of the object
2: Change new type
Return new object data
2.1.1: Send information for view
2.1.1.1: Return HTML view

Figure 8. Sequence diagram changing the type of the object

Figure 8 is a sequence diagram which displays what happens when changes the type of the object. First the user chooses wanted object to change the type. View tells the controller that the type of the object will change. The controller asks the model to change the type of the object. The model then returns the changed data information to the controller which sends the information to the HTML to be displayed.

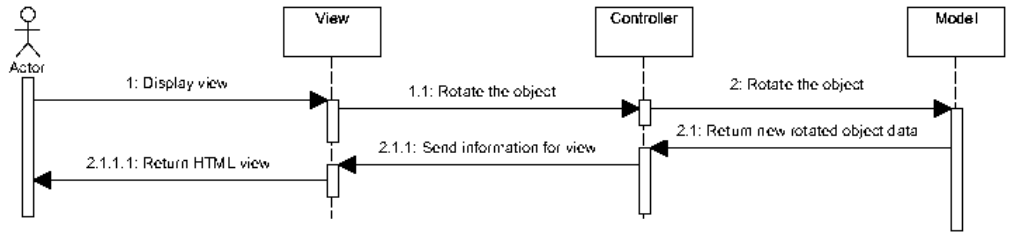

Actor
View
Controller
Model
1: Display view
1.1: Rotate the object
2: Rotate the object
2.1: Return new rotated object data
2.1.1: Send information for view
2.1.1.1: Return HTML view

Figure 9. Sequence diagram for rotating the object

Figure 9 is a sequence diagram which displays what happens when the object is rotated. First the user chooses the wanted object to rotate. The view tells the controller that objects' will be rotated. The controller asks model to rotate the selected object. The model then returns changed data information to the controller which sends the information to the HTML to be displayed.

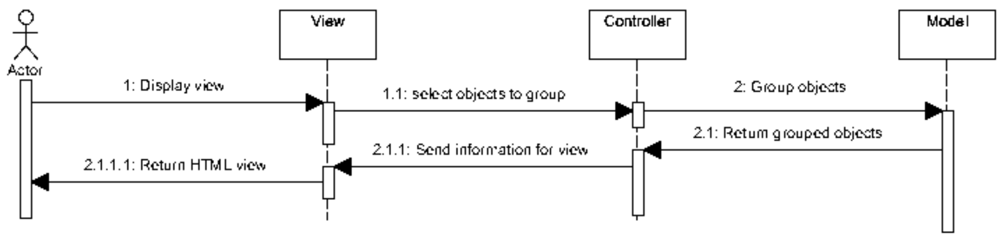

Actor
View
Controller
Model
1: Display view
1.1: select objects to group
2: Group objects
2.1: Return grouped objects
2.1.1: Send information for view
2.1.1.1: Return HTML view

Figure 10. Sequence diagram for grouping objects

Figure 10 is a sequence diagram which displays what happens when objects are grouped. First the user chooses wanted objects to group. The view tells the controller that the objects will be grouped. The controller asks model to group selected objects. The model then returns changed data information to the controller which sends the information to the HTML to be displayed.

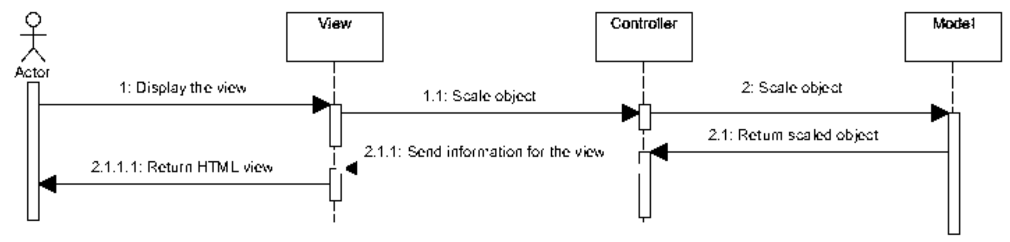

Actor
View
Controller
Model
1: Display the view
1.1: Scale object
2: Scale object
2.1: Return scaled object
2.1.1: Send information for the view
2.1.1.1: Return HTML view

Figure 11. Sequence diagram for scaling the object

Figure 11 is a sequence diagram which displays what happens when the object is scaled. First the user chooses the wanted object to be scaled. The view tells the controller that the object will be scaled. The controller asks the model to scale the selected object. The model then returns changed data information to the controller which sends the information to the HTML to be displayed.

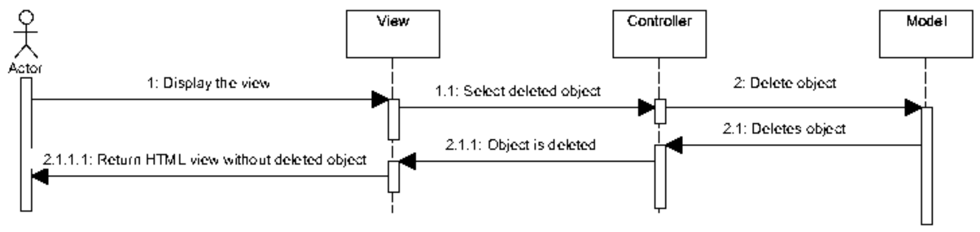


Figure 12. Sequence diagram for deleting the object

Figure 12 is a sequence diagram which displays what happens when the object is deleted. First the user chooses the wanted object to be deleted. The view tells the controller that the object will be deleted. The controller asks the model to delete the selected object. The model then returns changed data information to the controller which means that model does not send the deleted objects information anymore.

The controller sends the information about not having the deleted object anymore to the HTML view.

5 EXECUTION OF THE THESIS

This chapter presents all the issues that were relevant in the thesis. All the issues are included in the requirement specifications. Chapters have a brief introduction to the issues and the snippet codes are provided for them. All these issues were prioritized as important and must-have.

5.1.1 Use of a JSON API and SVG graphics

The layout editor must use a JSON API and use SVG graphics. This issue is standard and will be present in all the other issues. The usage on JSON API and SVG graphics is crucial for the view to work. Genius Dashboard already uses these, so if the added features will not use them, the site won't work.

JSON API is a specification for how a client should request that resources be fetched or modified as well as how a server should respond to those requests. Using JSON API the client can take advantage of its features around efficiently caching responses. JSON API covers responses, as well as creating and updating resources. JSON API is modelled to reduce the number of requests and the amount of data transmitted between clients and servers. This issue is fixed without compromising readability, flexibility or discoverability. /11/

SVG graphics defines vector-based graphics in XML format for web applications. In this project x and y –coordinates are defined with SVG technology. There are many advantages in using SVG. For example SVG images are scalable, zoomable and they can be printed at any resolution. /12/

5.1.2 Standard configurations available at starting point

Standard configurations must be available so they can be used as starting point.

When a user starts Genius Dashboard, standard configurations must be available. This means that when an object is clicked, there should be basic configurations. Functions are moving, rotating, cloning, scaling, grouping and deleting object. Also when layout is clicked, the user can select layout model, whether it is a ship or power plant. Choices of objects must be available as well.

```
<div ng-show="settingsEquipment.type === 'selected'" class="set-
tings-equipment__selected">

            <div class="settings-equipment-transition-buttons">

                  <button class="button button--primary"

                        ng-click="settingsEquipment.move()"

                        ng-class="{'button-active' : settingEquip-
      ment.selected.data.move }">

                  Move

            </button>

            <button class="button button--primary" ng-click="set-
      tingsEquipment.rotateCW()">Rotate CW</button>

            <button class="button button--primary" ng-click="set-
      tingsEquipment.rotateCCW()">Rotate CCW</button>

            <button class="button button--primary" ng-click="set-
      tingsEquipment.cloning()">Clone</button>

            <button class="button button--primary" ng-click="set-
      tingsEquipment.changeType()">Change Type</button>

            <br><br><button  class="button  button--primary"  ng-
      click="settingsEquipment.rotateCW()">Rotate CW</button>
```

```
<button class="button button--primary" ng-click="settings-
Equipment.changeSize()">Resize</button>

</div>
```

Code Snippet 4. HTML-code for buttons

Code Snippet 4 is a part of the HTML-code. In the code object’s functionalities buttons are written and actions for the buttons are assigned.

5.1.3 Snapping and spreading of objects to grid

Snapping of objects to the grid is an important issue, this guarantees that objects will group correctly. This was solved by a code which was applied when the object was let go. After letting go of the object, the code would set different x- and y-coordinates in the grid. The grid was given points that objects would spread to. Code 5 shows the coordinates updated values.

```
vm._getGridPositions = (amount) => {

            let positions = [];

            for (var i = 0; i < amount; i++) {

                  positions.push(i * 100 / amount);

            }

            return positions;

      };
```

```
        vm._roundedCoordinate = (amount,value) => {

            let positions = vm._getGridPositions(amount);

            return   positions[Math.round(value/   (100   /
    amount))];

        };
```

Code Snippet 5. Snapping and spreading objects

In Code Snippet 5 the coordination of the objects are defined. In the program snapping and spreading of objects to the grid is wanted, so in the code the coordinates are adjusted to the grid's already assigned points. Basically, when the object is dragged to the grid, this part of the program will adjust the object to a specific position.

5.1.4 Grouping objects together

In the grid, the user must have a choice of grouping objects via a button. For example an engine, flysteel and turbo must be handled as one component. This will help the user's work, if the layout has multiple groups of objects.

5.1.5 Actions on objects

The layout should have functionalities. There are four requirements for these functionalities. The first is that objects have to have an option to select between the same types of objects. For example, it must be possible to change turbocharger type without removing or re-adding it. Code Snippet 6 shows how the changing part was done by updating the element’s class value. Code Snippet 7 is an example of how data is transferred to another file. Code Snippet 7 is linked to the button, so when the button ‘Change Type’ is clicked Code Snippet 7’s code is ran by. It scopes to Code Snippet 6’s code, to function vm.changePart. That is where the selected div’s class is updated by one. When the part is last one to update, code will start updating the div’s class from the start.

```
vm.changePart = function (event, part) {

      if (part.id !== vm.data.id) return;

      if(vm.data.part === 'engine-block') {

            if(angular.element(targetElement[0].querySelec-
tor('.engine-block')).hasClass('engine-block--9L')){

                  angular.element(targetEle-
ment[0].querySelector('.engine-block')).addClass('engine-
block--6L'));

                  angular.element(targetEle-
ment[0].querySelector('.engine-block')).removeClass('en-
gine-block--7L engine-block--8L engine-block--9L'));

                  return;

            }

            elseif(angular.element(targetEle-
ment[0].querySelector('.engine-block')).hasClass('engine-
block--8L')) {

            var type = 9;
            var new_class = 'engine-block--' + type + 'L';

            angular.element(targetElement[0].querySelec-
tor('.engine-block')).addClass(new_class));

                  }

      Else   if(angular.element(targetElement[0].querySelec-
tor('.engine-block')).hasClass('engine-block--7L')) {

            var type = 8;

            var new_class = 'engine-block--' + type + 'L';

            angular.element(targetElement[0].querySelec-
tor('.engine-block')).addClass(new_class));

      else   if(angular.element(targetElement[0].querySelec-
tor('.engine-block')).hasClass('engine-block--6L')) {

            var type = 7;

            var new_class = 'engine-block--' + type + 'L';

            angular.element(targetElement[0].querySelec-
tor('.engine-block')).addClass(new_class));

            angular.element(targetElement[0].querySelec-
tor('.engine-block')).removeClass('engine-block--6L');

            }

      }

      };
```

Code Snippet 6. Changing the type

In Code Snippet 6 is shown how to change type of the selected object. This part of the code is linked to controller which is displayed in Code Snippet 7 and that controller is connected to HTML –button. First the written code will clarify the selected object's part. After that it'll verify which type is currently displayed and via button click, the type will change.

```
    vm.differentType = function () {

          $rootScope.$emit('changePart', vm.selected.data);

    }
```

Code Snippet 7. Controller for changing the type

In Code Snippet 7 is the connection with Code Snippet 6 and Code Snippet 7. The object's type will be changed in Code Snippet 6 and in Code Snippet 7, there is the connection between them. Div's data is scoped and the vm.differentType function is triggered, when HTML-button is clicked.

Secondly the object has to have a possibility of rotating the selected component. This was already implemented in Genius Dashboard, but my work could not harm this feature.

Third requirement was to scale component or object. This was implemented into the project via a button.

```
Vm.resizeObjects = function (event, part)

      if(part.id !== vm.data.id) return;
      if(angular.element(targetElement[0].querySelector(.’engine-
      block’)).hasClass(‘engine-block)){

      var object_id = “-“ + vm.data.id + “-“ + vm.data.part + “-“
      + vm.data.type;

      if(angular.element(document.getElementById(ob-
      ject_id))[0].style.height == “”) {

            var   height_new   =   angular.element(document.getEle-
            mentById(object_id))[0].scrollHeight *2;

            var   width_new   =   angular.element(document.getEle-
            mentById(object_id))[0].scrollWidth *1.5;

            angular.element(document.getElementById(ob-
            ject_id))[0].style.height = height_new + “px”;

            angular.element(document.getElementById(ob-
            ject_id))[0].style.width = width_new + “px”;

                  return;

            }

      if(!angular.element(document.getElementById(ob-
      ject_id))[0].style.height == “”) {

            angular.element(document.getElementById(ob-
            ject_id))[0].style.height = “”;

            angular.element(document.getElementById(ob-
            ject_id))[0].style.width = “”;

            return;

            }

      }
```

```
      if(angular.element(targetElement[0].querySelec-
tor(.turbo)).hasClass(‘turbo’)){

var object_id = “-“ + vm.data.id + “-“ + vm.data.part + “-
L“;

if(angular.element(document.getElementById(ob-
ject_id))[0].style.height == “”) {

      var   height_new   =   angular.element(document.getEle-
      mentById(object_id))[0].scrollHeight *1.5;

      var   width_new   =   angular.element(document.getEle-
      mentById(object_id))[0].scrollWidth *1.5;

      angular.element(document.getElementById(ob-
      ject_id))[0].style.height = height_new + “px”;

      angular.element(document.getElementById(ob-
      ject_id))[0].style.width = width_new + “px”;

            return;

      }

if(!angular.element(document.getElementById(ob-
ject_id))[0].style.height == “”) {

      angular.element(document.getElementById(ob-
      ject_id))[0].style.height = “”;

      angular.element(document.getElementById(ob-
      ject_id))[0].style.width = “”;

      return;

      }
}

      else{

var object_id = “-“ + vm.data.id + “-“ + vm.data.part + “-“;

if(angular.element(document.getElementById(ob-
ject_id))[0].style.height == “”) {

      var   height_new   =   angular.element(document.getEle-
      mentById(object_id))[0].scrollHeight *0.75;

      var   width_new   =   angular.element(document.getEle-
      mentById(object_id))[0].scrollWidth *0.75;

      angular.element(document.getElementById(ob-
      ject_id))[0].style.height = height_new + “px”;

      angular.element(document.getElementById(ob-
      ject_id))[0].style.width = width_new + “px”;

            return;

      }

if(!angular.element(document.getElementById(ob-
ject_id))[0].style.height == “”) {

      angular.element(document.getElementById(ob-
      ject_id))[0].style.height = “”;

      angular.element(document.getElementById(ob-
      ject_id))[0].style.width = “”;

      return;

      }

}
```

Code Snippet 8. Resizing the objects

In Code Snippet 8 is a code which will allow the object's resizing. First the object has to be identified and after that the current values are multiplied by some value. After this, the new width and height values are implemented to the object and this will cause object to change the height and width. Currently the object has two different size options which will be displayed by turns via button click.

The fourth requirement was to clone an object or component. This was implemented into the project via an HTML-button. When the button is clicked the selected object will be cloned into the top corner of grid. The function that 'Clone' –HTML-button triggered is vm.cloning (). In the vm.cloning () function new objects were created. This was done in the code in Engines.add method, where the needed parameters were passed from the selected objects. This made it possible to cloning the right object. Cloning needed the object's part, type and wanted coordinates to drop the object to the grid. The cloning function is shown in Code Snippet 9. At first there is a definition of the drop zone aka grid. Then the part and type are sorted out, so the selected object can be cloned. Engines.addPart () will actually make the clone and place it to the assigned coordinates.

```
vm.cloning = function(){

            let dropzoneRect = interact.getElementRect(event, tar-
      get);

            let part = this.stack.selected.data.part;

            let type = this.stack.selected.data.type;
```

```
        let x = 1;

        let y = 2;


        Engines.addPart({

              part: part,

              type: type,

              coordinates: {

                    x: x,

                    y: y,

                    x_absolute:  (x  /  dropzoneRect.width  *
  100).toFixed(4),

                    y_absolute:  (y  /  dropzoneRect.width  *
  100).toFixed(4)

              }

        });

  };
```

Code Snippet 9. Cloning the object

5.1.6 Configuration for overlays of component or object

The objects in the grid should not be able to lay on each other on the grid. This is an important feature to have, so the objects will not be on top of each other. To make this change, all the objects had to be looped through, so that the user can see their coordinates. This allowed for the configuration for overlays to be possible.

5.1.7 Possibility to do customer specific configurations

One of the requirements was that the user can do customer specific configurations, for example changing colour in Genius Dashboard. This will help the configuration to be more individual and unique.

5.1.8 Final look of the Genius Dashboard

In this chapter the final results are shown in the improved Genius Dashboard.

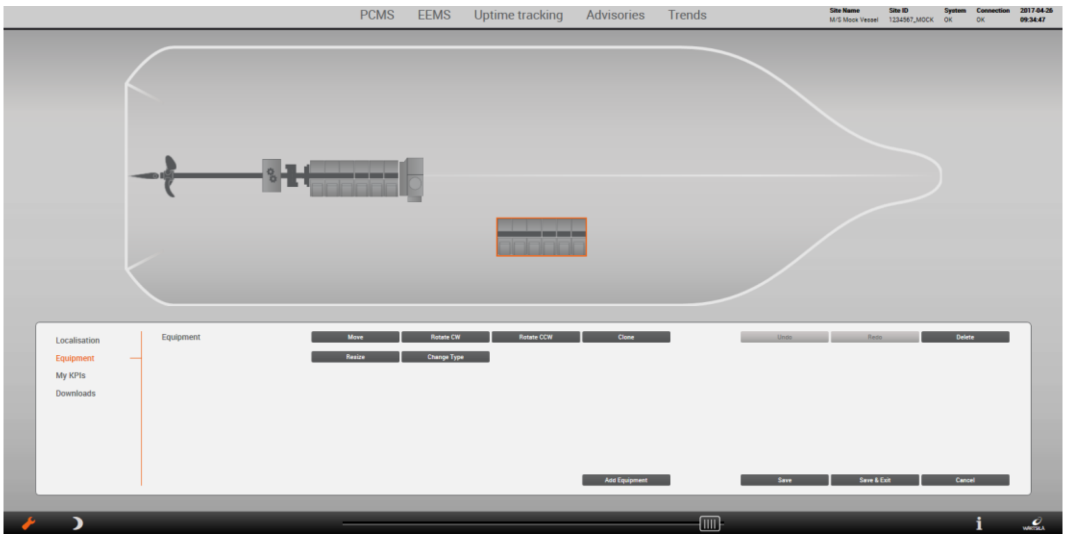


Figure 13. Added features to the Genius Dashboard

The added features to the Genius Dashboard are displayed in figure 13. The elected object has orange frames so that the user knows which object is selected. All the features are listed after one another. Added visible features are cloning, resizing and changing the type.

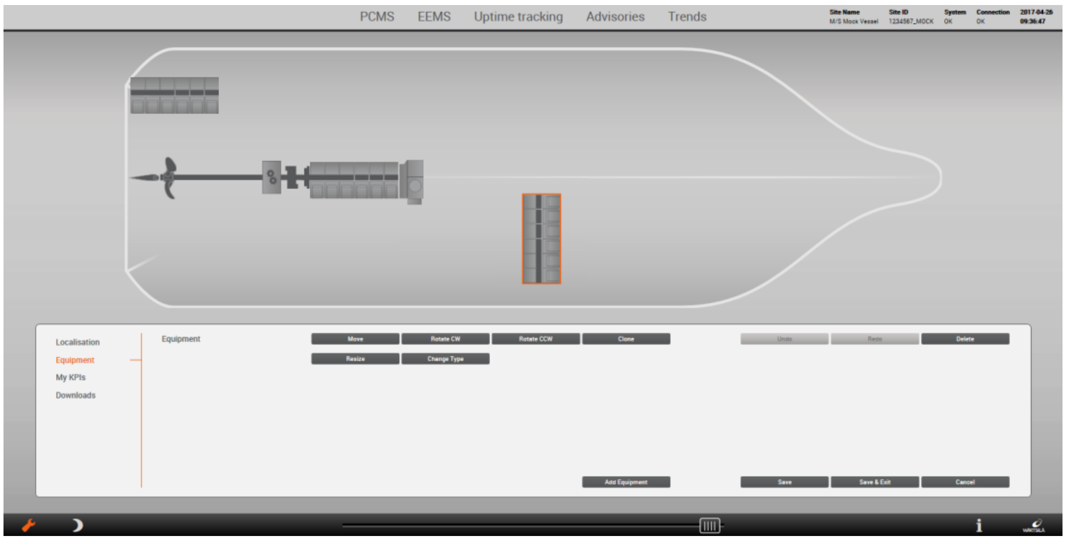


Figure 14. Selected object rotated and cloned

In Figure 14 the selected object is rotated and cloned. The clone of the selected object is displayed in the grid's top-left corner. This is the final look of the improved Genius Dashboard.

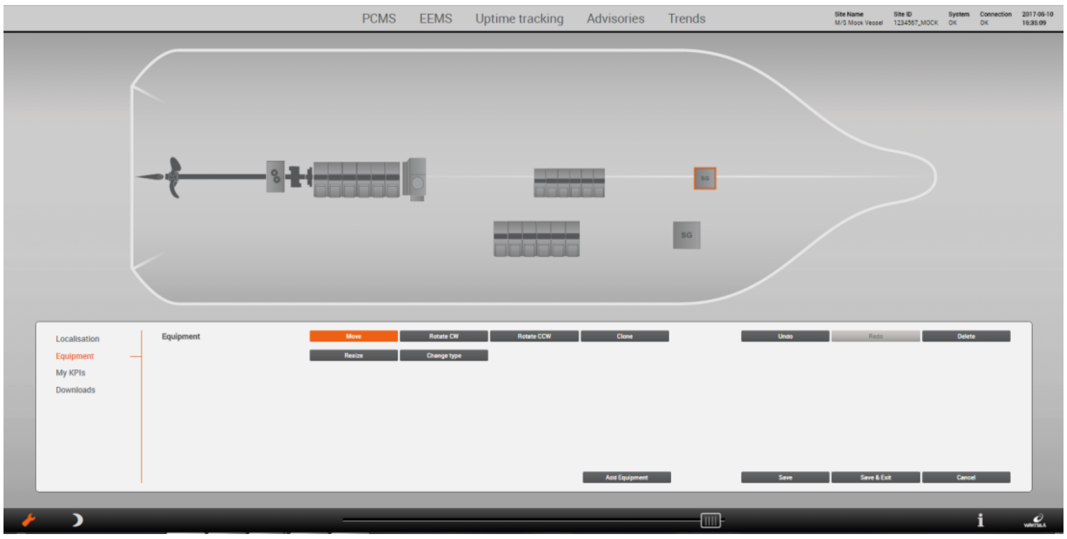


Figure 15. Resizing the objects

In Figure 15 is displayed how the figures are resized. The object will be half the size of the original components when they are resized. In Figure 15, there are two objects resized next to the normal sized objects. The difference is easy to spot, since the resized objects are a lot smaller.

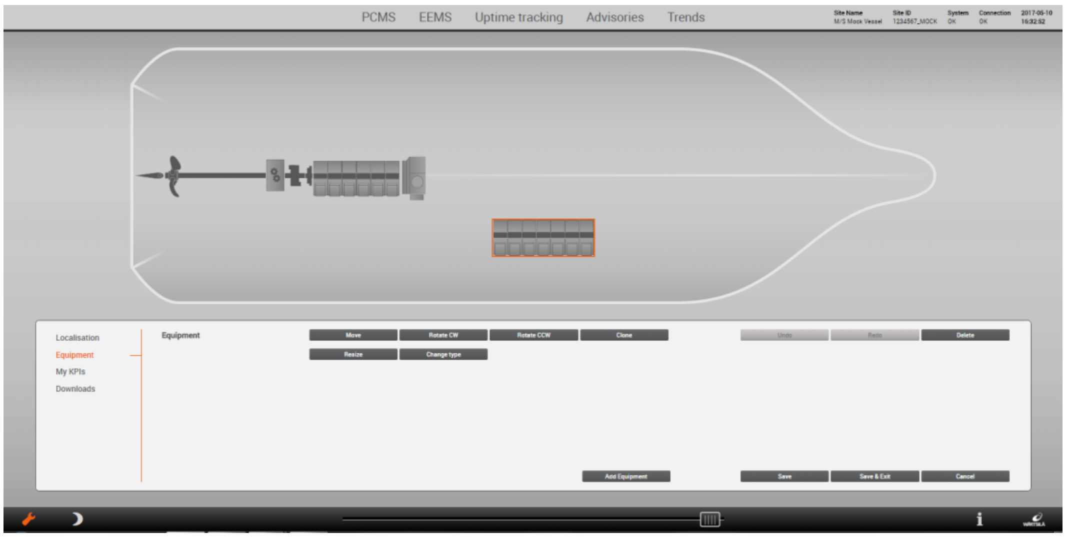


Figure 16. Selected object change type once

In figure 16 is shown what the type of the changed object looks like, when the “Change type” –button is clicked once. This is one of many sizes specifically for the selected object.

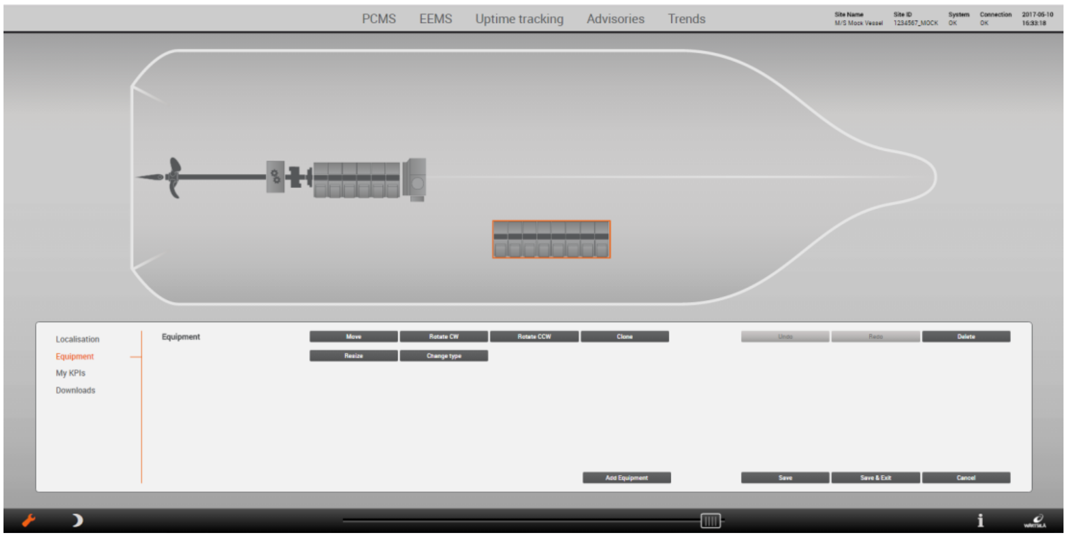


Figure 17. Selected object change type twice

In figure 17 is shown the next type of the specific element. The type is bigger this time and it will grow via the "Change Type" –button until it comes to the biggest type. After that it will start the loop all over again.

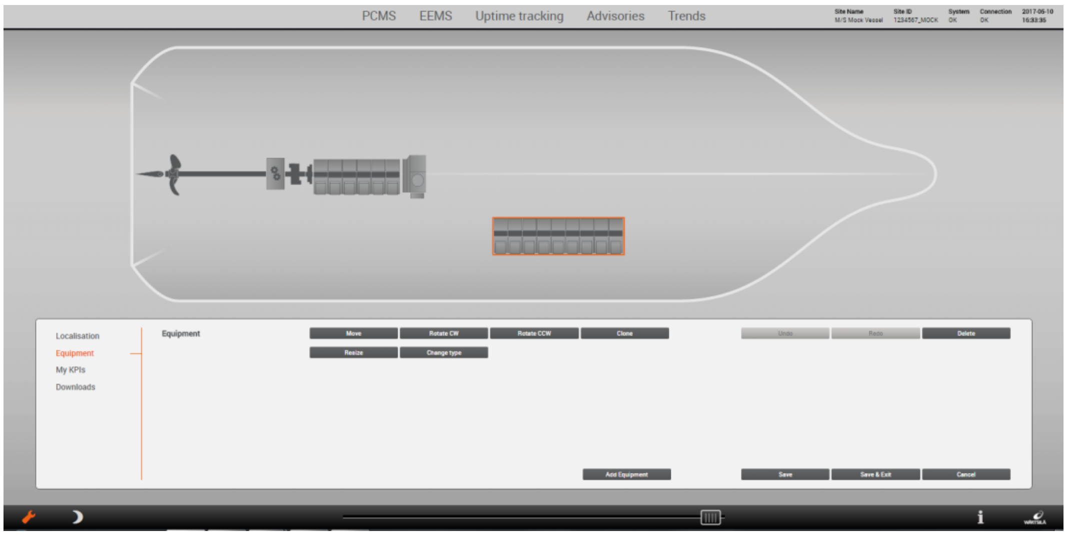


Figure 18. Selected object change type third time

In figure 18 is displayed yet another type of the selected object.

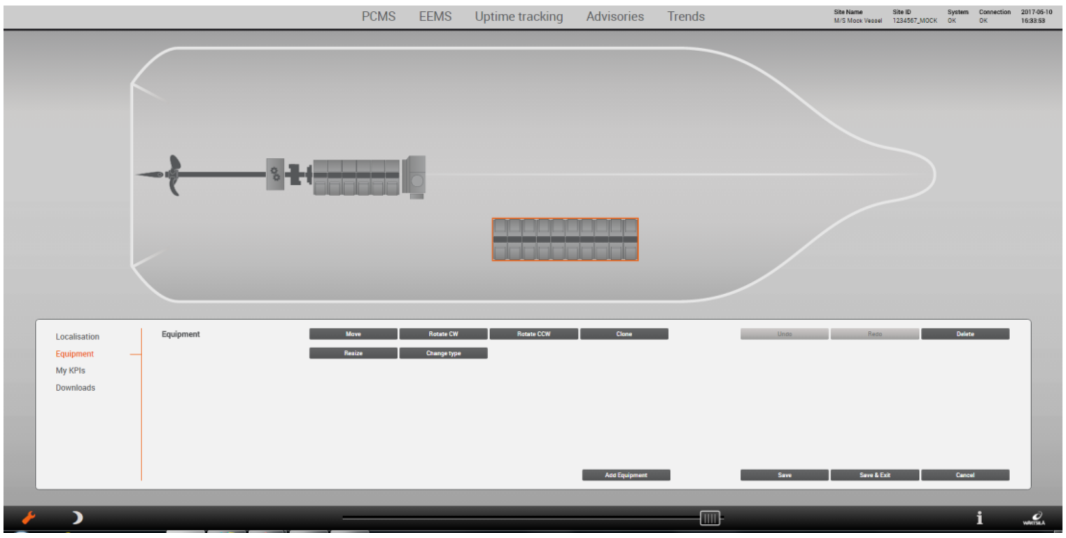


Figure 19. Selected object change type last time

In figure 19 is shown the biggest size of the specific element’s type. After this size element will go back to the original one.

6 TESTING

Testing is a way to evaluate a system and its contents with the aim to find whether system is doing what it is supposed to or not. Testing will provide information of what is happening in the program. Testing is used to identify gaps, errors or missing demands. /13/

There are many ways to test programs, for example there is manual and automation testing. Manual testing is when testers test programs manually, basically without any automated tools or script. The tester takes the role of end-user and tests software to determine any bugs or not wanted behaviour. /13/

Automated test or test automation, is when the tester writes scripts and uses automated tools to test program. This involves automation. Redundant test cases are performed manually, quickly and often. /13/

In this thesis the program was tested using black-box and unit testing methods. The written codes were tested to make sure that the changes worked as wanted and did not harm any previous code. The testing was executed throughout the project when the programming has been completed piece by piece. After finishing every issues unit tests were ran to make sure Genius Dashboard still worked perfectly. The code written in this thesis has been tested through the project. This made it easier to implement every feature, because the Genius Dashboard worked desirably before working with the issues. Multiple browsers were used to make sure that the written codes work.

After all the issues were implemented, the whole thing was tested one more time. It was crucial to make sure that everything works fine individually and as a whole project.

7 SUMMARY

The final result of the thesis was a functional and refined Genius Dashboard. All necessary additions, which were in the requirement specification, were completed. The most important thing was to have successfully implemented the functionalities.

The work was challenging, even though most of the technologies were familiar. AngularJS was a new framework so in order to work with it, studying it was necessary. The projects advancement was varied, it was hard to find help with the issues, because they were so specific. This meant much of help was not available from the internet. After all, the project was interesting and challenging.

In the future, I do not think that Genius Dashboard will be updated, as the unit I was working in Wärtsilä was discontinued at the end of January. However, this thesis was finished. If the unit’s products will be released to production sometime in the future, the project’s results will probably be released as well. At the moment, changes done to the Genius Dashboard will remain in version control.

REFERENCES

/1/ http://www.wartsila.com/

Last accessed 9.3.2017

/2/ https://docs.angularjs.org/guide/introduction

Last accessed 9.3.2017

/3/ https://www.w3schools.com/js/js_intro.asp

Last accessed 13.3.2017

/4/ https://developer.mozilla.org/en US/docs/Learn/JavaS-cript/First_steps/What_is_JavaScript

Last accessed 20.3.2017

/5/ https://www.jetbrains.com/webstorm/

Last accessed 20.3.2017

/6/ https://www.sourcetreeapp.com/

Last accessed 20.3.2017

/7/ http://www.tieke.fi/pages/viewpage.action?pageId=3441242

Last accessed 21.3.2017

/8/ https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

Last accessed 21.3.2017

/9/ http://www.cisco.com/c/en/us/about.html

Last accessed 2.4.2017

/10/ https://support.office.com/fi-fi/article/UML-sekvenssikaavion-luominen-6feaf97d-26ff-4a0e-80e2-327958850a6f

Last accessed 2.4.2017

/11/ http://jsonapi.org/

Last accessed 2.4.2017

/12/ https://www.w3schools.com/graphics/svg_intro.asp

Last accessed 3.4.2017

/13/ http://www.tutorialspoint.com/software_testing/

Last accessed 3.4.2017

/14/ https://www.w3schools.com/html/

Last accessed 3.4.2017

/15/ http://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-cli-ents/index.html

Last accessed 3.4.2017

/16/ https://git-scm.com/

Last accessed 10.4.2017

/17/ https://advancedkittenry.github.io/koodaaminen/arkkitehtuuri/index.html

Last accessed 21.4.2017

/18/ https://www.sitepoint.com/10-reasons-use-angularjs/

Last accessed 21.4.2017