



VAASAN AMMATTIKORKEAKOULU  
VASA YRKESHÖGSKOLA  
UNIVERSITY OF APPLIED SCIENCES

Maria-Corina Sibinescu

# SOPHISTICATED ORDERING TOOL FOR WRM PRODUCTS

Technology and Communication  
2014

VAASAN AMMATTIKORKEAKOULU  
UNIVERSITY OF APPLIED SCIENCES  
Degree Programme of Information Technology

## **ABSTRACT**

Author	Maria-Corina Sibinescu
Title	Sophisticated Ordering Tool for WRM Products
Year	2014
Languages	English
Pages	51
Name of Supervisor	Timo Kankaanpää

---

The process of ordering a product should be convenient for both the customer and the seller. As a means to simplify the ordering of a highly customizable product with a large number of possible features, it is useful to have an efficient and interactive ordering tool that can guide the customer through placing a correct and complete order. The aim of this thesis is to develop a user interface for a web-based ordering tool, which allows easily customizing the order and saving the data into a designated database.

The user interface is implemented using HTML, CSS and the AngularJS framework. Furthermore, choosing the desired features of the ordered product makes use of the REST API of an existing order configurator tool. The user interface communicates with the REST API using the CORS technology, constantly exchanging JSON data to customize the order. Lastly, when the order is fully configured, the data is processed using PHP, being sent to the concerned sales personnel within the company, as well as saved in an OpenERP database for further use.

The ordering tool is a useful method of simplifying the sales process and ensuring that a placed order is valid and fully configured.

---

Keywords	Ordering Tool, Order Configurator, AngularJS, Dynamic User Interface
----------	--

## TIIVISTELMÄ

Tekijä	Maria-Corina Sibinescu
Opinnäytetyön nimi	Sophisticated Ordering Tool for WRM Products
Vuosi	2014
Kieli	englanti
Sivumäärä	51
Ohjaaja	Timo Kankaanpää

---

Tilausprosessi on parhaimmillaan joustava sekä asiakkaalle että myyjälle. Monimutkainen ja hyvin modulaarinen tuote, jossa on suuri määrä mahdollisia valintoja voi helposti olla sekä asiakkaalle vaikea ostaa että myyjälle vaikea myydä. Asian helpottamiseksi tarvitaan tehokas ja interaktiivinen tilaustyökalu, joka ohjaa asiakasta luomaan oikean ja täydellisen tilauksen. Tämän opinnäytetyön tarkoituksena oli toteuttaa käyttöliittymä selainpohjaiselle tilaustyökalulle, joka mahdollistaisi monimutkaisen tuotteen tilaamisen helposti ja tiedon tallettamisen sille tarkoitettuun tietokantaan.

Käyttöliittymän ohjelmointi toteutettiin HTML:llä, CSS:llä ja AngularJS:llä. Lisäksi tilattavien tuotteiden toiminnallisuuksien näyttämisessä ja valitsemisessa tilaustyökalu kommunikoi Summium myynticonfiguratorin REST API:a vasten CORS-teknologia käyttäen välittäen JSON-muotoista tietoa. Lopuksi kun tuotteen valinnat on tehty, tieto prosessoidaan PHP:tä käyttäen ja lähetetään tarvittaville myyntihenkilöille sekä talletetaan OpenERP:n tietokantaan jatkokäsittelyä varten.

Opinnäytetyössä toteutettu tilaustyökalu helpottaa tilausprosessia ja varmistaa asiakkaiden tekemien tilausten oikeellisuuden.

## CONTENTS

LIST OF FIGURES AND TABLES.....	7
LIST OF ABBREVIATIONS.....	9
1.INTRODUCTION.....	10
1.1. Client Organization (Wapice Oy).....	10
1.1.1 WRM(Wapice Remote Management).....	10
1.1.2. Summium.....	11
1.2. Current Situation.....	11
1.3. Project Objectives and Outcome.....	11
2.USED TECHNOLOGIES.....	12
2.1. HTML.....	13
2.2. CSS.....	13
2.3. JavaScript.....	14
2.3.1. jQuery.....	14
2.3.2. AngularJS and the MVC development model.....	15
2.4. JSON (JavaScript Object Notation) and JSON parsers.....	15
2.4.1. JSON data.....	15
2.4.2. Choosing an appropriate JSON parser.....	16
2.5. REST.....	16
2.6. CORS.....	17
2.7. CMS.....	17
2.8. OpenERP.....	18
2.9. Used Software Tools.....	19
3.SYSTEM DESCRIPTION.....	19
3.1. Use Cases.....	20
3.1.1. Use Case Diagram.....	20
3.1.2. Register.....	21
3.1.3. Login.....	22

3.1.4. Place an Order.....	22
3.1.5. Modify user details.....	23
3.1.6. View Order Status.....	23
3.1.7. View Order History.....	24
3.2. Software requirements specification.....	24
3.3. Functional specification.....	25
3.3.1. Application Flow.....	26
3.3.2. Sequence Diagrams.....	28
3.4. Technical Specification.....	31
3.4.1. Application architecture.....	31
3.4.2. Detailed design.....	33
3.5. Deployment Diagram.....	36
3.6. Testing Specification.....	36
3.6.1. Invalid login credentials.....	36
3.6.2. Invalid data format in registration.....	37
3.6.3. Non-numerical values in quantity and cable length fields.....	37
3.6.4. Changes to Summium model.....	37
3.6.5. XML-RPC connection broken.....	38
3.6.6. Connection to REST API broken.....	38
4. IMPLEMENTATION AND DEPLOYMENT.....	39
4.1. AngularJS Application Structure.....	39
4.2. GUI Design and Implementation.....	40
4.2.1. AngularJS implementation – used concepts and syntax.....	41
4.2.2. Login View.....	44
4.2.3. Registration View.....	45
4.2.4. Product Selection View.....	47
4.2.5. Product Configuration View.....	47
4.2.6. Order Confirmation View.....	52
4.2.7. Order Placement View.....	53
4.3. Communicating with the REST Services.....	54

4.4. Registering the Order.....	54
4.4.1. Registering the order through an email.....	54
4.4.2. Inserting Data into OpenERP.....	55
4.5. Application Deployment.....	56
5. TESTS, RESULTS AND ANALYSIS.....	57
5.1. Application Testing.....	57
5.1.1. Invalid Login Credentials.....	57
5.1.2. Invalid Data Format in Registration.....	57
5.1.3. Non-numerical values in quantity and cable length fields.....	58
5.1.4. Changes to Summium model.....	58
5.1.5. XML-RPC connection broken.....	59
5.1.6. Connection to REST API broken.....	59
5.2. Implementation Results and Analysis.....	60
5.2.1. Implementation limitations.....	60
5.2.2. Future work and usage.....	61
6. CONCLUSIONS.....	62
7. REFERENCES.....	63

## **LIST OF FIGURES AND TABLES**

Figure 1. Use case diagram for the WRM Ordering Tool	p21
Figure 2. WRM Ordering Tool mock-ups	p27
Figure 4. Registration process sequence diagram	p29
Figure 5. Login process sequence diagram	p29
Figure 6. Product selection process sequence diagram	p30
Figure 7. Product configuration process sequence diagram	p30
Figure 7. Order registration process sequence diagram	p31
Figure 8. Ordering Tool Package diagram	p33
Figure 9. Controllers detail implementation	p35
Figure 10. Deployment diagram	p36
Figure 11. Structure of the “js” directory	p39
Figure 12. Structure of the root directory of the ordering tool	p40
Figure 13. Using the “ng-app directive”	p41
Figure 14. Example of AngularJS routing	p41
Figure 15. Example of a view implementation in AngularJS	p42
Figure 16. Declaration of an AngularJS module	p43
Figure 17. Syntax of an AngularJS controller	p43
Figure 18. Screenshot of the login view	p44
Figure 19. Login view with wrong credentials	p45
Figure 20. Registration view screenshot	p46
Figure 21. Product selection view screenshot	p47
Figure 22. Configuring a WRM247 with 3 connectors product	p49
Figure 23. Configuring a WRM247 with 5 connectors product	p50
Figure 24. Configuring a WRM247+ product	p51
Figure 25. Configuring a WRM365 product	p52

Figure 26. Order confirmation view example	p53
Figure 27. Order placement view example	p53
Figure 28. OpenERP GUI screenshot	p56
Figure 29. Required field empty error	p56
Figure 30. Incorrect format error	p58
Figure 31. Non-numerical value error	p58
Figure 32. Valid number	p58
Figure 33. Error registering the order	p59
Figure 34. Connection to REST API broken error	p59
Table 1. Use case “Register”	p22
Table 2. Use case “Login”	p22
Table 3. Use case “Place an order”	p22
Table 4. Use case “Modify user details”	p23
Table 5. Use case “View order status”	p23
Table 6. Use case “View order history”	p24
Table 7. Ordering tool requirements	p25



## **LIST OF ABBREVIATIONS**

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CMS	Content Management System
CORS	Cross Origin Resource Sharing
CSS	Cascading Style Sheets
ERP	Enterprise Resource Planning
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
MVC	Model View Controller
PHP	Hypertext Preprocessor
REST	Representational State Transfer
RPC	Remote Procedure Call
SPA	Single Page Application
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WRM	Wapice Remote Management
XAMPP	X(Cross) Apache MySQL PHP Perl
XML	Extensible Markup Language

# 1. INTRODUCTION

The application implemented in this thesis was developed as a project for the Finnish company Wapice Oy, to aid in the selling of their WRM products.

## 1.1. Client Organization (Wapice Oy)

Wapice Oy is a Finnish company established in 1999 in Vaasa. The company provides expertise in both software and electronics, as a subcontractor and through their own products, WRM(a remote management solution) and Summium(a sales configurator).

A large number of clients of the company come from the energy sector including large, well-known companies such as ABB and Wärtsilä.

Wapice Oy currently employs over 260 IT experts and the number is constantly growing. The company is present in 6 locations around Finland(Vaasa, Tampere, Seinäjoki, Oulu, Hyvinkää and Jyväskylä ). The headquarters are located in Vaasa.

### 1.1.1 WRM(Wapice Remote Management)

WRM(Wapice Remote Management) is a complete remote management solution, suited for a wide range of features, including remote control, location tracking and diagnostics.

WRM electronics are small devices that can be programmed and can communicate with the WRM server. Currently, 4 types of devices are available:

- WRM247 with 3 connectors
- WRM247 with 5 connectors
- WRM247+ (this device has 8 connectors)
- WRM365 (two types of device are available)

An order of a WRM device requires choosing the following characteristics:

- Possible cable lengths and end connectors for each cable in the case of WRM247 (each connector on a product can support one such cable)

- Type of device in case of WRM365
- Accessories to be shipped with the product
- The possibility of choosing a template, when the ordering company is a previous customer and existing templates have been assigned to it. Since the WRM product templates do not currently exist, choosing a template will not be part of this thesis implementation.

### **1.1.2. Summium**

Summium is a sales configurator product developed by Wapice Oy. In this project, Summium is used to configure a product, as well as manage the users of the ordering tool.

Summium allows for rules and dependencies to be added to the product. Therefore, when configuring an order through Summium, each choice of the client triggers the display of new, appropriate data.

The ordering tool communicates to Summium constantly for managing the user information and configuring the ordered product.

### **1.2. Current Situation**

Currently, placing an order for a WRM product is done by sending an order email from the customer to the relevant persons inside the company. However, this method is not the most convenient and efficient for both the customers and the company's employees involved in the sales of the WRM products.

### **1.3. Project Objectives and Outcome**

The company is looking to develop a sales tool to be used by their clients for placing orders for the WRM products. The tool will provide a clear and easy to use interface for the clients, allowing them to place and customize an order in a more efficient way. Furthermore, the tool will be based on the company's Summium product, and will use Summium's Representational State Transfer (REST) services to provide the functionality. At the end of the ordering process, the order will be placed by sending

an email with the data configured through the user interface, as well as inserting the data to OpenERP.

The scope of this thesis is to develop the user interface of the ordering tool. The REST services will not be implemented as part of this final project.

## **2. USED TECHNOLOGIES**

The ordering tool's user interface will be built as a part of the WRM's website, on top of a content management system. The user interface (UI) will be implemented using HTML and CSS.

The functionality will be implemented using JavaScript and the UI will communicate with the Summium REST services through JSON data.

After the order confirmation by the customer, an email will be sent to the concerned personnel and the data will be inserted to OpenERP. This action will be performed using PHP.

### **2.1. HTML**

HyperText Markup Language(HTML) is the language in use for publishing web documents. The current most commonly used version is HTML 4.01, along with HTML5, which is in use, but not fully defined.

HTML defines the elements that make up a document, such as paragraphs, links, tables, titles, embedded objects etc. Furthermore, HTML uses stylesheets for presentation purposes (although certain HTML attributes can be used for the same purpose) and scripts for creating dynamic, interactive web documents. /12/

### **2.2. CSS**

Cascading Style Sheets(CSS) is a system of rules which allows the styling, spacing and positioning of the content on a web page. /12/

A CSS rule contains a selector, which defines to what elements the rule is applied, and a set of pairs of properties and their assigned values, enclosed in braces ({}). Each property-value pair is followed by a semicolon.

CSS can be used to format a web document in three different ways:

- Inline style sheets (used directly inside an HTML tag through the “style” attribute)
- Internal style sheets (used in the head of the web document inside the tag <style>)
- External style sheets (separated documents imported through the tag <link>)

The main advantage of using CSS is separating the structure of the document from their presentation (layout, positioning and custom look). External CSS files will be used in the implementation of this thesis, as they offer more possibilities to customize a document by simply linking a different file.

### **2.3. JavaScript**

JavaScript is a scripting language that can be executed by all modern browsers. JavaScript is inserted into a web document using the <script> tag.

Scripts can be placed in the head or body of a web document, as well as linked from an external file with the extension .js.

JavaScript uses a type of object-oriented programming, called prototype-based programming. Prototype based programming does not contain classes, as defined in standard object-oriented programming. Instead, the role of a class is filled in by functions. A function can be instantiated to create an object using the keyword “new”. Furthermore, the function itself serves as the constructor, and does not need any method explicitly defined for this purpose. Other methods can be added using the “prototype” keyword, while properties(variables) are added using the “this” keyword.

Prototype-based programming makes classic object-oriented programming, such as inheritance, encapsulation and polymorphism, available in JavaScript. /4, 6/

JavaScript presents as an advantage an increased speed and reduces the load on the server, being run in the client's browser. While security issues arises because of JavaScript being run on the client computer, modern web standards reduce the threat significantly. Several libraries, as well as add-ons are available, making the development process easier and more customizable. /5/

### 2.3.1. jQuery

“jQuery is a fast, small, and feature-rich JavaScript library” /7/.

jQuery is a popular and powerful JavaScript library that may be used extensively to provide the functionality of the user interface. Firstly, jQuery can be used in this project to provide animations, as well as layout changes in response to user's actions.

In addition, jQuery can be used for making AJAX calls to the REST API of the sales configurator and process the received data, as well as send data modified by the customer through the user interface.

### 2.3.2. AngularJS and the MVC development model

AngularJS is a powerful JavaScript framework developed by Google. It extends HTML, allowing easy retrieval of data and a fast, responsive interface. AngularJS is particularly suitable for Single Page Applications (SPAs), where the content of the page changes dynamically based on user interactions. /1/

Furthermore, AngularJS allows the use of the MVC (Model-View-Controller) development model. The MVC development model easily separates an application into 3 functional parts:

- Model: a representation of the data structure manipulated by the application
- View: defines the page that is presented to the user
- Controller: connects the view and the model and provides necessary functionality

The possibility of using the MVC allows for clarity of the implementation and ease of maintenance.

Overall, AngularJS appears to be a suitable solution for the implementation of the application and its main alternative is the use of the jQuery library.

## 2.4. JSON (JavaScript Object Notation) and JSON parsers

Data exchange throughout the application is done using the JSON data format.

### 2.4.1. JSON data

JSON is a light-weight format meant for data exchange, based on a subset of JavaScript. JSON permits quick and simple data exchange, is completely language independent and easy to read/generate by both humans and computers.

JSON construction is based on two main features:

- An object as a collection of 0 or more pairs of name and value. Objects are enclosed in braces ({}). A colon (:) is used between the name and the corresponding value, while the name-value pairs are separated by comma (,).
- An array as a collection of ordered values(0 or more). Arrays are enclosed between square brackets ([]) and separated by commas (,).

A value can be:

- A primitive data type(string, Boolean, number, null)
- A structured data type(object, array)

A name can only be a string and names must be unique in an object.

Clear advantages of JSON include data simplicity, human readability and less need for a specialized software for data exchange. /3, 10/

### 2.4.2. Choosing an appropriate JSON parser

Modern browsers incorporate a JSON parser, and if the native parser exists, it will be automatically preferred to any external parser.

For older browsers, which include browsers currently in use, such as IE7, a fall-back parser must be used. This also appears as a priority since mobile browsers have relatively recently started to implement native JSON parsers.

During this project, JSON will be parsed as a JavaScript object, since having knowledge of the structure of the JSON data allows relatively easy access to any value by use of the key associated with it.



For example, when using the JSON object named “object1”, whose value is “{'name': 'name1', 'value': 'value1' }”, the value 'name1' can be accessed by using the key 'name' in the following way: object1.name.

In this situation, no specific JSON parser needs to be used.

## **2.5. REST**

REST (Representational State Transfer) is an architectural style designed for Web services. REST services are stateless, use HTTP methods, show URI (Uniform resource identifier) structure and transfer XML and/or JSON./12/

The ordering tool's REST services implementation is beyond the scope of this thesis, but the user interface communicates with the services, transferring data constantly.

In order to be able to send and receive JSON data from a REST service, it is sufficient that the service's URI is known. The necessary URIs for data retrieval have been described in the REST services' specification.

## **2.6. CORS**

CORS(Cross-Origin Resource Sharing) allows an API to make cross-origin requests on the client-side, a behaviour that is normally disabled because of security issues. In order to allow cross-origin data transfer, a number of headers must be set accordingly in the request of the client and the response of the service on the server.

CORS allows the resource to be shared if the “Origin” header of the request is present and it specifies a case-sensitive match of any of the locations that are enabled to access the resource. In this case, the “Allow-Control-Allow-Origin” header will be set to the value of the “Origin” header of the request and, if credentials are used, the header “Access-Control-Allow-Credentials” is set to “true”. If no credentials are used, the “Allow-Control-Allow-Origin” header can be set to “\*” as well, which indicates that all origins are allowed to access the resource. /2, 11/

## **2.7. CMS**

A CMS(Content Management System) is an application that allows users to create and manage content without the need of programming languages./9/

Due to the convenient way of adding, connecting and editing data, CMSs are a popular solution for web pages that are constantly being updated. CMSs are based on

a webpage template, dynamically generating content using PHP and storing data in a database (MySQL).

Using a CMS, creating a new page for a website or updating a menu can be done without any web development knowledge, through a graphical user interface.

A CMS is usually comprised of a front-end (the website available to the public) and a back-end (used for website administration and accessible through a valid username and password combination).

Popular CMS solutions include Wordpress, Joomla! and Drupal.

## **2.8. OpenERP**

OpenERP is an open-source ERP (Enterprise Resource Planning).

An ERP is a type of software that combines the functionalities of applications designed for each major department in a company and allows for close collaboration and data sharing between the departments.

As a final step in the development of this project, data will have to be inserted through JavaScript to OpenERP. OpenERP allows data to be inserted into one of its databases from a client application by use of XML-RPC services.

An XML-RPC service resides at a specific URL address and can be easily accessed if the URL is known. Access to XML-RPC services in OpenERP is granted through the URL `openerpservername/xmlrpc`. Several methods are accessible at this URL in order to cover a large range of functionality. /8/

XML-RPC clients can be implemented in a large number of programming languages.

Due to simplicity and large-scale support, PHP was chosen to implement the XML-RPC client in this application. Furthermore, PHP was a natural choice considering that the other part of the order registration (sending an email to the interested

personnel within the company) will be implemented using this programming language.

In order to use PHP to call XML-RPC services, an XML-RPC library should be added to the server and imported into the application. In this application, as per OpenERP's official documentation's recommendation, the library named “XML-RPC for PHP” was used. /8, 13/

## **2.9. Used Software Tools**

For writing HTML/CSS/JavaScript, any basic text editor can be used. Due to its simplicity and familiarity to the tool, Notepad++ was chosen for this purpose.

Testing the user interface can be done easily in any browser. However, it is a known fact that CSS might work differently depending on the browser. Therefore, in order to ensure that the application looks identical for all (or an extremely high percentage of) the users, it must be tested at least in the following browsers:

- Chrome 32.0 and higher
- Mozilla Firefox 19.0 and higher
- Safari 6.0 and higher
- Internet Explorer 10 and higher (Internet Explorer < 10 do not fully support CORS and are therefore not suitable for this application)
- Opera 20.0 and higher

In order to run PHP files, a PHP-enabled server must be available. For the purpose of implementing the application, a XAMPP server has been configured locally. A XAMPP server offers a wide range of features, including an Apache server with PHP support necessary for the development of this project. Upon completion of the implementation, the application will be deployed on a publicly available Apache server with PHP support belonging to Wapice Oy.

### **3. SYSTEM DESCRIPTION**

Before starting the implementation of a software project, the requirements of the project must be gathered and analysed. This process results in the project specification, which gives the client and the developers a clear understanding of the project to be developed, the methods to be used and the results to be expected.

The current chapter describes the system to be developed, as resulted from the requirements gathering and analysis process.

#### **3.1. Use Cases**

Since a user interface is mainly based on the interaction between a machine and a human, identifying the use cases allows getting a clear picture of the functionality the interface must provide.

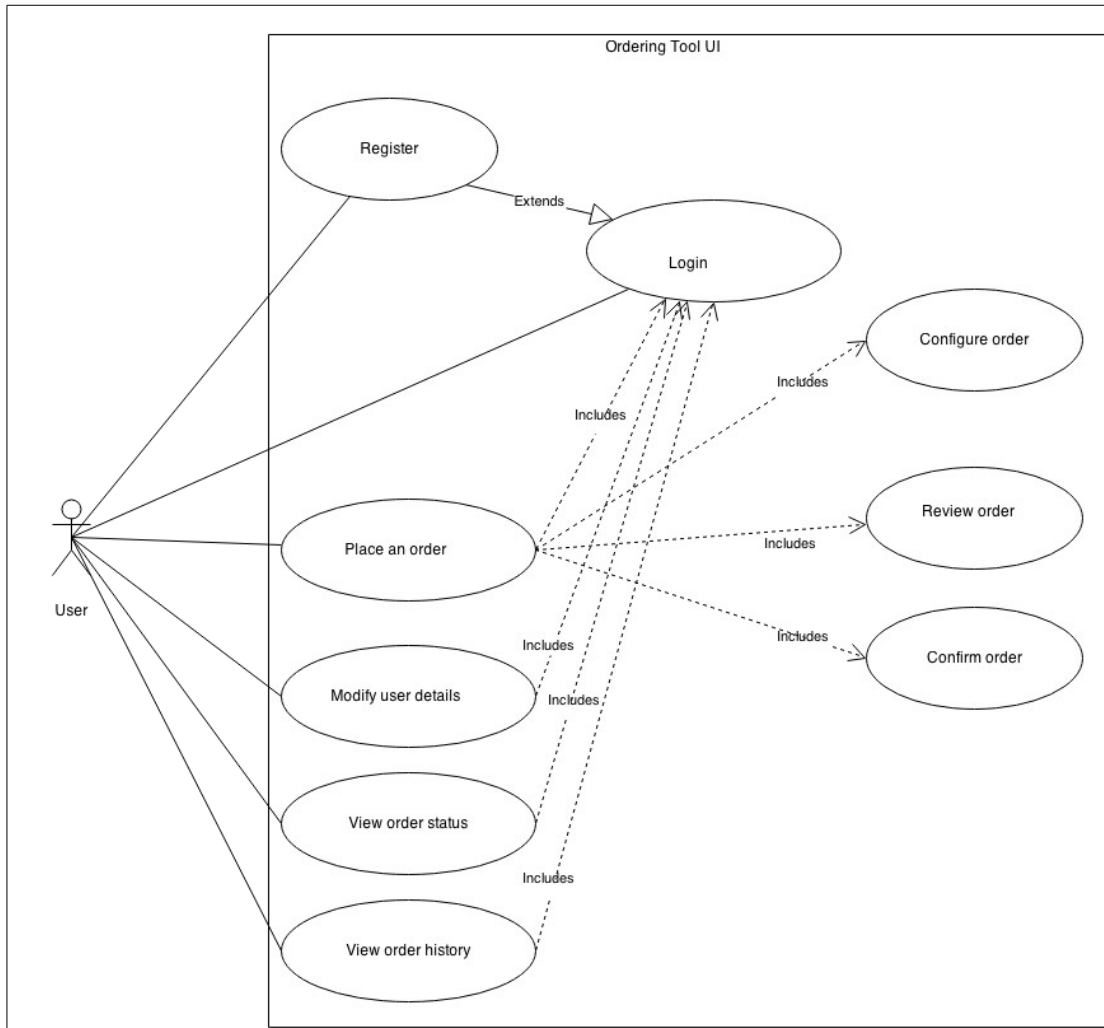
Each use case describes an action that can be performed by a user of the application. All use cases are gathered in a use case diagram of the application.

The use case diagram, as well as details about each specific use case, can be found below.

##### **3.1.1. Use Case Diagram**

In Figure 1, the use case diagram of the WRM Ordering Tool user interface is presented.

The actor in the use case is the customer(or application user), that performs all the actions that trigger an event in the user interface.



**Figure 1:** Use case diagram for the WRM Ordering Tool.

### 3.1.2. Register

In Table 1, the main characteristics of the use case “Register” are described. The customer performs a registration to gain a valid combination of username and password for using the ordering tool. Furthermore, the customer provides during the registration process all the necessary data for the sales process.

**Table 1:** Use case "Register".

Preconditions	-
Input	All the necessary user data: username, password, name, phone number, email, shipping address and invoicing address
Description	User registers to be able to use the ordering tool
Exceptions and errors	Registration cannot be completed; missing information
Result and outputs	Login page is loaded; user can now login with previously registered username and password

### 3.1.3. Login

In Table 2, the main characteristics of the use case "Login" are described.

A user can only log in if valid credentials have previously been obtained through the registration process.

**Table 2:** Use case "Login".

Preconditions	User must be registered
Input	Username and password
Description	User logs in to use the ordering tool
Exceptions and errors	Username and/or password not valid
Result and outputs	Access to the ordering tool's functionality

### 3.1.4. Place an Order

In Table 3, the main characteristics of the use case "Place an order" are described.

**Table 3:** Use case "Place an order".

Preconditions	User must be logged in
---------------	------------------------

Input	All the necessary data for order configuration; button click for confirming the order
Description	User places an order by completing three steps: configuring the order in detail according to their needs, reviewing the order and as a final step, confirming that the order is correct and can be placed
Exceptions and errors	Invalid or missing data; problems in communicating with the order configurator
Result and outputs	An order is placed: an email is sent to the concerned personnel and order data is inserted into OpenERP

### 3.1.5. Modify user details

In Table 4, the main characteristics of the use case “Modify user details” are described.

**Table 4:** Use case "Modify user details".

Preconditions	User logged in
Input	All the data that needs to be modified (contact information, shipping and invoicing addresses, password)
Description	User can modify his or her own data
Exceptions and errors	Information cannot be updated
Result and outputs	Information updated and confirmation message; error message

### 3.1.6. View Order Status

In Table 5, the main characteristics of the use case “View order status” are described.

**Table 5:** Use case "View Order Status".

Preconditions	User logged in; order previously placed
---------------	---

Input	Order code
Description	User views the status of his/her order
Exceptions and errors	Cannot retrieve data
Result and outputs	Information about the order status is displayed

### 3.1.7.View Order History

In Table 6, the main characteristics of the use case “View order history” are described.

**Table 6:** Use case "View Order History".

Preconditions	User logged in
Input	-
Description	User view his/her order history
Exceptions and errors	Cannot retrieve data
Result and outputs	Information about the order history is displayed

## 3.2. Software requirements specification

The ordering tool's UI must fulfil the requirements described in Table 7.

The requirements are divided into three groups based on their priority level:

- Must-have requirements, numbered “1”, are the most basic features the application must implement
- Should-have requirements, numbered “2”, are important features that should be present for the user experience to be complete
- Nice-to-have requirements, numbered “3”, are extra features and the success of the project does not depend on their implementation



**Table 7:** Ordering Tool requirements

No.	Description	Priority
1	User is able to register to use the ordering tool	2
2	User is able to log in to use the ordering tool	1
3	User is able to place an order by configuring, reviewing and confirming it	1
4	The ordering tool communicates with the order configurator's REST API	1
5	An email of the order is sent to the concerned personnel within the company	1
6	Order data is inserted to OpenERP	3
7	User can modify their personal details	2
8	User can view the status of their order	3
9	User can view their order history	3

### **3.3. Functional specification**

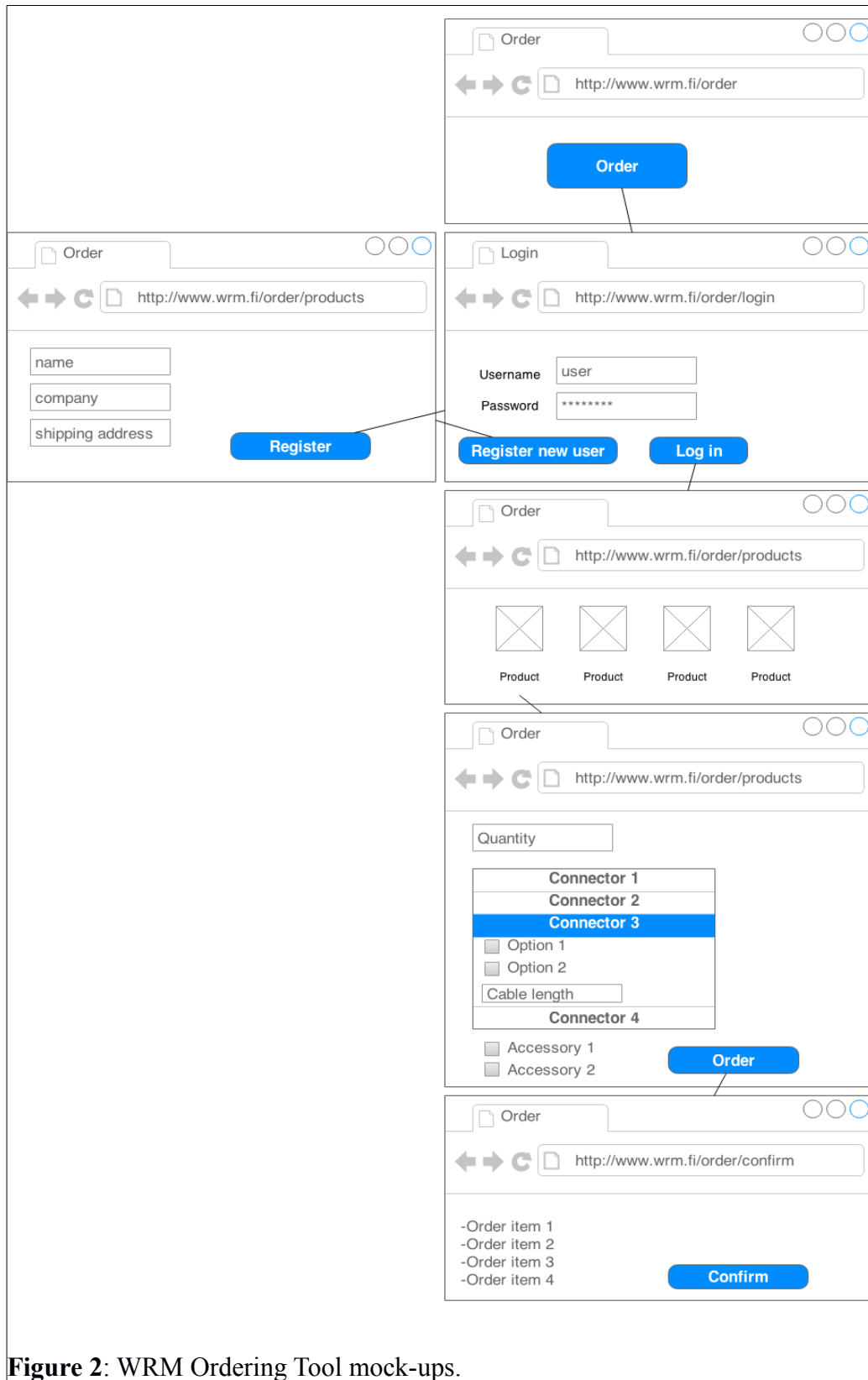
In regard to the functionality of the user interface, the following options are available to the user:

- Registering option through filling in a detailed form
- Login option by using a username and a password. Login succeeds only if the user has previously registered and the username/password combination is correct
- Modifying own user details. A user can modify their own details, including the password, phone number, shipping and invoicing addresses

- Viewing the status of an existing order. A user can view the status of an order he/she has previously placed
- Viewing the order history. A user can view his/her own previous order history
- Placing an order. The order placement option is available to a user after a successful log in and it requires a number of steps to complete, as detailed below:
  - User can choose the product to be ordered from a list of available products
  - User can choose the desired features of the product (in the case of a WRM product, these are cables and connectors, accessories and, in the future, templates). The configuration of the product is done through an intuitive interface, that comprises of checkboxes, textboxes and drop-down lists. The user receives feedback when the selection is invalid or an error occurs.

### **3.3.1. Application Flow**

In order to better understand the flow of the application, the application mock-ups in Figure 2 can be used. The mock-ups provide a basic outline of the user interface.



**Figure 2:** WRM Ordering Tool mock-ups.

As it can be seen from the image above, several pages are presented to the user:

- A specific WRM website page from where the ordering process starts
- Login page
- Register page
- Product selection page(accessible only to logged in users)
- Product configuration page(loaded after the user chooses a product on the product selection page)
- Order confirmation page(presented to the user after the customization process is complete)

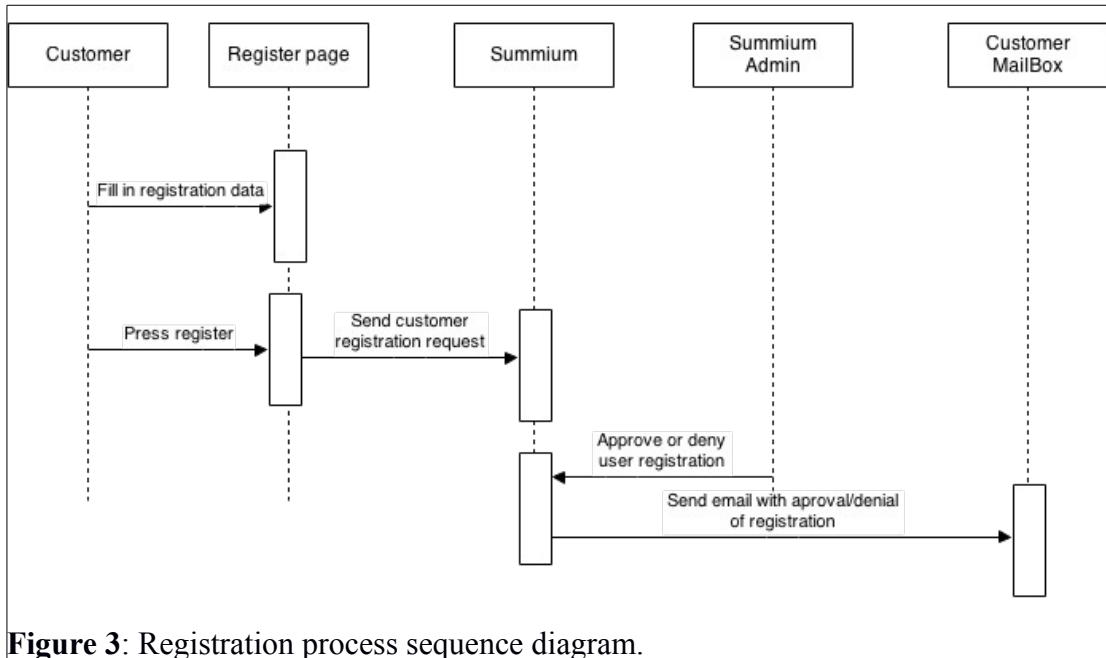
The division of functionality into these pages ensures that the user is guided through the ordering process in a natural manner, each page corresponding to one use case and therefore one user action.

### **3.3.2. Sequence Diagrams**

Sequence diagrams are used to describe the sequence of events taking place in the application.

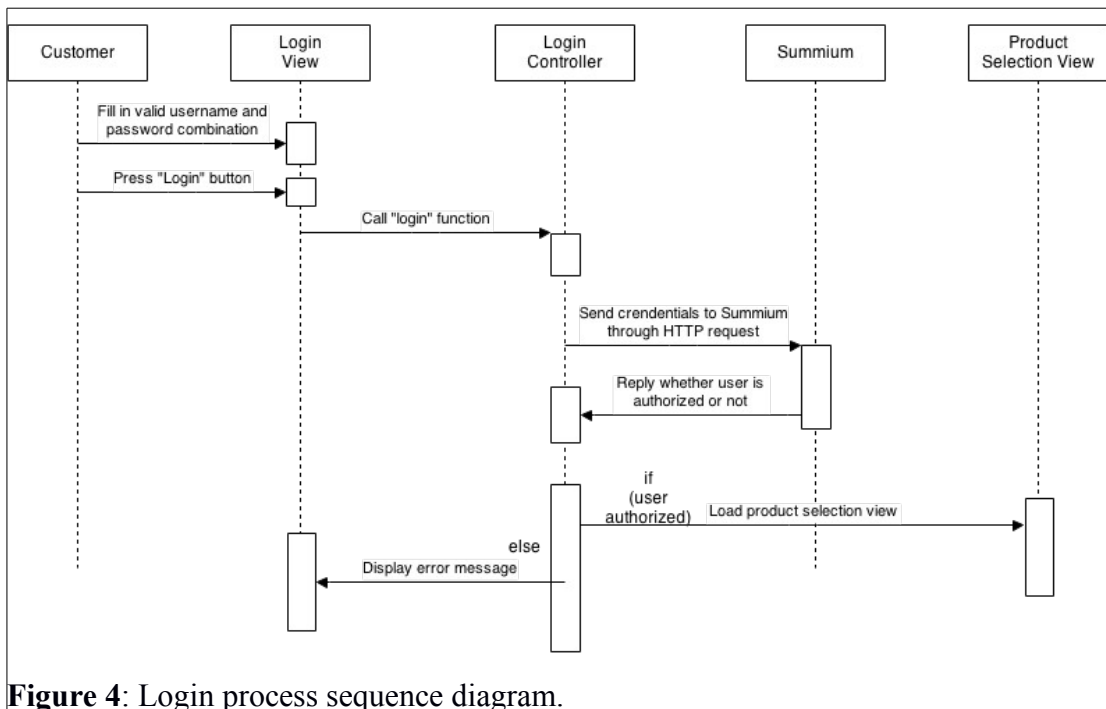
The figures below detail the sequence of the ordering process, from the user login until the registration of the order after confirmation. It can be seen how each user interaction affects the behaviour of the application, as well as when and how the requests to the REST API are made.

Figure 3 details the actions involved in the user registration process. It can be seen that a Summium administrator must approve the registration. This significantly lowers the risk of spam accounts being created.



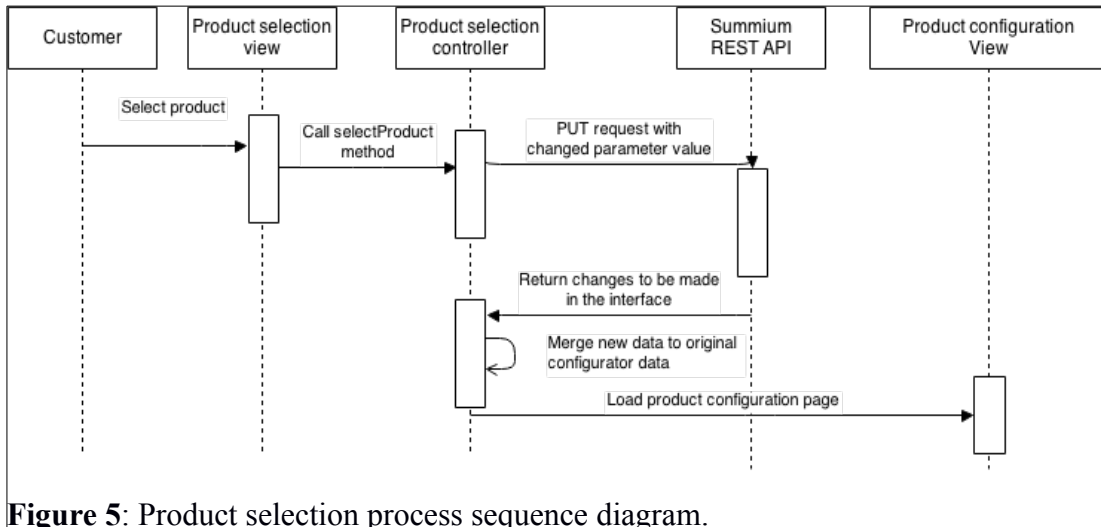
**Figure 3:** Registration process sequence diagram.

Figure 4 presents the login sequence diagram of the ordering tool. It can be seen that the username and password are validated through Summium REST API.



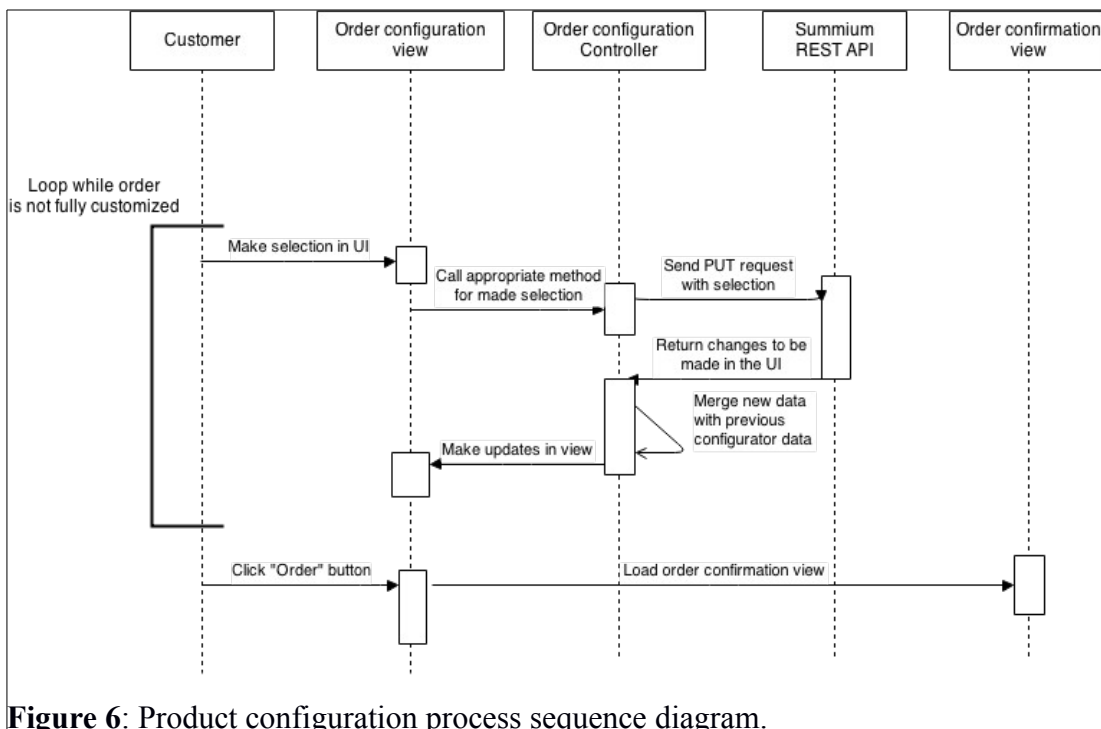
**Figure 4:** Login process sequence diagram.

Figure 5 presents the sequence through which a use makes a product selection, the first step in placing an order.



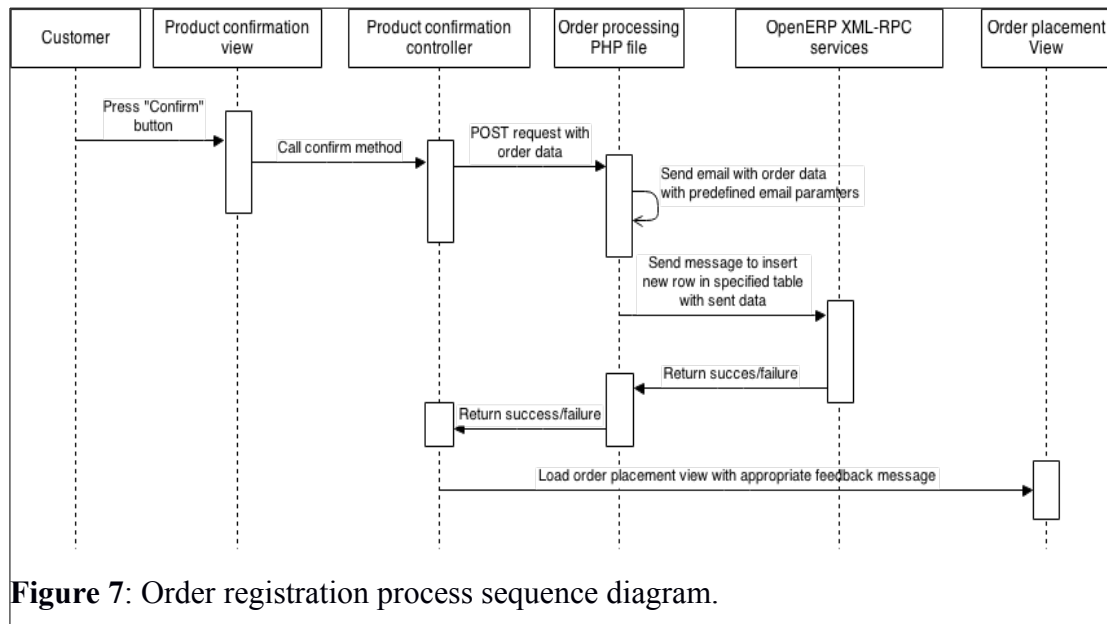
**Figure 5:** Product selection process sequence diagram.

Figure 6 presents the product configuration sequence diagram.



**Figure 6:** Product configuration process sequence diagram.

Figure 7 presents the order registration sequence. The order registration includes the processing of the order through PHP (email sending and OpenERP data insertion).



**Figure 7:** Order registration process sequence diagram.

### 3.4. Technical Specification

This chapter describes the technical details of the ordering tool development, presenting the general view of the application's architecture, as well as the details of the design and planned implementation.

#### 3.4.1. Application architecture

The general architecture of the ordering tool is divided into three packages, as described below:

- User interface – implemented in AngularJS and following the MVC model and thus divisible into three parts:
  - Model: in AngularJS, the model is stored in a context called “\$scope”, from where it is accessible to both the view and the controller. Each controller automatically creates a “\$scope” object for itself and the

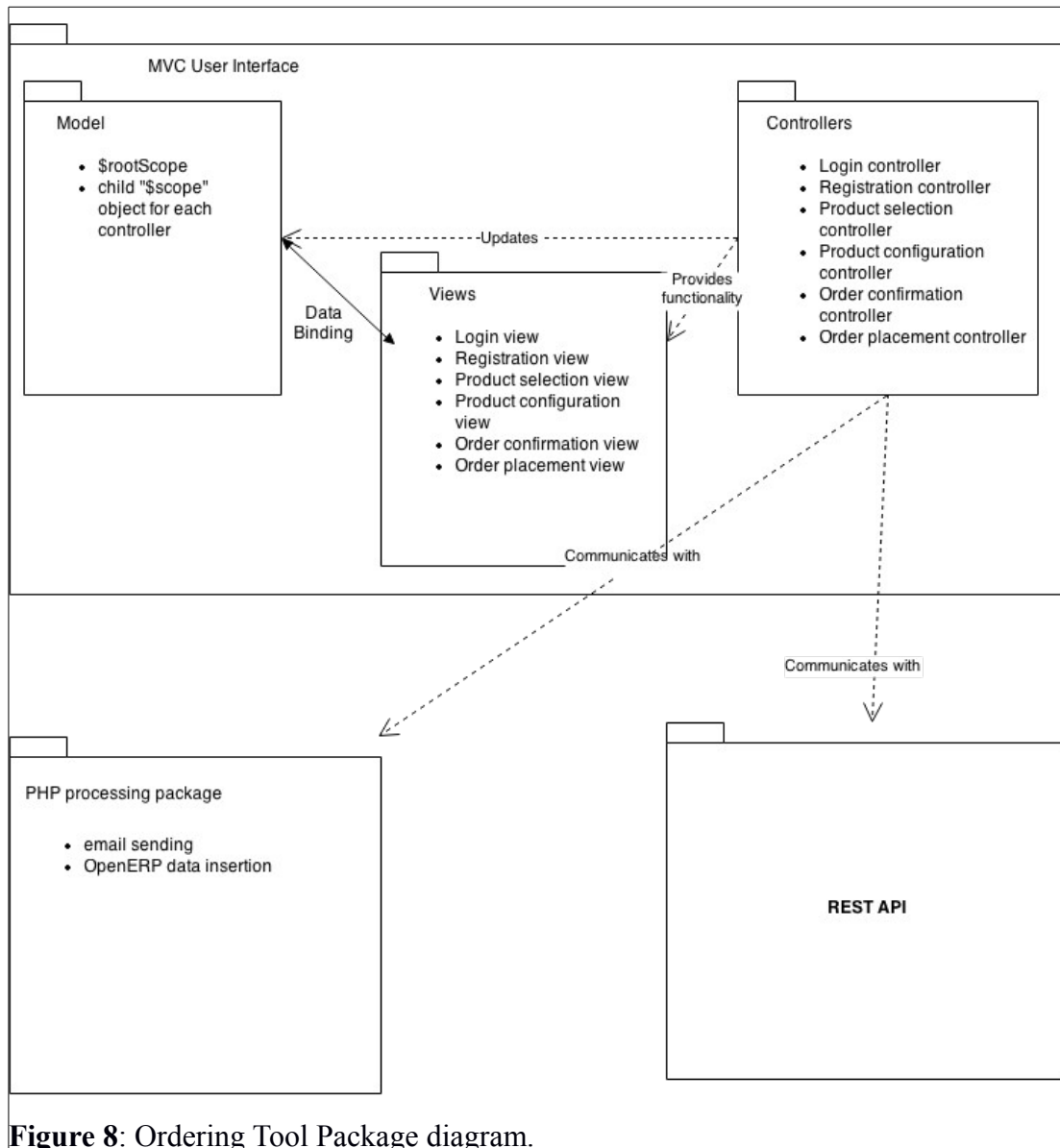
associated view. An object named “\$rootScope” is the parent of all “\$scope” objects and is accessible from anywhere in the application. However, the usage of “\$rootScope” is not recommended.

- View: represented by HTML files also containing AngularJS directives for loops, conditional display of data and data binding
- Controller: updates its associated “\$scope” object, communicates to the REST API and provides the behaviour of the application
- REST API – permits the use of the Summium configuration. The implementation of the REST API is outside the scope of this thesis.
- PHP order processing files – provides a way of registering an order through email sending functionality and ERP data insertion

The controllers handle the communication to both the REST API and the PHP order processing packages.

The package diagram in Figure 8 shows the packages needed in the ordering tool application.





**Figure 8:** Ordering Tool Package diagram.

### 3.4.2. Detailed design

In AngularJS, a controller is defined as a JavaScript constructor function.

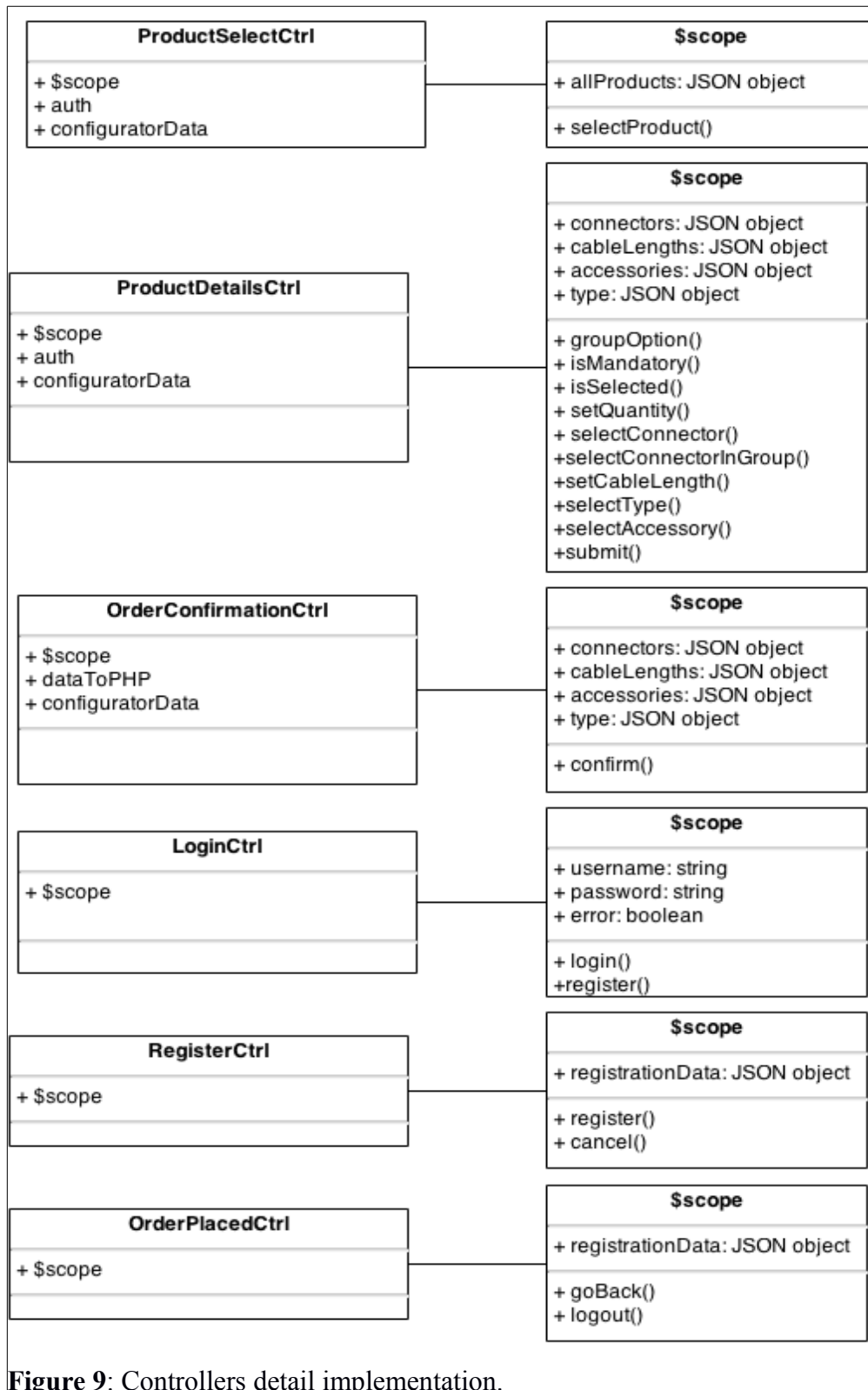
A controller automatically creates a “\$scope” object for handling the model. The controller sets up and modifies the \$scope object. The \$scope object itself can have attached properties(any kind of object) and behaviour(methods). If a property or method is attached to the \$scope object, it is accessible in both the view and the

controller. Therefore, the methods handling user interactions will be attached to the \$scope object.

Furthermore, the REST API specification requires a client to locally store a copy of the previously fetched JSON data and merge it with any newly fetched data from the configurator. For that purpose, each controller dealing with order placement (choosing a product, configuring a product, order confirmation) will contain a variable to handle the configurator JSON data.

In addition, a variable to handle user credentials is necessary in all controllers that manage users or configure the order using the Summium REST API.

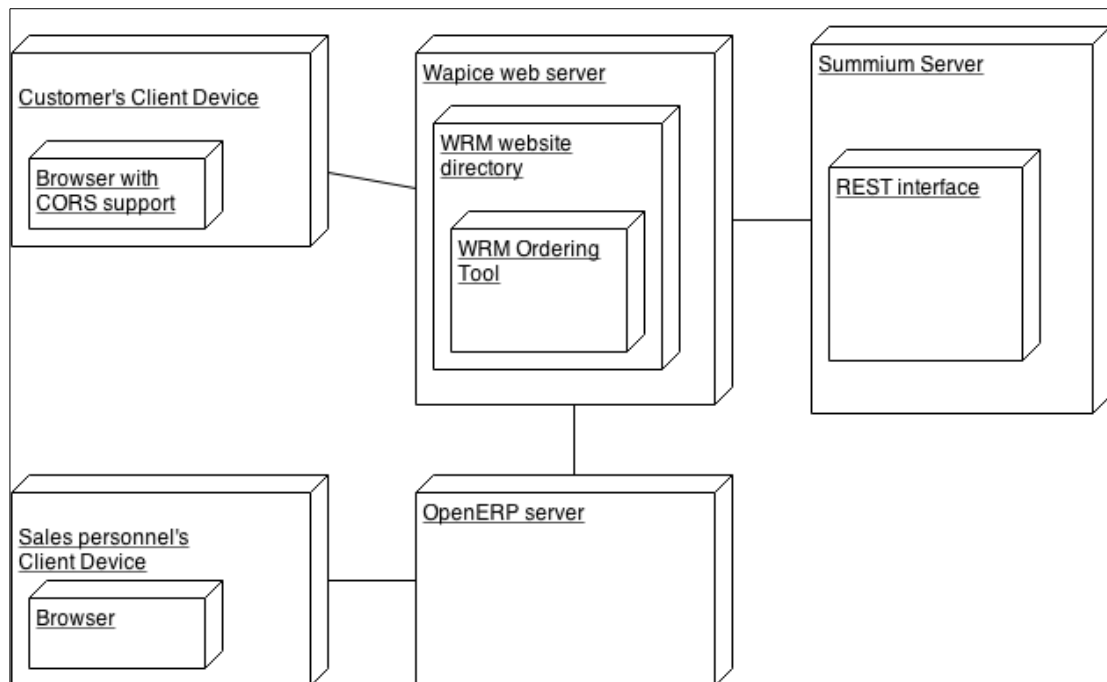
Figure 9 details the methods and properties of each controller, as well as the methods and properties attached to the \$scope object associated with the controller.



**Figure 9:** Controllers detail implementation.

### 3.5. Deployment Diagram

The application will be deployed as a part of the WRM product's website. The REST API will be available on a Summium server, in a different domain. The situation is described by the deployment diagram in Figure 10.



**Figure 10:** Deployment diagram.

### 3.6. Testing Specification

Several aspects of the user interface will be tested thoroughly to ensure the quality of the implementation. The tests will be performed regularly during the implementation phase, as well as in the final stage of the development.

The following test cases will be used:

#### 3.6.1. Invalid login credentials

In order to ensure that only authorized users have access to placing an order, the application will be tested with the following combinations:

- Invalid username and/or password
- Missing username and/or password

### **3.6.2. Invalid data format in registration**

Certain input fields in the registration form require a specific data format. Tests will be run so that the application does not proceed with the user registration in the following situations:

- Data is missing
- Data added to email field is not a valid email
- Data added to phone field is not a valid phone number format
- Data added to the two password fields is not the same

### **3.6.3. Non-numerical values in quantity and cable length fields**

On the product configuration page, the quantity and cable length fields should only accept numerical values.

### **3.6.4. Changes to Summium model**

As long as the data retrieved from the Summium API follows the agreed format, the application should work within the normal parameters after the following changes are made to the Summium model, considering the JSON data structure remains as previously defined in the REST API specification:

- Adding a parameter
- Removing a non-essential parameter
- Changes to the values in the JSON object, as long as the keys remain as previously defined

### **3.6.5. XML-RPC connection broken**

Upon breaking the XML-RPC connection to OpenERP, the data cannot not be inserted to the database. The application should be able to handle this situation.

### **3.6.6. Connection to REST API broken**

Upon breaking the connection to the REST API, the user management and order configuration cannot be handled. The application should be able to register the situation and inform the user accordingly.

## 4. IMPLEMENTATION AND DEPLOYMENT

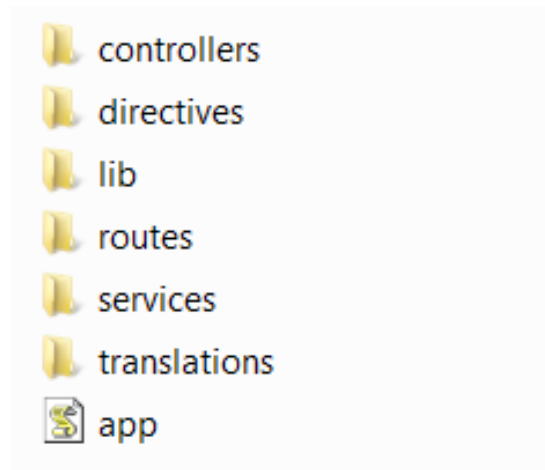
The implementation of the user interface uses the AngularJS JavaScript framework, while the order registration is implemented in PHP.

### 4.1. AngularJS Application Structure

Due to using AngularJS for implementing the user interface, a specific directory structure was created, consistent with the requirements of the framework and providing the base structure of the application.

AngularJS requires only one HTML file to be downloaded to the user's browser. This file is commonly named "index.html". Naming the file "index.html" also ensures it will be the first file to be run in the application. The "index.html" file should be placed in the root directory of the application.

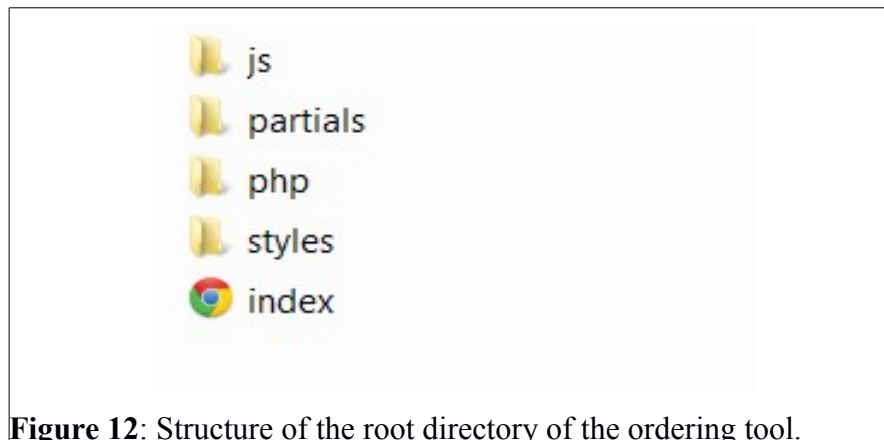
Since the user interface is JavaScript-based, a specific directory was created to hold the necessary JavaScript files, also located in the root directory of the application. The directory has been denoted "js". The "js" directory contains the "app.js" file, which initializes the application, as well as directories holding various types of JavaScript files, such as controllers or libraries. The contents of the "js" directory in this application can be seen in Figure 11.



**Figure 11:** Structure of the "js" directory.

The “app.js” file declares the paths used to navigate through the user interface, as well as link each path to a view and a controller.

Located in the root directory as well, a directory named “partials” holds HTML files corresponding to each view of the application, while a directory named “styles” holds all data referring to the look and design of the user interface. Figure 12 presents the content of the root directory of the AngularJS ordering tool application.



**Figure 12:** Structure of the root directory of the ordering tool.

In the case of this application, a directory named “php” has also been added to the root folder of the application, to hold the files necessary for order registration.

#### **4.2. GUI Design and Implementation**

In the early stages of the implementation, the styling applied to the user interface is minimal and very little CSS declarations have been used. However, as the implementation continues, the look of the webpages will be improved and refined to match the design of the WRM products.

The user interface follows the elements described in the mock-ups presented in the system analysis.



#### 4.2.1. AngularJS implementation – used concepts and syntax

According to AngularJS requirements, the “index.html” file serves to link the external CSS and JavaScript files, as well as provide a container for the views presented to the user. Each view represents one page presented to the user and is described in the respective HTML file located in the “partials” directory. The views are loaded inside an HTML tag containing the directive “ng-view” as an attribute.

The “index.html” file contains the “ng-app” directive as well, to specify that the ordering tool user interface is an AngularJS application. For this purpose, the “ng-app” directive is attached to the “html” tag, as in Figure 13.

```
<html ng-app="WRMOrderApp">
```

**Figure 13:** Using the "ng-app" directive.

Each view receives data from the model through the use of a controller. The controller is also responsible for sending and receiving data from the REST API, and updating the model accordingly.

Views are mapped to an URL in the “app.js” file, using AngularJS's routing functionality. Figure 14 shows an example of a mapping: when the URL is “applicationURL/login”, the view in use is “login.html” and the controller associated with it is “LoginCtrl”.

```
$routeProvider.  
  when('/login', {  
    templateUrl: 'partials/login.html',  
    controller: 'LoginCtrl'  
  }).
```

**Figure 14:** Example of AngularJS routing.

In the views, AngularJS directives have been used for creating loops and conditional display of data, extending the HTML functionality.

The example in Figure 15 shows the implementation of the product selection view.

```
<div>
  <h2>Select your product</h2>
  <div class="product" ng-repeat="productValue in allProducts">
    <a ng-click="selectProduct(productValue.id)" >
      
      <p>{{productValue.name}}</p>
    </a>
  </div>
  <div class="error" ng-if="communicationBroken">
    A communication error has occurred. Please try again later.
  </div>
</div>
```

**Figure 15:** Example of a view implementation in AngularJS.

An AngularJS directive is inserted as an attribute into an HTML element. Each directive provides a certain functionality. In the example above, the directive “ng-repeat” has the functionality of a “foreach” loop: in each iteration, the current element in the “allProducts” array is denoted by “productValue” and a link HTML element is created containing the image and name associated with the element. In addition, the error text will appear only if the “communicationBroken” variable is set to true in the controller.

In AngularJS, an element of the view can also be mapped directly to an element of the model, which simplifies the passing of user events from the view to the controller. For this purpose, the “ng-model” directive is used.

In addition, the ordering tool user interface implementation makes extensive use of the following directives:

- ng-click (assigns an on-click listener)
- ng-change (assigns an on-change listener)

- ng-show (shows or hides an element of the view based on a condition)
- ng-if (includes an element of the view conditionally)
- ng-checked (selects a checkbox conditionally)
- ng-disabled (disables a view element conditionally)

In AngularJS, controllers belong to a module. A module gathers related functionality in an application. Larger application declare multiple modules, but a small-scale application such as the ordering tool UI may only use one. Figure 16 illustrates the declaration of a module:

```
var WRMOrderControllers = angular.module('WRMOrderControllers', []);
```

**Figure 16:** Declaration of an AngularJS module.

In addition, figure 17 serves as an example of the syntax to be used in implementing a controller.

```
WRMOrderControllers.controller('NameOfCtrl', ['$dependency1', '$dependency2', function ($dependency1, $dependency2) {
    $scope.propertyName = value;

    $scope.methodName = function(parameters) {
        //to do
    }
}]);
```

**Figure 17:** Syntax of an AngularJS controller.

A controller implementation provides the following:

- The name by which the controller will be identified in the application
- Dependencies (for example, AngularJS services, such as \$http)
- Properties and methods associated to the \$scope object and accessible from the view

- Methods and properties not linked to the \$scope object

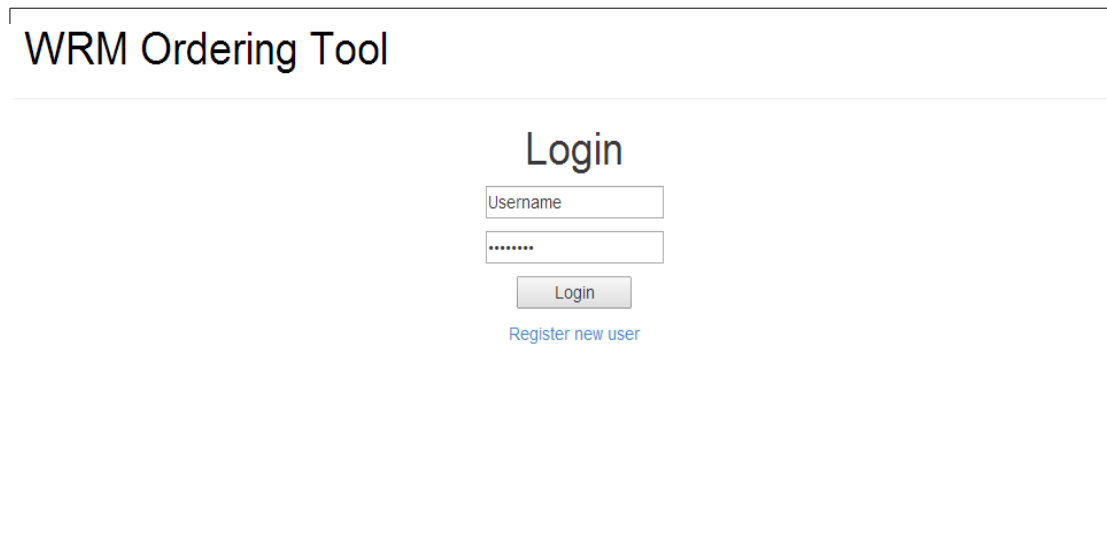
#### 4.2.2. Login View

The login view is the first view presented to the user, upon starting the ordering process. It contains fields for username and password, a button for submitting the credentials, as well as a link to the registration page.

In case of the user submitting incorrect credentials, the login view displays an error message.

In case of the user submitting correct credentials, the product selection page will be loaded.

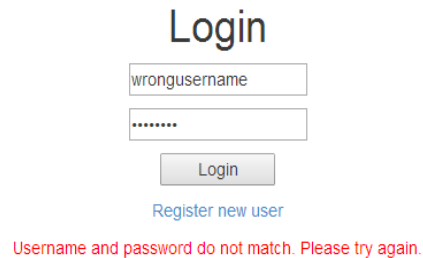
A screenshot of the login view can be seen in Figure 18.



**Figure 18:** Screenshot of the login view.

A screenshot of the login view after wrong credentials have been used can be seen in Figure 19.

## WRM Ordering Tool



The screenshot shows a login form titled "Login". It contains two input fields: the first contains the text "wrongusername" and the second contains a series of dots representing a password. Below the fields is a "Login" button. Underneath the button is a blue hyperlink that says "Register new user". At the bottom of the form, there is a red error message: "Username and password do not match. Please try again."

**Figure 19:** Login view with wrong credentials error.

### 4.2.3 Registration View

The registration view is loaded when a user follows the “Register” hyperlink in the login view. The registration view consists of a form for user data and a button to submit the registration.

The form fields require valuable information for the ordering process:

- Username (a valid email address)
- Password (having two fields for the password minimizes the risk of user misspellings and errors)
- Personal data of the user (name and personal contact information)
- Shipping address (company name and address)
- Invoicing address (can be the same as the shipping address)

Upon clicking the “Register” button, the user information will be send to the REST API, in order for a new user to be created in Summium, if the information is confirmed as valid by an admin.

A screenshot of the registration view can be seen in Figure 20.

The screenshot shows a web form titled "WRM Ordering Tool" with a sub-header "Register to order". The form is organized into three main sections: "Customer information", "Shipping address", and "Invoicing address".

**Customer information**

- User Name
- Password
- Confirm Password
- Name
- Phone
- Fax
- Email

**Shipping address**

- Company
- Address 1
- Address 2
- State
- City
- Zip

**Invoicing address**

**Same as shipping address**

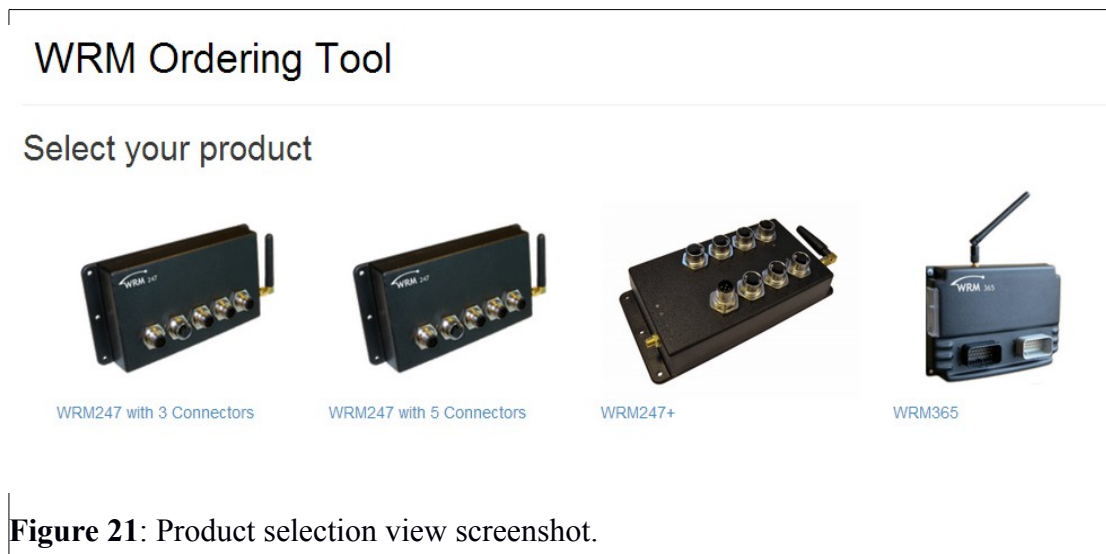
At the bottom of the form are two buttons: "Cancel" and "Submit".

**Figure 20:** Registration view screenshot.

#### 4.2.4. Product Selection View

The product selection view allows the customer to choose between the four available WRM products, each of them being represented by an image and a name. Clicking each product will load the product configuration view corresponding to the product.

A screenshot of the product selection view can be seen in Figure 21.



**Figure 21:** Product selection view screenshot.

The product selection view is built based on the data retrieved from the Summium REST API through a POST request. The POST request creates a new configurator instance to be used for customizing the order and returns detailed data for creating the user interface. Upon selection of a product, a PUT request will be sent to the REST API, signalling the selection and retrieving the data concerning the necessary changes in the interface.

#### 4.2.5. Product Configuration View

When loading the product configuration view, the user interface will display all the necessary fields for customizing the previously selected product, as defined by the JSON data retrieved from the REST API.

The product configuration view will always contain a field for the quantity of the ordered product, as well as choices for the accessories to be shipped with the product, if any fit the current selection. Furthermore, the product configuration view will offer the option of choosing the type of WRM365 products and the cables (cable length and end connectors) for the WRM247 series.

The number of cables and the possible end connectors are defined in the JSON data received from the REST API. Mandatory and mutually exclusive end connectors may exist. Default values for the customization fields can also be found from the JSON data.

Any user event in the product configuration view triggers a PUT request to the REST API, which responds with the changes to be made to the model in JSON format. The response is merged with the previously stored JSON data and the user interface is updated automatically.

The product configuration view is designed to ease the customization process, through the use of text fields, checkboxes and radio boxes. For a cleaner and more attractive user interface, an accordion is used for displaying the customizable cables for each connector. This is particularly useful in case of products with a large number of connectors, reducing the size of the page displayed in the browser.

A set of default values is received from the initial state of the configuration and displayed upon page load. Mandatory end connector values (for example, the power input) are displayed as disabled checked checkboxes.

The images below represent the product configuration view for each of the available products. The name of the product can be seen at top of the page.

In Figure 22, the configuration of a WRM247 with 3 connectors is presented.



# WRM Ordering Tool

## Select product details

### WRM247 with 3 Connectors

Quantity

## Select connectors

Connector1

Connector2

Connector3

1-wire

Analog IN 2

Current loop 1(20mA)

Digital IN 3,4

Length of cable

## Accessories

DC Power

GSM Antenna

**Figure 22:** Configuring a WRM247 with 3 connectors product.

In Figure 23, the configuration of a WRM247 with 5 connectors is presented.

# WRM Ordering Tool

## Select product details

### WRM247 with 5 Connectors

Quantity

## Select connectors

Connector1

Connector2

Connector3

Connector4

Connector5

- Current loop 2
- CAN 2
- Digital OUT 2,3,4
- RS-232

Length of cable

## Accessories

- DC Power
- Combo Antenna

**Figure 23:** Configuring a WRM247 with 5 connectors product

Figure 24 presents the configuration page for a WRM247+ (8 connectors available).

## WRM Ordering Tool

---

### Select product details

#### WRM247+

Quantity

### Select connectors

Connector1

Connector2

Connector3

USB

Digital OUT 2-4

Length of cable

Connector4

Connector5

Connector6

Connector7

Connector8

### Accessories

DC Power

**Figure 24:** Configuring a WRM247+ product.

Figure 25 presents the product configuration view for a WRM365 product. The type option appears instead of the customizable cables for each connector.

The screenshot shows a web form titled "WRM Ordering Tool". Below the title is a section "Select product details" with the product name "WRM365" and a "Quantity" input field containing the number "1". The next section is "Select type", featuring two radio button options: "Serial port" (which is selected) and "ADC". Below that is an "Accessories" section with a checked checkbox for "DC Power". At the bottom of the form is a "Submit" button.

**Figure 25:** Configuring a WRM365 product.

Once the customization is complete, the customer can choose to proceed by using the “Submit” button. This action will load the order confirmation view.

#### **4.2.6. Order Confirmation View**

The order confirmation view allows the user to review the customized order and confirm it with the press of a button present on the page. The complete details of the order will be displayed on the page.

Upon confirmation, the customer will be redirected to another view displaying a feedback message, while the application will process the order by sending an email to interested personnel and inserting the data into OpenERP.

An example of the contents of the order confirmation view is displayed in Figure 26.

## WRM Ordering Tool


### Confirm your order

Thank you for your order

Product name	WRM365
Quantity	1
Type	Serial port

### Accessories

Accessories	DC Power
-------------	----------



**Figure 26:** Order confirmation view example.

#### 4.2.7. Order Placement View

The order placement view is loaded upon confirmation of an order by a customer and informs that the order has been placed and is being processed. The order placement view contains a way for the customer to return to the start of the ordering process.

An example of the order placement view can be seen in Figure 27. The example presents a situation where the order registration (email and OpenERP data insertion) has been successful.

## WRM Ordering Tool

Your order has been placed and will be processed shortly.

**Figure 27:** Order placement view example.

### **4.3. Communicating with the REST Services**

The user interface communicates with the REST API by sending POST and PUT HTTP requests.

In the application, the communication is handled by the controllers, though AngularJS's \$http service. The \$http service requires the method of the HTTP request, as well as the URL of the server being used.

When connecting to the REST API, basic HTTP authentication is used. Each request contains the username and password in its header.

Data is passed and received in JSON format. Received data is merged with a local copy of previously received data to define the contents of the application model.

### **4.4. Registering the Order**

After the order has been confirmed, it must be registered within the company and to OpenERP. Order registration is a two-step process: an email is sent to the concerned personnel and the order data is inserted to OpenERP for further use.

The order registering process is triggered by a call from within the user interface to the PHP file that handles the process. Similarly to the other parts of the application, the call uses AngularJS's \$http service and sends a POST request, passing and receiving JSON data. The response received specifies whether the order registration was completed correctly and based on the result, the customer will receive feedback on the order processing page.

#### **4.4.1. Registering the order through an email**

As the first step in the order registration sequence, data representing the fully-configured order is gathered from the application. The data is inserted in the body of an email in human-readable format.

The email is sent using the mail function of PHP. The sender's email address and the email title are chosen to reflect the origin: the WRM ordering tool. The addressees are Wapice personnel involved in the sales of WRM products.

#### **4.4.2. Inserting Data into OpenERP**

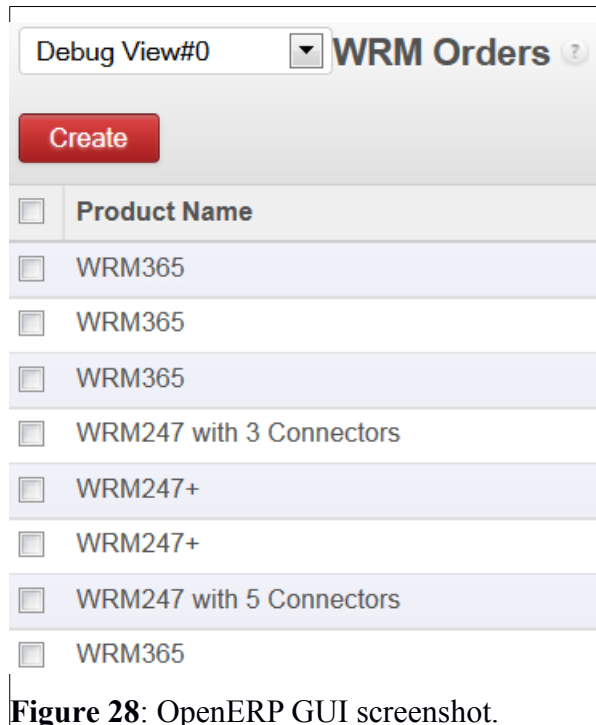
The final step of the order registration is inserting the data into OpenERP, in a table specifically created for storing WRM order data.

For this step, OpenERP must be installed on a server at a known location. A valid username and password combination to the OpenERP installation must be available for the application to use.

Inserting the data into OpenERP is done using the “XML-RPC for PHP” library. Through the methods available in that library, a “create” request is sent to OpenERP. The database and table name, username and password, as well as the data to be inserted are passed as parameters. The connection is made to the URL “http://openERPservername/xmlrpc/object”.

As a result, data is stored into the designated OpenERP table and will be available through the graphical user interface.

In Figure 28, the designated table for WRM orders can be seen in the OpenERP graphical user interface, containing a small number of orders created during application implementation and testing.



**Figure 28:** OpenERP GUI screenshot.

#### 4.5. Application Deployment

The ordering tool will be deployed as a part of the WRM product website. The WRM website is built on top of the Joomla! CMS. A menu item will guide a potential customer to the ordering tool's page, where they can start the ordering process by logging in or registering.

The REST API is available from a Summium server.



## 5. TESTS, RESULTS AND ANALYSIS

Testing is a major part of the software development process, to ensure the correct look and functionality of an application.

Furthermore, the developed process has provided, in addition to a product necessary to the company, useful information about a number of technologies to be used in this type of application. The following chapter details the findings.

### 5.1. Application Testing

The application has been extensively tested during all the stages of the development, according to the testing specification. Tests have been performed for each functional module of ordering tool, as well as the application as a whole.

Below the test results for each test case defined in the specification can be found.

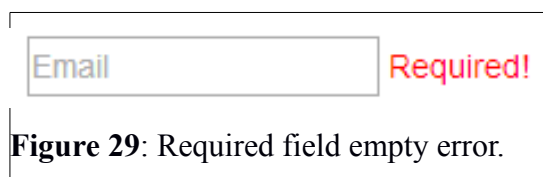
#### 5.1.1. Invalid Login Credentials

It has been found that in case of inputting invalid login credentials (missing data or wrong username and password combination), the application will not proceed to the product selection view. Instead, the application will display an error message, informing the user the used credentials are not valid.

#### 5.1.2. Invalid Data Format in Registration

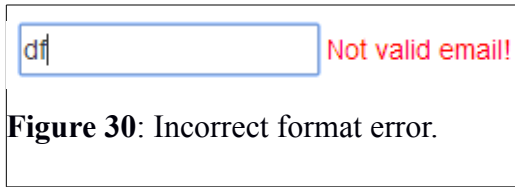
Form validation is implemented using AngularJS. Data will not be sent to the REST API for user creation unless the data format is correct.

If data is missing in a required field, an error message will appear beside it, as illustrated by Figure 29.



**Figure 29:** Required field empty error.

If data inserted into a form field is not in the correct format, a message will appear beside it, as illustrated by Figure 30.



### 5.1.3. Non-numerical values in quantity and cable length fields

If a non-numerical value is inserted into the quantity or cable length field on the product configuration page, an error message will appear beside it, as illustrated by Figure 31.

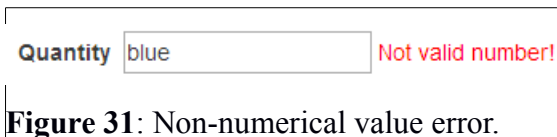
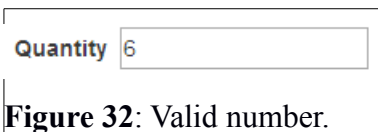


Figure 32 illustrates the situation where a valid number is added to the form field.

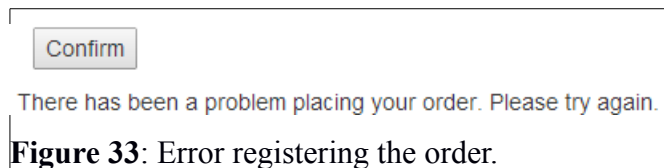


### 5.1.4. Changes to Summium model

The application functions normally if the Summium model is modified, as long as the essential data (for example, product selection options) continues to exist and the JSON data retrieved from the REST API keeps its predefined structure.

### 5.1.5. XML-RPC connection broken

In the early stages of the development, if the XML-RPC connection is broken, or in the case of any other error while registering the order through PHP, the application will continue displaying the order confirmation page, while a feedback message is displayed for the user. The feedback informs that the order could not be placed and advises the user to try again in a short time. Figure 33 illustrates the situation.

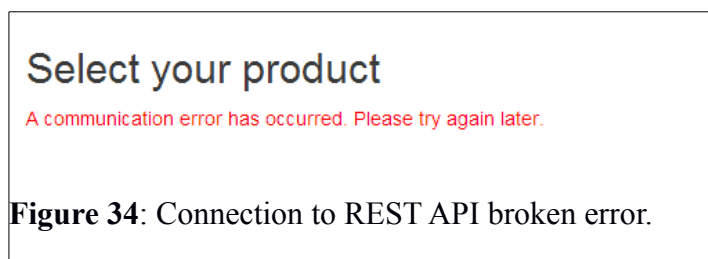


**Figure 33:** Error registering the order.

A useful improvement to this situation would be to develop a way through which the order data will be saved for a specific amount of time during which the application will automatically attempt OpenERP data insertion. In this case, a temporary break in connection would be solve automatically.

### 5.1.6. Connection to REST API broken

If the connection to the REST API is broken, the user will be informed of the situation and advised to try again later, as illustrated by Figure 34.



**Figure 34:** Connection to REST API broken error.

## **5.2. Implementation Results and Analysis**

The implementation resulted in the creation of a Summium-based WRM ordering tool, as well as a configurator implementation using the newly developed Summium REST API.

The initial version of the user interface was developed using the jQuery library and was later changed to use the AngularJS framework.

While the jQuery library provided all the tools necessary for the correct display of data and communication to the REST API, AngularJS was found to be a cleaner, more appropriate implementation method. The change resulted in shorter, more compact and more maintainable code.

With the development of the ordering tool, the ordering process is simplified for the customer and reduced to simply making a number of selections, while the options are presented in a straightforward manner. Previously, a customer would need to study the specification of each product and collect the desired options to email to the WRM sales personnel within Wapice.

In addition, WRM sales personnel is now provided with an improved way of managing the orders, through the use of the OpenERP software.

### **5.2.1. Implementation limitations**

Due to the fact that the WRM templates have yet to be fully defined, the implementation of the ordering tool was limited. The possibility to choose between predefined WRM templates will be added as a product configuration feature in the future.

Furthermore, due to limitations in the current version of the REST API, the user management has yet to be implemented fully implemented.

### **5.2.2. Future work and usage**

In addition to an ordering tool for WRM products, the application provides a base for developing Summium-based ordering tools using the REST API.

For that purpose, it is important that the correct functionality of the ordering tool depends as little as possible on the Summium model. In other words, the application should work if the product family is changed, with minimum to no changes to the implementation.

## **5. CONCLUSIONS**

The creation of the WRM ordering tool provided benefits to the development of both Wapice's Summium and WRM products.

The WRM product gained from having available an intuitive and easy to use ordering tool to be used by its customers. The ordering tool allows for the quick placement of complete and correct orders for the WRM products. As the product grows, more configurable features can easily be added to the ordered products. Furthermore, the order registration process ensures that the concerned personnel is aware that an order has been placed and that the order data is stored safely and can be easily managed through OpenERP.

The main benefit for the Summium product was the implementation of a configurator using the newly developed REST API. The ordering tool serves as an example of such an application and verifies that the REST API functions correctly.

The WRM ordering tool has the potential to grow into a more complex application, as the WRM product is being developed and new features will be available for ordering.

Features such as managing user content, tracking order status and displaying product availability can be added to improve the user's experience.

## 6. REFERENCES

/1/ AngularJS by Google. Last accessed 18.05.2014

<https://angularjs.org/>

/2/ Enable CORS. Last accessed 02.04.2014

<http://enable-cors.org/>

/3/ Introducing JSON. Accessed 21.02.2014

<http://www.json.org/>

/4/ Introduction to Object-Oriented JavaScript. Accessed 26.02.2014

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction\\_to\\_Object-Oriented\\_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript)

/5/ JavaScript: Advantages and disadvantages. Accessed 26.02.2014

<http://www.jsripters.com/javascript-advantages-and-disadvantages/>

/6/ JavaScript Overview. Accessed 26.02.2014

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/JavaScript\\_Overview](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/JavaScript_Overview)

/7/ jQuery website. Last accessed 26.04.2014

<http://jquery.com/>

/8/ OpenERP Documentation v7.0. Last accessed 17.05.2014

<https://doc.openerp.com/> (12)

/9/ Software development Resources. Accessed 28.02.2014

<http://docforge.com/> (11)

/10/ The JavaScript Object Notation (JSON) data interchange format. Accessed 21.02.2014

<http://tools.ietf.org/html/rfc7159>

/11/ Using CORS. Last accessed 03.04.2014

<http://www.html5rocks.com/en/tutorials/cors/>

/12/ World Wide Web Consortium website. Last accessed 24.02.2014

<http://www.w3.org/>

/13/ XML-RPC for PHP. Last accessed 19.05.2014

<http://gggeek.github.io/phpxmlrpc/>