

Lauri Seppälä

# Developing an Auction Module for Online Gaming Service

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

29 November 2017

Author Title	Lauri Seppälä Developing an Auction Module for Online Gaming Service
Number of Pages Date	32 pages 29 November 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor	Markku Karhu, Principal Lecturer
<p>The goal of this final year project was to develop an auction module for online gaming service as well as a management system for this and other future models. It was also decided that the development should follow a specific set of practices so that it would be easier to extend the system in the future.</p> <p>A large number of different web technologies and tools were used as part of the process which are all described in this thesis. For front end development, common technologies like HTML, CSS and JavaScript were used. In addition, JavaScript's jQuery library and Twitter's Bootstrap framework were utilized. Bootstrap, for example, enables bringing together all the other front end techniques and provides many components for a faster development process. In the back end, the main tool was a PHP framework called Laravel which utilizes a development architecture style known as MVC. Also, MySQL databases were an essential part of the project. The main ideas of all these technologies are explained in detail where necessary.</p> <p>As a result of this project, a content management system was created and an auction module was implemented as a part of this system. This system has been built on a profound foundation to support multiple web sites and future implementations. The auction module works as a fully functional prototype but will still be thoroughly tested in the future before its final release. The management system on the other hand is already in use and it will be utilized with all the future modules. Laravel web application framework was very useful in this, as it turned out to be highly efficient and provided a steady environment for the development process.</p>	
Keywords	HTML, CSS, JavaScript, PHP, Laravel, web development

Tekijä Otsikko	Lauri Seppälä Huutokauppamoduuli verkkopelipalvelulle
Sivumäärä Aika	32 sivua 29.11.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotuotanto
Ohjaaja	Yliopettaja Markku Karhu
<p>Insinööriytyön tarkoituksena oli kehittää huutokauppamoduuli verkkopelipalvelulle. Lisäksi kehitettiin hallintajärjestelmä tälle ja muille jatkossa luotaville moduuleille. Alussa päätettiin myös, että kehityksessä seurattaisiin sovittuja toimintatapoja helpomman jatkokehityksen mahdollistamiseksi.</p> <p>Moduulin kehittämisessä käytettiin suuri määrä erilaisia web-kehityksen teknologiota ja työkaluja. Näkyvällä puolella käytettiin yleisiä teknologiota, kuten HTML-esityskieltä, CSS-tyylikieltä ja JavaScript-skriptikieltä. Näiden lisäksi käytettiin avuksi vielä JavaScriptin jQuery-kirjastoa ja Twitterin Bootstrap-ohjelmistokehystä. Bootstrap esimerkiksi mahdollistaa muiden näkyvän puolten teknologioiden yhdistämisen yhdeksi kokonaisuudeksi ja tarjoaa samalla useita kehitystä nopeuttavia komponentteja. Palvelinpuolella taas käytettiin pääasiassa PHP-ohjelmointikieleen perustuvaa Laravel-ohjelmistokehystä, joka hyödyntää kehityksessä MVC-arkkitehtuuria. Oleellisena osana työtä olivat myös MySQL-tietokannat.</p> <p>Työn tuloksena saatiin aikaiseksi sisällönhallintajärjestelmä ja huutokauppamoduuli osana tätä järjestelmää. Koko järjestelmä on rakennettu vakaalle pohjalle, jotta se toimisi yhdessä useiden eri web-sivustojen kanssa ja tukisi jatkokehitystä. Huutokauppamoduulista saatiin tehtyä toimiva prototyyppi, jota testataan kuitenkin vielä tulevaisuudessa ennen sen lopullista julkaisua. Moduulienhallintajärjestelmä taas on jo käytössä ja sitä hyödynnetään jatkossa kaikkien moduulien hallinnointiin. Laravel osoittautui kaiken kaikkiaan erittäin tehokkaaksi ohjelmistokehykseksi, ja se tarjosi vakaan kehitysympäristön työn suorittamiseen.</p>	
Avainsanat	HTML, CSS, JavaScript, PHP, Laravel, web-kehitys

## Contents

1	Introduction	1
2	Front end development	3
2.1	Document Object Model	3
2.2	HTML	4
2.3	CSS	6
2.4	JavaScript	6
2.5	Twitter Bootstrap	9
3	Back end development	11
3.1	PHP	12
3.2	Databases	12
3.3	Laravel	14
3.3.1	MVC-architecture	15
3.3.2	Artisan interface	19
4	Project implementation	21
4.1	Project environment setup	22
4.2	GamesFactory	24
4.3	Auction module	28
5	Conclusion	32
	References	33

## Abbreviations

URL	Uniform Resource Locator, reference to a web resource
PHP	PHP: Hypertext Preprocessor, popular open source scripting language
HTML	HyperText Markup Language, language that is used to create web pages in the internet
MVC	Model-View-Controller, architectural software development technique
GUI	Graphical User Interface, visual way of interacting with websites and web applications
DOM	Document Object Model, logical structure of a web document
CSS	Cascading Style Sheets, language to define the presentation of a web document
API	Application Programming Interface, set of tools to access the application and its functions
XML	Extensible Markup Language, language to store and transport data
W3C	World Wide Web Consortium, main international standards organization for the World Wide Web
OOP	Object Oriented Programming, programming paradigm based on objects
SQL	Structured Query Language, standard programming language for relational databases

## 1 Introduction

Lotto24 is an online gaming service where the user can play various lotto type games for free and get currency called pig coins to buy real existing products such as gaming consoles and cosmetics. Additionally, the user has a possibility of upgrading his account from basic to plus type in order to have access to new games and some extra coins. Lotto24 has, during the recent years, spread across Europe and is actively used in the Nordic countries, the United Kingdom and France. The Finnish version of this website can be found in [www.lotto24.fi](http://www.lotto24.fi) and the others countries' versions in a similar fashion by changing the ending of the URL to the corresponding country code.

The aim of this final year project is to design and implement an auction module for Lotto24 as well as the module management system called GamesFactory for this and all the future modules. GamesFactory would then later be used exclusively to handle all kinds of management activities focused on Lotto24 and the related websites. Lotto24 is owned and the project initialized by a Danish company called PlusService ApS. One part of the work consist of exploring the technologies and implementations already used at PlusService and then utilizing those as the backbone for the project itself. PlusService wanted to have this auction module developed for them to try some new innovations on their already proven concept. The same strategy has also been used in developing similar websites to Lotto24 but with new, updated graphics. In future, if the auction module is a success, it will also be used in these new websites. GamesFactory on the other hand will be implemented to use them from the very beginning.

The main technologies used in this project are PHP programming language and a framework built for it called Laravel but various other tools are also used and explained. PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely used open source scripting language that is especially well suited for web development thanks to the possibility of embedding it into HTML (HyperText Markup Language) code. [1.] PHP is being developed in a collaborative public manner giving it a huge advantage compared to some of its competitors as this means it will regularly get updates provided by the active community. Laravel, also based on open source principles, is an extensive but easy to learn framework meant for all-purpose web application development. But unlike PHP, Laravel is being developed independently with the addition of a few select-

ed components that are removed from the official release and now being updated by the community. [2.] It is based on MVC-architecture (Model-View-Controller) which, broadly speaking, divides the application to three more understandable components that all have their own specific purposes.

The second section of this thesis describes the front end development techniques that are used to build the auction module and the related administration service. The third section explains what happens behind the scenes, in the back end, and fourth section shows the implementation of the theory. In the end, there is a conclusion that goes over what has been achieved during the project.

## 2 Front end development

Web development is typically divided into two different categories: front end development and back end development. Front end development, also known as client-side development, is designing and developing websites and web applications using technologies that run on the web platform. The purpose of these sites and applications is to work as interfaces, also known as Graphical User Interfaces (GUI), for the users. The standard front end technologies include HTML, CSS (Cascading Style Sheets) and JavaScript. In addition, DOM (Document Object Model) is also often counted in these technologies as it has become an intrinsic part of web development. Quite often one can hear an example where front end technologies are described in relation to a person where HTML is the skeleton, CSS the skin and JavaScript all the actions or functions that a person can perform. Here DOM could be compared to basic human anatomy. All these techniques are constantly evolving so a good developer should always be following what are the newest standards and additions. The main objectives of front end development are to provide users with interfaces that are easy to use, fast to load and hold relevant content. On top of this the developer should take care that sites are responsive for different screen sizes and resolutions plus compatible with various browsers, operating systems and devices. [3.]

### 2.1 Document Object Model

The starting point for front end development is the Document Object Model which is a tree-like way of describing how a website or web application is structured and how all the different pieces, or objects with their own specific identities, are related to one another. It was originally meant to allow making scripts and programs portable for the web browsers but has since then evolved to become an API (Application Programming Interface) for the HTML and XML (Extensible Markup Language), meaning that it gives tools to access and manipulate these types of documents. HTML and XML are similar ways of describing data in documents and with DOM it is possible to modify their structure and content, or even create new documents from the scratch. DOM has been standardized by the W3C (World Wide Web Consortium) to accept all the different programming languages. Thanks to this, it is possible to use DOM in various set of environments and applications. [4.] An illustration of DOM tree is shown in figure 1.



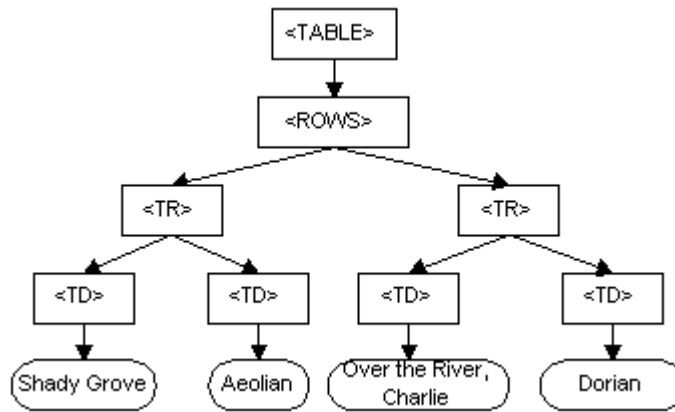


Figure 1. Document Object Model [4]

Figure 1 illustrates a typical document object model representation of an example table. The table has a tree-like structure where all the following elements are branching out to the very deepest levels of the data.

For a web developer, Document Object Model gives the opportunity to view changes instantly on the browser making testing easier and speeding up the development process. The work of transforming an HTML document to document object representation is made automatically by the browser. Inspecting this model can give a better understanding of the underlying structure and functions of the web page. Interactions with the document object on the browser are handled with JavaScript. [5.]

## 2.2 HTML

When accessing a website, the client (browser) receives an HTML file from the web server (host computer). Browsers can be accessed from multiple different category devices such as computers, tablets and mobile phones which all might have their own customised files on the server. After receiving the file, it is up to the browser to interpret it and create a web page accordingly. There can also be other files that the server is sending such as CSS for styling the page, JavaScript for functionality and content including resources such as images, audio and video. As the web techniques evolve it is important that also the browser manufacturers keep their software up to date to minimize compatibility problems. The current version of HTML is HTML5 which is already supported by all the main browsers. [6,6-8.]

When creating web pages, the developer uses HTML to build structure with elements such as headings, tables and forms. Elements can also include various attributes such as, in case of input elements, data types and usually with every new version of HTML comes new additions. The latest update came with new elements for graphics and media as well as better semantics. [7.] A typical way of starting an HTML document is shown in listing 1.

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.    <head>
4.      <title>This is the Page Title</title>
5.    </head>
6.    <body>
7.      <h1>This is the Main Heading</h1>
8.    </body>
9.  </html>
```

Listing 1. HTML document example

As shown in listing 1, the HTML document starts with a type declaration on row 1. Doc-type html here means specifically HTML5 and for other versions it would be necessary to use different declaration. Starting on row 2 is the HTML element with language attribute set to English. The element continues all the way until the last row and all the other elements have been set inside its opening and closing tags. The HTML element almost always consists of head and body. Some information about the document itself is placed inside the head element and all the visible elements inside the body element.

HTML elements can be grouped together with class attributes or specified with id attributes in order to have easier access to them when using CSS and JavaScript. A group of various elements can all be given the same class attribute for unified selection or, in similar manner, a single heading can be given a personal id for more specific selection. Using these attributes makes the code more readable and, because of this, easier to approach after spending some time away from it. Other good practices in making the code more human friendly include commenting, indenting and smart naming of classes and ids.

## 2.3 CSS

Cascading Style Sheets (CSS) are used to make web pages look more alive by adding styles to any individual or grouped HTML elements. They can be included in three different ways: inline, internally or externally. Inline means simply writing style attributes for the elements inside an HTML document and internal writing CSS rules inside a style element. External, on the other hand, means including an additional CSS files dedicated for the styles. These styles are applied by giving a valid value for any available property that an element or element group has. [8.] For example, the developer can make an element red by changing the value of its color property to red as in listing 2.

```
#red-heading {  
    color: red;  
}
```

Listing 2 CSS color example

Listing 2 illustrates how an example element is selected and its color property is set to red. The element is selected by using its id value in combination with a hashtag. Normal elements can be selected by using only the element selector without hashtag, like “p” for all the paragraphs, and classes with a dot notation in front of the class name.

Not all styles work in all environments, meaning that what looks good on a big screen does not necessarily give as good impression on tablets and mobile phones. Therefore, different rules for styles can be described for different circumstances like desktop computers, mobile phones and printing. It is also a common practice to share a single CSS file with multiple web pages to reuse some things like color palettes and to save time. [6,359.]

## 2.4 JavaScript

HTML and CSS are used to make a web page look good but sometimes that is not enough. This is where JavaScript comes in. It is used to make a website alive by adding interactivity as well as changing content and styles on the fly for less static and more dynamic experience. JavaScript is a wide purpose language and can be used in both front end (client-side) and back end (server-side) development. In client-side, it is

run after the browser is done processing the given HTML and CSS documents and it offers possibilities like manipulating the page with DOM, saving data to variables and launching events. [9.] An example of event is shown in listing 3.

```
<button onclick="clickAlert()">Click here</button>
<script>
    function clickAlert() {
        alert("Button was clicked!");
    }
</script>
```

Listing 3. JavaScript event example

Listing 3 shows an example which, when inserted in HTML code, provides a button with added JavaScript functionality. The button element here triggers an event when clicked and calls for the user defined function named as “clickAlert”. This function is then found inside the script element and it in turn triggers an alert method provided by the core JavaScript. Alert is used to launch messages on top of the current open window and, in this case, it will show a message to the user that the button was clicked. In the example, the scripts are included in the HTML document internally and, just like with CSS, JavaScript can be included also inline and externally.

What makes JavaScript even more useful is the abundant number of external libraries, frameworks and APIs built for it. They are all a bit different but mainly provide the developer with tools to do various tasks without the need of having to build the code from beginning. For example, Application Programming Interfaces give the user functions to access and manipulate external data for the benefit of one’s own application. The earlier mentioned Document Object Model is just one example of a JavaScript API and the others include likes of Google Maps API for custom maps and Canvas for 2D and 3D graphics. Frameworks, on the other hand, make it possible to use application skeletons for faster builds and libraries are generally just collections of ready-made functions. [9.]

An important part of modern JavaScript development is a library called jQuery. jQuery is used by 89% of the top million websites in the world, such as Amazon and Instagram to name a few, making it one of the most popular JavaScript libraries available. [10.] Its main purpose is to make JavaScript programming, which is generally thought to be

quite difficult, easier by simplifying a lot of the code structure. On their official website, jQuery is described in the following way:

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. [11.]

Using vanilla JavaScript often results in having to write multiple lines of code for the simplest of functionalities making it cumbersome to use. In many cases, the programmer must also write various implementations of a single function to cover all the different browsers and their versions. jQuery tackles this problem by offering functions and statements to cover a wide range of possible implementations. [12.] The difference between JavaScript and jQuery has been demonstrated in figures 2 and 3.

```
1  var d = document.getElementsByClassName("goodbye");  
2  var i;  
3  for (i = 0; i < d.length; i++) {  
4      d[i].className = d[i].className + " selected";  
5  }
```

Figure 2. Adding classes with JavaScript [13]

Figure 2 shows how, with JavaScript, it takes 5 lines of code to go through elements with a specific class in order to add a new class for all of them. The same functionality written with jQuery is shown in figure 3.

```
1  $(".goodbye").addClass( "selected" );
```

Figure 3. Adding classes with jQuery [13]

As seen in figure 3, jQuery uses less space and clearly has a structure that is easier to comprehend. For the same task, it uses only one row and as much as 80% less space which demonstrates exactly how powerful it can be. This power comes from saving the time of the programmer and providing him with tools that are both easy to use and make more sense than the original ways of implementation.

## 2.5 Twitter Bootstrap

Twitter Bootstrap, or simply Bootstrap, is a front end web development framework that brings together HTML, CSS and JavaScript. Before its release in 2011, no generally accepted approaches to front end development existed and instead programmers were forced to begin their projects from practically nothing. Today, 6 years and 4 version numbers later, Bootstrap is widely spread and has become one the most popular frameworks in the whole web development business. It includes, for example, templates, various new ready-to-use components, style labels and grids to make beautiful layouts as well as practices on how to make the site friendlier for different browsers and devices. All these tools help new projects for a quick and solid start. [14,1-2.] An example of admin dashboard template from bootstrap website is shown in figure 4.

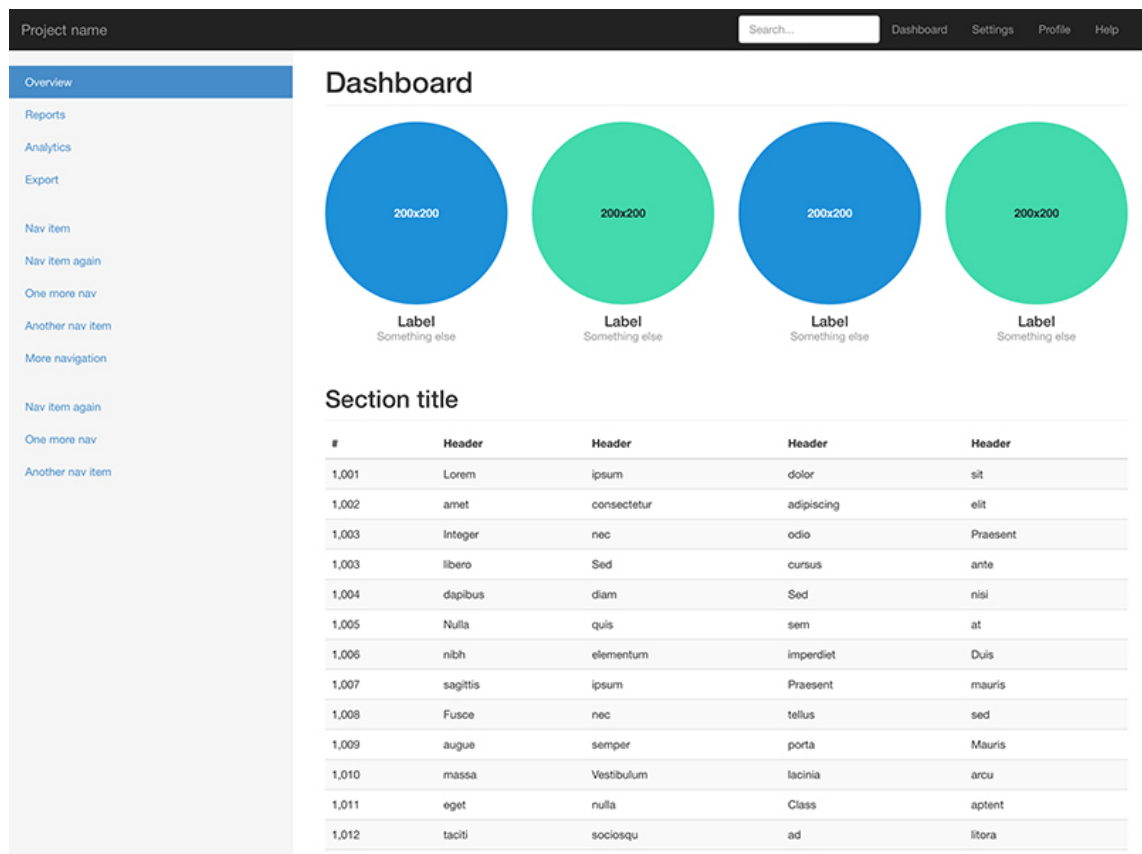


Figure 4. Bootstrap admin dashboard template [15]

The code for the template in figure 4 is possible to get from the official Bootstrap website and it is fully based on their own techniques. For example, navigations and list groups have been used to showcase easy approaches for building common dashboard components. Many of these components in Bootstrap are also interactive which is why

jQuery is included in the standard installation. [16.] An example of Bootstrap's interactive modal component is shown in figure 5.

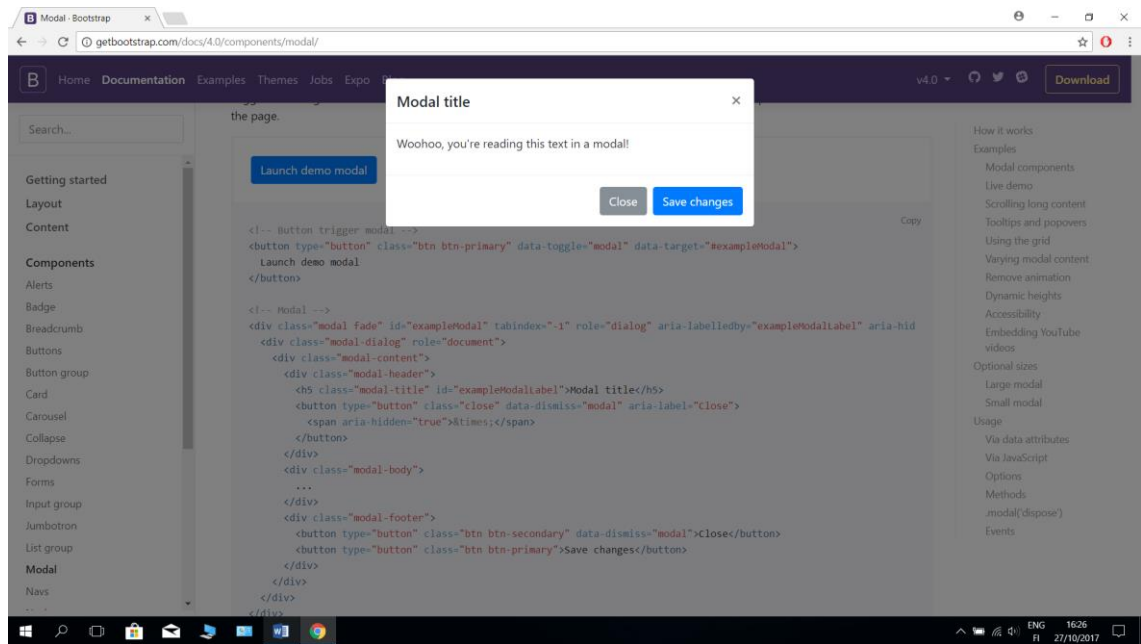


Figure 5. Bootstrap modal example [16]

The example in figure 5 illustrates how it is possible to build pop-up like modals with Bootstrap components. The framework is used by adding Bootstrap classes, such as the modal classes in the example, to basic HTML elements. The style and functionality of these classes are then determined by Bootstrap's own CSS – and JavaScript files.

### 3 Back end development

Back end development, also known as server-side development, is web development behind the scenes. If front end development is about making everything that the user can see or interact with, then back end is exactly the opposite. It is about the underlying technology that makes sure the site is up and running smoothly. [17.] The difference and interaction between front end – and back end development is further explained in figure 6.

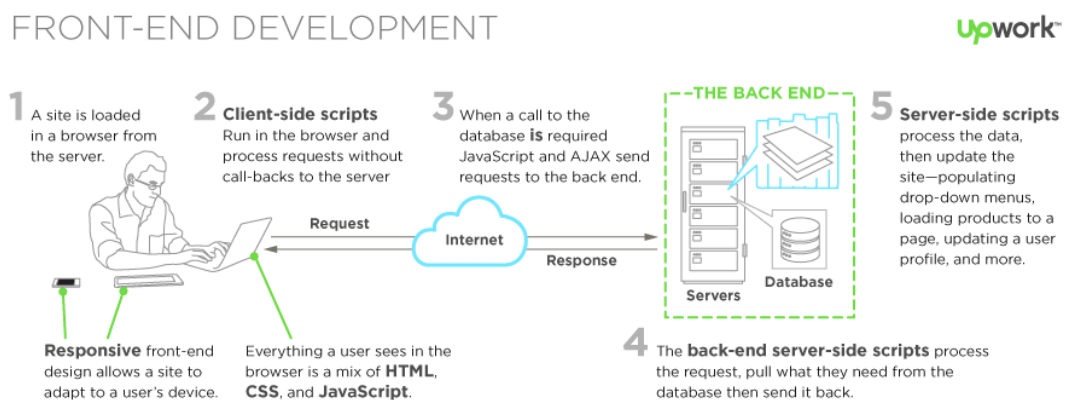


Figure 6. Front end – and back end development [17]

Figure 6 shows how front end – and back end development are connected between points 3 and 4. JavaScript or AJAX are used to send request from front to back end and then response is returned after the necessary processing. This processing in back end can include database operations, API calls or other ways of accessing data. In a nutshell, back end is about providing the necessary resources and computation for the front end. When choosing the programming language for back end development, the programmer has a lot of different options. Some popular languages include PHP, Python, Ruby and JavaScript. Many developers choose JavaScript in order to become full-stack JavaScript developers and have less languages to work with. Other factors making languages popular are the supported frameworks. Frameworks are tools that provide, for example, techniques and reusable parts of code for faster development process. [17.]



### 3.1 PHP

PHP is used in the project as the main back end language mostly because it is very easy to learn the basics and, on the other hand, it has a lot of depth for experienced developers. It was also already used in PlusService or more importantly they were using a PHP framework called Laravel. What makes PHP more suitable for web development than some other server-side languages is the possibility of embedding it inside HTML code allowing the mixing between front end and back end. [1.] An example of using PHP with HTML is shown in listing 4.

```
1.      <?php
2.          if(isset($string)) {
3.              echo "<p>".$string."</p>"
4.          }
5.      ?>
```

Listing 4. Embedded PHP example

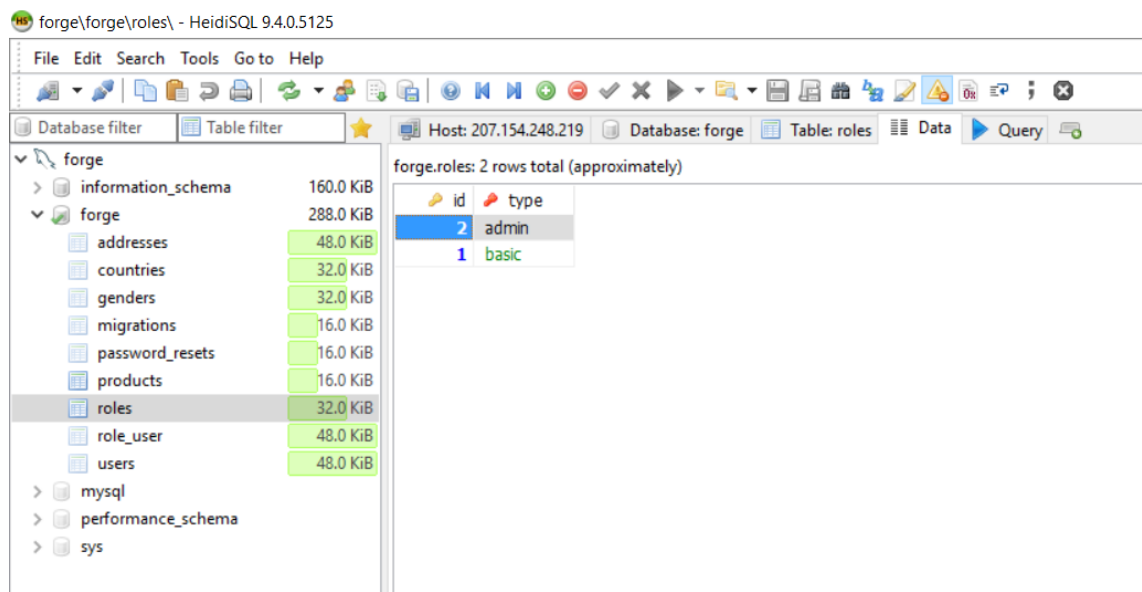
As seen in listing 4, PHP block is started with an opening tag on line 1 and ended with a closing tag on line 5 in similar manner to normal HTML elements. Inside this block it is possible to write PHP code freely. The example listing checks if variable “\$string” exists in which case it is echoed inside a paragraph element.

It is fairly easy to get started with PHP as it is compatible with all of the main operating systems and web servers plus it supports both procedural – and object-oriented programming (OOP). As a server-side language, PHP can also be used to output other files than HTML such as images, PDF and Flash. In addition, it also supports a wide range of various databases and interconnections with other programming languages. [18.]

### 3.2 Databases

Databases are a common practice in web development. They are used to store data or information efficiently and make it possible to access, update or delete them when necessary. Different kinds of databases exist but the most used type is a relational database. These databases consist of tables which resemblance real objects or things

such as users, products and transactions but can also be used for some other purposes. These tables are then divided to columns and rows where columns can generally be thought as the various attributes or data categories such as name, description or creation time. Rows, on the other hand, are then the individual records and specify their own values for each of the columns. [19.] A simple example of a database table is shown in figure 7.



forge\forge\roles\ - HeidiSQL 9.4.0.5125

File Edit Search Tools Go to Help

Host: 207.154.248.219 Database: forge Table: roles Data Query

forge.roles: 2 rows total (approximately)

id	type
2	admin
1	basic

Figure 7. Database table example

Figure 7 illustrates a simple database table named as roles. They can be thought of as the possible user roles that any imaginable website could contain. The table has two columns: “id” and “type”. The former represents a unique identification number that each record has and the latter, as the name says, the type of the role. The table includes two records, or rows, and they both have their individual values.

The concept of relational database also includes primary keys and foreign keys. Primary keys are used as unique indexes for the records such as the “id” column in the example. They enable faster access to data as it is more efficient to go through only one column and look for a specific value when making queries. Foreign keys then link to primary keys from different tables. For example, a user in users table could have a column called “role\_id” which would link to a specific “id” in roles table. This would then specify the role of the user. [20.]

Relational databases have their own standardized programming language called SQL (Structured Query Language) which is supported by most of these databases. The statements in SQL are quite rational and simple, and are divided into four main categories:

- Queries: Used to fetch data with statements like SELECT, FROM, WHERE and ORDER BY.
- Data manipulation language: Used to manipulate data with INSERT, DELETE, UPDATE and few more for controlling transactions.
- Data definition language: Used for table management and includes statements like CREATE, ALTER, TRUNCATE and DROP.
- Data control language: Used for database administrations purposes with, for example, GRANT and REVOKE which are used for permissions. [21.]

For example, all the roles in figure 7 could have been fetched with one simple query shown in listing 5.

```
SELECT * FROM roles;
```

Listing 5. SQL query example

The query in listing 5 shows how to fetch all records from “roles”-table. As can be seen, SQL has very clear structure which is easy to read and understand as all the statements are named after their functionality.

### 3.3 Laravel

Laravel is a PHP framework designed for fast and effective development. It provides many things out of the box but also has a fully adjustable environment. Laravel is suitable for all sizes of projects as it is possible to choose between a light installation or a full one. In addition, the framework offers various other packages to start with. [22.] In their own words, Laravel has been described the following way:

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable, creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching. [22.]

As mentioned in the description, Laravel makes it easier to use many of the common web development practices. The framework is based on MVC-architecture and the three letters represent the main areas of Laravel development: Model, View and Controller. Laravel has dozens of other useful features as well such as the Artisan command-line interface, the Blade templating engine and the earlier mentioned routing feature.

### 3.3.1 MVC-architecture

MVC-architecture is a common development pattern which divides the application into three separate main components: Model, View and Controller. They all have their specific purposes and are interconnected in a way that builds a base for an effective development environment. [23.]

The model represents an object, as in Object Oriented Programming, and is responsible for all the data-related logic. It has access to database to manage the corresponding records and then moves this information between views and controllers. For example, a user model would have access to users-table. [23.] In Laravel the models are based on Eloquent ORM and are called, respectively, Eloquent Models. They are extended versions of normal models and offer additional features to work in Laravel environment. Eloquent models have easy access to database after the connection has been configured. It eliminates the need for SQL queries and statements, and instead uses Eloquent model methods for data management. The initial model tables and their attributes, meaning columns, may be inserted and updated with migration files which are also a special feature provided by Laravel. Migrations can be used to handle all the data definition functionalities from SQL. An example of a migration file is shown in figure 8.

```

1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateUsersTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->increments('id');
18             $table->string('first_name')->nullable();
19             $table->string('last_name')->nullable();
20             $table->date('birthday')->nullable();
21             $table->string('email')->unique();
22             $table->string('phone')->nullable();
23             $table->string('password');
24             $table->rememberToken();
25             $table->timestamp('last_login_at')->nullable();
26             $table->timestamps();
27             $table->softDeletes();
28         });
29     }
30
31     /**
32      * Reverse the migrations.
33      *
34      * @return void
35      */
36     public function down()
37     {
38         Schema::dropIfExists('users');
39     }
40 }

```

Figure 8. Laravel migration file

As seen in figure 8, Laravel migration files have two main methods: “up” and “down”. They are used when creating a table or dropping it. In figure 8, a “users”-table is created with Laravel’s Schema-façade and it is filled with attributes. The primary and foreign keys can also be specified in these migrations and, afterwards, to build relationships between Eloquent models. If no other primary key has been set, Laravel uses “id” as the default value for it. Relationships are used to access linked models with methods for various purposes. [24.]

The second component of MVC-architecture, View, is used to provide logic for the front end. Views include all the possible UI components and they can be mixed with back end functionalities. [23.] In Laravel, views are not normal PHP-files but are instead extended with the Blade templating engine. Blade is, in the end, compiled to plain PHP but before that it offers simpler and more suitable syntax. It is possible to Blade inside the views for things like template inheritance, various components, displaying data and control structures. [25.] These are all explained in the following list:

- Template inheritance: Can be used to extend layouts such as the main navigation bar in order to include it to every page and avoid code repetition.
- Components and slots: Similar to template inheritance but is used in smaller scale to provide reusable blocks such as an alert box to display errors.
- Displaying data: Blade has a simple syntax for displaying data and to avoid writing cumbersome PHP blocks.
- Control structures: As with displaying data, Blade provides a simpler way of using control structures such as conditional statements and loops. [25.]

These four are the main, but not all, benefits of using Blade templates. An example of Blade is shown in listing 6.

```
@foreach($users as $user)
    <p>{{ $user->name }}</p>
@endforeach
```

Listing 6. An example of foreach loop using Blade

Listing 6 illustrates how to loop through a collection of users with Blade and inserting each of their names to their own paragraphs. For comparing, the same code written in PHP is shown in listing 7.

```
<?php
    foreach($users as $user) {
        echo "<p>".$user->name."</p>";
    }
?>
```

Listing 7. An example of foreach loop using PHP

As can be seen from listings 6 and 7, Blade has a much cleaner syntax compared to PHP. By using Blade the programmer can avoid PHP entirely inside views.

Controller, the last part of MVC-architecture, is used as the interface between views and models. It handles the interactions from views and manipulates data with models when needed. [23.] The logic of controllers in Laravel is connected to a system called routing. When the user enters a valid URL, a request is sent to the server. The server is then responsible in handling the request and returning a response. In Laravel, routing files are used to determine the corresponding course of action. For data management, the request may also use different kinds of methods which in turn could alter the response. [26.] The most common methods in Laravel are explained in the following list:

- GET: The default method which is used to fetch data from the server.
- POST: Sends data to the server when creating new records.
- PUT/PATCH: Sends data to the server when updating existing records.
- DELETE: Sends data to the server in order to locate and delete a record. [26.]

For every available route, a specific action can be defined. The most common action is to send the request to the controller. Usually, when the controller is connected to a specific model it uses CRUD (Create-Read-Update-Delete) operations, which in turn are connected to the request methods listed before. A controller that uses CRUD operations is called a resource controller. Resource controllers have seven basic methods for managing models. [27.] These methods are explained and compared to the request methods in the following list:

- Index (GET): Used for the default view. For example, to list all the created instances of a model.
- Show (GET): Used to show a specific instance of a model.
- Create (GET): Used to return a view for creating a new instance.
- Store (POST): Used to store a new instance to the database.
- Edit (GET): Used to return a view for editing an existing instance of a model.

- Update (PUT/PATCH): Used to update an existing instance of a model.
- Destroy (DELETE): Used to delete an existing instance from the database. [27.]

As can be seen, resource controllers utilise the request methods in order to interact with the models in the corresponding way. The CRUD operations are not necessary when using controllers, though. For example, a controller does not have to be connected to a specific model, in which case it has no reason to list these specific methods either. Controllers, in general, are used to handle whatever logic necessary for the views. [27.]

### 3.3.2 Artisan interface

An important part of Laravel development environment is the Artisan command-line interface which provides useful commands to handle various tasks within the framework. If necessary, new commands may also be created by the user. For example, artisan can be used to create templates of things such as models and controllers or run migration files to update the database. [28.] Figure 9 shows a list of some of the artisan commands and their explanations.



```

Available commands:
clear-compiled  Remove the compiled class file
down           Put the application into maintenance mode
env           Display the current framework environment
help          Displays help for a command
inspire       Display an inspiring quote
list          Lists commands
migrate        Run the database migrations
optimize       Optimize the framework for better performance (deprecated)

)
preset        Swap the front-end scaffolding for the application
serve         Serve the application on the PHP development server
tinker        Interact with your application
up            Bring the application out of maintenance mode

app
app:name      Set the application namespace
auth
auth:clear-resets  Flush expired password reset tokens
cache
cache:clear   Flush the application cache
cache:forget  Remove an item from the cache
cache:table   Create a migration for the cache database table
config
config:cache  Create a cache file for faster configuration loading
config:clear  Remove the configuration cache file
db
db:seed       Seed the database with records
event
event:generate  Generate the missing events and listeners based on registration
ide-helper
ide-helper:generate  Generate a new IDE Helper file.
ide-helper:meta      Generate metadata for PhpStorm
ide-helper:models    Generate autocompletion for models
key
key:generate        Set the application key
make
make:auth            Scaffold basic login and registration views and routes
make:command         Create a new Artisan command
make:controller      Create a new controller class
make:event           Create a new event class
make:factory         Create a new model factory
make:job             Create a new job class
make:listener        Create a new event listener class
make:mail            Create a new email class
make:middleware      Create a new middleware class
make:migration       Create a new migration file
make:model           Create a new Eloquent model class
make:notification    Create a new notification class
make:policy          Create a new policy class
make:provider        Create a new service provider class
make:request         Create a new form request class
make:resource        Create a new resource
make:rule            Create a new validation rule
make:seeder          Create a new seeder class
make:test            Create a new test class
migrate
migrate:fresh        Drop all tables and re-run all migrations
migrate:install      Create the migration repository
migrate:refresh      Reset and re-run all migrations

```

Figure 9. A partial list of commands provided by the Artisan command-line interface

As can be seen from figure 9, Artisan provides several useful commands for managing the Laravel development environment. As with most of the things in Laravel, it is not necessary to use the Artisan interface but, if used the right way, it can vastly enhance the overall development process.

## 4 Project implementation

The project was carried out as part of a bigger development cycle so it was necessary to follow specific principles and standards set by PlusService. There were some additional tools that were utilised as part of the process. For example, Git version control system was used alongside BitBucket in order to follow the progress and review code. Version control is generally used for tracking the changes in source code and it enables the possibility of backups in case of mistakes. BitBucket provides a distributed version control with the possibility to review code before pushing it available. It also works as a repository where the whole history of the source code lives in. Another tool that was used to track progress was Jira which is used as a dashboard for all the current tasks. When a certain task has been finished it needs to be marked as completed. In PlusService, the software developers were following agile development principles where tasks are set for the next few weeks. This time period is called a sprint. The goal is to finish all the set tasks during a sprint and review the progress afterwards. For communication, an instant messaging service HipChat was utilised. It works like any other messaging platform but has also a built-in support for BitBucket and Jira. HipChat can be used to show important notifications from both of these platforms.

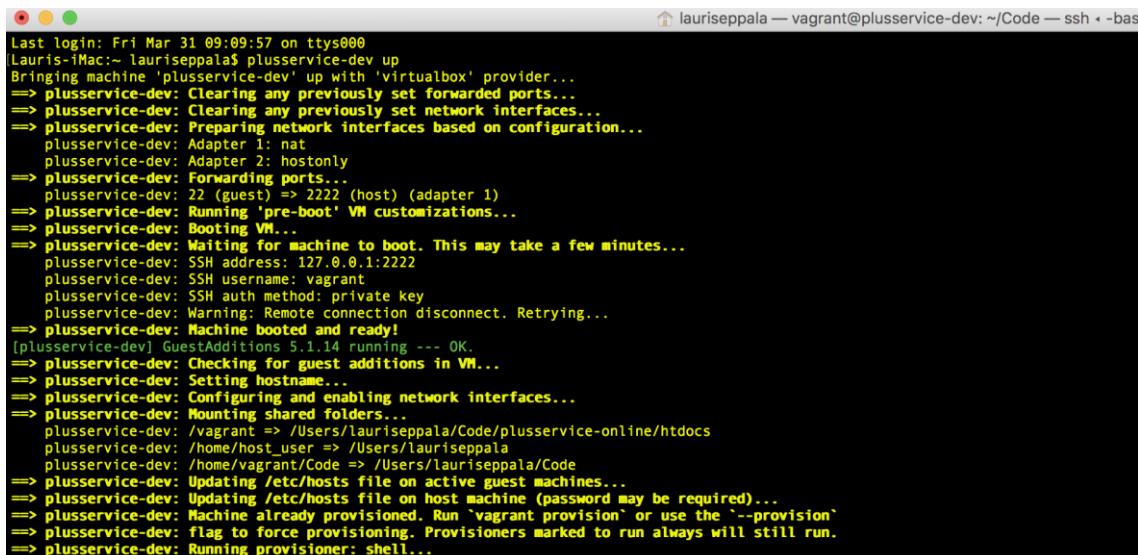
There were also a few tools that were used in direct relation with the development process itself. When working with databases, a common practice is to design and somehow illustrate the future tables before actually creating them. MySQL Workbench was used for this purpose. It can be used to design tables and determine the columns, attribute data types and the relations with primary – and foreign keys. MySQL Workbench can also be used as a tool for database administration purposes as it offers a wide variety of management functionalities. This did not become necessary during the project, however. Instead a simple database access software, HeidiSQL, was used. It is used as a Graphical User Interface for basic database operations like viewing the existing records and altering them. Lastly, PhpStorm was used as the main code editor as it has a vast amount of integrated features like code completion, keywords and Git tracking. Additional features may also be included by installing any of the free community created plugins. It supports many different languages like HTML, CSS, JavaScript and SQL as well as all the newest PHP versions.

## 4.1 Project environment setup

Before getting started with building the project, a Laravel specific development environment needed to be setup. Laravel can be set up by installing all the necessary tools and packages locally or by using the provided Vagrant box. Vagrant is a software that enables users to share pre-packaged development environments and run them through a virtual machine. Virtual machines are required since the Vagrant box might be using a different operating system or other incompatible settings. These machines make it possible to emulate the workings of other systems. This union of a virtual machine and Vagrant box make it easy to install and destroy environments as necessary. Laravel provides their own Vagrant box called Homestead. It is, of course, compatible with the all the tools that Laravel has to offer. The newest version of Homestead includes, for example, the following software:

- Ubuntu 16.04
- Git
- PHP 7.1
- MySQL
- Composer
- Node (With Yarn, Bower, Grunt and Gulp) [29.]

If necessary, an older version of Homestead maybe installed instead or the box can be customized by the user. For example, a customized version of Homestead was installed for the project since it was necessary to run some additional tools required by PlusService. The basic installation process is quite straight forward and well documented on the official Laravel website. When all the settings are done, a single Vagrant command can be used to run the setup process. A console view of running a setup is shown in figure 10.



```

Last login: Fri Mar 31 09:09:57 on ttys000
Lauris-iMac:~ lauriseppala$ plusservice-dev up
Bringing machine 'plusservice-dev' up with 'virtualbox' provider...
=> plusservice-dev: Clearing any previously set forwarded ports...
=> plusservice-dev: Clearing any previously set network interfaces...
=> plusservice-dev: Preparing network interfaces based on configuration...
    plusservice-dev: Adapter 1: nat
    plusservice-dev: Adapter 2: hostonly
=> plusservice-dev: Forwarding ports...
    plusservice-dev: 22 (guest) => 2222 (host) (adapter 1)
=> plusservice-dev: Running 'pre-boot' VM customizations...
=> plusservice-dev: Booting VM...
=> plusservice-dev: Waiting for machine to boot. This may take a few minutes...
    plusservice-dev: SSH address: 127.0.0.1:2222
    plusservice-dev: SSH username: vagrant
    plusservice-dev: SSH auth method: private key
    plusservice-dev: Warning: Remote connection disconnect. Retrying...
=> plusservice-dev: Machine booted and ready!
[plusservice-dev] GuestAdditions 5.1.14 running --- OK.
=> plusservice-dev: Checking for guest additions in VM...
=> plusservice-dev: Setting hostname...
=> plusservice-dev: Configuring and enabling network interfaces...
=> plusservice-dev: Mounting shared folders...
    plusservice-dev: /vagrant => /Users/lauriseppala/Code/plusservice-online/htdocs
    plusservice-dev: /home/host_user => /Users/lauriseppala
    plusservice-dev: /home/vagrant/Code => /Users/lauriseppala/Code
=> plusservice-dev: Updating /etc/hosts file on active guest machines...
=> plusservice-dev: Updating /etc/hosts file on host machine (password may be required)...
=> plusservice-dev: Machine already provisioned. Run 'vagrant provision' or use the '--provision'
=> plusservice-dev: flag to force provisioning. Provisioners marked to run always will still run.
=> plusservice-dev: Running provisioner: shell...

```

Figure 10. Running a Vagrant box setup process

As can be seen in figure 10, Vagrant runs a series of automated commands which makes the installation process painless for the user. The green line in the middle of figure 10 runs some additional commands included to this specific Vagrant box by PlusService.

One of the most important tools included in the Homestead installation is the Composer package manager. It can be used inside the application to manage project specific dependencies and required libraries. Once a library has been required within Composer, it will automate the process of installing and updating. Unlike some other package managers, Composer does not install anything globally but instead puts them in a folder inside the project. [30.] Composer can actually be downloaded by the user and used to install Homestead itself to further simplify the environment installation process. New projects within Homestead are also created with Composer. For example, the command shown in listing 8 was used to initiate GamesFactory.

```
composer create-project --prefer-dist laravel/laravel
gamesfactory
```

Listing 8. Creating a new Laravel project with Composer

The “--prefer-dist”-option in listing 8 installs dependencies without the version control history. This is the suggested option when using a package as part of a bigger project.

## 4.2 GamesFactory

After installing the development environment, the process of building GamesFactory was started. The idea was to create a simple administration dashboard where the user would be able to create and modify modules. Later GamesFactory would be improved to work as the content management system for all the websites within PlusService brand. For this reason, also the modules would have to support a white label attribute which specifies the supported websites.

The first thing to do was to create an authentication system with user registration and a login page. Laravel makes this task very easy as it provides a command that automatically builds this system. The command is shown in listing 9.

```
php artisan make:auth
```

Listing 9. Artisan command for creating an authentication system

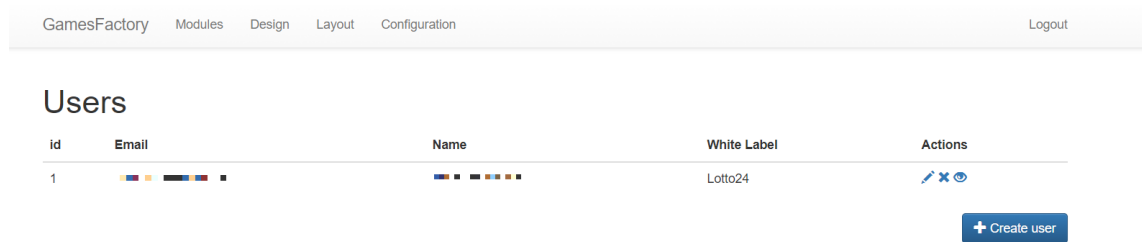
The command shown in listing 9 creates an authentication system following the basic Laravel building practices. For example, it creates migrations for database tables, the models related to these tables, views for all the necessary pages, controllers for all the logic between the models and the views as well as the route configurations. At this point, the only thing left to do was to run the migrations in order to have a fully functioning authentication system. [31.] Migrations are run with the command shown in listing 10.

```
php artisan migrate
```

Listing 10. Artisan command for running the migrations

In most cases, creating an authentication system with the command shown in listing 10 is enough but all of its functionality can of course be modified as necessary. The next step in the project was to extend the newly created user models. The user model came with basic attributes like email and password which were needed for the authentication. In addition, some other attributes such as the white label and deletion time were added. White label is used to see to which sites does the user have administration rights for. Deletion time, on the other hand, is a special Eloquent attribute known as timestamp. It creates a model-like instance of the time with its own attributes and methods by using

Carbon library. Carbon is an extended version of PHP's DateTime class and enables easy management of time-based attributes. With the user models done, the next step was to add functionality to create and modify users from the dashboard. First, a simple dashboard view was taken from the Twitter Bootstrap examples and modified to have the necessary navigation for GamesFactory. Then users page was created to list all the existing users with some basic information and management links. The links pointed to different pages to show more information about a specific user, to modify their information or to delete them. A view of the users page is shown in figure 11.

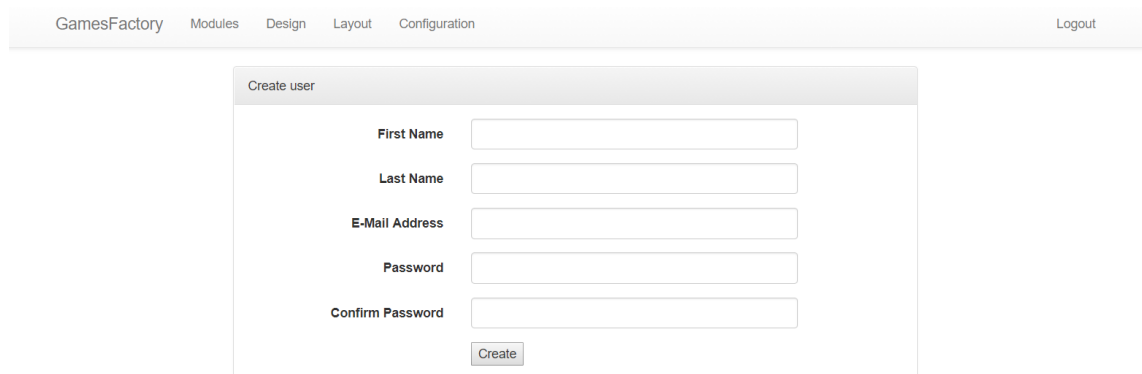


id	Email	Name	White Label	Actions
1	[blurred]	[blurred]	Lotto24	[edit] [delete] [view]

[+ Create user](#)

Figure 11. GamesFactory users page view

The email and name of the created user have been blurred in figure 11 but otherwise it shows the final outcome of the users page. In the bottom right corner, there is a button to create more users. A view of the user creation view is shown in figure 12.



GamesFactory Modules Design Layout Configuration Logout

### Create user

First Name

Last Name

E-Mail Address

Password

Confirm Password

Figure 12. GamesFactory user creation view

The form to create users in figure 12 has been implemented using Laravel Collective's HTML & Forms package which aims to simplify the creation of forms within Laravel. It allows, for example, the programmer to decide which HTTP-method to use since normally only GET and POST methods are available. Also, the CSRF-token has been automatically included in Laravel Collective forms which is a type of security measure against hacking attempts known as Cross-Site Request Forgery. Finally, it enables a simple form model binding technique which makes it possible to populate forms based on the contents of a model. [32.] An example of form model binding with Laravel Collective has been shown in listing 11.

```
{{ Form::model($user, [
    'method' => 'patch',
    'route' => ['dashboard.users.update', $user->id],
    'class' => 'form-horizontal'
]) }}
```

Listing 11. Form model binding using Laravel Collective

Listing 11 shows the opening tag of a form using Laravel Collective. The user model has been bound to the form in the first row. The rest of the rows are the attributes of this form. First, the method has been set as PATCH which means that the form is used to update an existing user. This can be confirmed in the next row where the route is set to the update page. Also, the user id has been injected to the route here. Lastly, a Bootstrap attribute has been added as a class to give the form some styling. GamesFactory's routing has been shown in figure 12 to explain more the concept of routing within Laravel.

Method	URI	Name	Action
GET HEAD	/	home	\App\Http\Controllers\HomeController@index
GET HEAD	_debugbar/assets/javascript	debugbar.assets.js	Barryvdh\Debugbar\Controllers\AssetController@js
GET HEAD	_debugbar/assets/stylesheets	debugbar.assets.css	Barryvdh\Debugbar\Controllers\AssetController@css
GET HEAD	_debugbar/clockwork/{id}	debugbar.clockwork	Barryvdh\Debugbar\Controllers\OpenHandlerController
GET HEAD	_debugbar/open	debugbar.openhandler	Barryvdh\Debugbar\Controllers\OpenHandlerController
GET HEAD	api/auth/{user}	api.login	\App\Http\Controllers\Auth\LoginController@apiLogin
GET HEAD	dashboard	dashboard.index	\App\Http\Controllers\DashboardController@index
POST	dashboard/configuration	dashboard.configuration.save	\App\Http\Controllers\Dashboard\ConfigurationController@save
GET HEAD	dashboard/configuration	dashboard.configuration	\App\Http\Controllers\Dashboard\ConfigurationController
POST	dashboard/converter	dashboard.converter	\App\Http\Controllers\Dashboard\ConverterController@convert
GET HEAD	dashboard/converter	dashboard.converter	\App\Http\Controllers\Dashboard\ConverterController
GET HEAD	dashboard/design	dashboard.design	\App\Http\Controllers\Dashboard\DesignController@design
GET HEAD	dashboard/layout	dashboard.layout	\App\Http\Controllers\Dashboard\LayoutController@layout
GET HEAD	dashboard/modules	dashboard.modules.index	\App\Http\Controllers\Dashboard\ModulesController@index
POST	dashboard/modules/auctions	dashboard.modules.auctions.store	\App\Http\Controllers\Dashboard\Modules\AuctionController@store
GET HEAD	dashboard/modules/auctions	dashboard.modules.auctions.index	\App\Http\Controllers\Dashboard\Modules\AuctionController@index
GET HEAD	dashboard/modules/auctions/bids	dashboard.modules.auctions.bids	\App\Http\Controllers\Dashboard\Modules\AuctionController@bids
GET HEAD	dashboard/modules/auctions/create	dashboard.modules.auctions.create	\App\Http\Controllers\Dashboard\Modules\AuctionController@create

Figure 13. Console view of GamesFactory's routes

Figure 13 shows a list view of GamesFactory's routes from the console. It was printed out using an Artisan command and shows important attributes about the routes. The figure has been cut to fit the document so it is missing some information. The first column on the left shows which method the route is using. The next two columns are the URI (Uniform Resource Identifier) and name which are the main components of routing. URI is used to identify the wanted resource and is normally part of the page address, URL (Uniform Resource Locator). The name, on the other hand, is a Laravel attribute which can be defined in the routes file. It can be used instead of the URI to write simpler code. The last shown column in figure 13 specifies which action this route uses. It points to a specific method and a controller. The controllers in GamesFactory utilize more or less the basic CRUD operations. Out of the figure, there is one more column that shows which middleware the route uses. Middlewares work as permissions deciding in which cases are the routes accessible.

After building all the necessary functionality for the users, the auctions' pages were created using similar practices. There were more models needed with different attributes but otherwise everything was done more or less the same way. Figure 14 shows the designs of the necessary tables in MySQL Workbench.

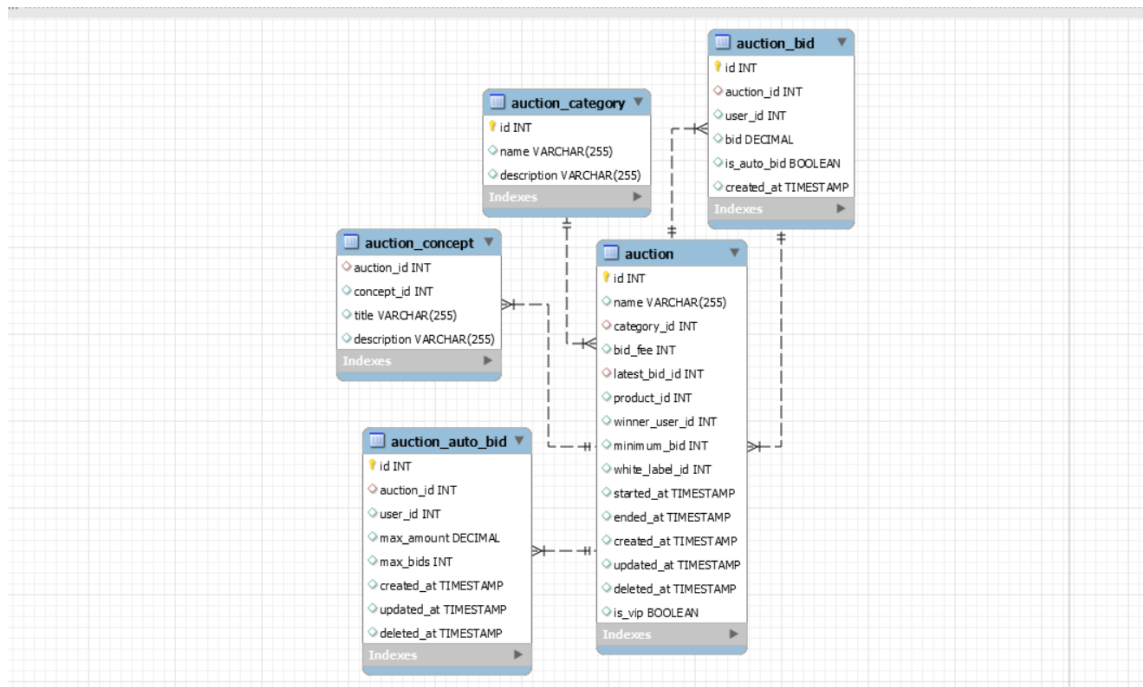


Figure 14. Designs of auction related tables



### 4.3 Auction module

As seen in figure 14, the auction module has quite a few different tables. The main auction table holds all the relevant information about itself but, in addition, is linked with two different kinds of bids: normal bid and an automatic one. All the auctions also belong to a single category which is used to group them. Categories can be created within GamesFactory. Lastly, auctions relate to various concepts to specify in which sites they belong to.

The main functionality of an auction is that it includes a product that will be bought by the highest bidder after the auction closes. Special type of tokens are used as the main currency in all the auctions. Also, the auction may have a minimum price if wished and it can be open for the public or only for the bonus members. The normal bids work in similar manner to buying a product but instead only an offer is made. For the automatic bids, it was necessary to include both the maximum price that the user is willing to offer as well as the maximum amount of bids before the automatic bid is deleted. The number of bids were limited because making a bid could have its own price depending on the auction. Categories are used in the main view for grouping and sorting the auctions.

In GamesFactory, the auction module was included into Lotto24 so that a page for auctions could be built for it. First, a grid to hold all the auctions was created using front end techniques and following the style of the existing shop page in Lotto24. Then the logic to fetch all the auctions and populate the grid with them was created within the auction controller. A simple sorting mechanism was also added to order the products according to starting time, price or customer group. Lastly, a manual of the basic functionality was written and linked to the main page of the auctions. The view of the auctions page is shown in figure 15.

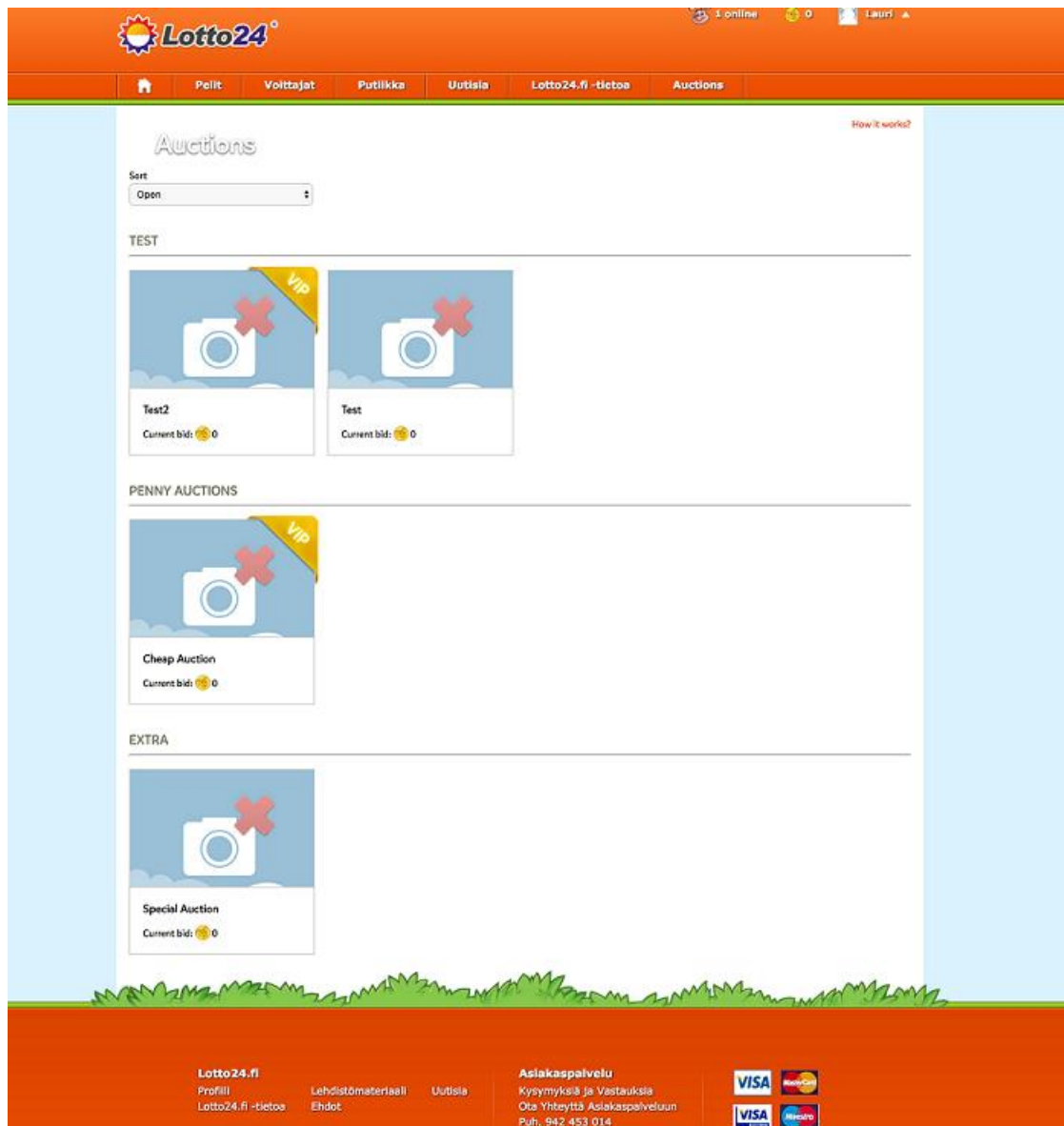


Figure 15. The auctions page in Lotto24

As can be seen in figure 15, the auctions are grouped by their categories and sorted within these groups. Some of the auctions marked with a VIP-tag are available for only bonus members. When the user hovers mouse over the auction an information box will slide up showing some additional information such as a live clock showing the time until start. If the auction is then clicked an additional modal will pop up and show all the necessary information as well as allow the user to make a bid. This modal is shown in figure 16.

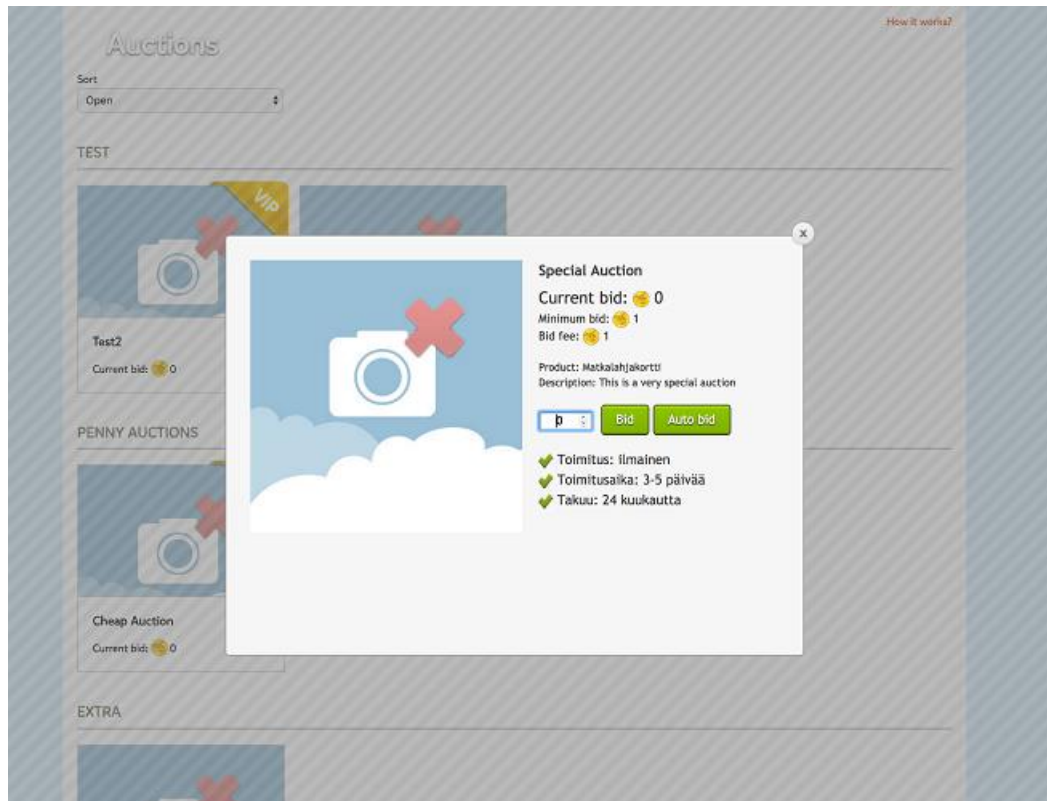


Figure 16. Auction modal in Lotto24

The auction modal shown in figure 16 shows all the necessary information about the auction. Since the localization for the module was not built, some of this information is hard-coded in English and the rest in the current set language of the web site, in Finnish. The user is able to either make a normal bid for the auction or an automatic one which will raise itself every time someone else goes higher. As many bids can be created simultaneously, the validity of a bid is checked in real time with JavaScript before accepting it. If successful, the bid is sent to a controller where it is created. Part of the bidding logic is shown in figure 17.

```

117
118 // Handle and refund old bid if necessary
119 if (isset($old_auto_bid)) {
120     // Check if old auto bid can bid again
121     if ($old_auto_bid->max_amount >= $input_value && $old_auto_bid->max_bids > 0) {
122         $this->auctionCreateBid($auction->id, $old_auto_bid->user_id, $input_value);
123         $old_auto_bid->decrement('max_bids', 1);
124
125         // Delete new auto bid if it was created
126         if (isset($new_auto_bid)) {
127             $new_auto_bid->delete();
128         }
129         $this->auctionRefundTokens($user_id, $input_value);
130     } else {
131         $old_auto_bid->delete();
132         $this->auctionRefundTokens($old_auto_bid->user_id, $old_auto_bid->max_amount);
133     }
134 } else if (isset($old_bid)) {
135     $this->auctionRefundTokens($old_bid->user_id, $old_bid->bid);
136 }
137
138 return $this->successResponse([
139     'redirectTo' => route_url('public.auctions'),
140 ]);
141 }
142

```

Figure 17. Part of the bidding logic

A small part of the bidding logic is shown in figure 17. The code checks if an automatic bid exists and a series of other alternatives before creating a new one. It also refunds the user that created the old bid, if necessary. The refunding is done because tokens are reserved every time a new bid is made. In the end, a success response is returned with the next route destination.

The last thing to do in the project, was to implement a task that runs automatically and checks for the ended auctions. For each of these auctions, an order is created and email sent to the winning bidder. This was implemented using Laravel's task scheduler which automatically runs previously created commands between set time intervals. For the project, it was decided to check the auctions every five minutes. The command shown in listing 12 was used to include the tasks to the automated process.

```

$schedule->command('cron:handle-ended-auctions')
    ->withoutOverlapping()
    ->everyFiveMinutes()
    ->appendOutputTo(storage_path('cron/auctions.log'));

```

Listing 12. Scheduling a task to check ended auctions

The command in listing 12 sets the task to run every five minutes without overlapping with itself. It also writes to a log file to help with possible errors.

## 5 Conclusion

The goal of this final year project was to implement an auction module for an online gaming service as well as a management system for the service and all the future modules. The project was carried out in Denmark for a local company and built partly on top of their existing platform. The work was carried out individually but with frequent checks from rest of the team to follow the progress and give feedback. Three months were given to finish the project which turned out be just enough, although thorough testing and releasing the final product would take place in a later time. Studying the theoretical aspects was also started well before the practical implementation.

The main tool for building the project was a PHP framework called Laravel which was extensively used in every part of the practical implementation. Although slow to get started with, it turned out be a highly efficient platform that provided useful tools to speed up the development process.

The development process was divided into two main parts, the first one being GamesFactory. GamesFactory was created to become the future content management system for PlusService so it was necessary to build a profound foundation that would be easy to extend later. For this reason, strict specifications were followed when creating critical aspects like routing and models. For example, the models were built from beginning to be compatible with the models from other projects carried out in PlusService.

In the second part, a prototype implementation of the auction module was created. It was built on top of an existing online gaming service, Lotto24. The auction module could also be attached to any other web sites using GamesFactory. The final result was an auction module with support for some additional features such as automatic bidding.

All in all, the project was a success and a working product was created within the set period of time. PlusService will continue to test and develop the GamesFactory platform in the future as well as enable auctions and other modules to their various web-sites as seen fit.

## References

- 1 What is PHP? [online]. PHP official website.  
URL: <http://php.net/manual/en/intro-what-is.php>  
Accessed 1 April 2017
- 2 About The Collective [online]. Laravel Collective official website.  
URL: <https://laravelcollective.com/about>  
Accessed 2 April 2017
- 3 What Is a Front-End Developer? [online]. Front-End Developer Handbook 2017.  
URL: <https://frontendmasters.com/books/front-end-handbook/2017/what-is-a-FD.html>  
Accessed 4 April 2017
- 4 What is the Document Object Model? [online]. World Wide Web Consortium (W3C) official website.  
URL: <https://www.w3.org/TR/WD-DOM/introduction.html>  
Accessed 6 April 2017
- 5 The HTML DOM Document Object [online]. W3Schools official website.  
URL: [https://www.w3schools.com/jsref/dom\\_obj\\_document.asp](https://www.w3schools.com/jsref/dom_obj_document.asp)  
Accessed 7 April 2017
- 6 Duckett J. HTML & CSS Design and Build Websites. Indianapolis, IN: John Wiley & Sons, Inc; 2011
- 7 HTML5 New Elements [online]. W3Schools official website.  
URL: [https://www.w3schools.com/html/html5\\_new\\_elements.asp](https://www.w3schools.com/html/html5_new_elements.asp)  
Accessed 10 April 2017
- 8 HTML Styles – CSS [online]. W3Schools official website.  
URL: [https://www.w3schools.com/html/html\\_css.asp](https://www.w3schools.com/html/html_css.asp)  
Accessed 12 April 2017
- 9 What is JavaScript? [online]. Mozilla Developer Network official website.  
URL: [https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First\\_steps/What\\_is\\_JavaScript](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript)  
Accessed 14 April 2017
- 10 jQuery Usage Statistics [online]. BuiltWith official website.  
URL: <https://trends.builtwith.com/javascript/jQuery>  
Accessed 16 April 2017
- 11 What is jQuery? [online]. jQuery official website.  
URL: <https://jquery.com>  
Accessed 17 April 2017
- 12 Bibeault B, Katz Y, De Rosa A. jQuery in Action, 3<sup>rd</sup> Edition. Manning Publications; 2015

- 13 Castiglione C. jQuery vs. JavaScript [online]. One Month official website; 19 September 2016  
URL: <https://learn.onemonth.com/jquery-vs-javascript/>  
Accessed 19 April 2017
- 14 Cochran D. Twitter Bootstrap Web Development How-To. Birmingham, UK; Packt Publishing Ltd; 2012
- 15 Examples – Custom components [online]. Bootstrap official website.  
URL: <http://getbootstrap.com/docs/4.0/examples/>  
Accessed 22 April 2017
- 16 Modal [online]. Bootstrap official website.  
URL: <http://getbootstrap.com/docs/4.0/components/modal/>  
Accessed 23 April 2017
- 17 A Beginner's Guide to Back-End Development [online]. Upwork official website.  
URL: <https://www.upwork.com/hiring/development/a-beginners-guide-to-back-end-development/>  
Accessed 1 May 2017
- 18 What can PHP do? [online]. PHP official website.  
URL: <http://php.net/manual/en/intro-whatcando.php>  
Accessed 3 May 2017
- 19 Database (DB) [online]. TechTarget official website.  
URL: <http://searchsqlserver.techtarget.com/definition/database>  
Accessed 5 May 2017
- 20 Primary and Foreign Key Constraints [online]. Microsoft official website.  
URL: <https://docs.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints>  
Accessed 6 May 2017
- 21 Structured Query Language (SQL) [online]. Techopedia official website.  
URL: <https://www.techopedia.com/definition/1245/structured-query-language-sql>  
Accessed 7 May 2017
- 22 Introduction [online]. Laravel official website.  
URL: <https://laravel.com/docs/4.2/introduction>  
Accessed 1 September 2017
- 23 MVC Framework – Introduction [online]. Tutorialspoint official website.  
URL: [www.tutorialspoint.com/mvc\\_framework/mvc\\_framework\\_introduction.htm](http://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm)  
Accessed 3 September 2017
- 24 Eloquent: Getting Started [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/eloquent>  
Accessed 4 September 2017
- 25 Blade templates [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/blade>  
Accessed 5 September 2017

- 26 Routing [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/routing>  
Accessed 6 September 2017
  
- 27 Controllers [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/controllers>  
Accessed 7 September 2017
  
- 28 Artisan Console [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/artisan>  
Accessed 9 September 2017
  
- 29 Laravel Homestead [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/homestead>  
Accessed 3 October 2017
  
- 30 Introduction [online]. Composer official website.  
URL: <https://getcomposer.org/doc/00-intro.md>  
Accessed 4 October 2017
  
- 31 Authentication [online]. Laravel official website.  
URL: <https://laravel.com/docs/5.5/authentication>  
Accessed 6 October 2017
  
- 32 Forms & HTML [online]. Laravel Collective official website.  
URL: <https://laravelcollective.com/docs/master/html>  
Accessed 7 October 2017