Bachelor’s thesis

International Information Technology

2019

Tony Kallio

FULL STACK DEVELOPMENT FOR A SMALL BUSINESS

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

International Information Technology

2019 | 31, x

Tony Kallio

FULL STACK DEVELOPMENT FOR A SMALL BUSINESS

The purpose of this thesis was to build a website for a small business featuring the necessary pages with content and a reservation system while discussing the technologies used in order to achieve this.

The first half of the thesis contains the technical information about technologies used on both the front-end and back-end development in addition to a brief overview of Content Management Systems. The second half features the goals laid out by the commissioner as well as discussion by the author on how goals were achieved and the technologies used in order to achieve them.

Overall the results were successful, and the website was deployed with changes to the final plan as per the commissioner’s requests. Those reading the thesis should gain a basic understanding of the methods and an example of the workflow used to attain the result.

KEYWORDS:

Full-stack, Front end, Back end, HTML, CSS, JavaScript, PHP

CONTENT

LIST OF ABBREVIATIONS (OR) SYMBOLS

Abbreviation	Explanation of abbreviation (Source)
CMS	Content Management System
CSS	Cascading Style Sheets
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
OS	Operating System
URL	Uniform Resource Locator
SQL	Structured Query Language
W3	World Wide Web

1 INTRODUCTION

An Oulu based hair and beauty salon Hiushuone SP Oy is changing ownership and wishes to deploy a modern web platform offering more services than just information on the location and opening hours. The commissioner currently has a simple website hosted by Weebly featuring information on location, open hours, and prices. Due to the hosting method, the website has low web visibility and search engine rankings resulting in low overall web traffic. Time reservations are currently handled through calling their phone or by visiting on-site.

The purpose of this thesis is to develop and deploy a modern web platform featuring additional services compared to those currently available, including an online booking system and mobile compatibility. Secondary goals will include attempting to raise the website's traffic and search engine visibility specifically on Google through the use of search engine optimization and the website along with Google's AdSense.

The theoretical portion of this thesis will delve into the technologies used to develop the costumer's web platform as well as research and discuss methods used to attempt to raise the web platform's visibility and Google search engine rankings based on specific keywords. This is then continued into the practical section where the new website is deployed, and web traffic is monitored, finally rounding out into the conclusion where the final results and thoughts as a whole on the project will be supplied.

A majority of the work will be production-oriented focusing on usability and visibility of the web platform but will also include research elements related to implementation technologies and digital marketing, as well as researching similar projects that have been undertaken.

2 FULL STACK DESIGN AND TECHNOLOGIES

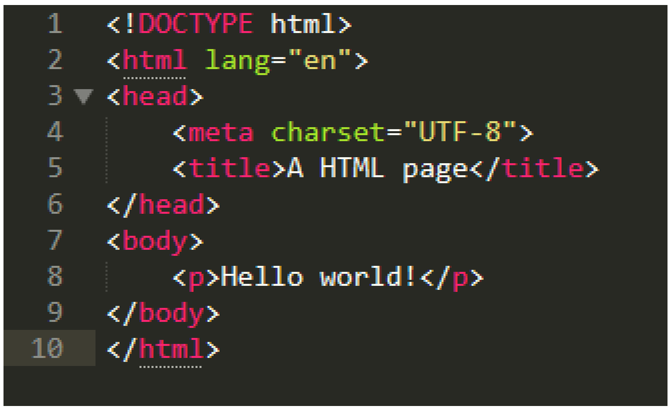
In web and software development, a full stack developer refers to someone both familiar and able to execute tasks on any level of the software “stack” they are involved with. More specifically, in web development, these layers are split in the front end or “visual” layer, which end users can see and interact with and the back end “data” layer (Kumar, 2018).

2.1 Front End

Front end development is the work done focusing on the end-user experience and mostly focuses on the software used as the framework of a website, namely HTML, CSS, and JavaScript. Tasks of front end developers include the planning and execution of the visual design aspects and design of websites as well as navigation between pages and simple interactivity and animations within pages. Front end developers are responsible for page performance and accessibility, including ensuring functionalities on various platforms including mobile devices.

2.1.1 HTML

Hypertext markup language, abbreviated as HTML. is a coding language designed to work as the base framework for a website. HTML consists of tags that are the basic elements that define the general structure of the website. Tags typically follow the format of and opening tag such as “<p>”, and a closing tag denoted with the same tag beginning with a slash “</p>”. The content that falls within these tags is the element that is displayed on the webpage. HTML pages can be identified with the starting <!DOCTYPE HTML> declaration at the start of the page. While not specifically an HTML tag, this declaration helps browsers identify that document is an HTML file (World Wide Web Consortium (W3C), 2017). An example of a simple HTML page can be seen in figure 1.



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>A HTML page</title>
</head>
<body>
    <p>Hello world!</p>
</body>
</html>
```

Figure 1. A sample HTML page featuring a singular body element

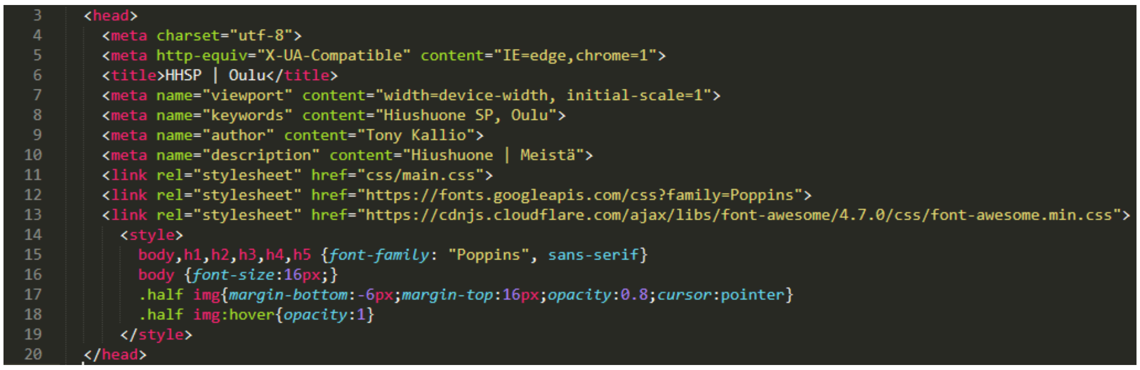
Tim Berners-Lee originally pioneered HTML in the late 1980s and by October 1990 had written three cornerstones of the internet that are still in use today, namely;

1. HyperText Markup Language, the formatting used for most websites today
2. Uniform Resource Identifier: a unique “address” used to identify web resources
3. HyperText Transfer Protocol, allowing for retrieval of linked web resources (W3 Consortium, 2017)

Though HTML has evolved over the years, most of the core characteristics have remained from the original version. The most recent version of HTML is HTML5, initially released in October 2014. This is the version supported by most modern browsers and is all the version the current project is being done in. The current adaptation at the time of writing is HTML 5.2, released in December 2017.

In general, HTML pages can be split into two sections, the head and body, both of which are contained within the <html></html> tags that define the entirety of the webpage.

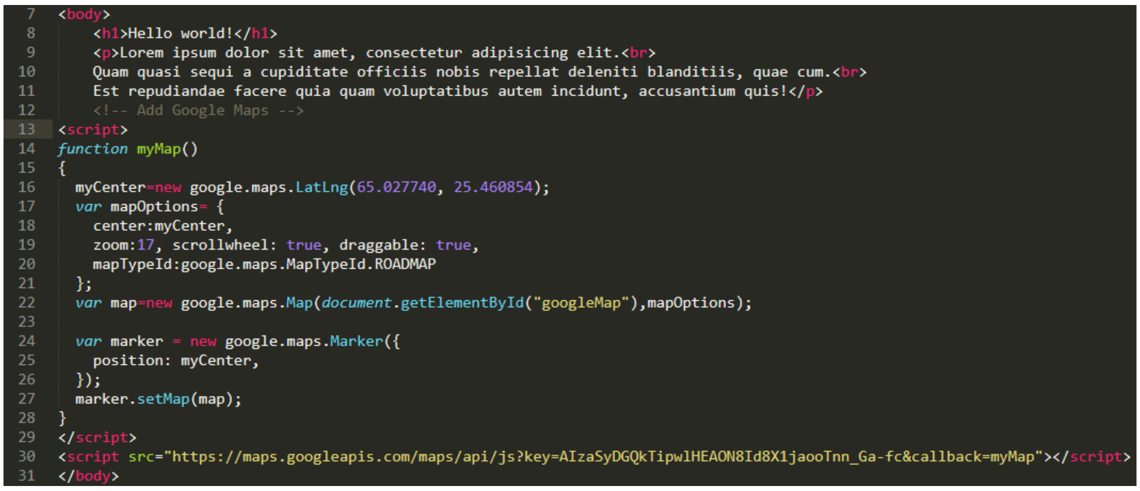
The HTML head contains the title and meta tags as well as links to possible CSS and other resources that may be called upon within that page. An example of an HTML head section can be found in Figure 2. HTML heads can be easily identified with the opening and closing <head> </head> tags.



```html
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>HHSP | Oulu</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="keywords" content="Hiushuone SP, Oulu">
  <meta name="author" content="Tony Kallio">
  <meta name="description" content="Hiushuone | Meistä">
  <link rel="stylesheet" href="css/main.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Poppins">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <style>
      body,h1,h2,h3,h4,h5 {font-family: "Poppins", sans-serif}
      body {font-size:16px;}
      .half img{margin-bottom:-6px;margin-top:16px;opacity:0.8;cursor:pointer}
      .half img:hover{opacity:1}
    </style>
</head>
```

Figure 2. An example of a complete HTML head section. The first and last line demonstrates the opening and closing head tags.

The HTML body is what is displayed to the end-user and contains all the individual elements that physically make up the page, such as headers, navbars, footers, and other elements. The HMTL body is also where scripts for elements on the page are contained. The body makes up the bulk of most pages, and like the head can be identified with its unique body tags. A small example can be found in Figure 3, featuring some sample heading, paragraph and a small code snippet (World Wide Web Consortium (W3C), 2017).

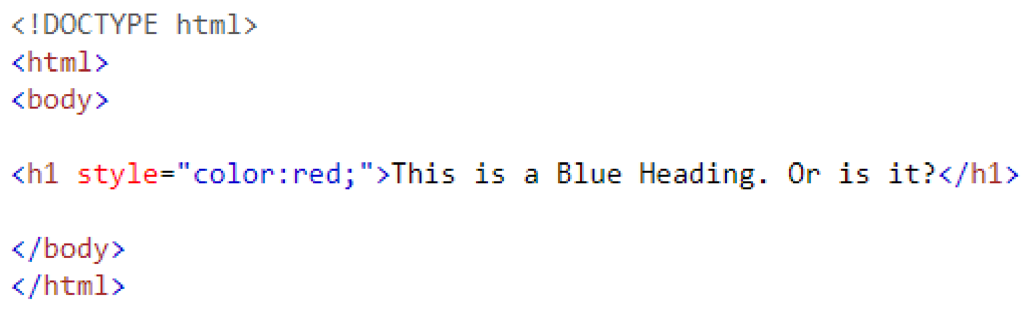


```html
<body>
    <h1>Hello world!</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.<br>
    Quam quasi sequi a cupiditate officiis nobis repellat deleniti blanditiis, quae cum.<br>
    Est repudiandae facere quia quam voluptatibus autem incidunt, accusantium quis!</p>
    <!-- Add Google Maps -->
<script>
function myMap()
{
  myCenter=new google.maps.LatLng(65.027740, 25.460854);
  var mapOptions= {
    center:myCenter,
    zoom:17, scrollwheel: true, draggable: true,
    mapTypeId:google.maps.MapTypeId.ROADMAP
  };
  var map=new google.maps.Map(document.getElementById("googleMap"),mapOptions);

  var marker = new google.maps.Marker({
    position: myCenter,
  });
  marker.setMap(map);
}
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDGQkTipwlHEAON8Id8X1jaooTnn_Ga-fc&callback=myMap"></script>
</body>
```

Figure 3. An example of an HTML body section, featuring a single header, paragraph, and a simple script to display an interactive google map.

2.1.2 CSS

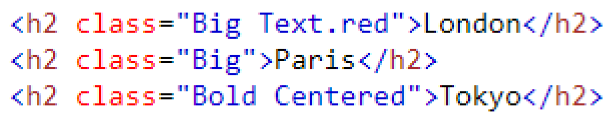
CSS refers to Cascading Style Sheets, a language used to define the layout of HTML elements and other media found within webpages. CSS can be either coded inline within HTML code as seen below in figure 4,



```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:red;">This is a Blue Heading. Or is it?</h1>

</body>
</html>
```

Figure 4. HTML with inline CSS giving h1 the red text property.

or alternatively, linked as a separate entity in order to more efficiently apply CSS attributes to multiple elements using the “Class” characteristic in order to apply the CSS elements attributed to that class onto the element defined by the class such as below in Figure 5.

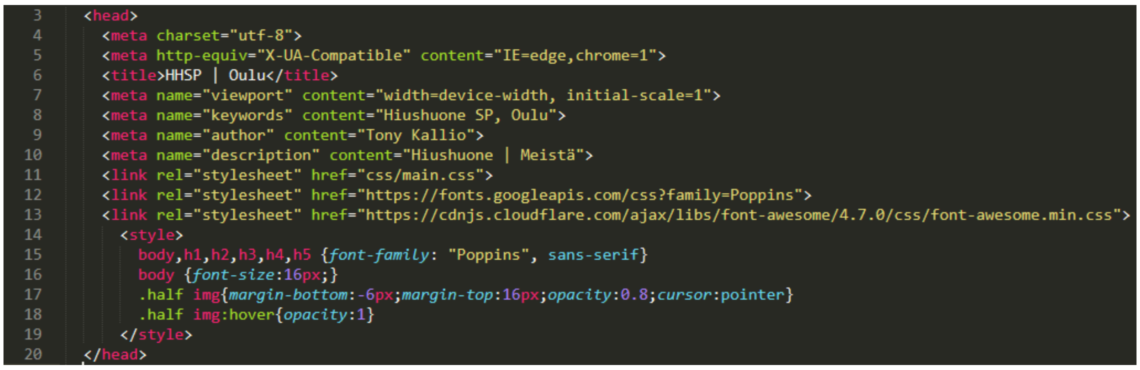


```
<h2 class="Big Text.red">London</h2>
<h2 class="Big">Paris</h2>
<h2 class="Bold Centered">Tokyo</h2>
```

Figure 5. 3 H2 elements with different classes attributed to them giving different CSS characteristics.

These classes are given attributes within the .CSS style sheet that is then applied to all elements with that specific class tag. HTML elements can have multiple class tags (W3 Schools, 2018).

Individual elements can also be defined within the HTML header with the “<style>” tag in order to apply the specific styling those elements, as seen below in Figure 6.



```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>HHSP | Oulu</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="keywords" content="Hiushuone SP, Oulu">
  <meta name="author" content="Tony Kallio">
  <meta name="description" content="Hiushuone | Meistä">
  <link rel="stylesheet" href="css/main.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Poppins">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <style>
      body,h1,h2,h3,h4,h5 {font-family: "Poppins", sans-serif}
      body {font-size:16px;}
      .half img{margin-bottom:-6px;margin-top:16px;opacity:0.8;cursor:pointer}
      .half img:hover{opacity:1}
    </style>
</head>
```

Figure 6. The <style> tag within the HTML header defines the used font, font size, and margins for elements within the “body” and “h1-5” HTML elements, as well as defining image margins for images (Chris Mills, 2018).

CSS was released initially in December of 1996 by Håkon Wium in order to improve content accessibility and improve web design flexibility by allowing the editing the outlook of an entire website through the use of a single file. Nowadays, it is still maintained by Håkom in addition to the W3 Consortium. Overall it has gone through 2 major revisions, as can be seen below in Figure 7 (Bos, 2016).

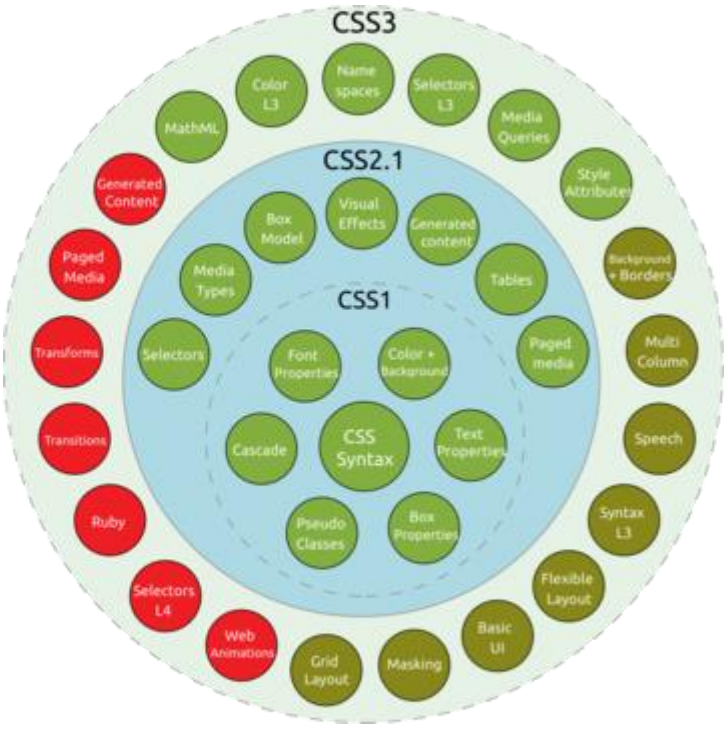


Figure 7. An image highlighting the features implemented with each new revision. Red circles indicate a working draft version, while dark green represents W3 consortium candidate revisions or those added after the base version (Mavrody, 2015).

2.1.3 JavaScript

JavaScript is a programming language used to add or modify the behavior of a webpage, and is one of the core three technologies in web front end development along with HTML and CSS, and is used in over 90% of websites as of June 2018 (W3 Techs, 2018). JavaScript's key feature is to enable users to interact with a website on the front end, though there are also libraries of JavaScript that run back end services. Most modern web browsers have dedicated JavaScript engines in order to execute code.

JavaScript was initially designed as a feature or Netscape Navigator 2.0 in September 1995 by Brandan Eich and was called LiveScript before undergoing minor updates and a name change to the current JavaScript in December of the same year. In 1996 is was also adapted by Microsoft to work with their then Internet Explorer 3. Along with being slowly adapted by other browser companies the June 1998 revision was adapted to conform with ISO/IEC 16262 international standards and on December of 1999 the basis for the current version of JavaScript was released to the public under the name of ECMA script 3. (Aston, 2015) The current version of JavaScript is based around ECMAScript 2017, released in June 2017 (Morelli, 2017).

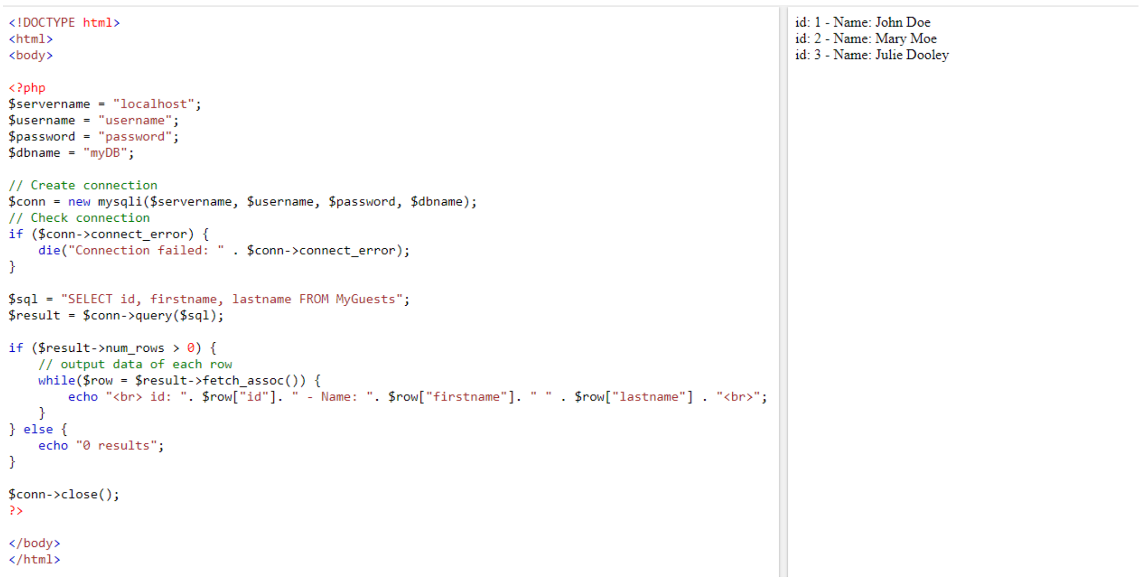
JavaScript has multiple libraries to enable different features to be added to pages, some enabling it to also function as the primary back end language of a web service such as Node.JS, enabling server-side JavaScript execution (W3 Schools, 2018).

2.2 Back end

Back end frameworks refer to the server-side languages used by web platforms. These include both databases and software not run client-side. The back end is frequently also referred to as the data access layer as it is the site elements that interact with and augment data within the web service. Common web service backend languages include C, C#, Node.JS, PHP, Python, and Ruby for applications and languages such as SQL and its many variants for database management.

2.2.1 SQL

SQL or Structured Query Language is the most commonly used backend language related to the storage and manipulation of relational databases. SQL features a relatively simple syntax and can easily be incorporated into other languages in order to manipulate databases when required (SQL Course, 2014), such as below in Figure 8.



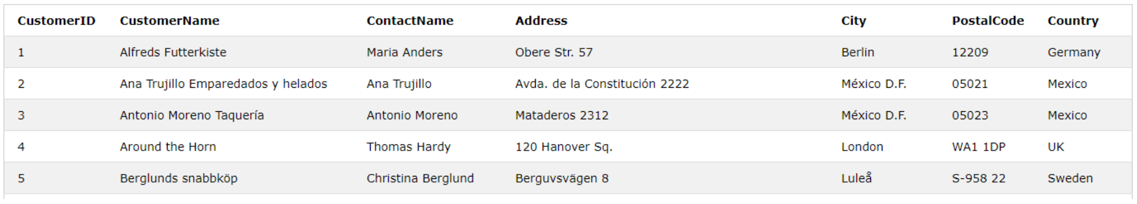
```
<!DOCTYPE html>
<html>
<body>

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<br> id: ". $row["id"]. " - Name: ". $row["firstname"]. " " . $row["lastname"] . "<br>";
    }
} else {
    echo "0 results";
}

$conn->close();
?>

</body>
</html>
```

```
id: 1 - Name: John Doe
id: 2 - Name: Mary Moe
id: 3 - Name: Julie Dooley
```

Figure 8. A simple SQL command ran within a PHP script retrieving and showing a list from a linked SQL database.

SQL was initially developed in 1974 at IBM, with the original people working on it being Donald D. Chamberlin and Raymond F. Boyce. Original called SEQUEL, the language was designed to function with their then in use relational database known as System R. Released publicly in 1979, SQL quickly gained popularity before being both ISO and ANSI certified in 1986 as the “Standard Database Language SQL” (Laine, 2000). New versions are published quite often, with the most recent release being SQL:2016 in December of 2016. It is currently being updated and maintained cooperatively by both ISO and IEC (Microsoft, 2018).

SQL databases are stored in a form known as a table, featuring entries in rows and columns known as fields. Each column represents a specific entry of information such as name, phone number, unique ID, or other identifiers. Rows are unique individuals or elements. An example of a simple table can be seen below in Figure 9.



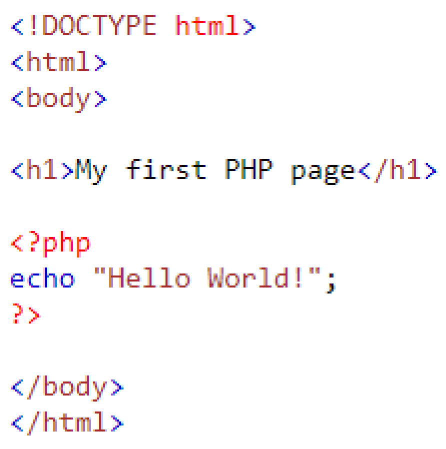
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Figure 9. A simple table featuring seven columns and five rows. Note the unique customerID on the first row for every individual.

A key feature in SQL tables is the use of a primary key. This primary key is a constraint that is used to uniquely identify separate records in a table and, therefore, cannot have duplicates within that database. Each table may only contain one primary key, but a second key known as a foreign key can be used to identify a row within another table to establish a unique relationship (Steve Stein, 2017). An example of such can be given using the CustomerID primary key from one table as a reference to a separate table featuring all purchases from a store to quickly identify purchases a customer has made.

2.2.2 PHP

PHP: Hypertext processor or PHP for short is a commonly used backend language with scripts being run server-side. PHP files often include HTML, CSS, JavaScript, and PHP snippets but are executed on the backend or server-side, allowing the front-end user to view pages as a standard HTML page. PHP was made accessible by its comprehensive support of OS and database platforms, making it compatible with most servers. PHP can be used to code a variety of server-side tasks such as database and file management, encryption, cookie management and generation of dynamic content, to name a few (Mkhitaryan, 2017). As stated previously, PHP can easily be incorporated into HTML pages, such as below in figure 10.



```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

Figure 10. A Simple echo PHP script to display the line “Hello World!” in an HTML page.

PHP development began in 1994 by Rasmus Lerdorf by writing several programs in C for his homepage. He continued working a specific one called Personal Home Page/Forms Interpreter extending its feature to incorporate the ability to work with forms and modify databases. This early form could be used to build simple back end applications but was still a far cry from the basis of modern PHP. Rasmus publicly released PHP Tools version 1 in June 1995, where the web development community quickly embraced it and started to grow into its unique language. In 1997 PHP was rewritten into its modern version with the help of Zeev Suraski and Andi Gutmans, forming the base of PHP 3.0, also renaming it to its current convention. (The PHP Group, 2004) What followed was multiple years of additional features and revisions with the latest full release being PHP 7.2 in November 2017 (The PHP Group, 2018).

2.3 CMS

Recently, content management systems (CMS) have been on the rise compared to traditionally coded websites. Critical features of CMS are often user friendly interfaces and the separation of contents from designs to enable a simple management of a page’s layout without having to edit the pages’ source code. Usually, CMS come packaged with multiple themes that allow the user to select the visual design of their site and add modules or plugins in order to modify the content. CMS are also web or cloud-based allowing multiple users access in order to edit and manage the sites without the use of repositories such as Git. These features allow users with minimal web development experience to launch fully-fledged and interactive sites.

There have been some negative aspects surrounding CMS. Usually, since elements are stored individually and retrieved on request rather than supplying a premade page with HTML, CMS sites tend to load slower and, as a result, may suffer from lower SEO scores. CMS also rely on the underlying code to be updated regularly in order to maintain the functionality of current and future plugins and modules and thus are seen as a higher maintenance alternative to HTML/CSS based web platforms (AmberD, 2017).

3 PLANNING AND IMPLEMENTATION

3.1 Goals

The overall objective of this thesis was to design and deploy a full-stack environment for a small business featuring a reservation system. At the beginning of the project, different front-end services were investigated in order to be able to deliver on what was promised to the end client effectively. The two outstanding choices were the standard HTML/CSS based self-coded site and CMS services offering rapidly-built and managed websites and services such as WordPress and Wix. After researching the available options, it was decided to work on the more traditional HTML/CSS based site for several reasons most prominently;

1. More self-sustained once deployed as there are fewer plugins and updates to track and manage.
2. Search Engine rankings: An additional goal for this thesis was trying improving the site's SEO rankings if time allowed.

3.1.1 Commissioner Objectives

Initially, the commissioner supplied simple directives for the design of the website. It had to be both simple in design and easy to navigate, as the business already had an established clientele, of which the majority being of an older generation who typically would view it on a desktop PC rather than on their smartphone. Based on these guidelines, multiple sample websites were created. Designs included a single page design with a consistently top floating navigation bar, and a more traditional website featuring multiple pages again with a top floating navigation bar.

The commissioner decided upon using the more traditional multiple page setup stating reasons of familiarity and ease of navigation, as only information pertinent to the current tab would be displayed, resulting in a more straightforward and easier to use end-user experience.

3.1.2 Personal

The creator of this personal thesis objective mostly revolved around the betterment of their skills in web development. As they already had quite a solid understanding of the front-end side of full-stack development, most of the project learning would be based around the back end, specifically about PHP and SQL as these are some of the most common languages that reservation systems are based on and ended up being used in this project.

3.2 HTML/CSS

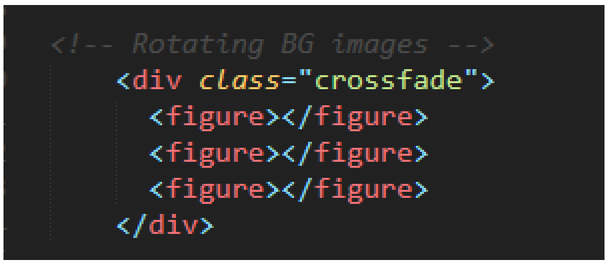
Logically work started from the base framework of the site, notably the front-end design; similar business sites were researched and investigated to get a general framework of how typical websites in the field were made in terms of appearance and navigation. Based and these and the directives outlined by the commissioner previously, rough drafts were sketched out and presented for appraisal. Some basic descriptions and preferences for the website were provided, and work and the HTML portion could begin.

Rough drafts of several sample frameworks were submitted for approval, and more information was given by the commissioner about the structure and content to be added later to the project. Multiple suggestions and variants such as single page designs were shown and eventually ruled out until a final base design was chosen with certain elements incorporated from previous versions.

Work began on the CSS outline of pages featuring inline syntax on each page. Once pages were finished and approved, a stylesheet for each was made using classes that were made in order to clean up the HTML code.

From here onwards, it was solely a matter of polishing up the design by making a singular CSS sheet sharing elements and classes for all the pages encompassed by the project and removing unused snippets, artifacts, and old elements from the previous versions of the page.

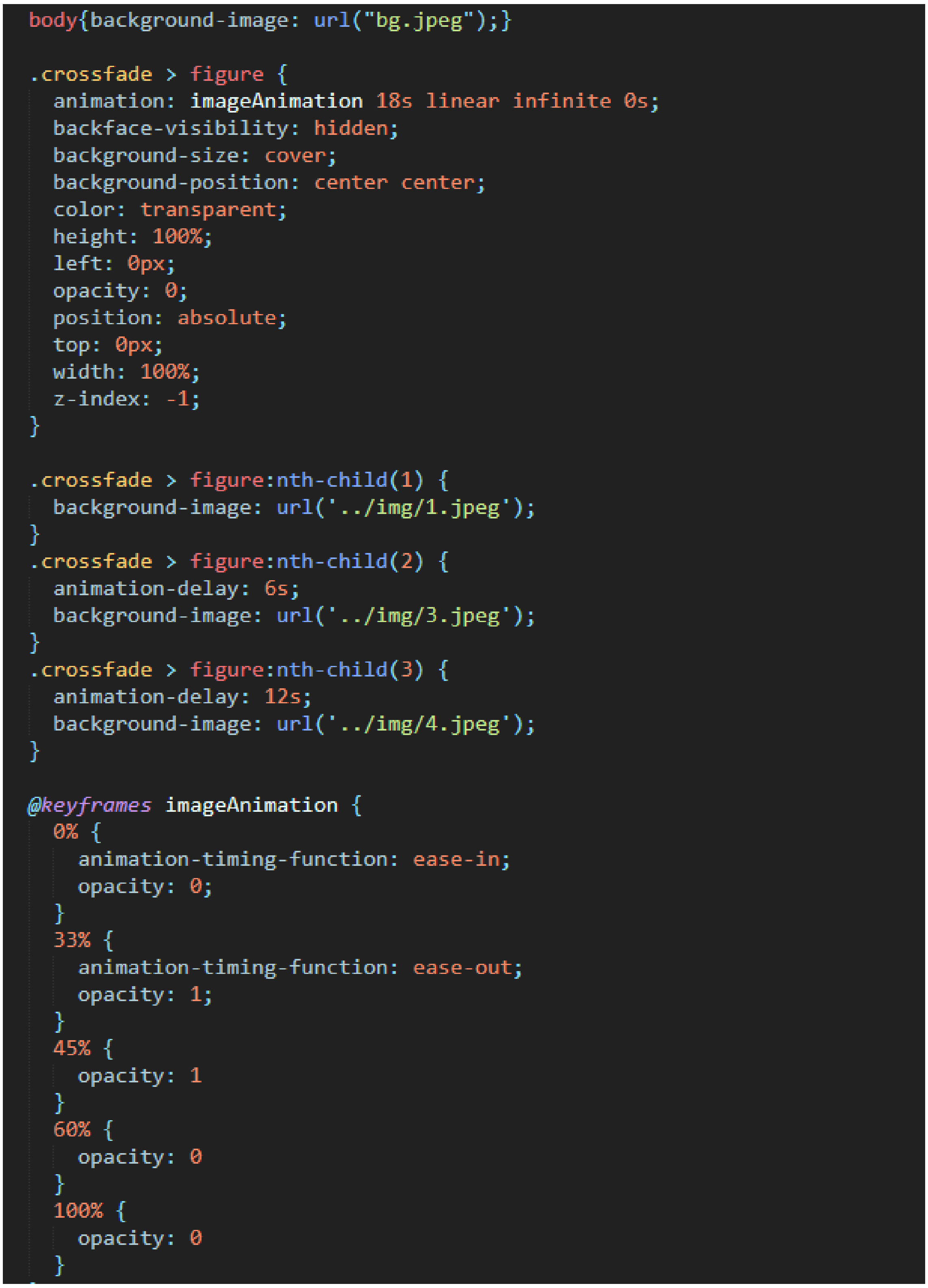
The original idea was to use JavaScript-based animation libraries for an animated background on a specific though due to issues discussed later, it was instead decided to adapt and use the CSS3 libraries in-built animation. Some research pulled up a few promising examples and eventually, the author adopted a version created based on CSS.Materialise libraries by Mary Lou (Lou, 2012). The source code for this ended up being rather simple and can be viewed below in Figure 11.



```
<!-- Rotating BG images -->
    <div class="crossfade">
      <figure></figure>
      <figure></figure>
      <figure></figure>
    </div>
```

Figure 11. The entirety of the HTML code for the rotating images.

The original code was modified to contain fewer images with a longer time and slightly altered fade time, shown below in Figure 12.



```css
body{background-image: url("bg.jpeg");}

.crossfade > figure {
  animation: imageAnimation 18s linear infinite 0s;
  backface-visibility: hidden;
  background-size: cover;
  background-position: center center;
  color: transparent;
  height: 100%;
  left: 0px;
  opacity: 0;
  position: absolute;
  top: 0px;
  width: 100%;
  z-index: -1;
}

.crossfade > figure:nth-child(1) {
  background-image: url('../img/1.jpeg');
}
.crossfade > figure:nth-child(2) {
  animation-delay: 6s;
  background-image: url('../img/3.jpeg');
}
.crossfade > figure:nth-child(3) {
  animation-delay: 12s;
  background-image: url('../img/4.jpeg');
}

@keyframes imageAnimation {
  0% {
    animation-timing-function: ease-in;
    opacity: 0;
  }
  33% {
    animation-timing-function: ease-out;
    opacity: 1;
  }
  45% {
    opacity: 1
  }
  60% {
    opacity: 0
  }
  100% {
    opacity: 0
  }
```

Figure 12. The CSS code used to fade in and out background images.

Despite changing most of the timers, most of the core code is the CSS3 animate kit found in the original code. Despite no using a JavaScript-based application such as with Slick, the overall result was still smooth and was approved for the final version of the website.

3.3 Mobile Optimization

Finally, a personal project of the author began where they started working on making the website more mobile-friendly, as information released about Google's search engine optimization protocols reveals that these type of web pages rank higher on Google results. (Google, 2018) Search engine optimization is discussed later on in its own section in more depth. The viewport scale command was included in all site pages to enable element scalability, and individual elements were also made adaptive based on-screen resolution.

The vertical navbar was exchanged for a simple collapsing menu when the width of the viewport went below a specific size was implemented using a lightly modified JavaScript example acquired from W3schools. This was a simple script featuring three unique appearances, the standard full-screen menu, and the open and closed small screen variants on sub-1000 pixel pageviews, a size commonly found on mobile devices and tablets (W3 Schools, 2018). This was then tested using Bluestacks to simulate multiple different phones and viewport sizes, and the final menu design was approved along with the final draft of the website. Screenshots and source code of this menu in action can be viewed in the appendices section under the Menu topic.

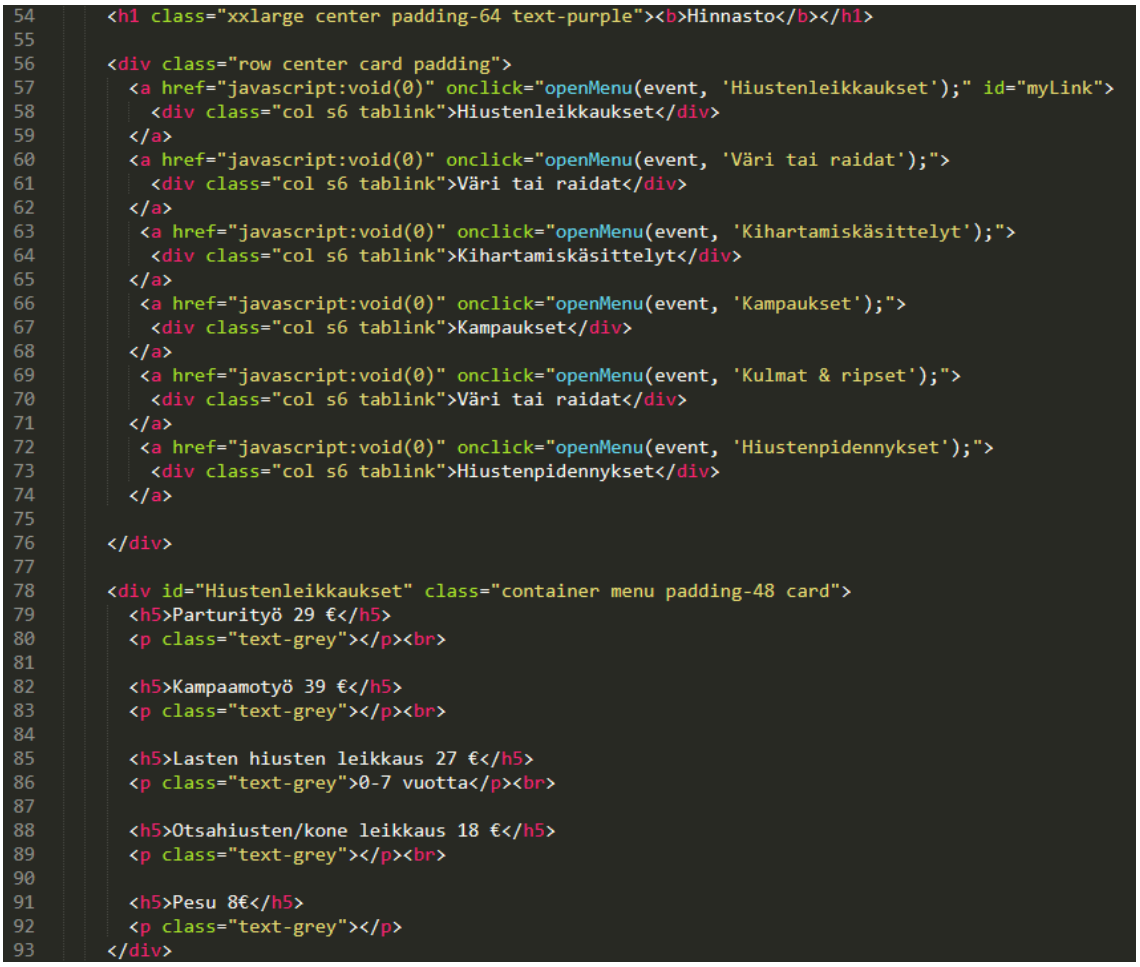
3.4 JavaScript

With the basic structure of the website finalized the last stages of modifying the appearance to suit the commissioners' preferences. Colors were decided on and edited using the main CSS file. A background image was requested for most pages, excluding a few with significant critical elements in order to retain clarity, an example being the information page, including a map.

The commissioner had initially requested a carousel-based solution on an individual page featuring a rotating selection of images. Different design suggestion was supplied, and a solution featuring no buttons and an automatically changing set of images agreed upon.

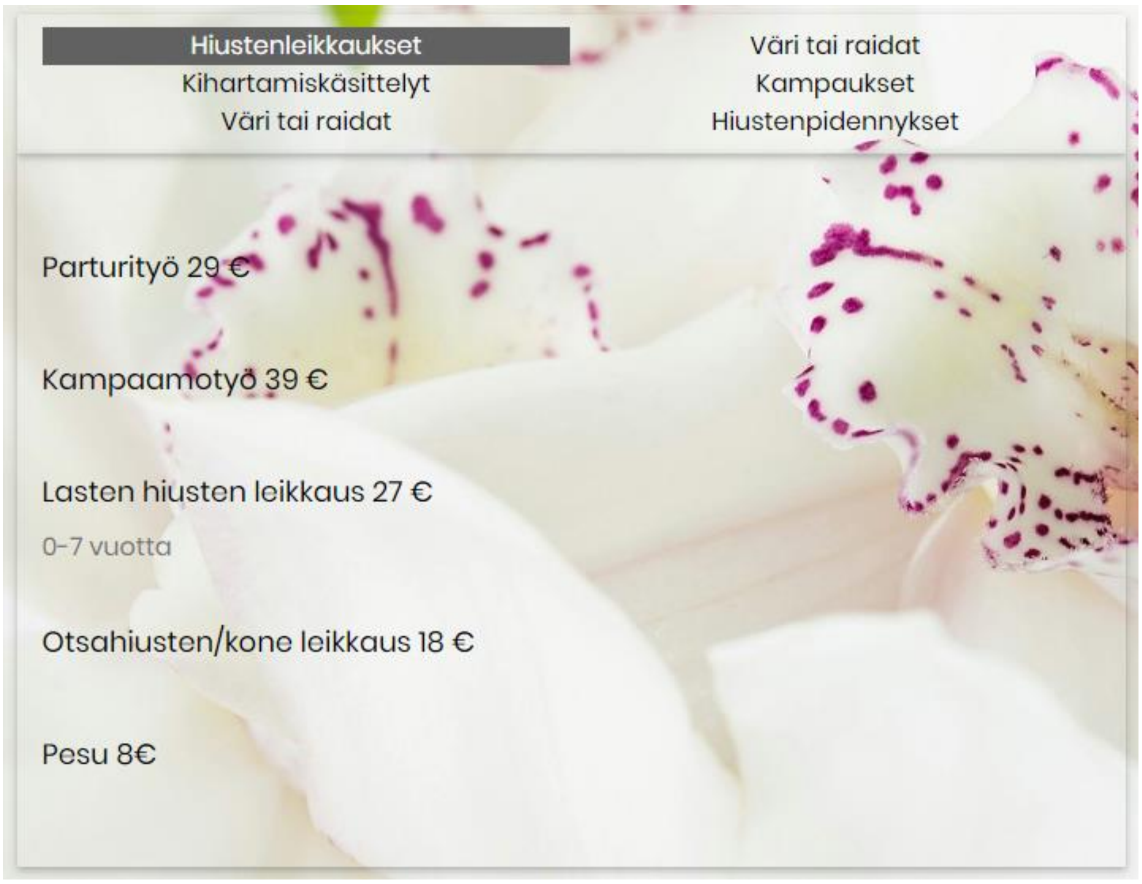
Work began with researching different methods to execute to this, and the author first began with trying to implement the JavaScript-based “Slick” (Wheeler, 2018). This ended with a working automated carousel; however the JavaScript elements worked poorly with positioning, especially breaking with the menu and other elements on the z-axis resulting in the navigation elements being either covered or otherwise rendered inoperable. This carousel was replaced with the more straightforward CSS solution discussed earlier in the HTML/CSS Section.

From here on out work consisted mostly of filling out the content required for each page. A small issue of page length occurred on pages involving long lists of prices, which was remedied with the use of a small variation of JavaScript container, the code, and an example of such supplied below in Figures 13 and 14, respectively.



```html
    <h1 class="xxlarge center padding-64 text-purple"><b>Hinnasto</b></h1>

    <div class="row center card padding">
      <a href="javascript:void(0)" onclick="openMenu(event, 'Hiustenleikkaukset');" id="myLink">
        <div class="col s6 tablink">Hiustenleikkaukset</div>
      </a>
      <a href="javascript:void(0)" onclick="openMenu(event, 'Väri tai raidat');">
        <div class="col s6 tablink">Väri tai raidat</div>
      </a>
       <a href="javascript:void(0)" onclick="openMenu(event, 'Kihartamiskäsittelyt');">
        <div class="col s6 tablink">Kihartamiskäsittelyt</div>
      </a>
       <a href="javascript:void(0)" onclick="openMenu(event, 'Kampaukset');">
        <div class="col s6 tablink">Kampaukset</div>
      </a>
       <a href="javascript:void(0)" onclick="openMenu(event, 'Kulmat & ripset');">
        <div class="col s6 tablink">Väri tai raidat</div>
      </a>
       <a href="javascript:void(0)" onclick="openMenu(event, 'Hiustenpidennykset');">
        <div class="col s6 tablink">Hiustenpidennykset</div>
      </a>

    </div>

    <div id="Hiustenleikkaukset" class="container menu padding-48 card">
      <h5>Parturityö 29 €</h5>
      <p class="text-grey"></p><br>

      <h5>Kampaamotyö 39 €</h5>
      <p class="text-grey"></p><br>

      <h5>Lasten hiusten leikkaus 27 €</h5>
      <p class="text-grey">0-7 vuotta</p><br>

      <h5>Otsahiusten/kone leikkaus 18 €</h5>
      <p class="text-grey"></p><br>

      <h5>Pesu 8€</h5>
      <p class="text-grey"></p>
    </div>
```

Figure 13. The source code for the price list container



Hiustenleikkaukset
Kihartamiskäsittelyt
Väri tai raidat
Väri tai raidat
Kampaukset
Hiustenpidennykset

Parturityö 29 €

Kampaamotyö 39 €

Lasten hiusten leikkaus 27 €
0-7 vuotta

Otsahiusten/kone leikkaus 18 €

Pesu 8€

Figure 14. The final appearance of the price container with the background.

A more flexible tab-based container would have been desirable as it would have kept a fixed page size, but due to technical issues, lack of in-depth JavaScript know-how, and time constraints, a simple cell-based solution used. In terms of execution, the simpler version still was valid at the given goal of reducing the particular page from a lengthy multiple page scroll into a simpler one-page version.

3.4.1 Google Map API

The next step of the project was to begin work on Google’s map API integration to the information page in order to give viewers an interactive map in order to help find the business. This was done by registering the business to Google’s cloud platform and registering for a unique Geo-location API for the user. This is necessary in order to monitor and, if necessary throttle map loads in order to combat potential DOS attacks by using bots or scripts to load maps thousands of times per second. Google itself supplies the example HTML snippet with accompanying JavaScript code to integrate the map into a client's website where all that has to be done in substitute for the API key that is given out by them when registering for the service (Google, 2018). Some tweaking was done for clarity's sake, in addition to making the map be the significant element of the page by adjusting the size and zoom level of the map. User control for the map was also enabled in order to help users better query the location of the business. A screenshot of this along with the source code can be found below in Figures 15 and 16.

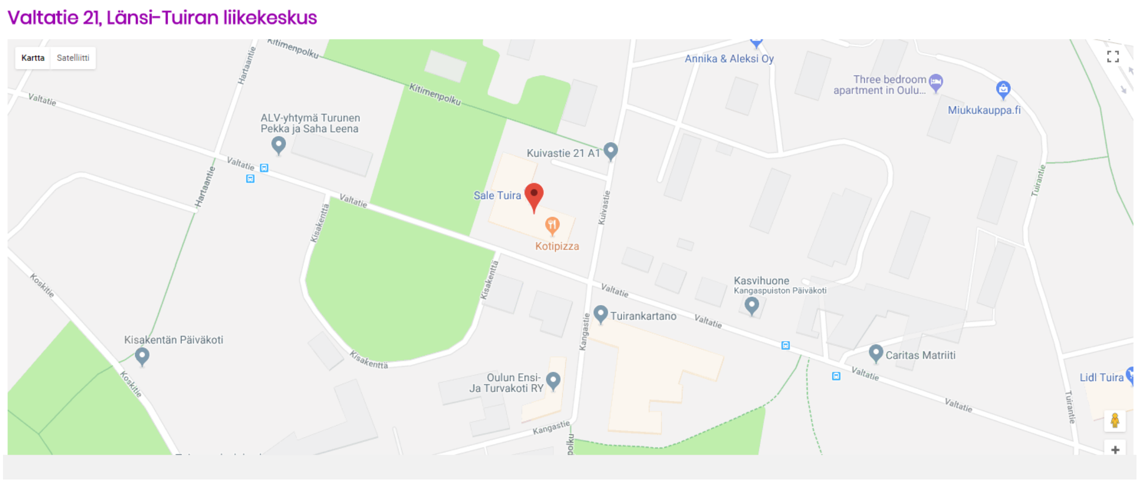


Figure 15. The adjustable Google Maps elements

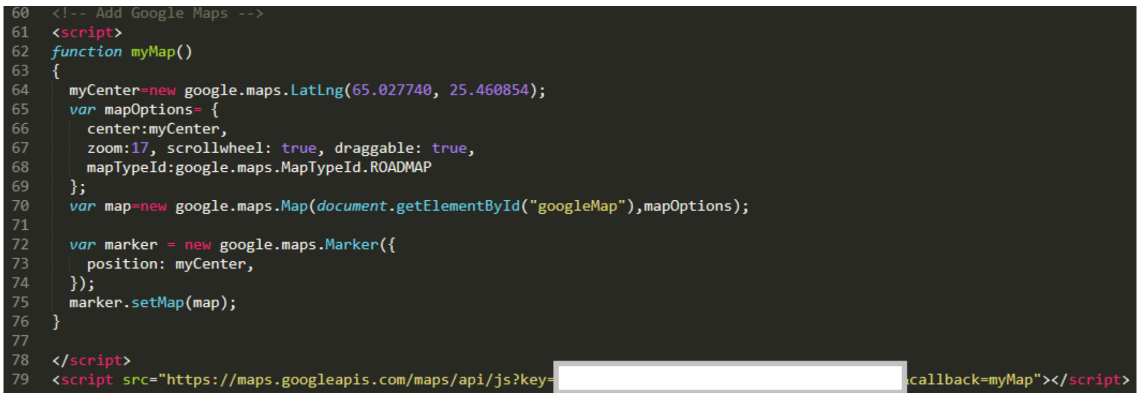

```
<!-- Add Google Maps -->
<script>
function myMap()
{
  myCenter=new google.maps.LatLng(65.027740, 25.460854);
  var mapOptions= {
    center:myCenter,
    zoom:17, scrollwheel: true, draggable: true,
    mapTypeId:google.maps.MapTypeId.ROADMAP
  };
  var map=new google.maps.Map(document.getElementById("googleMap"),mapOptions);

  var marker = new google.maps.Marker({
    position: myCenter,
  });
  marker.setMap(map);
}

</script>
<script src="https://maps.googleapis.com/maps/api/js?key=[redacted]callback=myMap"></script>
```

Figure 16. The source code for the map element with the personal API key removed.

3.5 Backend

Backend work on the page was focused solely on the deployment of a reservation system. Many suggestions as to the method were put forth as to methods and the consensus was made to make it simple to navigate as possible with either the use of a client-side confirmation or the use of a double confirmation where client reservations were only confirmed after being checked by an employee, due to their current use of a physical calendar to mark reservations in order to avoid overlaps.

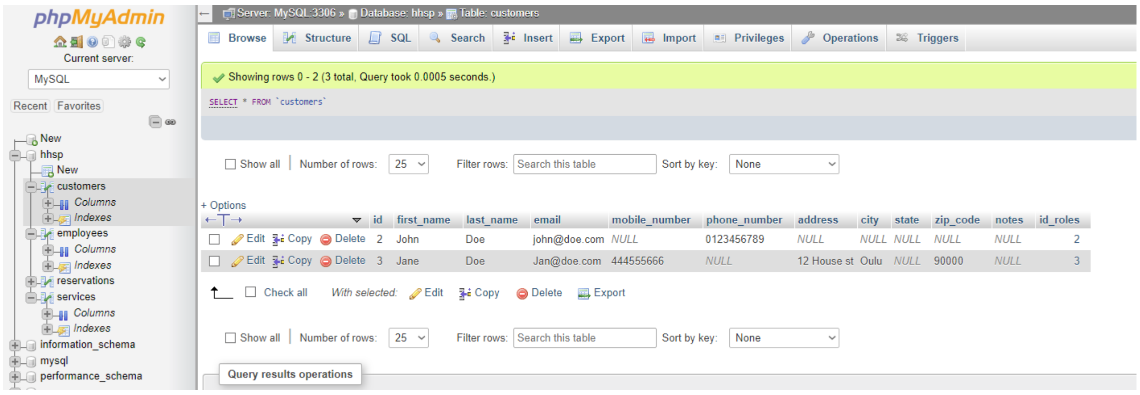
The commissioner was presented with two plans. The first featuring a model where first the day is selected, following being presented services available for the selected day. Timeslots were shown to the user with unavailable timeslots being highlighted red. The alternative was a system where a view of the current week was shown with a dropdown below each day. From the dropdown services were highlighted based on availability for that day and users would be able to select the service and time that would best suit them. The commissioner decided that displaying service first was the better alternative and thus, work began on the backend services with this information in mind.

There was some discussion about whether to use a login-based solution or have it based on database entries with a unique identifier such as a phone number being used, with the latter being adopted due to user-friendliness of the system requiring no login credentials or registration necessary, despite the possibility of malicious intent with the use of fake reservations.

3.5.1 SQL

After coming up with the proposed design for the booking system, work began on first creating the required databases. The author chose to work with MySQL as they had some previous knowledge working with and its ease of implementation into different web environments. Another factor playing a role in this decision was the use of WAMP/LAMP servers as a testbench for the web platform before final release which both feature native MySQL Support.

Using phpMyAdmin, an opensource software enabling the manipulation of databases with the help of a GUI, 4 sample tables were created for the project, namely Customers, Employees, Services and Reservations. A screenshot of these can be found below in Figure 17.



Server: MySQL:3306 » Database: hhsp » Table: customers

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

```
SELECT * FROM `customers`
```

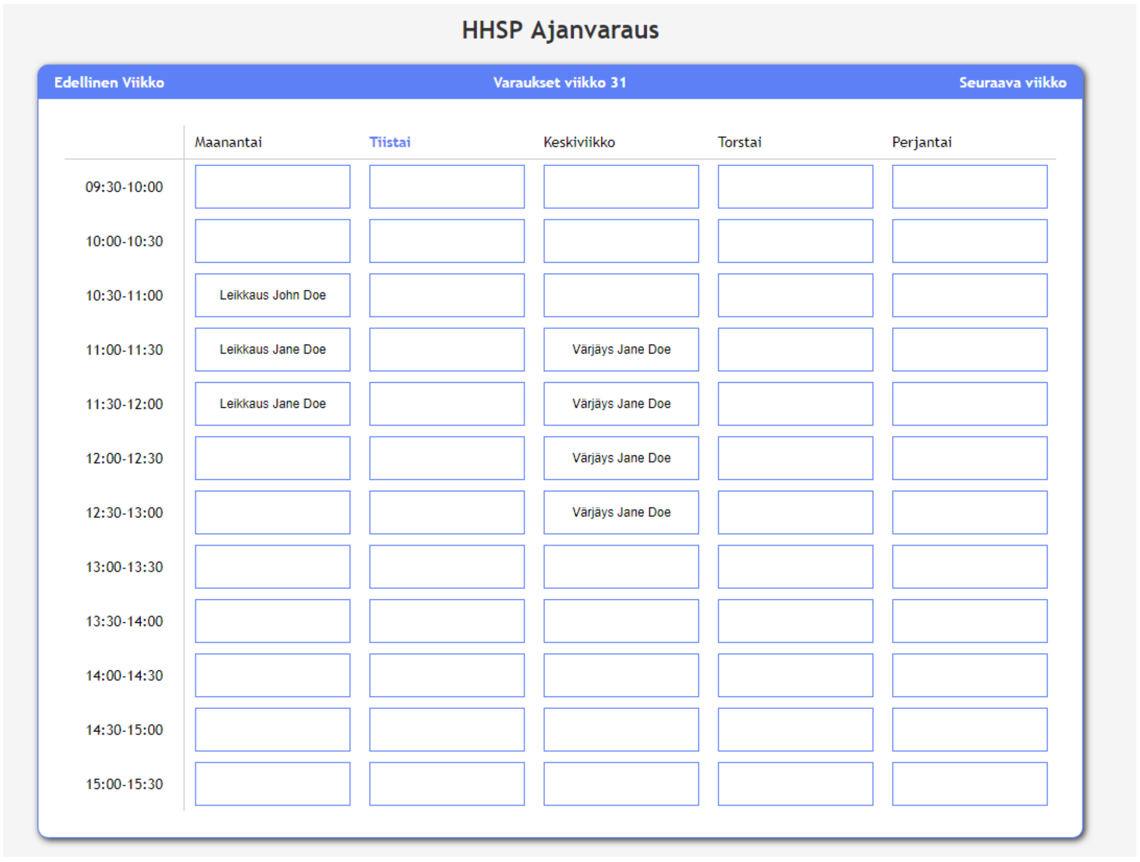
id	first_name	last_name	email	mobile_number	phone_number	address	city	state	zip_code	notes	id_roles
2	John	Doe	john@doe.com	NULL	0123456789	NULL	NULL	NULL	NULL	NULL	2
3	Jane	Doe	Jan@doe.com	444555666	NULL	12 House st	Oulu	NULL	90000	NULL	3

Figure 17. Showing the 4 tables with the Customers table open showing the rows and columns featuring a few test entries.

Upon further thought and discussion, two additional tables were added featuring service providers to link employees to services as some customers had employee preferences and by doing so allowed users to prefer employees along with services. Additionally, a confirmation table was added in order to view and track if users confirmed reservations effectively.

3.5.2 PHP

As the author had limited previous experience working with PHP, simple designs were started on featuring a calendar with boxes that users were able to click on after entering their details and selecting a service from a dropdown. As seen below in Figure 18.



HHSP Ajanvaraus

Edellinen Viikko | Varaukset viikko 31 | Seuraava viikko

	Maanantai	Tiistai	Keskiviikko	Torstai	Perjantai
09:30-10:00					
10:00-10:30					
10:30-11:00	Leikkaus John Doe				
11:00-11:30	Leikkaus Jane Doe		Värjäys Jane Doe		
11:30-12:00	Leikkaus Jane Doe		Värjäys Jane Doe		
12:00-12:30			Värjäys Jane Doe		
12:30-13:00			Värjäys Jane Doe		
13:00-13:30					
13:30-14:00					
14:00-14:30					
14:30-15:00					
15:00-15:30					

Figure 18. The first iteration featuring a calendar where users themselves had to select times.

Overall the design was primitive and not very user-friendly. Users had to select the correct time window manually and everyone's reservations were visible to everyone else raising concerns of privacy for both the business and other clients, on additional to not being very user-friendly, which was one of the primary goals.

It was decided with the commissioner that the next version would require more clarity and user-friendliness with the bare minimum being that users would not have to manually select times according to the selected service and would instead only select the start time. Based on the ideas shown earlier, work began on a system that showed a calendar view with a dropdown menu showing services highlighted green or red based on availability. From here on users would select the service provider followed by entering necessary information including an email or phone in order to confirm the reservation manually. An early version of this can be seen below in Figure 19.

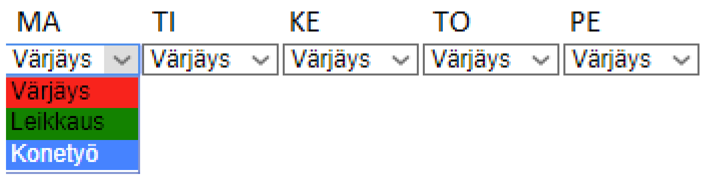


Figure 19. A very early version of the dropdown calendar model. Note how the first option does not have suitable time for the selected day. Mouseover currently shows selection as blue.

Several issues arose with this design as linking service provider, and availability showed to be rather tricky along with the overall design relying a lot on viewing outside updates for an accurate calendar.

As time was running out on the proposed deadline, the author opted to look at open-source alternatives and decided to modify a prebuilt application called Easy!Appointments (Tselegidis, 2018). This application featured an admin login with calendar view and easy to modify resources allowing for previously created elements such as the currently existed database to be imported and modified. The admin login features also proved to be helpful as management and tracking of confirmations proved to be much simpler than the previously made model.

3.6 Testing and Installation

Throughout the project, the site was tested on multiple occasions on a variety of platforms and browsers to confirm compatibility. Testing tools included Bluestacks to simulate various android environments as well as multiple different browsers in order to ascertain a similar experience to all users regardless of their choice of the browser software. Depreciated versions of browsers, however, were not tested.

3.6.1 LAMP/WAMP

Before deploying the website and services on a web hosting platform, test installations were performed and both local machines were running virtualized Microsoft Windows web servers as well as a Linux Debian based LAMP server running off of a Raspberry

Pi. General SQL server and backend functions were tested and confirmed functioning on both platforms in addition to general website functionality tests using Google Webmaster tools to confirm that no issues were present on commonly used platforms. Both setups were running the most recent stable releases of their operating system in addition to WAMP/LAMP software bundles (WampServer 3.0.6 64-bit running on Windows 10 version 1803, Linux Debian 9.4, Apache 2.4.32, MySQL 8.0.11, PHP 7.2.7).

3.7 Deployment

The website was finally deployed and published in July with the booking system initially offline. The decision to release without the booking system was caused by some employees still being on holiday, and the manager wanted them briefed on how it works and where things would show up as well as update the employee registrar for it.

Deployment was done via FTP to a third parting web hosting service featuring support for all services run within this software stack in addition to providing domain name based emails. The .fi domain name had already been previously acquired for the commissioner through a separate third-party vendor selling them on behalf of the Finnish Communications Regulatory Authority.

3.8 SEO

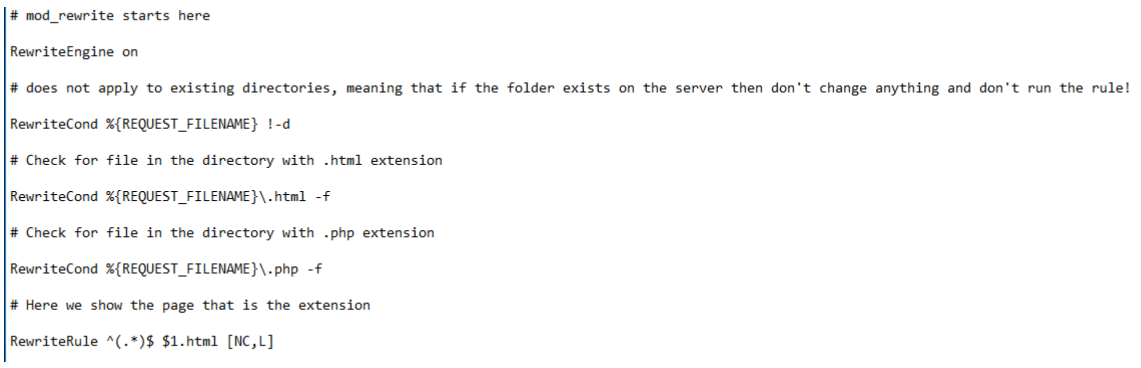
Search engine optimization is a marketing tool used to bring higher visibility and thus drive more traffic into a user's site. This is done automatically by search engines using bots called crawlers to navigate and index web pages. To simplify this, an example is website ranking on Google’s search index. Pages with more traffic and more content seen by search engine crawlers as relevant to the provided keywords show up in earlier page results. Though there are multiple ways to go about this, generally, SEO tactics can be split up into white hat and black hat methods.

White hat SEO methods are considered as techniques approved by a search engine that targets a specific audience that contains simple things such as clear, concise URLs, proper page meta tags, proper use of links, and relevant content, to name a few. Generally, these techniques cause slow natural search engine ranking growth in regards to keywords or phrases that a site might be associated with.

Black hat methods are techniques used to fool search engine crawlers into seeing the web site as more popular or relevant than it is in actuality. Techniques such as links farms, hidden text that only crawlers can see, scraping, or plagiarism or high rated content are just some of the methods used to try to climb search engine listings rapidly. Black hat methods attain ratings fast but are generally seen as a short term strategy as they feature a high risk of having the domain become delisted from search results of discovered. (Agrawal, 2015)

3.8.1 Methods

As there previously had been a sub-hosted website already that showed up already page one with the relevant keywords, the author decided to drive traffic from there by first setting up a 403 Redirect from the old site linking to the new pages' home URL. All headers were rechecked for consistency and relevancy within their meta tags, and titles were slightly modified to improve visibility when viewing from a tab. Also added to the web host were a robots.txt telling crawlers where to look and a .htaccess in order to remove the .html and .php ending from browser URLs and give it a cleaner look., found below in Figure 20.



```
# mod_rewrite starts here

RewriteEngine on

# does not apply to existing directories, meaning that if the folder exists on the server then don't change anything and don't run the rule!

RewriteCond %{REQUEST_FILENAME} !-d

# Check for file in the directory with .html extension

RewriteCond %{REQUEST_FILENAME}\.html -f

# Check for file in the directory with .php extension

RewriteCond %{REQUEST_FILENAME}\.php -f

# Here we show the page that is the extension

RewriteRule ^(.*)$ $1.html [NC,L]
```

Figure 20. The .htaccess file removing .html and .php endings from the browsers URL bar.

Google analytics were incorporated into all pages in order to monitor site traffic and easily observe how and where traffic was coming from and seeing if the 403 was working from the previous site. Using Google's Webmaster tools the site was checked for errors and crawled on both desktops and mobile versions.

4 CONCLUSION

The end goal of this thesis was to provide a new website with an integrated booking system to the commissioner while discussing the full-stack technologies and techniques used to reach this goal. To this end, the commissioner received and had deployed the web platform with all the required content. Results show that the website functions as planned and should, over time, rise to replace its previous version in search rankings due to the SEO work and 403 redirect from the previous website.

Work began slowly as the commissioner had trouble deciding and what kind of page design was wanted, and the authors limited UI/UX design experience leads to having to create multiple HTML versions, so demonstrate different page models. Tools and languages used in this thesis are among the most commonly used in all web development, and though most of the front-end work was rather straightforward revision to the author thanks to skills acquired from previous courses and work, the back-end supplied plenty of new information to learn including most of the challenges that arose during the overall work the project.

Challenges in the project mostly related to the backend provided some difficulty to the author due to limited experience, though end goals were met with the aid of web forums such as Stack Overflow. It can still be stated that they acquired valuable insight and experience when working with similar projects in the future.

References

Agrawal, H., 2015. *BLACK HAT SEO VS WHITE HAT SEO.* [Online] Available at: https://www.shoutmeloud.com/black-hat-seo-vs-white-hat-seo-understand-the-difference.html
[Accessed 2018].

AmberD, 2017. *Static HTML websites vs. Dynamic CMS.* [Online] Available at: https://www.amberddesign.com/static-html-vs-cms/
[Accessed July 28].

Aston, B., 2015. *A brief history of JavaScript.* [Online] Available at: https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17
[Accessed July 2018].

Bos, B., 2016. *A brief history of CSS until 2016.* [Online] Available at: https://www.w3.org/Style/CSS20/history.html
[Accessed July 2018].

Chris Mills, S. H. R. Z. R. D.-H. J. P., 2018. *How CSS works.* [Online] Available at: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/How_CSS_works
[Accessed July 2018].

Google, 2018. *Maps Javascript API.* [Online] Available at: https://developers.google.com/maps/documentation/javascript/tutorial
[Accessed July 2018].

Kumar, S., 2018. *Ace Web Academy.* [Online] Available at: https://www.acewebacademy.com/blog/what-is-full-stack-development-and-how-to-become-full-stack-web-developer/
[Accessed July 2018].

Laine, H., 2000. *History of SQL.* [Online] Available at: https://www.cs.helsinki.fi/u/laine/tuelip/sql_material/sql_history.html
[Accessed July 2018].

Lou, M., 2012. *Fullscreen Background Image Slideshow with CSS3.* [Online]
Available at: https://tympanus.net/codrops/2012/01/02/fullscreen-background-image-slideshow-with-css3/
[Accessed July 2018].

Mavrody, S., 2015. *CSS3 taxonomy and status.* [Art].

Microsoft, 2018. *How to determine the version, edition, and update level of SQL Server and its components.* [Online]
Available at: https://support.microsoft.com/en-in/help/321185/how-to-determine-the-version-edition-and-update-level-of-sql-server-an
[Accessed July 2018].

Mkhitaryan, A., 2017. *Building A Website Today: What You Should Know About PHP?.* [Online]
Available at: https://medium.com/sololearn/building-a-website-today-what-you-should-know-about-php-8490e41391f4
[Accessed July 2018].

Morelli, B., 2017. *JavaScript—WTF is ES6, ES8, ES 2017, ECMAScript… ?.* [Online]
Available at: https://codeburst.io/javascript-wtf-is-es6-es8-es-2017-ecmascript-dca859e4821c
[Accessed 2018].

SQL Course, 2014. *What is SQL?.* [Online]
Available at: http://www.sqlcourse.com/intro.html
[Accessed July 2018].

Steve Stein, C. G. G. M. J. F. M. U., 2017. *Primary and Foreign Key Constraints.* [Online]
Available at: https://docs.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints
[Accessed July 2018].

The PHP Group, 2004. *History of PHP.* [Online]
Available at: http://php.net/manual/en/history.php.php
[Accessed July 2018].

The PHP Group, 2018. *GPG Keys.* [Online] Available at: http://php.net/gpg-keys.php [Accessed July 2018].

Tselegidis, A., 2018. *Easy!Appointments - Open Source Appointment Scheduler.* [Online]
Available at: https://github.com/alextselegidis/easyappointments [Accessed July 2018].

W3 Consortium, 2017. *Sir Timothy Berners-Lee OM, KBE, FRS, FREng, FRSA.* [Online] Available at: https://www.w3.org/People/Berners-Lee/Longer.html [Accessed 2018].

W3 Schools, 2018. *CSS Navigation Bar.* [Online] Available at: https://www.w3schools.com/css/css_navbar.asp [Accessed July 2018].

W3 Schools, 2018. *HTML class Attribute.* [Online] Available at: https://www.w3schools.com/tags/att_global_class.asp [Accessed July 2018].

W3 Schools, 2018. *JavaScript Tutorial.* [Online] Available at: https://www.w3schools.com/js/default.asp [Accessed July 2018].

W3 Techs, 2018. *Usage of JavaScript for websites.* [Online] Available at: https://w3techs.com/technologies/details/cp-javascript/all/all [Accessed July 2018].

Wheeler, K., 2018. *Slick.* [Online] Available at: http://kenwheeler.github.io/slick/ [Accessed July 2018].

World Wide Web Consortium (W3C), 2017. *HTML 5.2 W3C Recommendation, 14 December 2017.* [Online] Available at: https://www.w3.org/TR/html5/syntax.html [Accessed July 2018].