OULU UNIVERSITY OF
APPLIED SCIENCES

Joel Muittari

MODERN WEB BACK-END

What happens in the back end of the web application?

MODERN WEB BACK-END

What happens in the back end of the web application?

Joel Muittari
Bachelor’s Thesis
Spring 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author: Joel Muittari
Title of Bachelor´s thesis: Modern Web Back-End
Supervisor: Kari Jyrkkä
Term and year of completion: Spring 2020 Number of pages: 27

This thesis has no client and is the author's, Joel Muittari's, personal take on the web back end development as a mandatory part of his studies. The objectives of the thesis were to research how the web application back-end functions, compare current top back end frameworks and design a sauna reservation application with acquired knowledge.

The author applied some background knowledge learned in school about the development. The research happened on the Internet with Google and online library searches. A wide variety of articles and other sources were used as materials.

The main results are a description of how the back-end works, a comparison of frameworks and the implementation design for the sauna reservation application. The implementation design for the sauna reservation application had Express chosen as the framework and PostgreSQL as the database. The development of the web application designed in this thesis could take place as a further measure for this thesis.

Keywords:

web development, back-end development, back-end framework

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

CLI	Command Line Interface
CoC	Convention over Configuration
CSV	Comma-separated values
CSS	Cascading style sheets
DRY	Don’t repeat yourself
HTML	Hypertext markup language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MB	Megabyte
MEAN	MongoDB, Express.js, Angular.js and Node.js
MVC	Model-view-controller architecture
NPM	Node Package Manager
ORM	Object-Relational Mapper
PaaS	Platform as a Service
PHP	PHP: Hypertext Preprocessor
RoR	Ruby on Rails
UI	User interface
URL	Uniform Resource Locator
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language

1 INTRODUCTION

The topic of this thesis is the development of the web application's back-end. The web application is a sauna reservation system. There is no client for this project, but the object is that the author will learn more and improve himself as a web developer.

This thesis is going to cover the selection, comparison, and implementation of the back-end framework and the database. The sauna service application is going to be deployed to the webserver and this thesis also covers that process. The application is going to have a server-side that handles all the business logic and connects to the database where all the information about users, saunas, and reservations are stored. The server-side will serve all the information to the client-side where the user interacts with the application through the user interface.

The sauna reservation service application is going to have 3 different saunas in different locations that users can reserve for their private use an hour or multiple hours at the time. Users can log in to the system through the website and see, edit or cancel their reservation. Admin users can add and remove saunas and create, edit and cancel normal users' reservations. Also, admins can edit and remove users.

2 ARCHITECTURE OF WEB APPLICATION

Web application constructs from two parts: server-side and client-side. The front-end, or client-side, is the user interface on the website that users interact with. It is created with JavaScript, HTML, and CSS. The only front-end is needed for the application for it to function but then all the content would be static and same for everyone who visits the site. The back-end, or server-side, is where the magic happens. The back-end can be used to dynamically change the website for the individual users depending on their habits or preferences. It can also be used to interact with users through notifications or emails.

The web browser uses HTTP requests to communicate with the server-side when the user navigates on the web pages. These requests have a URL to identify the affected resource, a required action method, and may have additional information as POST method data, encoded as URL parameters or in related cookies. The server will, after processing the request, send an HTTP response that contains a status line telling whether the request succeeded, and successful requests will have response resources. A resource can be in example data, a new web page or an image. (1)

The required action method can be GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE or PATCH. The GET method requests a specific resource and should only return data. The HEAD is identical to a GET but without the response body. The POST method is used to submit data and often causes changes on the server. The PUT method is used to create a resource or overwrite it and a PUT is idempotent. The DELETE method is used to delete the specified resource. The CONNECT method creates a tunnel to the specified target server. The OPTIONS method describes the available communication options for the target. The TRACE method is used to perform a message loop-back testing along the path to the target. The PATCH method applies partial modification to the target resource. (2)

2.1 Static website

A static website is a basic website that always shows the same hard-coded content for the user. The web browser sends an HTTP “GET” request to retrieve the new page from the server whenever the user navigates on the site. The server will respond with the static document and success status. (See FIGURE 1). The response can also be an error if the file cannot be reached for whatever reason. Then the response will contain an error status. (1)

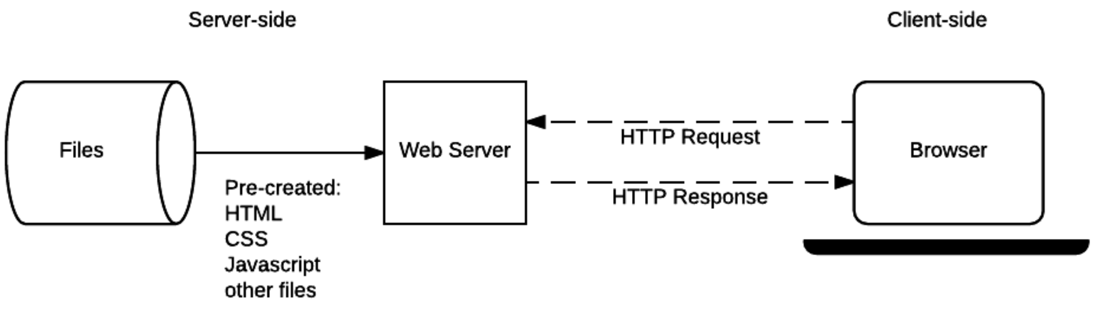

FIGURE 1. The architecture of a static site (3)

2.2 Dynamic website

On the dynamic website, the back-end creates the content when needed and it can be customized for the users or by the stored preferences. Normally, HTML pages on the dynamic website are created with HTML templates by inserting data into placeholders from the database. The back-end can also execute operations, such as sending notifications, when returning a response to the front-end.

See FIGURE 2 for a simple architect of a dynamic website. As in FIGURE 1 the browser sends HTTP requests to the server and the server responds appropriately with an HTTP response. Requests for static resources, which are CSS, JavaScript, images, PDFs, videos and other pre-created files, are handled just like in the static website. (1)

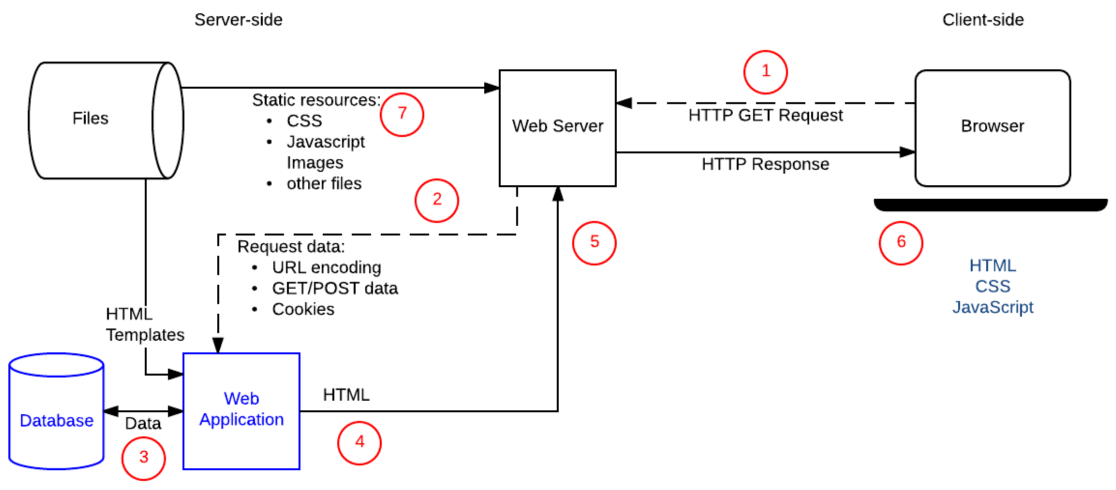

FIGURE 2. Basic architect of the dynamic website (4)

Requests for dynamic resources are forwarded to the server-side web application (see FIGURE 2 number 2). The application will read from or write to the database (see FIGURE 2 number 3) and process the HTML template by putting data into placeholders (see FIGURE 2 number 4) and then send the response containing the generated HTML to the client (see FIGURE 2 number 5). The browser will receive and process the response and send separate requests for CSS, JS or other files referenced on the HTML page (see FIGURE 2 number 6). The browser will get the static files directly from the web server based on configurations and URL pattern matching (see FIGURE 2 number 7). (1)

The sauna reservation application, which is being developed as part of this thesis, is a dynamic website. The server-side will tell the client-side to show the user what times are available for reservation and what times are reserved. Users are also able to see their reservations and modify them. The content of the website also changes when the admin adds new saunas to the system. Some pages, e.g. about page and pricing page, can be static. Images are always static resources.

2.3 Framework services

Writing code, handling server-side operations, maintaining and scaling the application are much easier to do with the structure that server-side frameworks, or in other words web application frameworks, provide for the developer. Mapping URLs for all the resources and pages is one of the most important tasks to be done on the server-side. Web frameworks provide simple mechanisms to map URLs to specific handler functions. They make it easy to define and maintain routing because the URL can be changed, but there is no need to change the handler function. Frameworks also provide tools and libraries for other common server-side tasks, such as database interactions, session handling, user authorization, security against malicious attacks and output formatting. Common output formats are CSV, HTML, JSON and XML.

As described in the previous chapter, the web browser communicates with a web server through the HTTP protocol: the server is waiting for HTTP requests from the browser and when it finally receives one, it will process it and return the needed information in an HTTP response. The web framework will make this task easier for the user by allowing them to write simplified code to interact with higher-level code: requests and responses instead of lower-level network primitives.
Web frameworks make it easy to access data in the HTTP request. This data can be encoded in many ways. An HTTP GET request can encode the required data in URL parameters or inside the URL structure. An HTTP POST request encodes data as POST data inside the body of the HTTP requests. Information about the current session or the user can also be included in the HTTP request as a client-side cookie. Web frameworks provide useful mechanisms to access information in the HTTP request. A request object contains methods and properties for accessing the target URL, the request type, GET or POST parameters, cookie and session data.

Most of the web frameworks offer templating systems. These systems allow to create a template for the output document: specifying the structure and using placeholders to add data when generating a page. Usually, the templates are used to generate HTML pages, but also other types of pages can be created. Often easy mechanisms to create other formats of data, e.g. JSON and XML, are provided inside the frameworks.

2.3.1 Database access

Web applications store a lot of data in the database: both to be shared with users, and information about the users. Most of the web frameworks provide a database layer for communication with the database. It abstracts read, write, query, delete and other database operations. This abstraction layer is called an Object-Relational Mapper or an ORM as short.

There are two benefits of using an ORM:

- The database can be changed, and the code does not necessarily need any changes. This allows easier optimization for a characteristic of a different database based on their usage.
- Implementations to validate the data can be done inside the framework. This way it is easier and safer to check that the data is in the correct format and goes to the correct type of field in the database. For example, an email address goes to the field for the email address. Validation can also protect from malicious data. Attackers can use certain patterns of code to delete records in the database or other malicious things.

Often, when framework provides an ORM, a framework can also provide models. Models are objects used to define the structure of the database records. It can specify the field types to be stored and provide some validation for the data. The field specifications can also set the maximum size, default values, selection list options, comments for documentation and label text for forms. The model does not affect the underlying database code which can be configurated and changed separately. (5)

3 BACK-END FRAMEWORKS

The digital transformation of businesses all over the world is going on and it has led to a great innovation of mobile and web applications. Businesses are spending huge amounts of money to impress the user with easy and addictive front-end experiences. But we should not forget that in all applications there is always the back-end, which is ineffective without an intelligent framework.

This chapter is going to look at the top frameworks, compare them and decide which one of them is the most suitable for this project. It is difficult to determine which framework would be the best because frameworks are created with different languages and with different intentions in mind, so it will be focused on finding a fitting framework. The application back-end has numerous tasks to be done and a good server-side framework makes solving these tasks convenient and hassle-free. An advanced framework will fast track the development with preconfigured tools and templates. It should not be limited to providing a structure or a methodology, but it should rather give freedom to build interoperable applications that can shoulder the scaling workload. (6)

There can be found endless amounts of frameworks in the world on different programming languages and many of them can help to develop a web application. It is important to choose the right technological stack for successful development. The selection of the most suitable back-end framework can be done with the following criteria:

- **Learning curve**: Some frameworks use strict naming conventions and structure rules for the project folders that the developer must understand to be able to avoid run-time errors over the smallest inconsistency.
- **Core Library**: Usually, web back-end frameworks provide a library that is meant to prevent developers from doing repetitive tasks. Some developers might want to choose a more flexible framework to have more control to manipulate the existing library elements.
- **Vulnerability**: A large community around the framework can take care of bugs and patches and keep fellow developers informed. Also, somebody might already have solved the problem you are facing. A still framework is an easier target for hackers.
- **Documentation**: Good documentation is vital for an early adaptation by power users. A wide variety of code snippets, examples and tutorial articles can help even the small and young framework to stand out from the others. (7)

- **Productivity**: This is the measure of how fast developers can create new features after getting familiar with the framework. It is impossible to write new features if the old ones are broken so both the effort to write and maintain should be considered when thinking about productivity. Productivity is like the learning curve: many of the factors affect the learning curve of the framework and here are some examples: documentation, community, programming experience, included helper libraries, and if the framework encourages good practices.
- **Performance of the framework and programming language**: Usually speed is not the largest factor when selecting the framework because even the programming language Python, which has a slow runtime, is more than good enough for medium web applications on moderate hardware. The speed benefits of faster language, such as C++, can be offset by the costs of learning and maintenance.
- **Accessibility**: Most frameworks are free, and they must be free for a wide audience of developers to use them. Also, the size of the framework is something to think about. It usually does not matter much but the huge framework can slow the application if advantage is not taken of its services. (5)

3.1 Comparison of frameworks

This part is the comparison of back-end frameworks: CodeIgniter, Django, Express.js, Flask, Laravel, Spring Boot and Ruby on Rails. These are some widely used back-end frameworks and all of them are free for developers to use.

3.1.1 CodeIgniter

CodeIgniter is an MVC framework created with PHP in 2006 and it is used for rapid application development. It provides a set of helper libraries for a database connection and various other operations. For example, sending emails, uploading files to the server and managing the user sessions. (8) CodeIgniter is designed to enable fast development of projects and minimizing the amount of code needed for the tasks. CodeIgniter is open source and it has 1.7k users watching, 17.9k stars and 7.8k forks in GitHub. (9)

CodeIgniter is a small framework. The whole source code is only 2MB and this is one of the advantages of it. The small size also makes it easier to master how the framework works and this simplifies the deployment and updating. CodeIgniter uses the Model-View-Controller architectural design as all other frameworks talked about in this thesis. The MVC pattern is the industry standard. CodeIgniter also has a good documentation online, a short learning curve and it is easily extendable with libraries, helpers or packages. (8)

Popular sites that have been developed using CodeIgniter include Casio Computers, Feedcump, The Mail & Guardian, Buffer, Nissan, Creditflux, McClatchy and Bonfire. (10)

3.1.2 Django

Django is an open-source framework created with one of the fastest-growing programming languages: Python. Django is widely used as a framework for agile development of APIs and high-end back-end applications. The framework is named after the famous guitarist, Django Reinhardt, and was created in 2003. Django supports quick development of the web applications with only very little of actual coding work. (6) Django has 2.2k users watching, 46.9k starts and 20.2k forks in GitHub. (11)

Django also provides some other good features for developers who use it in development. It provides security from malicious attacks with a stateless mechanism for authentication. Django can identify XML and JSON files for the user, which helps in managing a relational database efficiently. This framework is based on the “share nothing” architecture which allows new hardware being added by developers. Also, Django has large libraries included so it has needed tools for many different applications needs. (6)

Popular sites that have been developed using Django include Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest and Open stack. (5)

3.1.3 Flask

Flask is another widely used framework created with Python. It was created in 2004 by a Python developer team led by Armin Ronacher. Flask is incredibly lightweight, and it is based on the Werkzeug library and the Jinja 2 templating engine. They allow developers to build customized and scalable applications. (6) Flask has 2.3k users watching, 48.7k stars and 13.2k forks in GitHub. (12)

Flask is a WSGI (Short from Web Server Gateway Interface) compatible framework. The WSGI is a web server that forwards requests to web applications and frameworks in the Python language. Flask is quick to set up because it is lightweight and it lets developers choose the dependencies that they want to be included. This makes Flask very flexible. The framework has a built-in development server and debugger which makes the development and solving bugs faster. Flask is completely compatible with Google App Engine, provides a complete support for unit testing that allows testing production code, and offers detailed documentation which allows the plugging of any extension developers want. (6)

Flask is popular for developers who need to provide web services in restricted systems, such as Raspberry Pi or Drone controllers. (5) Popular sites that have reported using Flask in development include Netflix, Reddit, Lyft, Uber, MIT, Banksland, Zalando, Zillow and Keen IO. (13)

3.1.4 Express

Express is an open-source framework and one of the top frameworks running on Node.js. This framework is very non-restrictive on its nature and it is used widely by developers building APIs. Express was released in 2010 and it helps on creating web back-ends and mobile applications with a minimalist approach. It is an important part of the MEAN Stack development. (6) The MEAN stack is a software bundle used in web development and it consists of MongoDB, Express.js, Angular.js and Node.js. When using the MEAN stack the whole application, server-side and client-side, is written with JavaScript. (14) Also, many other server-side and full-stack frameworks are based on Express, including Feathers, ItemsAPI, KeystoneJS, Kraken, LoopBack and Sails. (5)

Express offers a wide variety of HTTP utility methods to build dynamic APIs. It offers routing functions based on URLs and HTTP methods. Preconfigured plug-ins and template engines significantly reduce the development time and testing supported. The framework also offers a quick execution of engines that enable for developers to create applications in comparatively less time.

Express has 1.8k users watching, 47.1k stars and 7.9k forks in GitHub. (15) Popular sites that are using Express in their development include Accenture, Exove, IBM, MuleSoft, Uber, Yandex, Hasura and Kuali. (16)

3.1.5 Laravel Lumen

Laravel is a PHP framework that is widely used on building big web applications. It offers a wide variety of web building components as it is a fully featured back-end framework.The Laravel Lumen framework is a Laravel based light-weight framework which allows developers to build microservices and well-performing APIs. It is more efficient on handling a lot of requests than the Laravel framework.

Laravel Lumen was released in 2015 and it minimizes the bootstrapping and offers a lot of flexibility. It is possible to quickly upgrade it to the Laravel if more functionality is needed. Lumen includes a Fast Route Library that can quickly send requests to appropriate handle functions to process. Lumen uses a user token-based system to authenticate multiple users at the same time. Also, it supports caching, error, and log-handling as well as Laravel. (6)

Laravel Lumen has 351 users watching, 6.6k starts and 883 forks in GitHub. (17) These numbers are drastically smaller than other frameworks described in this thesis. One reason for this can be that this framework is a more light-weight version, like a little brother, for the well-established and widely used Laravel framework which has 4.6k users watching, 57.4k stars and 17.8k forks in GitHub. (18) Popular sites that are using Lumen in their development include Geocodio, RatePAY Gm, Next, Autotrader, Fixico and PathMotion. (19)

3.1.6 Spring Boot Framework

Spring Boot is a Spring-based framework for developing production-grade back-end web applications in the Java programming language. This framework is microservices-ready, which allows a proper request handling across devices and platforms. It was initially released in 2002. (6) Its strength is in creating larger-scale applications with the cloud approach but it can solve smaller problems, too. It allows multiple applications running parallel to communicate with each other. Some of these can be providing a user interface and others an accessing database or other back-end work. (5)

Spring Boot runs automatically configurated libraries in Spring and also the third-party back-end frameworks. This framework comes with the Spring Boot CLI (short for Command Line Interface) which allows writing code with a simplified programming language called Groovy. Spring Boot does not require code generation nor the XML configuration. (6)

Spring Boot has 3.4k users watching, 45.2k stars and 28.6k forks in GitHub. (20) Popular companies that have reported using Spring Boot in their development include MIT, Intuit, OpenGov, Hepsiburada, PedidosYa, TransferWise and Trivago. (21)

3.1.7 Ruby on Rails

Ruby on Rails, or RoR in short, was released initially in 2004. It is written in the Ruby programming language which is considered as one of the most developer-friendly languages and this framework combines the capabilities of the language. RoR supports very quick development and offers out of the box almost all the components that a developer would be looking for in an advanced framework. (6) The design philosophy is very similar to the Django framework. (5)

RoR supports an approach called CoC (short for Convention over Configuration) which boosts flexibility when performing back-end tasks. Also, RoR supports the DRY (short for Don’t Repeat Yourself) approach which helps on building lightweight apps. This framework has detailed error-logs, support for search engine friendly URL development, and hassle-free integration of data handling libraries. RoR makes the development of the interactions with a third-party object easier by allowing each object to be able to possess unique attributes. (6)

RoR has 2.6k users watching in GitHub, 45k starts and 18.2k forks in GitHub. (22) Popular companies that have reported using RoR in their development include Airbnb, Ask.fm, Bloomberg, Dribbble.com, GitHub, Fiverr, Yellow Pages, Twitch, SoundCloud, Shopify, Basecamp and Cookpad. (23)

3.2 Speed Benchmarks

This part handles Web Framework Performance Benchmarks Round 18 (9.7.2019) by TechEmpower. They have tested 201 different web application platforms, full-stack frameworks and micro-frameworks including the ones that are being compared in this thesis: CodeIgniter, Django, Express, Flask, Laravel, Lumen, Spring Boot and Ruby on Rails. These tests are made with physical servers with realistic Linux setups. The test types are JSON serialization, Single query, Multiple queries, Fortunes, Data updates and Plaintext. The single query test is maybe the most important test for this project to look at because the application is going to be mostly creating and updating reservations or users. The speed is not a very important criterion for the selection of framework in this project.

3.2.1 Single query

This test was about fetching a single row from a simple database. The fetched row is then serialized as a JSON response. Performance column is the best database-access responses per second, a single query from 28 tests. Higher performance is better.

TABLE 1. Web Framework Performance Benchmarks Round 18 Single query test (24)

Rank	*Framework*	*Performance*	*%*	*Errors*	*Database*	*ORM classification*
1	spring-webflux-pgclient	139,334	100	0	Postgres	Micro
2	spring-webflux-jdbc	126,695	90.9	0	Postgres	Micro
3	spring-mongo	83,026	59.6	112	MongoDB	Full
4	spring	76,780	55.1	252	Postgres	Micro
5	spring-webflux-mongo	72,145	51.8	0	MongoDB	Full
6	spring-webflux-rxjdbc	68,543	49.2	0	Postgres	Micro

7	express-mysql	67,853	48.7	0	MySQL	Full
8	express-mongodb	46,786	33.6	0	MongoDB	Full
9	lumen-swoole	40,423	29.0	0	MySQL	Full
10	flask-raw	37,184	26.7	0	MySQL	Raw
11	laravel-swoole	29,744	21.3	0	MySQL	Full
12	django-py3	24,056	17.3	0	MySQL	Full
13	django-postgresql	20,421	14.7	0	Postgres	Full
14	django	19,409	13.9	0	MySQL	Full
15	flask-pypy2-raw	16,274	11.7	0	MySQL	Raw
16	flask	15,526	11.1	0	MySQL	Full
17	rails	13,057	9,4	0	MySQL	Full
18	rails-unicorn	12,789	9.2	0	MySQL	Full
19	express-postgres	11,355	8.1	0	Postgres	Full
20	lumen	9,407	6.8	0	MySQL	Full
21	flask-nginx-uwsgi	7,265	5.2	0	MySQL	Full
22	flask-pypy2	6,665	4.8	0	MySQL	Full
23	laravel	6,023	4.3	0	MySQL	Full
24	express-graphql-postgres	4,713	3.4	0	Postgres	Full
25	express-graphql-mysql	3,047	2.2	0	MySQL	Full
26	express-graphql-mongodb	2,241	1.6	0	MongoDB	Full
27	codeigniter	1,896	1.4	34,464	MySQL	Raw
28	spring-webflux	723	0.5	25,006	Postgres	Micro

3.3 Conclusions

Learning curve and productivity are hard to define. These frameworks are very similar as seen in TABLE 2 and there are no noticeable differences except the language that it has been created with. The biggest differences between frameworks are the abilities of the programming language that they are created with. On this project, the programming language is the most important criterion because of the skills of the developer and the author. The author has experience in developing web applications with JS and PHP and a bit with Python. The author has been using PHP and Python as back-end languages and has selected Express as the framework for this project because there is a benefit from learning it for the professional career as a web developer. Most of the web back-ends are built with PHP but the part of JS as the backend language is growing in popularity.

TABLE 2. Framework conclusions

	CodeIgniter	*Django*	*Flask*	*Express*	*Lumen*	*Spring*	*RoR*
Language	PHP	Python	Python	JS	PHP	Java	Ruby
Stars	17.9k	46.9k	48.7k	47.1k	6.6k	45.2k	45k
Users watching	1.7k	2.2k	2.3k	1.8k	351	3.4k	2.6k
Forks	7.8k	20.2k	13.2k	7.9k	883	28.6k	18.2k
Size	2 MB	8.6 MB	Not found	Not found	Not found	2 MB	57.8 MB
Speed	Slow	Medium	Medium	Fast	Medium	Very fast	Medium
Price	Free	Free	Free	Free	Free	Free	Free

4 THE APPLICATION IMPLEMENTATION

As mentioned in the previous chapter, Express was chosen to be the framework for this application. To develop the application with Express, Node and NPM are required to make the application work. Node is an open source, cross-platform JS runtime environment that allows JS to be run on the server-side of the application. Companies that use Node in their development include GoDaddy, LinkedIn, Microsoft, Netflix and PayPal. (25) NPM is the Node Package Manager which is used to manage all the libraries, packages and their dependencies in the application development. (26) The application is hosted in Heroku which is a cloud platform as a Service (PaaS). Heroku offers free hosting for small applications so it is very convenient for the project of this type. Heroku also offers PostgreSQL database hosting for a small database for free. (27) The front-end of an application that is created with Express can be done with the static HTML or Express also supports out-of-the-box many different templating engines, such as Pug, Blade and React. (28) Pug is widely used with Express and it was, therefore chosen for this application.

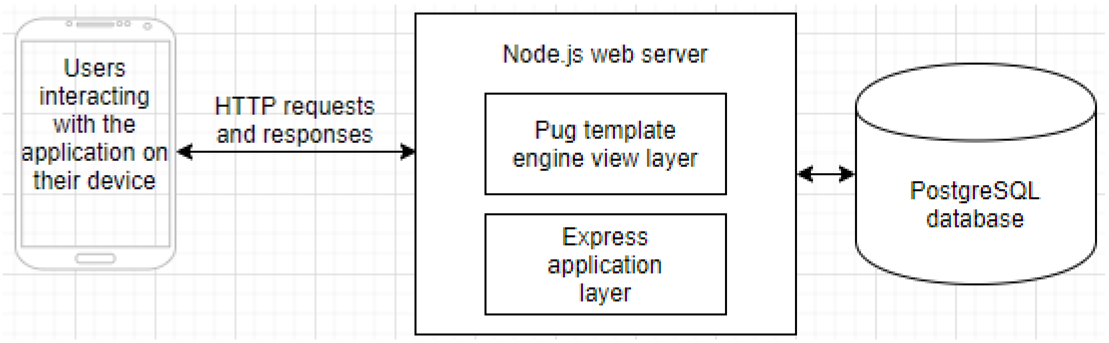
Users interacting with the application on their device
HTTP requests and responses
Node.js web server
Pug template engine view layer
Express application layer
PostgreSQL database

FIGURE 3. The application architecture diagram

FIGURE 3 shows the architecture of the sauna reservation application. The users (administrators and customers) interact with the application with their computers or mobile devices. The client device makes an HTTP request to the web server where the application layer handles the request. The postgreSQL database is accessed if needed. The view layer will generate the needed web page for the HTTP response.

The use case diagram for the sauna reservation application can be seen in FIGURE 4. The customers can browse the available saunas listed in the system. They must authenticate to be able to make a reservation and after making the reservation, they can make the payment. The customer is also able to edit or cancel the reservation that they have made.

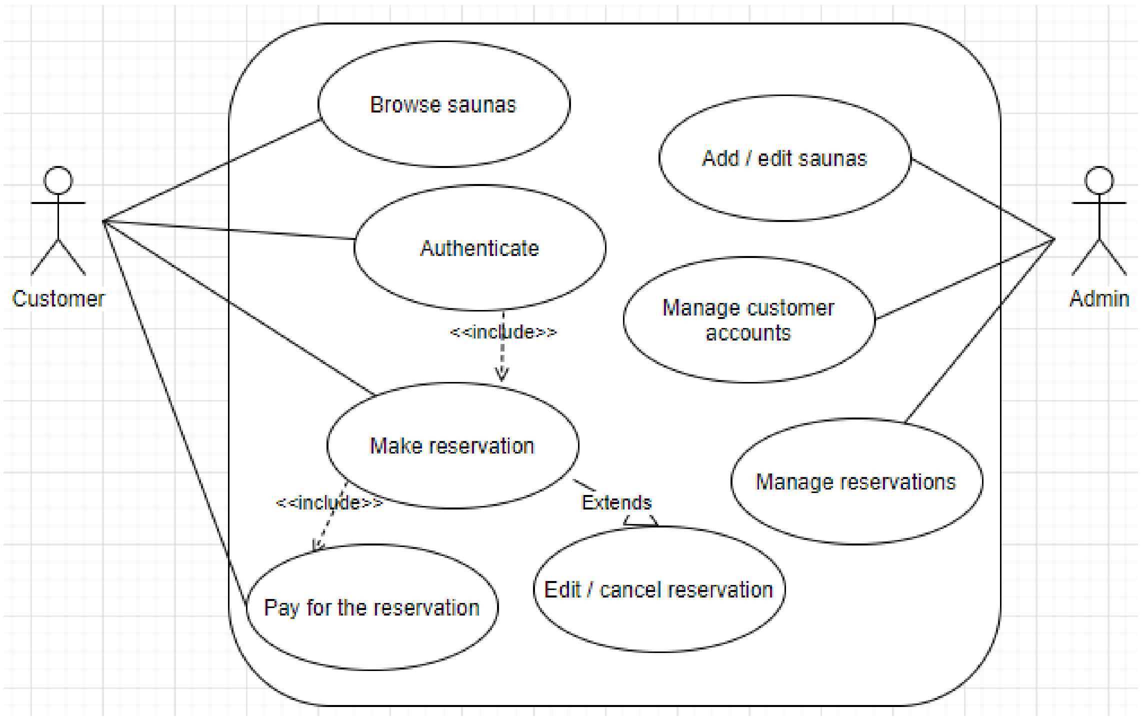

FIGURE 4. The use case diagram

As seen in FIGURE 4, the system administrators of the sauna reservation application can add and edit saunas, manage customer accounts and reservations. They can suspend accounts and cancel reservations if customers' behaviours show the need for that.

4.1 Database

PostgreSQL was chosen as the database for this project because Heroku, the cloud platform where this application is hosted, also offers free database hosting. It is easy for the development and queries are faster when everything is on the same host.

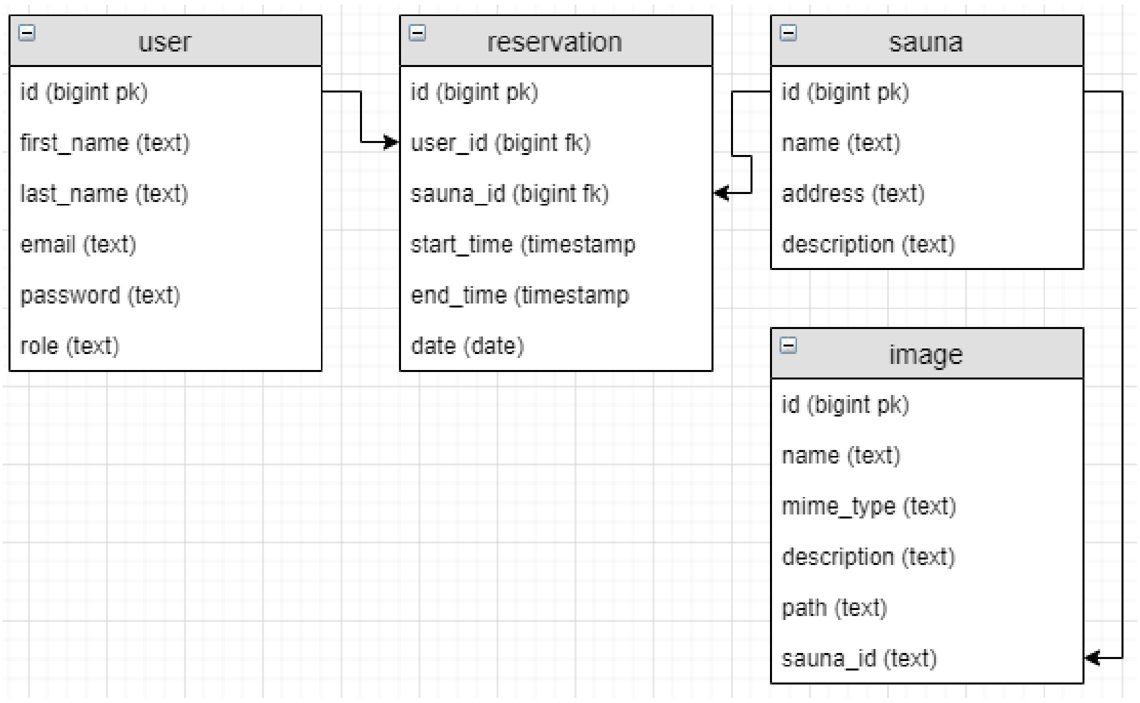

FIGURE 5. The database diagram

The plan for the database is to keep it simple as seen in FIGURE 5. There is a table for users that holds their name, email, password and role. The role row allows normal users and admins to be in the same table and they can have different permissions inside the applications. The sauna table has simple information about the sauna. The reservation table holds the information about each sauna reservation that users make. This table is linked with a foreign key to both the user table and the sauna table. And the last table in FIGURE 5 is the image table. It has the information about images in the application: their name, mime type, the path to the image on the server and the description that can be shown next to the image for the user. Also, there is a row for the sauna ID so the images can be linked to correct saunas.

4.2 Authentication system

The authentication system for this application is going to be built with Passport. Passport is middleware for Node.js that enables flexible authentication for web applications. When using Passport, the user can authenticate by using e.g. a username and a password, Facebook, Twitter, Google, LinkedIn, GitHub.. (29) In this application, the user can log in using an email address and a password, Facebook login or Google login. When a new user is created, the password will be stored into the database. For the security reasons the password is hashed first using salt with the bcrypt.js package and then the hash, not the user’s plaintext input, is stored into the database. Bcrypt.js is the node.js package that generates 60-character hashed strings from passwords and protects the application from the rainbow table and brute-force attacks. (30)

5 CONCLUSIONS

The objective of this thesis was to find out how the web application back end works and then find the best backend framework by researching and comparing for the sauna reservation application and designing, developing and deploying the application. During this thesis, the research of how the back end works was successful as was the comparison of the frameworks. The part of designing, developing and deploying the application became narrowed to design the application and the development was left out of the thesis and for the future.

Initially, this application was planned to be developed in cooperation with another thesis student and to be related to that thesis. This student would have developed the front end for this application, but that person decided to write about a different topic. At this point, the plan became to be to try to make a very basic front end for the application which would still allow the deployment. The development of the applications was later left out of the scope of this thesis. The researching part took a lot more time than initially expected which contribute to the decision of leaving out the development part. Instead of trying to scrap together in hurry a barely functional application, it would be better to push out the release date and finish the development. This applies well to the real world but in the world of thesis creation which cannot be extended forever, it was easier to just plan the application implementation properly and leave out the development part. The implementation design covers the whole application except how the payment method would be implemented. This was left for the future when it is needed as a functionality. Payment could also be done outside the application if the business owner would want that.

Software development is easier when the scope, resources and workload are well planned with real deadlines. This also applies to other parts of life: well-planned is half done. The learning during this thesis happened in the planning part when realising that developing a small web application is not that simple. Also, now the author has more knowledge about the functionality of the web back end and development options for it. The easy way to continue this thesis in the future would be to develop the application with the planning of the implementation done during this thesis. This could be a great application for someone who wants to start a sauna business. This type of application would be good to be accessible with computers using web browsers but also with mobile devices through the web browser or even a native mobile app. A native app can be a possible further measure if the business owner sees it as needed. It could make the application more accessible to normal people.

REFERENCES

1. MDN web docs, Introduction to the server-side. Cited 13.10.2019, https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction
2. MDN web docs, HTTP request methods. Citied 9.11.2019, https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
3. The architecture of the static site. Citied 1.11.2019, https://mdn.mozillademos.org/files/13841/Basic Static App Server.png
4. Basic architect of a dynamic website. Citied 4.11.2019, https://mdn.mozillademos.org/files/13839/Web%20Application%20with%20HTML%20and%20Steps.png
5. MDN web docs, Server-side web frameworks. Citied 21.11.2019, https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks
6. Kellton Tech. Top 7 Backend Web Development Frameworks 2019. Cited 14.9.2019, https://www.kelltontech.com/kellton-tech-blog/top-7-backend-web-development-frameworks-2019
7. Kellton Tech. Top 7 Backend Web Development Frameworks 2018. Cited 27.11.2019, https://www.kelltontech.com/kellton-tech-blog/top-7-backend-web-development-frameworks-2018
8. Guru99, What is CodeIgniter? How does it work? Cited 27.1.2020, https://www.guru99.com/what-is-codeigniter.html
9. GitHub, bcit-ci/ CodeIgniter. Cited 27.1.2020, https://github.com/bcit-ci/CodeIgniter
10. Maangatech, 8 Popular Websites and Apps Using CodeIgniter. Citied 31.1.2020, https://maangatech.com/popular-websites-and-apps-using-codeigniter/
11. GitHub, Django/Django. Cited 31.1.2020, https://github.com/django/django
12. GitHub, pallets/flask. Cited 31.1.2020, https://github.com/pallets/flask
13. Stackshare, Flask. Cited 1.2.2020, https://stackshare.io/flask
14. Wikipedia, MEAN (software bundle). Cited 1.2.2020, https://en.wikipedia.org/wiki/MEAN_(software_bundle)
15. GitHub, expressjs/express. Cited 1.2.2020, https://github.com/expressjs/express
16. Express, Companies using Express in production. Cited 1.2.2020, https://expressjs.com/en/resources/companies-using-express.html
17. GitHub, laravel/lumen. Cited 3.2.202, https://github.com/laravel/lumen

18. GitHub, laravel/laravel. Cited 3.2.2020, https://github.com/laravel/laravel
19. Stackshare, Lumen. Cited 3.2.2020, https://stackshare.io/lumen
20. GitHub, sping-projects/spring-boot. Cited 3.2.2020, https://github.com/spring-projects/spring-boot
21. Stackshare, Spring Boot. Cited 3.2.2020, https://stackshare.io/spring-boot
22. GitHub, rails/rails. Cited 5.2.2020, https://github.com/rails/rails
23. Netguru, Top 34 Companies Using Ruby on Rails in 2019. Cited 5.2.2020, https://www.netguru.com/blog/top-34-web-apps-built-with-ruby-on-rails
24. Tech Empower, Web framework benchmarks round 18. Cited 13.10.2019, https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=json&f=zik06f-zik073-zik0zj-zik0zj-zijbpb-zik0zj-zik0zj-zik0zj-yelngf-e7
25. Wikipedia, Node.js. Cited 28.3.2020, https://en.wikipedia.org/wiki/Node.js
26. Wikipedia, npm (software). Cited 28.3.2020, https://en.wikipedia.org/wiki/Npm_(software)
27. Wikipedia, Heroku. Cited 28.3.2020, https://en.wikipedia.org/wiki/Heroku
28. Express, Template engines, Cited 9.4.2020, https://expressjs.com/en/resources/template-engines.html
29. Passport. Cited 9.4.2020, http://www.passportjs.org/
30. npm, bcryptjs. Cite 9.4.2020, https://www.npmjs.com/package/bcryptjs