GARAGE WEB SYSTEM – FINAL PROJECT

By Fabio Marques Pimentel

Higher Diploma in Computer Science

CCT Dublin College

Dublin – Ireland

15th Aug 2020

# ACKNOWLEDGEMENTS

I would like to begin by expressing my sincerely thanks to all CCT College staff and lecture which has participated and helped me, on the way, to increase my skills and knowledge.

Thanks to give enough support to face all difficulties and be able to go through them.

In addition, I would like to thank to my study group, and classmates who has inspired me to keep going and growing. Thanks to the people which has been patience.

And at last, but not less important, I would like to mention specially my grandmother which passed away on December last year. She was a very important person.

Thanks to those people who are always by my side giving me the needed support during this period. I do appreciate all what them have been made. All of you are very special and owns my gratitude. Those people are the most important pillars of my life and career: my parents, friends, family and girlfriend.

Many thanks!!!

TABLE OF CONTENTS

[ACKNOWLEDGEMENTS 2](#_Toc48459017)

[ABSTRACT 6](#_Toc48459018)

[CHAPTER 1: INTRODUCTION 7](#_Toc48459019)

[PROJECT CONTEXT 7](#_Toc48459020)

[Main goal: 8](#_Toc48459021)

[Objectives: 8](#_Toc48459022)

[Areas to cover: 8](#_Toc48459023)

[ Java 8, object oriented language; 9](#_Toc48459024)

[ Java Persistence API, as Java software component to map the database using annotations, and manage queries on the DB; 9](#_Toc48459025)

[ Spring Boot as a Java framework in order to create web services; 9](#_Toc48459026)

[ Tomcat as HTTP web server provider; 9](#_Toc48459027)

[ Gradle, an open-source build automation system based on Apache Ant and Apache Maven; 9](#_Toc48459028)

[ Postman, a tool used to test RESTful APIs; 9](#_Toc48459029)

[ Ionic Framework to build desktop apps using web technologies integrated to Angular. 9](#_Toc48459030)

[ In terms of the project itself, the following areas will be covered: 9](#_Toc48459031)

[ SCRUM as Project Management Agile Methodology; 9](#_Toc48459032)

[ Project planning techniques such as Gantt Charts. 9](#_Toc48459033)

[Personal challenges 9](#_Toc48459034)

[PROJECT PLAN: 9](#_Toc48459035)

[Methodology: 9](#_Toc48459036)

[ REST Architecture; 13](#_Toc48459037)

[ Ionic / Angular; 13](#_Toc48459038)

[REST Architecture 14](#_Toc48459039)

[IONIC 14](#_Toc48459040)

[CHAPTER 3: SYSTEM ANALYSIS & DESIGN 14](#_Toc48459041)

[FUNCTIONAL REQUIRENMENTS 15](#_Toc48459042)

[REQUIREMENTS PER SUBSYSTEM: ACCESS TO THE SYSTEM 15](#_Toc48459043)

[Customer Register 15](#_Toc48459044)

[Customer Log-in 15](#_Toc48459045)

[Customer Booking 16](#_Toc48459046)

[Customer Booking View 16](#_Toc48459047)

[Staff Registration 16](#_Toc48459048)

[Staff Log-in 16](#_Toc48459049)

[Staff Booking View 17](#_Toc48459050)

[Administrator Registration 17](#_Toc48459051)

[Administrator Log-in 17](#_Toc48459052)

[Administrator Booking View 18](#_Toc48459053)

[Booking Workflow: Customer, staff and administrator 18](#_Toc48459054)

[Wireframes 18](#_Toc48459055)

[PUBLIC PAGE 18](#_Toc48459056)

[Log-in Page 19](#_Toc48459057)

[Login Page 19](#_Toc48459058)

[Register Customer Page 19](#_Toc48459059)

[PRIVATE PAGE 20](#_Toc48459060)

[Customer: 20](#_Toc48459061)

[Staff: 21](#_Toc48459062)

[Administrator: 22](#_Toc48459063)

[DATABASE 23](#_Toc48459064)

[Diagrams 23](#_Toc48459065)

[Database Data Definition Language (DDL) 24](#_Toc48459066)

[BACK-END 25](#_Toc48459067)

[Diagrams 25](#_Toc48459068)

[Java Persistence API 26](#_Toc48459069)

[Gradle Framework 27](#_Toc48459070)

[Spring Framework 28](#_Toc48459071)

[FRONT-END 29](#_Toc48459072)

[Spring Framework 32](#_Toc48459073)

[FRONT-END 33](#_Toc48459074)

[CHAPTER IV: SYSTEM IMPLEMENTATION 34](#_Toc48459075)

[ARCHITECTURE 34](#_Toc48459076)

[TECHNOLOGIES IMPLEMENTED 34](#_Toc48459077)

[FRONT END 34](#_Toc48459078)

[BACK END 34](#_Toc48459079)

[DATABASE 35](#_Toc48459080)

[DEVELOPMENT 35](#_Toc48459081)

[IMPLEMENTATION OF THE SYSTEM: EXPECTATION VS. REALITY 36](#_Toc48459082)

[Database 36](#_Toc48459083)

[Back-End 36](#_Toc48459084)

[Front-End 39](#_Toc48459085)

[CHAPTER V: TESTING & EVALUATION 42](#_Toc48459086)

[ACCESS TO THE SYSTEM TEST 43](#_Toc48459087)

[Register form 43](#_Toc48459088)

[Login Form 43](#_Toc48459089)

[CHAPTER VI: CONCLUSION & FURTHER WORK CONCLUSION 44](#_Toc48459090)

[FURTHER WORK 45](#_Toc48459091)

[APPENDIX A: SOURCE CODE 46](#_Toc48459092)

[DATABASE 46](#_Toc48459093)

[REFERENCES 56](#_Toc48459094)

# ABSTRACT

Ger runs a small garage where he carries out maintenance checks and he is supported by a small team of employees. The present document describes the Garage Web System designed based on Ger’s requirements. This application is to help Ger to better manage his garages so far he’s only had manual bookings, where the clients would come or call to the garage to book a service.

The new application has been being developed to support all stakeholders: clients can book their appointments, employees can consult them. The application provides login for staff and customers, staff and customer register, book, issue invoice and manage stock.

# CHAPTER 1: INTRODUCTION

## PROJECT CONTEXT

Ger manages a small team of 4 mechanics. All those employees are able to maintain different types of vehicle and perform any service or repair. In some cases, maintenance check will require parts or supplies (e.g. fluids, oil, tyres) which the staff will look for in the stock of supplies from the garage and invoice the customer for.

It is a small garage which carries out any type of services and repairs on cars, motorbikes, small vans and small buses. However, due to the high demand of bookings, Ger was facing, some issues which is going to be solved with the new application. The Web System allows customers and staff to login and register their details and customer can make appointment for their vehicles. The employees are able to view all booking assigned to them for the current day. Ger will also be able to manage and view all bookings and register new staff.

This web system is designed to assign automatically a booking to a staff member after a customer makes an appointment. Besides, the application allows Ger to print the schedule for any particular date, allocate costs to each booking or a basic fixed cost.

Taking into consideration the above scenario, the main services that need to be provided are:

* The customer can creates an account and books an appointment. After signing up and registering, the customer can visualise the last booking. The customer will also be able to update the following items:
  + User name and password, contact details and vehicle details;
* The staff will be able to check the schedule for the day, update booking status and complete information about services and prices.
* The administrator (Ger) will be allowed to create new staff, check and update bookings and monitor the stock.

Once registered, the customer can log-in and book a slot.

To book the appointment, the customer will be asked to pick a day, enter the vehicle’s details, the type of service and give details about the issue. The customer can’t make an appointment on Sundays as the garage is closed and also there will be a limit of services that can be booked per day.

The mechanic will need to update the booking status to either “Booked”, “In Service”, “Fixed / Completed”, “Collected” or “Unrepairable/Scrapped”.

Once the booking has been changed to “Completed” the application sends the invoice by email to the customer or can be printed from the screen.

Booking daily limit: the garage currently has 4 staff members who can carry out any kind of service. Each employee works at maximum on 4 vehicles per day. The total amount of services per day is the math between the number of staff members multiplied by the maximum number of services each employee can complete per day, so the daily limit for bookings is set to 16.

WHY IS IT A GOOD PROJECT?

Developing a web-based application enables us to go beyond the concepts we have learnt and apply them to a real-life case study. Also, it gives us the opportunity to articulate all the technologies we have studied during the course and others which we would have searched to implement this Web System.

The application has been built to serve small garages but can also attend to others businesses which purpose is to offer services (e.g hairdresser, computer maintenance, health consulting etc.). In addition, as it is web service based, new modules can be built and bigger garages could use this solution as well.

The system is a web service based architecture. It makes it easier to create new functionalities and integrate with other web applications. Also, troubleshooting and fixes are usually more straightforward because it is possible to isolate the problem. New modules can be added too (e.g. payments and e-invoices) with minimum effect, since they are independent blocks. The front-end is built using ionic calling the endpoints. In other words, to change the View layer only requires creating a new one and connecting to the same endpoints, without affecting the back-end.

## Main goal:

Create a web-based application where customers can book their vehicles to a garage service.

## Objectives:

* Research of technologies and review previous concepts;
* Define architecture and design diagrams;
* DB connection;
* Map JPA entities;
* Connect the Front-end with Back-end;
* Final report;

## Areas to cover:

To build this web-based application I’ve used the following technologies and methodologies:

### Java 8, object oriented language;

### Java Persistence API, as Java software component to map the database using annotations, and manage queries on the DB;

### Spring Boot as a Java framework in order to create web services;

### Tomcat as HTTP web server provider;

### Gradle, an open-source build automation system based on Apache Ant and Apache Maven;

### Postman, a tool used to test RESTful APIs;

### Ionic Framework to build desktop apps using web technologies integrated to Angular.

### In terms of the project itself, the following areas will be covered:

### SCRUM as Project Management Agile Methodology;

### Project planning techniques such as Gantt Charts.

### Personal challenges

It project development intends to address the following challenges:

* Front-end development

## PROJECT PLAN:

## Methodology:

Due to the complexity of working with and integrating a variety of stack technologies and also considering the size of the project, it has been managed using Scrum framework which is lightweight, simple to understand but difficult to master (Schwaber and Sutherland, 2017).

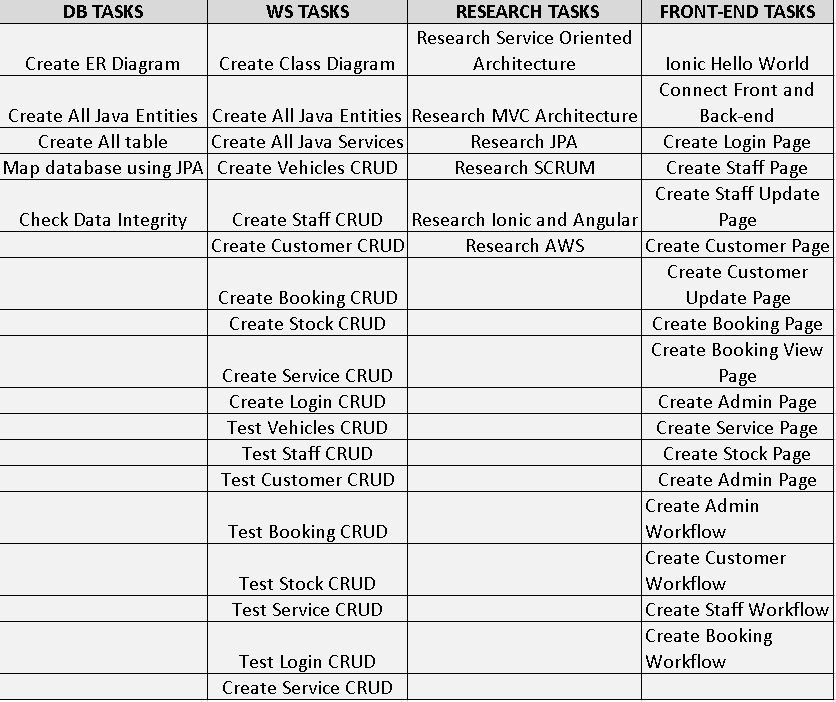
Scrum is a framework used to address complex adaptive problems, while productively and creatively delivering products of the highest possible value. It is for developing, delivering, and sustaining complex products and its definition consists of Scrum’s roles, events, artifacts, and the rules that bind them together (Schwaber and Sutherland, 2017).

There are three types of team stakeholders. Product Owner (PO), Scrum Master and Developer. As the Scrum creators stated the complexity of mastering it, this project has been managed by Product Owner and Developer role.

The PO role has produced the followings artifacts:

* Product Backlog items;
* Ordering Product Backlog to best achieve goals;

See below list of product backlog items:

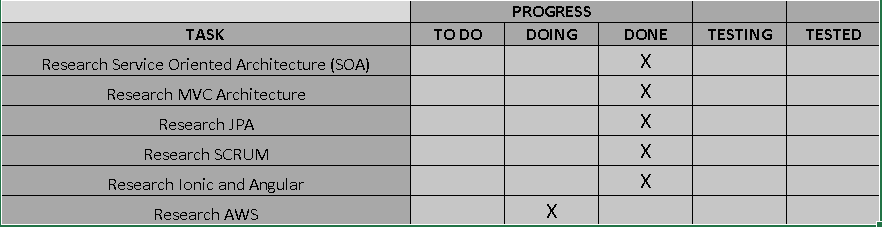


#### Product Backlog Item

Regarding the Development Team, it is the role which the stakeholder s work of delivering a potentially releasable Increment of “Done” product at the end of each Sprint (Schwaber and Sutherland, 2017).

At part of the project, the developer’s work has been supported by a Kanban board. According to (Scrum, 2020): “Kanban is a strategy for optimizing flow. The practices in the Kanban Guide for Scrum Teams help enhance and complement the Scrum framework and its implementation.”

See below list of activities on the Kanban board which the development team has used to follow and release the product backlog.



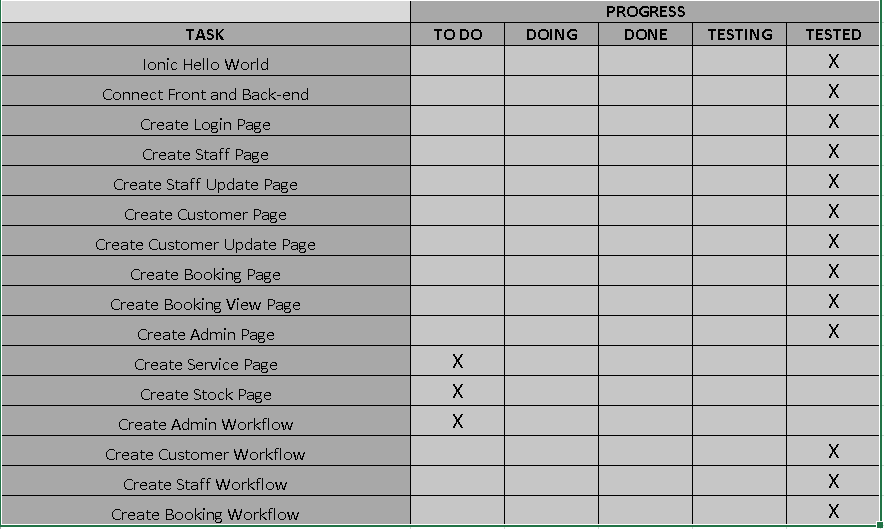
#### Kanban Board: Research Tasks



#### Kanban Board: Database Tasks

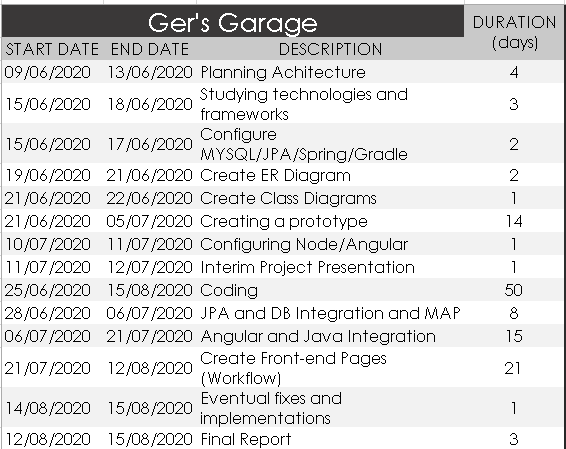


#### Kanban Board: Back-end Tasks

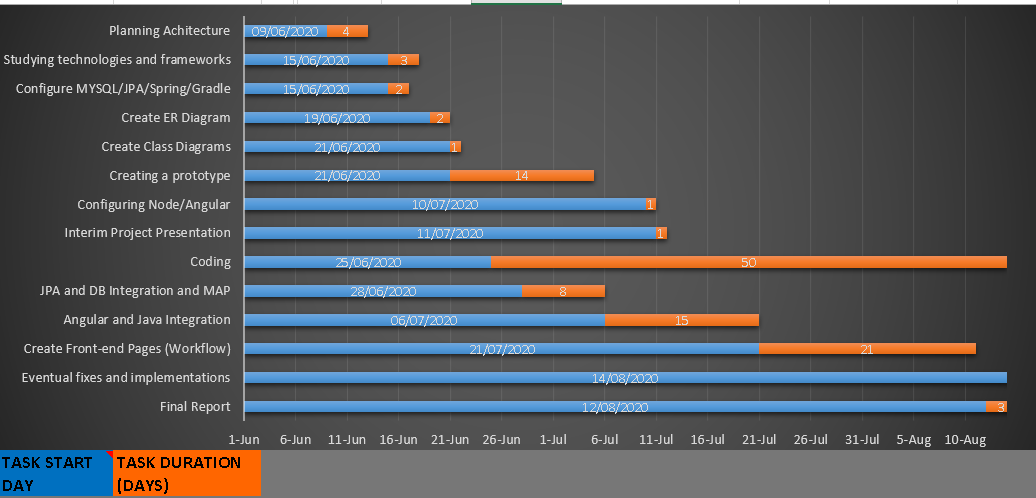


#### Kanban Board: Front-end Tasks

A Gantt Chart helped at the planning stage to outline and illustrate project schedule. It gives an overview of the project, the tasks and the deadlines.



#### Gantt chart: Timetable contents



#### Gantt chart: Bar Graph

This system was built over sprints of 1 week and using the Weekly Reflective Journal as a Sprint Review and Sprint Retrospective, which has given the possibility of revising Product Backlog and the Sprint Retrospective to create a plan for improvements.

# CHAPTER 2: LITERATURE REVIEW

The second chapter presents the academic research reviewed and used Due to the complexity of working with and integrating a variety of stack technologies and also considering the size of the project, it has been managed using Scrum framework which is lightweight, simple to understand but difficult to master this web application.

To achieve the project’s objectives, the following areas have been researched:

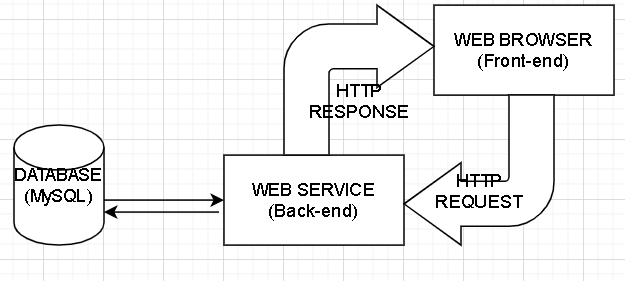
### REST Architecture;

### Ionic / Angular;

Although other topics and areas have been studied, those were the most challenging and required a considerable amount of research and time to complete this project.

## Representational State Transfer (REST)

REST is an architectural style which defines constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability and modifiability that enable services to work best on the Web (Oracle, 2013).



## REST Architecture

Each service in this architecture keeps its own code and is able to execute a complete functionality. Taking the Customer workflow as a reference, it is possible to visualise functionalities apart from each other. Through HTTP Request Methods (GET, PUT, POST and DELETE) and using JSON. For instance, customer registration is a functionality where the front-end (Browser) asks the back-end (Web Service) the method to register and it has been exposed through an endpoint.

## IONIC

Ionic is an open source, cross-platform framework used to develop web applications and hybrid mobile apps.

Ionic is a framework which helps to the frontend UX and UI interaction of an app, UI controls, interactions, gestures and animations. It is easy to integrate with other libraries or frameworks, such as Angular or React. Alternatively, it can be used standalone without any frontend framework using a simple script include (Ionic, 2020).

By integrating this framework with Angular libraries, using the command line (@ionic/angular), it gives all Angular core components and improves the development process.

# CHAPTER 3: SYSTEM ANALYSIS & DESIGN

This web system is designed to meet Ger’s mechanical garage needs. It is a small garage which carries out any type of services and repairs on cars, motorbikes, small vans and small buses as mentioned on the Chapter.

The third chapter will describe the application functional requirements and functional design. Also, it will show the diagrams that support both back-end and front-end.

## FUNCTIONAL REQUIRENMENTS

The application implies functional requirements for administrators, customers and mechanics. Moreover, pre-set up is required.

## PREREQUISITES

* At least one administrator and one staff member need to be set up;

## MAIN REQUIREMENTS

* The customer must sign up to book their vehicles for a service;
* After signing up, the customer must make an appointment for their service;
* The staff member needs to update the booking status, add a description of any extra service, use supplies from stock and charge for it if necessary;
* Either Staff or Administrator must be able to print the invoice for the customer.

## REQUIREMENTS PER SUBSYSTEM: ACCESS TO THE SYSTEM

## Customer Register

The customer must access the web system and create an account in order to be able to make an appointment for their vehicles. By going through the registration page the customer will find a form where the required details will need to be filled in.

Required information on the form:

* Username and login;
* First and last name;
* Email, phone number and PPSN;
* Vehicles details;
* Type;
* Maker;
* Model;
* Fuel;
* Manufacture;
* Licence number;

## Customer Log-in

To access the application the customer needs to provide the following:

* Username;
* Password;

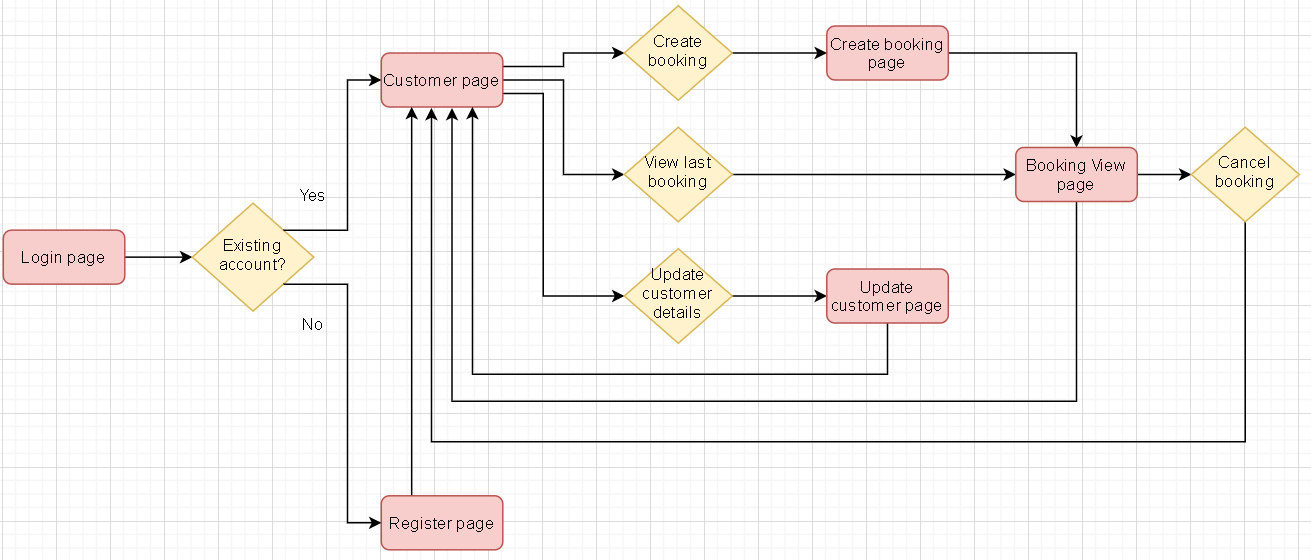
## Customer Booking

In order to book the vehicles for a service the customer must go to the booking page and enter some details:

* Booking date (date of appointment);
* Type of Service: Annual Service, Major Service, Repair / Fault and Major Repair);
* Details about the issue;
* The system will automatically load the customer and vehicle details.

## Customer Booking View

By accessing this page, the customer can visualise the last booking with the given details, plus the booking status and the mechanic attending the vehicle.



#### Customer Workflow

## Staff Registration

Each staff member will be created by the administrator with no need to go to the registration page like in the customer workflow. To create accounts for the staff, the administrator must enter some personal details that should be given by employees beforehand.

Required information on the form:

* Username and login;
* First and last name;
* PPSN;

## Staff Log-in

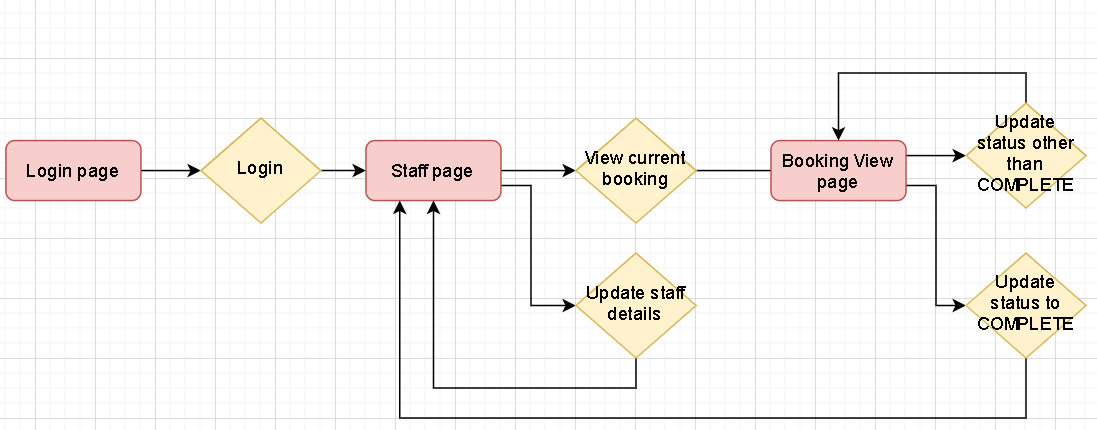
To access the application once the account is created the employee needs to enter the following:

* Username;
* Password;

## Staff Booking View

On the booking page the staff members must update the booking status for the services that have been assigned to them. In case the service requires to use stock supplies, the staff can add it on the notes and also add the value and a description for the extra service:

* Booking Status (Mandatory): Booked, In Service, Fixed, Collected and Unrepairable/Scrapped;
* Description Extra Service (Optional)
* Extra Service Price (Optional);



#### Staff Workflow

## Administrator Registration

As described in the prerequisites section, the web system admin will be set up at the very beginning through a database dump or a simple DB insert.

To create an admin account, the following details will be required:

* Username and login;
* First and last name;
* PPSN;

## Administrator Log-in

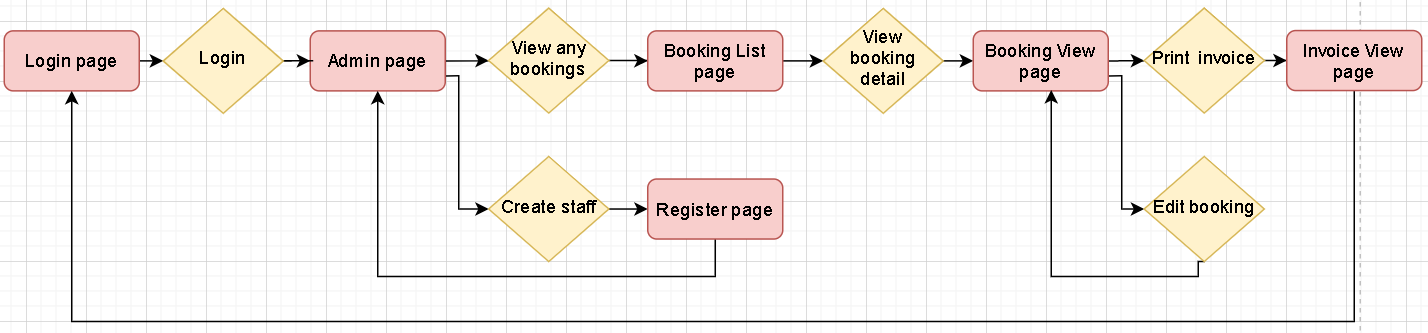
To access the application the admin role needs to provide the following:

* Username;
* Password;

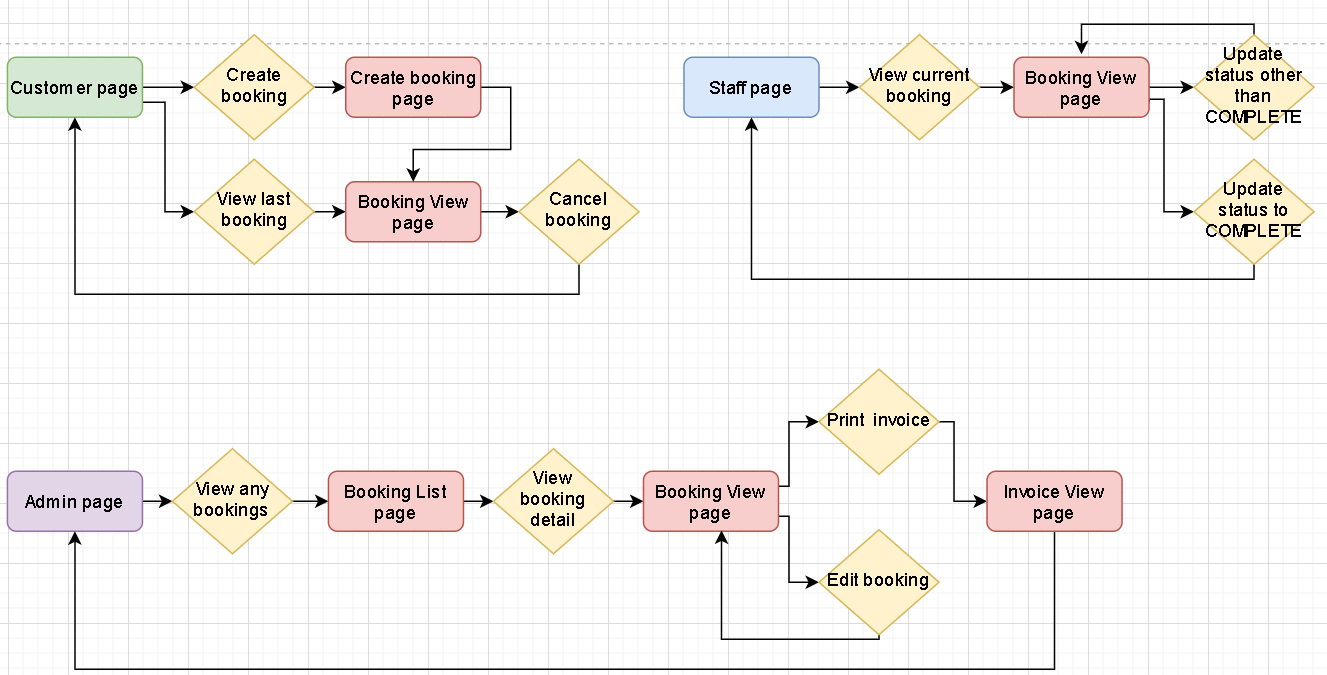
## Administrator Booking View

On the booking page the administrator can view any booking. In addition, he is able to update the booking status. In case the service requires to use stock supplies, the administrator can add it on the notes and also add the value and description for the extra service:

* Booking Status (Mandatory): Booked, In Service, Fixed, Collected and Unrepairable/Scrapped;
* Description Extra Service (Optional)
* Extra Service Price (Optional);
* Print Invoice (Mandatory);



#### Administrator Workflow



### Booking Workflow: Customer, staff and administrator

## Wireframes

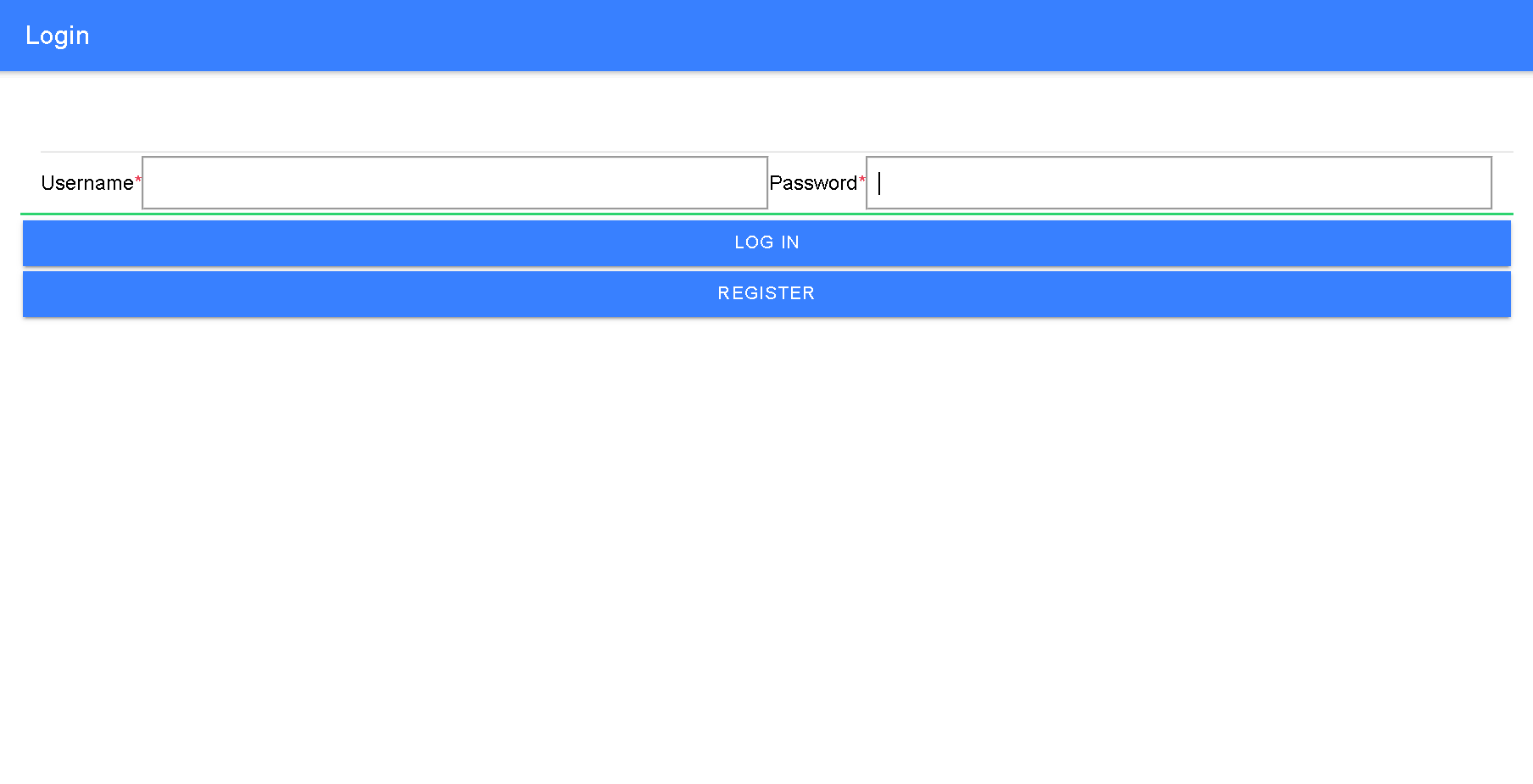
This section will depict the above workflows. For it, all requirements described on the previous section will be illustrated through wireframes.

## PUBLIC PAGE

The Log-in and Register are public pages meaning that anyone with the link can access it.

## Log-in Page

It allows customers to register and staff, administrator, and customers to sign in.



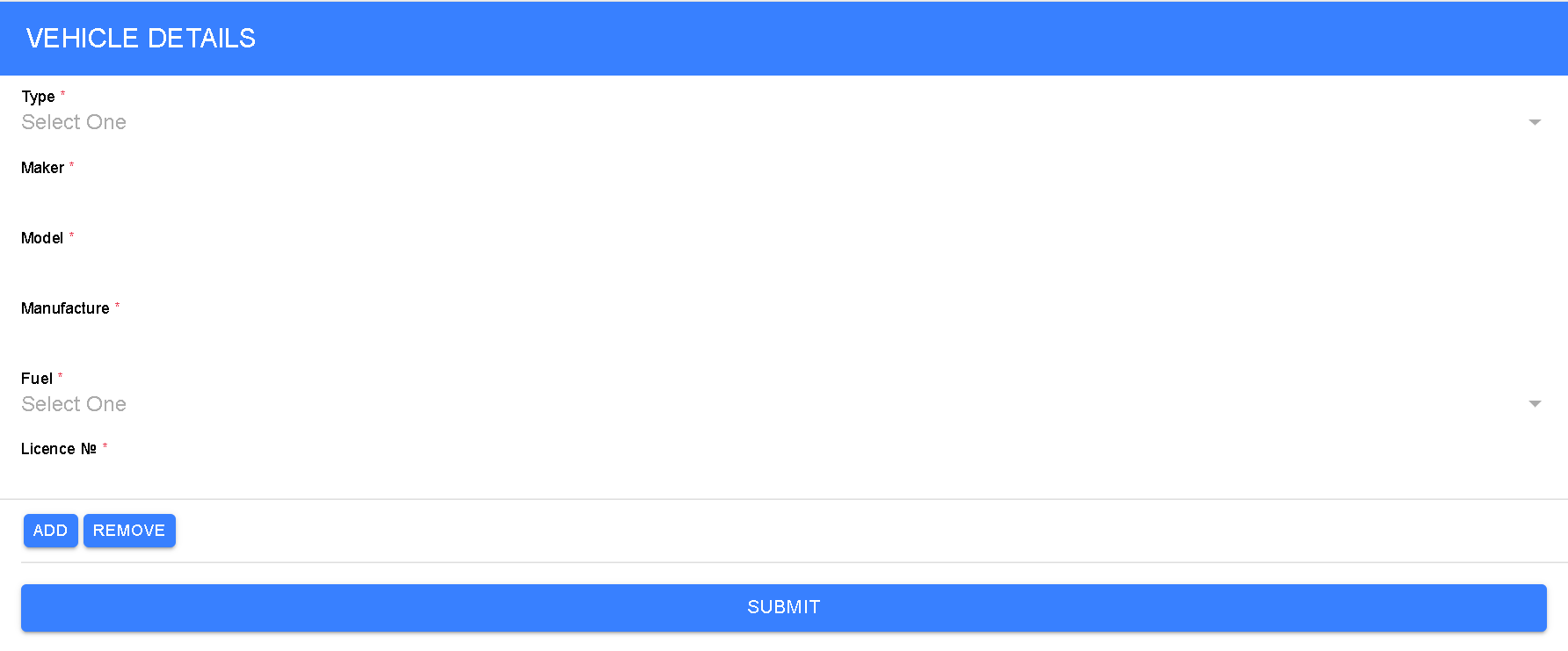
### Login Page

## Register Customer Page

Creation of a new customer account by filling up required fields and optional information.



#### Customer Register Page: User and Customer Details Section



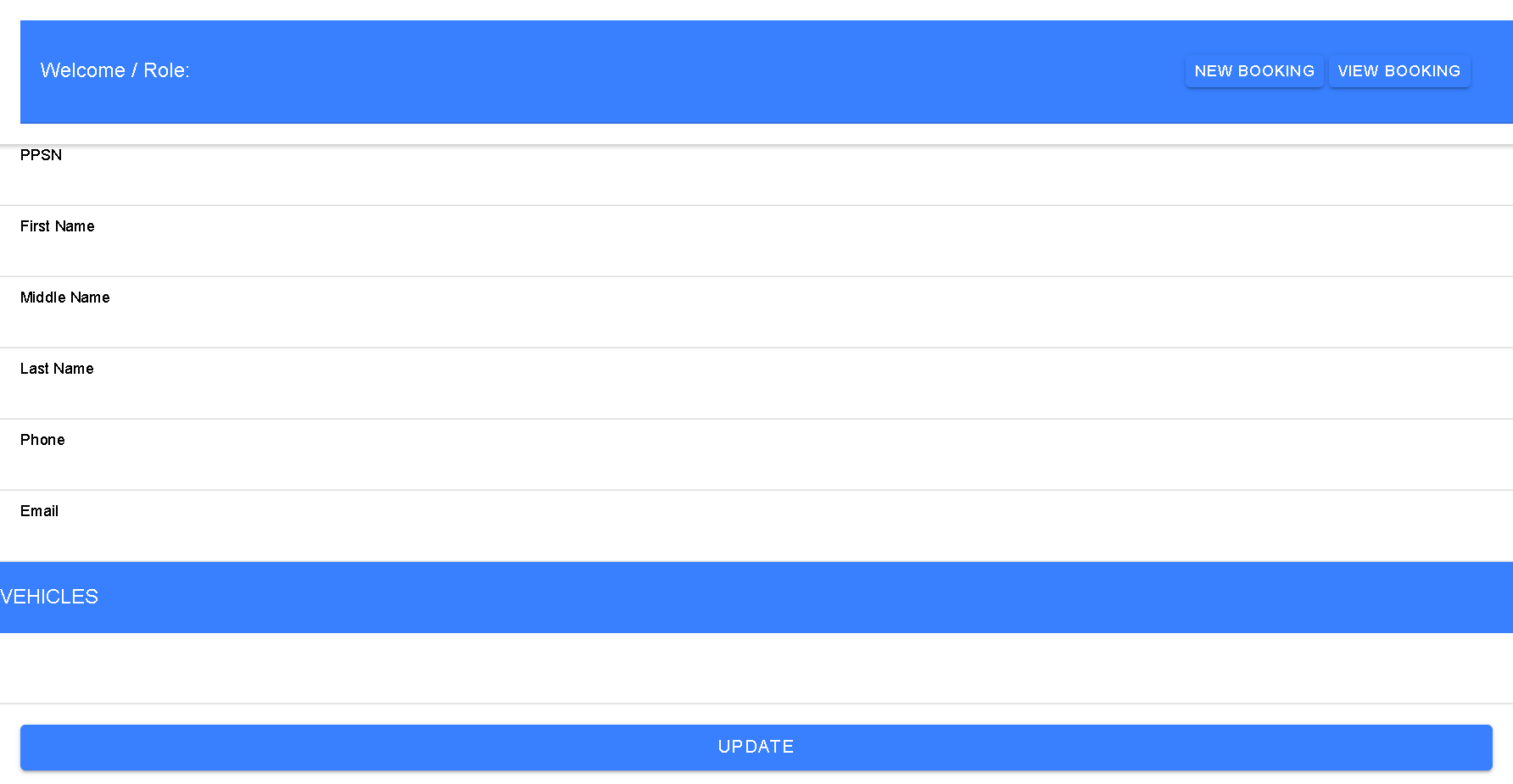
#### Customer Register Page: Vehicle Details Section

## PRIVATE PAGE

## Customer:

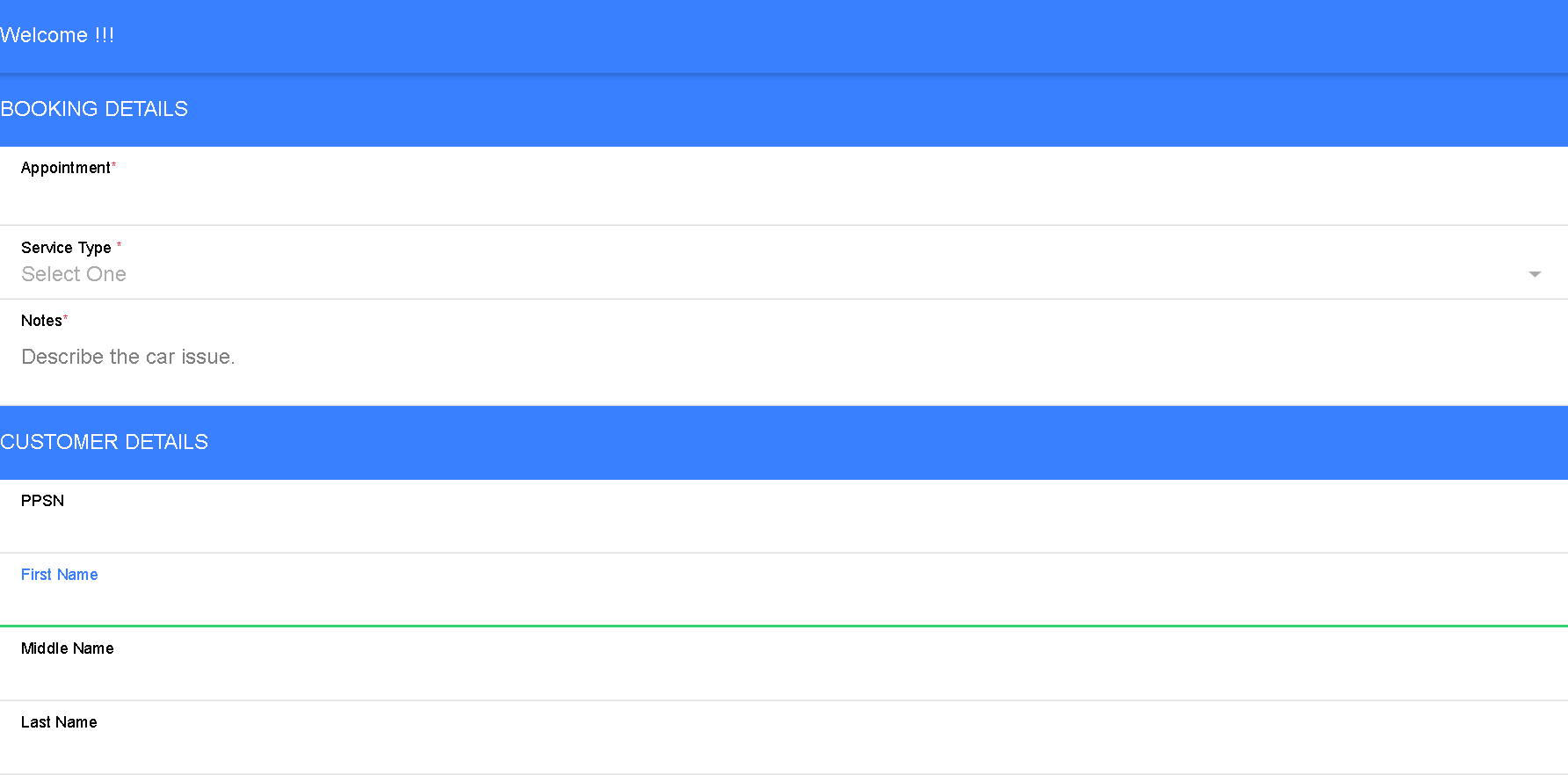
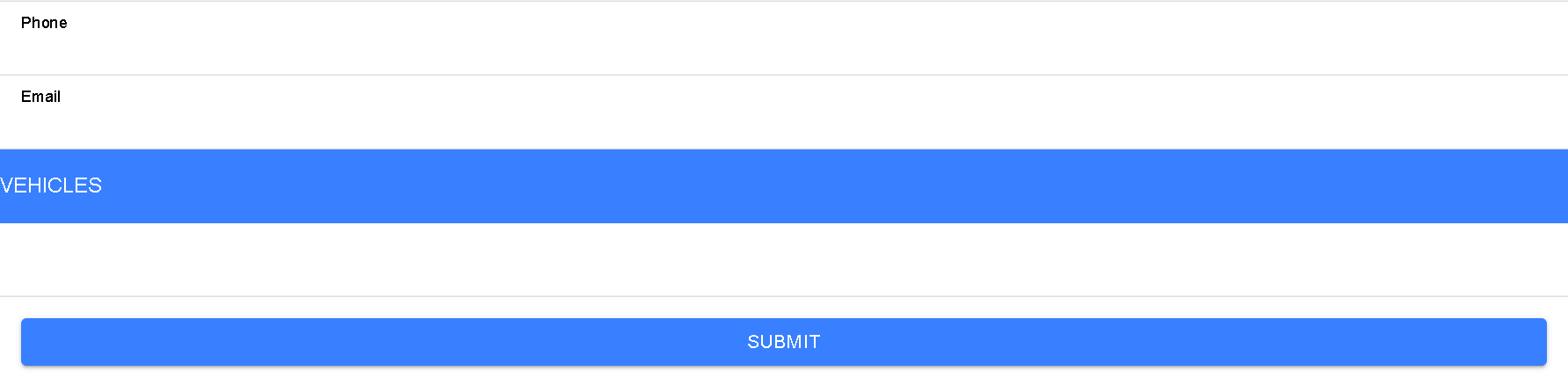
After creating an account, the customer can access the private page by using username and password on the log-in page. The customer page view will be loaded with their personal information and vehicle details.

At this point, the customer is able to update personal and vehicle details, make a new appointment and view the last one.



#### Customer Page

If the customer decides to access the new booking page, his details will be automatically populated and other details will be required on the booking form.

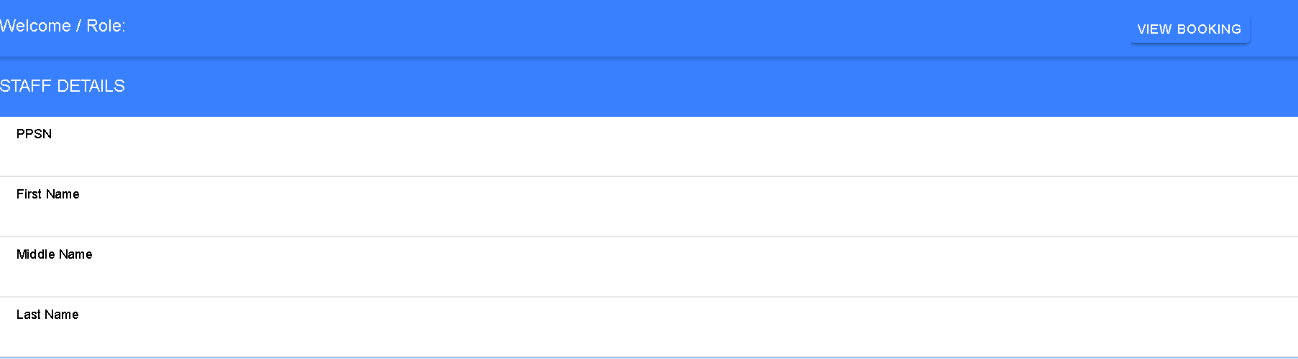
#### New Booking Page

In case they have already created a booking, customers can view the last booking page, which is very similar to the new booking page but brings additional information such as booking status and name of staff member who carried out the service.

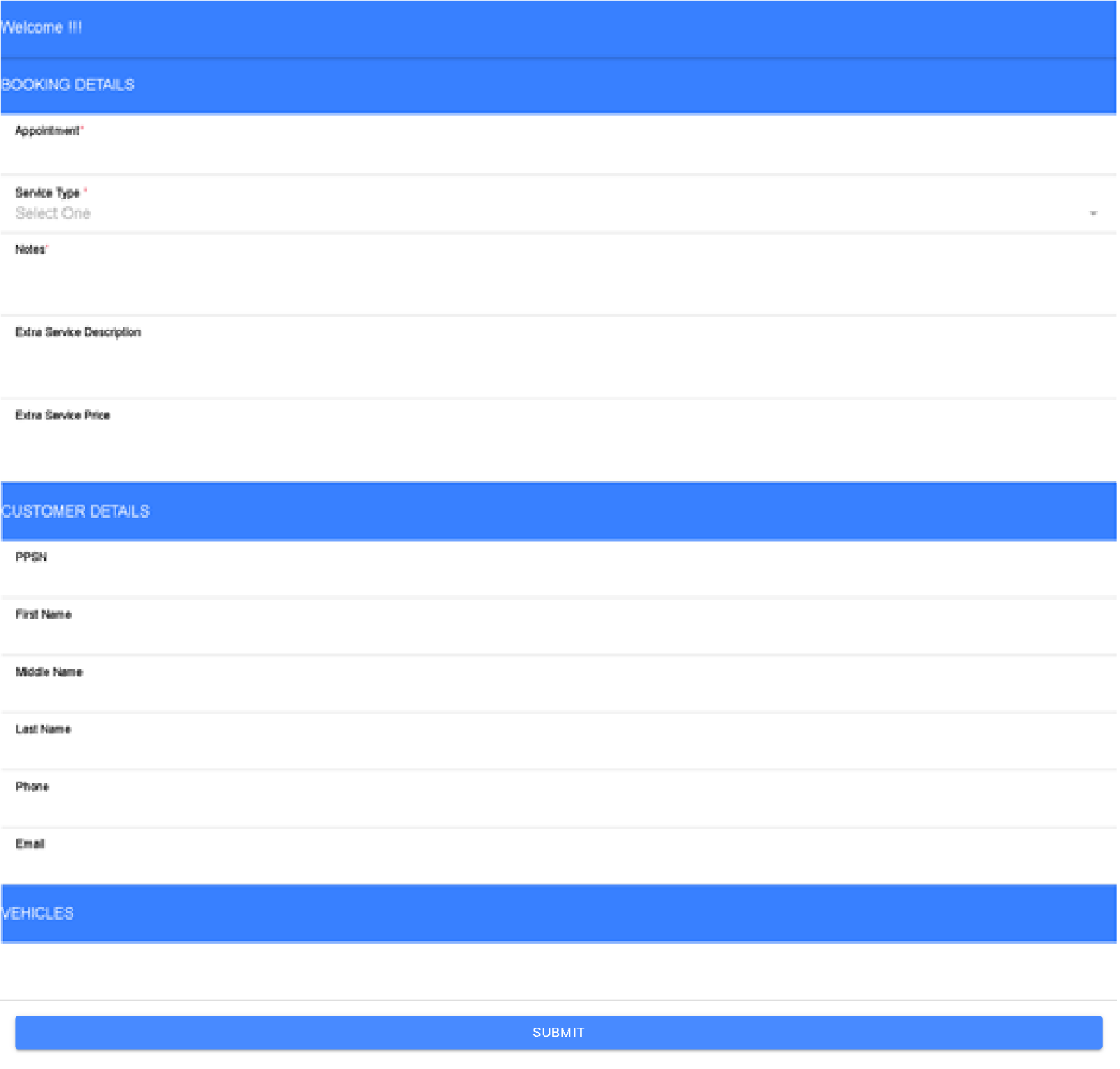
## Staff:

After getting their accounts created by the system admin, the staff members will access the private pages by entering their username and password on the log-in page and it leads to the staff page view where all personal data will be populated while the page is being loaded.

On this page, the mechanic is able to change personal details by using the button update. Also, it will be possible to view all the assigned bookings for the current day.

Staff Page

#### If a staff member wants to access the booking page, it will first show a booking list for the current day; then he can click on a specific booking and the next page will show the details of the booking. On this page, the employee can change the status of the booking. This page is very similar to the booking page illustrated on the customer section. The small changes are: Booking status drop-down menu will be available for selecting the status and extra inputs (Extra service description and extra service price.)



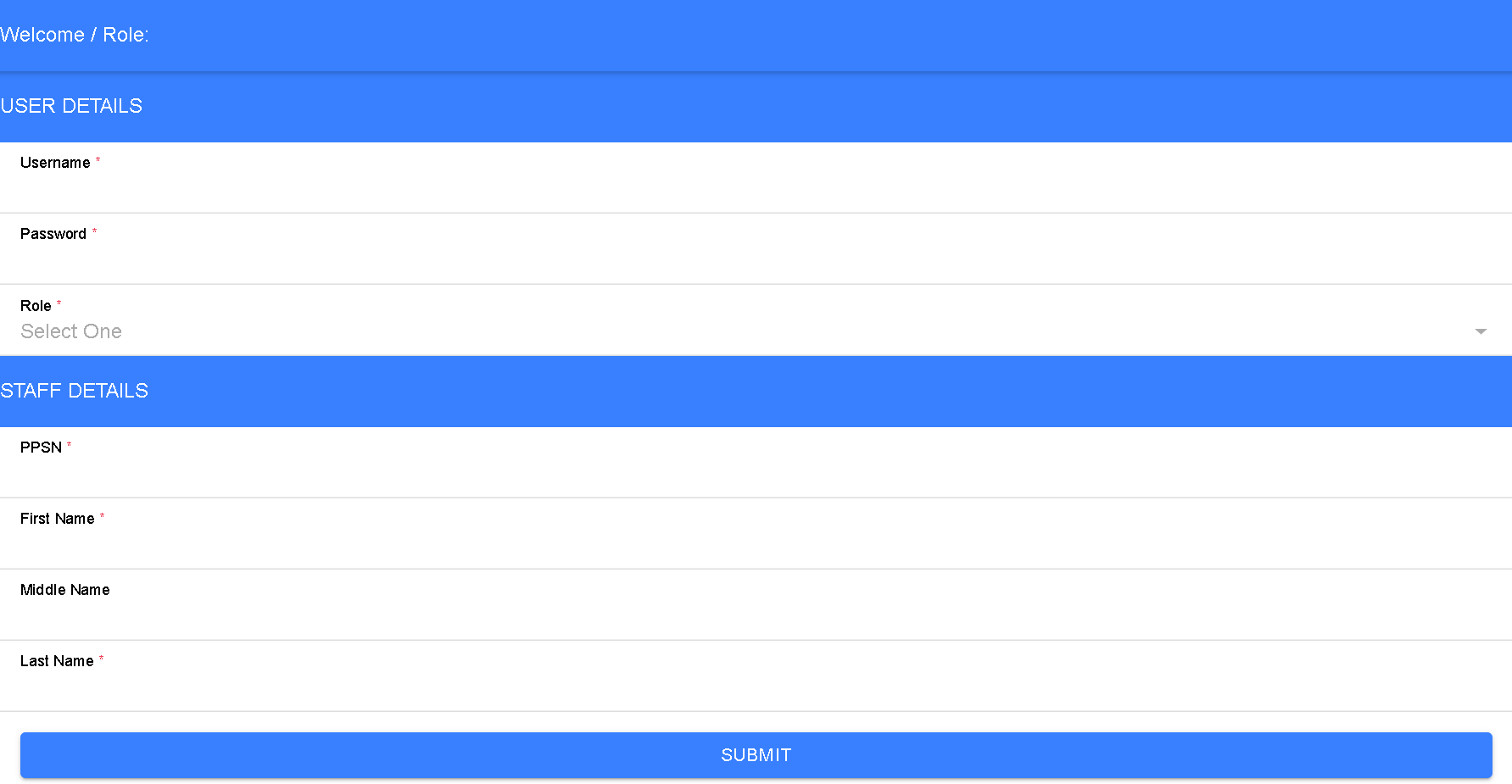
#### Booking View Page

Staff can also print the invoice for the customer.

## Administrator:

After having an account set up at the very beginning, the administrator will access the private page by logging-in. It will bring to the administrator page view where all personal data will be populated while the page is being loaded.

On this page, the system admin is able to create staff accounts and view any booking. When creating new staff in the system, the staff registration page will be loaded and the admin will enter the staff details as shown on the image below.



#### Staff Register Page

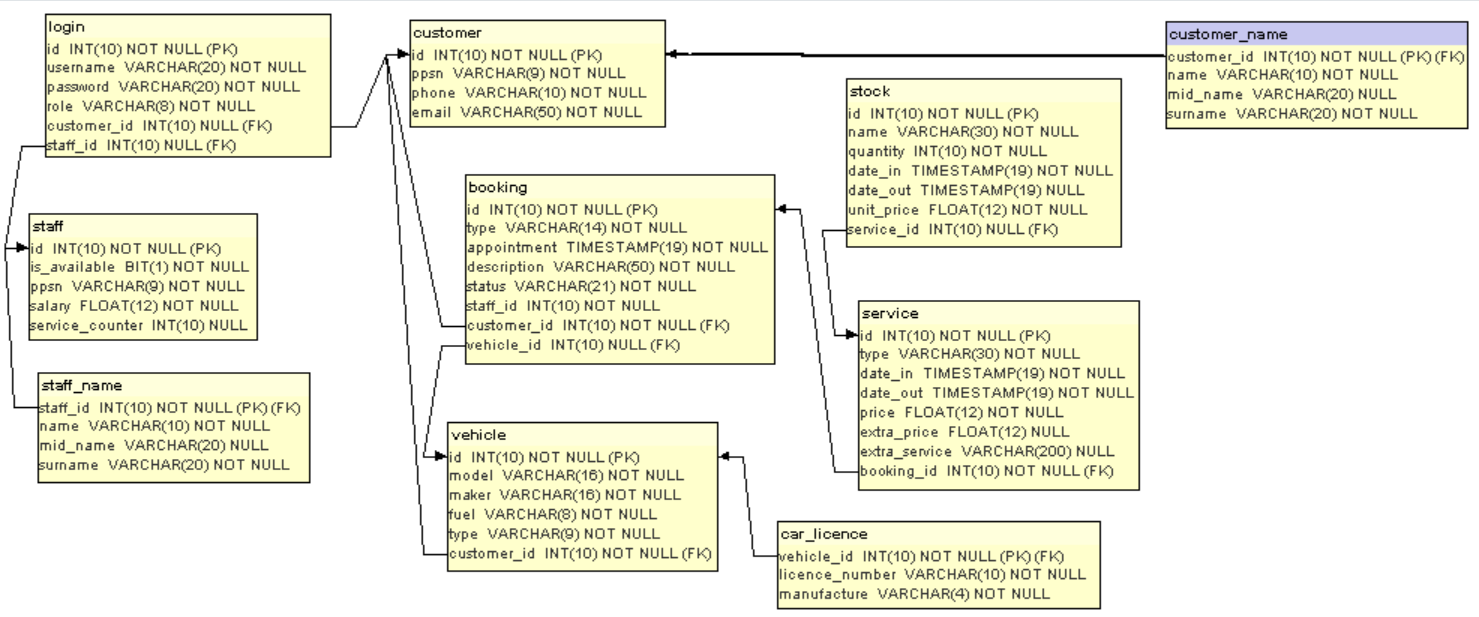
If instead of creating a new staff the administrator decides to view bookings, it leads to a list of bookings where it is possible also search per appointment date. By choosing one of them, the administrator will access the selected booking. It shows the same page and gives the e on the staff booking view page section.

## DATABASE

A relational database (MySQL) has been used to build this project in order to store all relevant data about Ger’s Garage Web System. Using this tool, required and additional tables have been created for a better working system. And to support the project with creation and drop tables and queries, a MySQL Workbench tool has been used.

## Diagrams

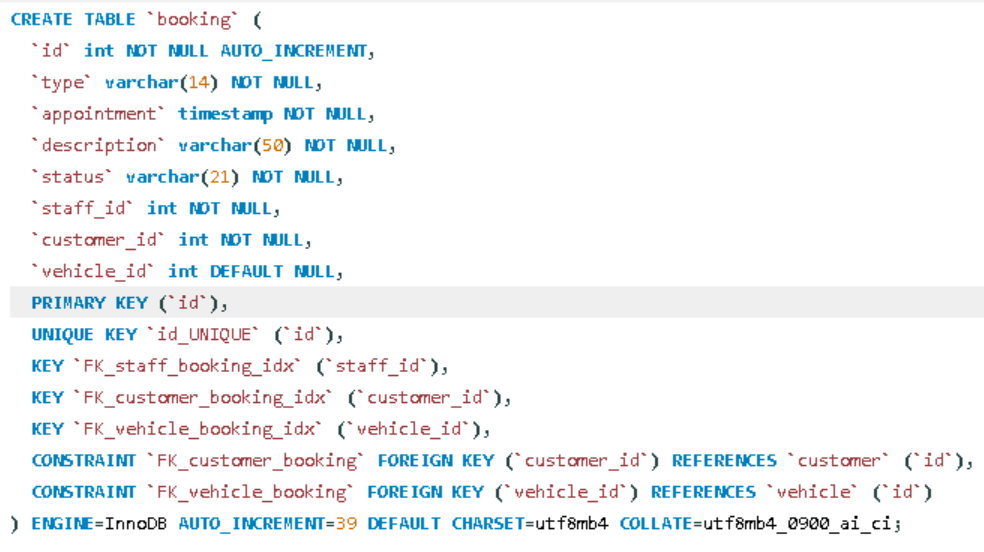
The project has been developed following the Entity Relation Diagram (ER). It depicts the database structure itself and helps on the development side. See the diagram below with the attributes, entities and relations.



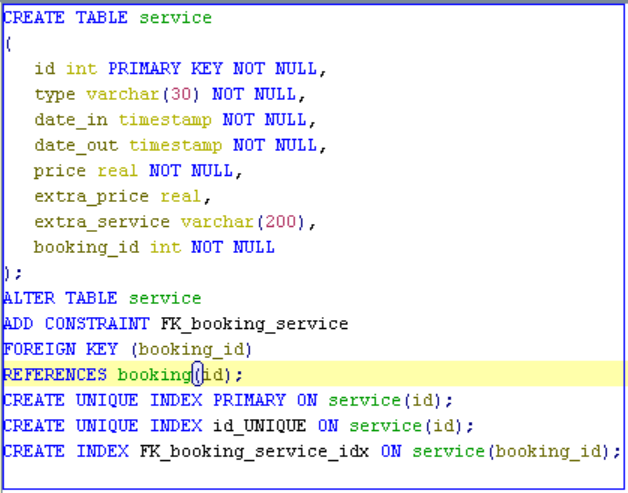
#### Entity Relational Diagram

## Database Data Definition Language (DDL)

DDL defines structures such as database, schemas, constraints amongst others. It helps to create and make changes on the database. See below images of the DDL of this project:



#### Booking Table DDL

It is possible to create a script for the table structure as well. It helps when replicating the database to another environment.

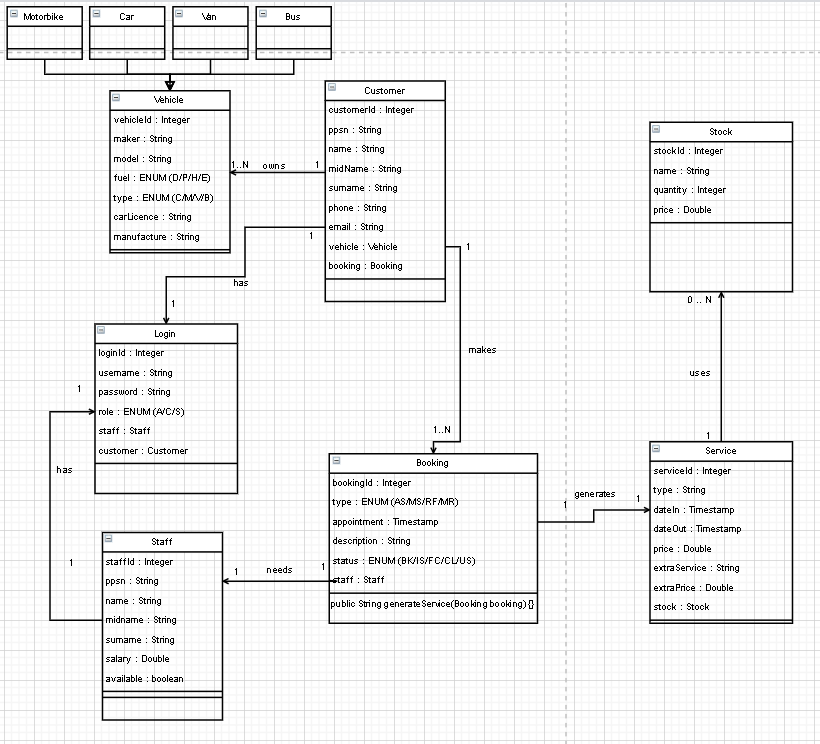
#### DB Script

## BACK-END

Java is an Object Oriented (OO) language. The decision of using Java led by the fact that it is widely spread, with big companies building their systems with this language, and many forums and communities to provide help and guidance. Also, there are many different frameworks that support Java development.

## Diagrams

In parallel with the ER diagram, a class diagram has also been produced to make it easier to view all classes, attributes, methods and relationships.



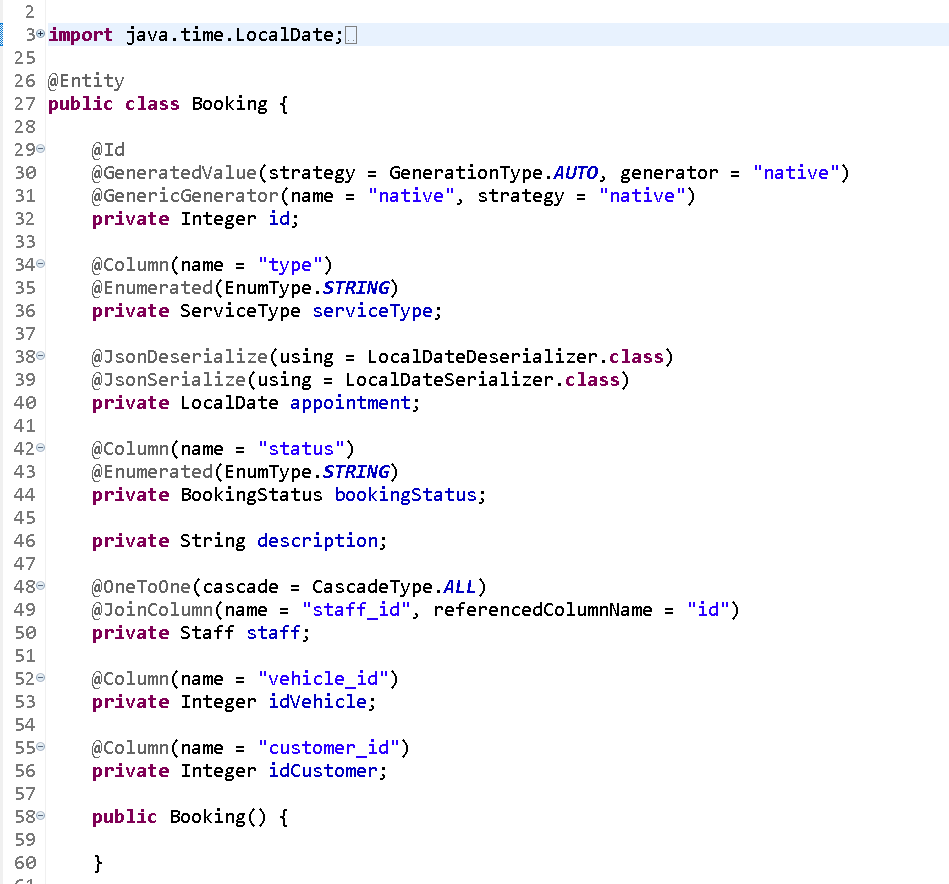
#### Class Diagram

This project will show some of the most known and used Java frameworks.

## Java Persistence API

Previously described in Chapter 1, JPA helped on the Java development dealing with queries and database connection.

This API helps mapping by using annotations. See below Booking class being mapped with those annotations. Notice that it reflects the database.



#### Booking Class using JPA annotations

## Gradle Framework

The back-end development was done with this framework. It has a build automation tool to control and download project dependencies. By adding dependencies on the “build.gradle” file, similar to POM.xml from Maven, the project downloads all libraries which will be used to develop.

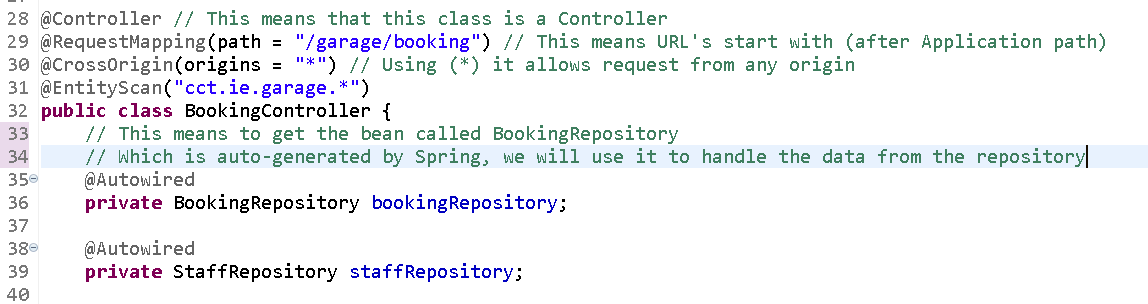


#### Build.gradle

## Spring Framework

It is a framework which allows to build web services. Springs annotations help on the development side. With those annotations, it becomes simple to create an HTTP Request and Response.

See below Booking Controller with some required annotation to build the web service.



#### Spring Annotations

Also, see below the Java Project Structure.

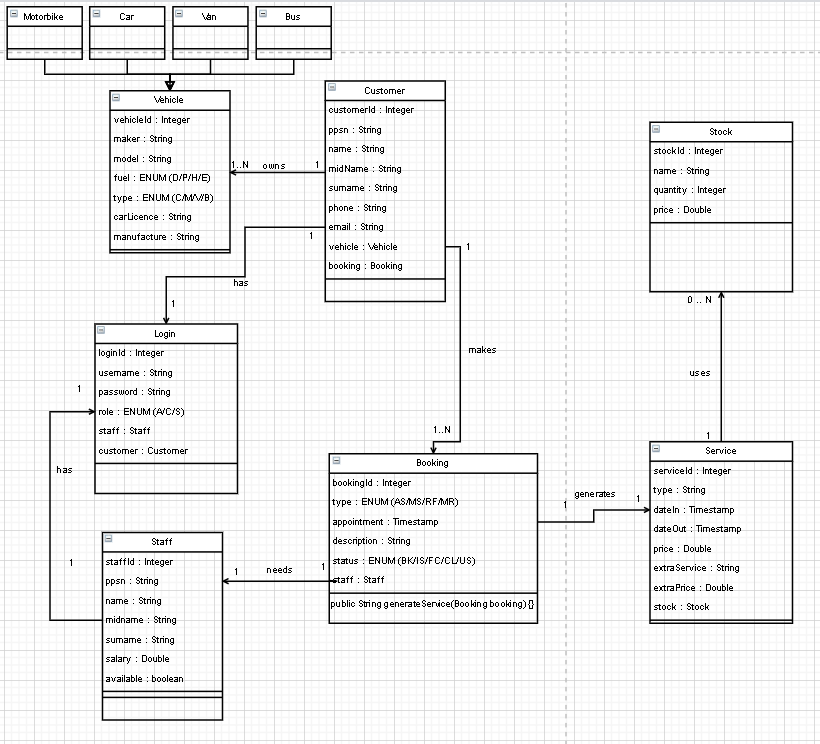


## FRONT-END

Ionic is the front-end language used for this application. Ionic combined with Angular is a powerful tool for front-end development. It has been built in an independent way, which makes it easier to customise the front-end layer.

## Diagrams

In parallel with the ER diagram, a class diagram also has been produced to view all classes, attributes, methods and relationships.



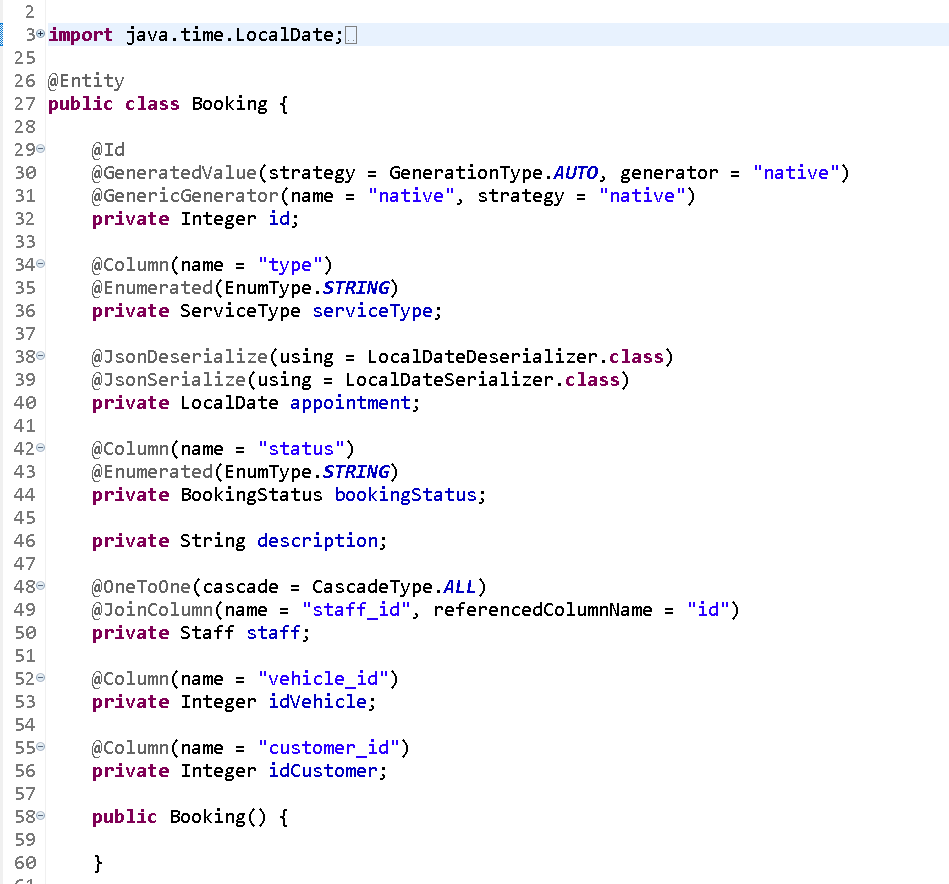
#### Class Diagram

Also, this project will show some of the most known and used Java frameworks.

## Java Persistence API

Previously described on the Chapter 1, JPA helped on the Java development dealing with queries and database connection.

This API supports on mapping the by using annotations. See below Booking class being mapped with those annotations. Notice that it reflects the database.



#### Booking Class using JPA annotations

## Gradle Framework

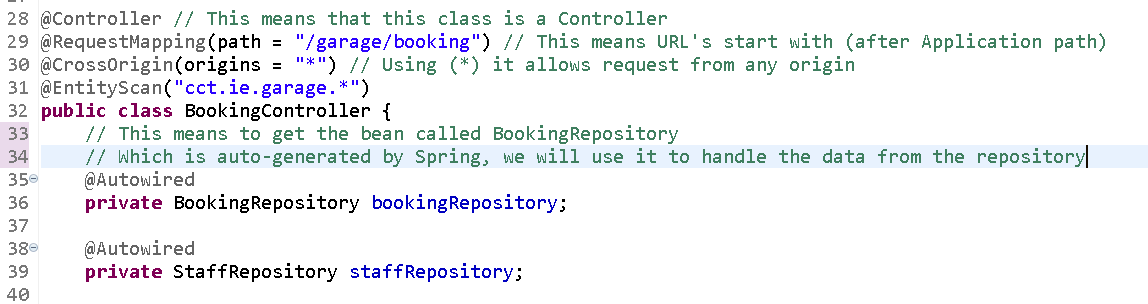
The back-end development started through this framework. It as a build automation tool to control and download project dependencies. With Gradle, by adding dependencies on the “build.gradle” file, similar to POM.xml from Maven, the project download all libraries which will be used to develop.



## Spring Framework

It is a framework which supports on building web services. Springs annotations supports on the development side. With those annotation, it becomes simple to create HTTP Request and Response.

See below Booking Controller with some required annotation to build the web service.



#### Spring Annotations

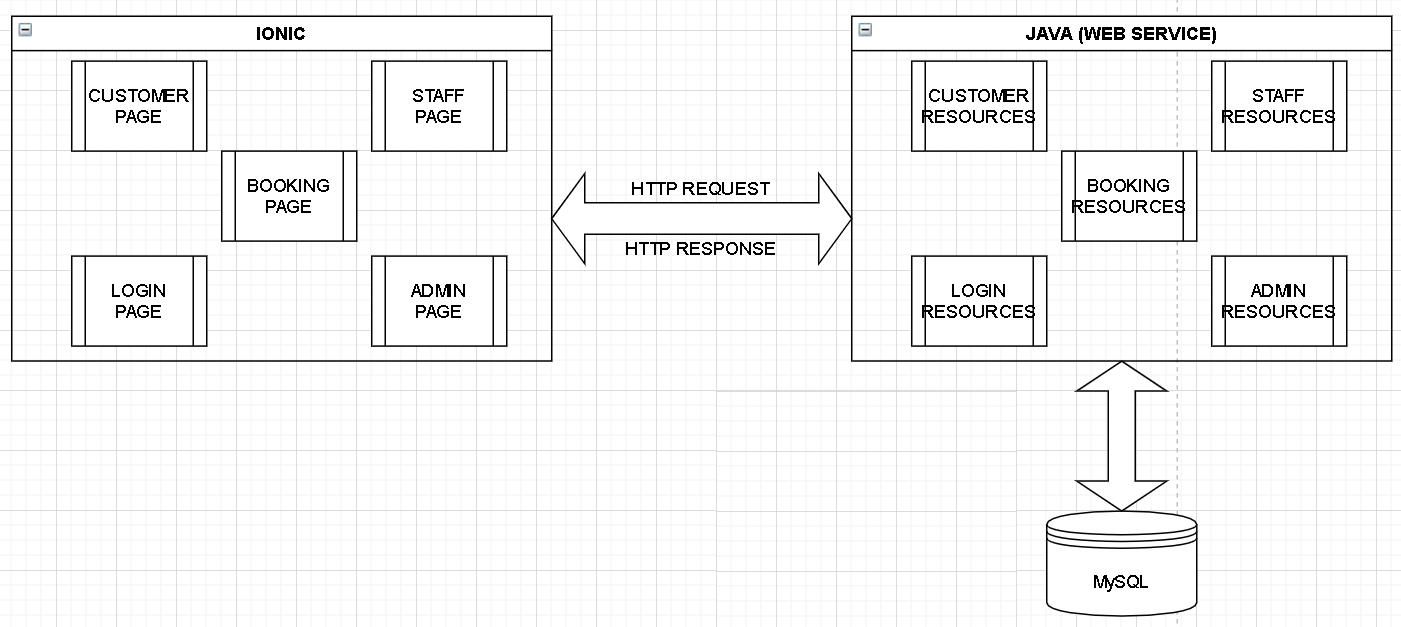
Also, see below the Java Project Structure.



## FRONT-END

Ionic is front-end language used to build this application and integrated with Angular gives a powerful tool to front-end development.

The front layer has been built in an indepent way, what makes easier to customise the front-end. Towards this, it is just necessary build another front-end and call the endpoints, with no effect on the back-end side.



#### Garage System Architecture

# CHAPTER IV: SYSTEM IMPLEMENTATION

As seen on Gantt chart through the specified tasks or on the product backlog on the previous chapters, the project life cycle has been split into 3 main phases:

* Management and Design;
* Development;
* Testing;

The previous chapters has explained about the first part of the life cycle and now, the development phase of the project will be shown.

## ARCHITECTURE

The project architecture is based on Restful and uses Java 8 to build the web service and make the resource available. Besides Java, JPA manages the database access, Spring to support to create the endpoints, MySQL as a data store and Ionic to create pages to user accessibility.

## TECHNOLOGIES IMPLEMENTED

## FRONT END

* Ionic;
* HTML;
* CSS;
* JavaScript;
* Angular;
* HTTP Methods;
* JSON;

## BACK END

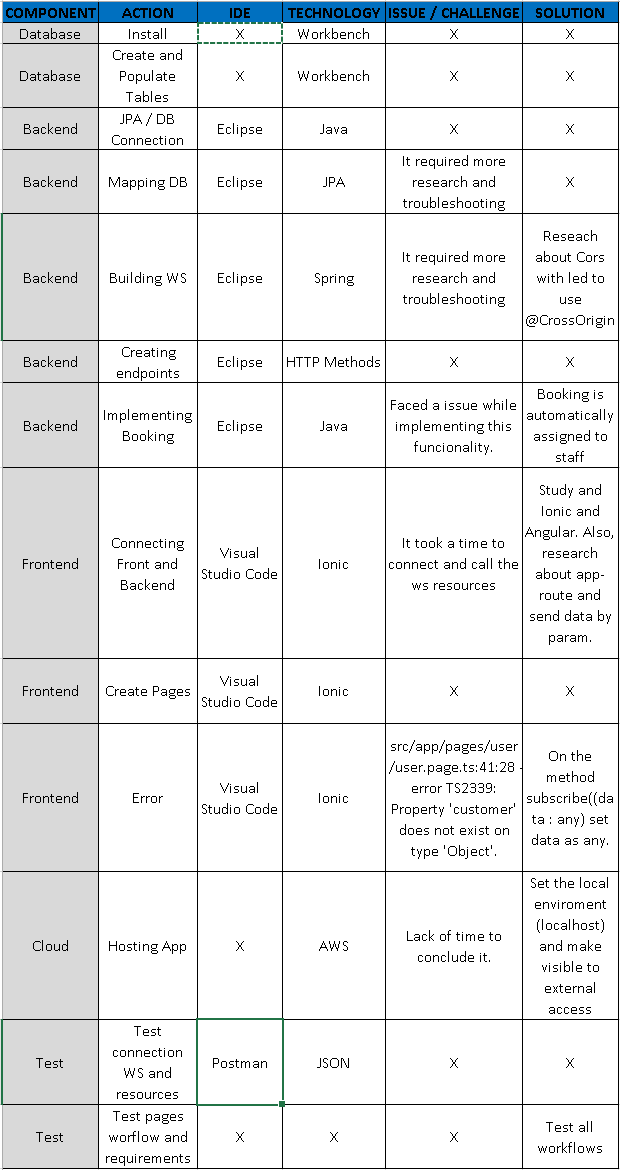
* Java;
* JPA;
* Tomcat;
* Spring Framework;
* Gradle;

## DATABASE

* MySQL;

## DEVELOPMENT

The following table summarizes all tasks on the development life cycle and includes issues, troubleshooting and solution accomplished on this project.

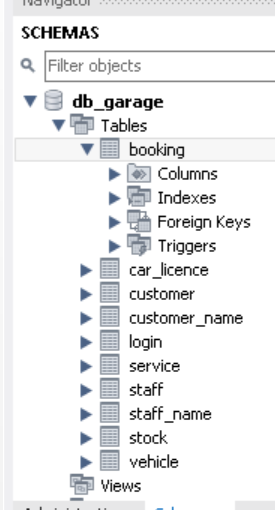


#### Issue / Solution Tracking Sheet

# IMPLEMENTATION OF THE SYSTEM: EXPECTATION VS. REALITY

## Database

At this stage the project achieved the requirements on time. Researches about which database ended up leading to use MySQL. All DB structures such as schema, tables, columns, constraints and at last the data insert.



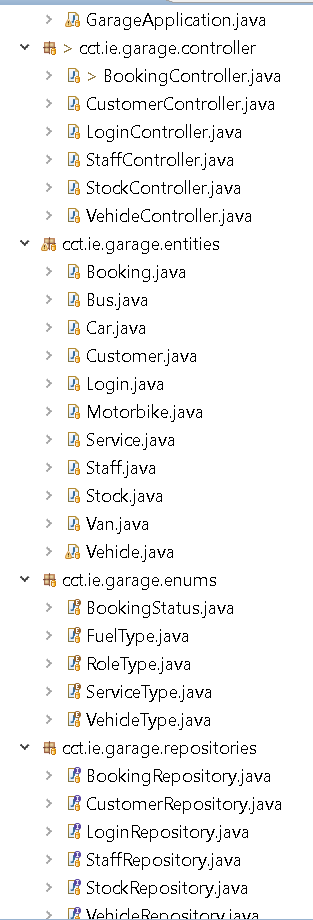
#### DB\_GARAGE SCHEMA

Relational Modelling, such as third normalisation, has been used and some modification were made to the web system improvement.

## Back-End

At the very beginning, due to previous knowledge on Java, Eclipse and database, the back-end development were as planned. Connection with database, JPA annotation and entity creation worked perfectly. However, while building the web service, few issues with Spring annotation and WS Controller have shown up. It affected the development process and led to the decision of only implement the main functionalities required for the project.

On the image below is going to show the Java WS structure.

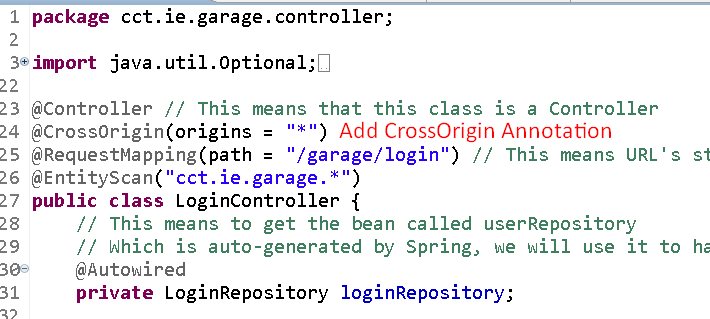


As a result of learning curve of Ionic and Angular, some functionalities have receive high priority and due to lack of timing others had to be left aside.

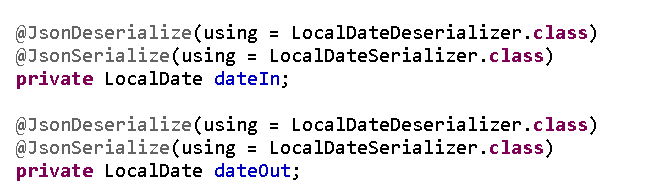
Functionalities not completed:

* Stock (Supplies);
* Admin View All Bookings;

When connecting the front-end to the back, few changes needed to be done.



On the image above one of the changes were add this annotation to allow the back-end to receive request from any origin. It can also be used with a specific origin. For example: @CrossOrigin(origins = "http://localhost:8100").



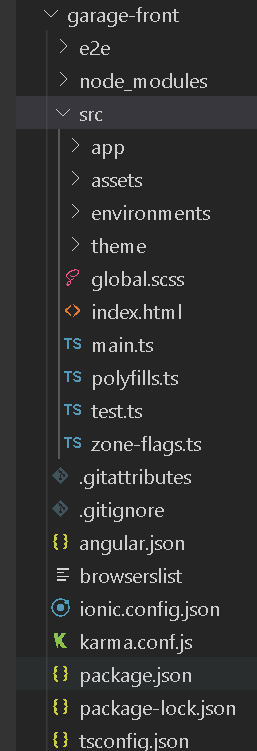
At this stage, a date serializable issue has been faced. Using the Jackson library and (@JsonDeserialize and @JsonSerialize) there is no need to worry about date serialise.

Despite some methods haven’t been implemented, the most of functionalities were covered. Plus, instead of the Admin (Ger) being asked to assign all booking, as requirements, after customer making an appointment, it is automatically assigned to the staff with the smallest amount of service.



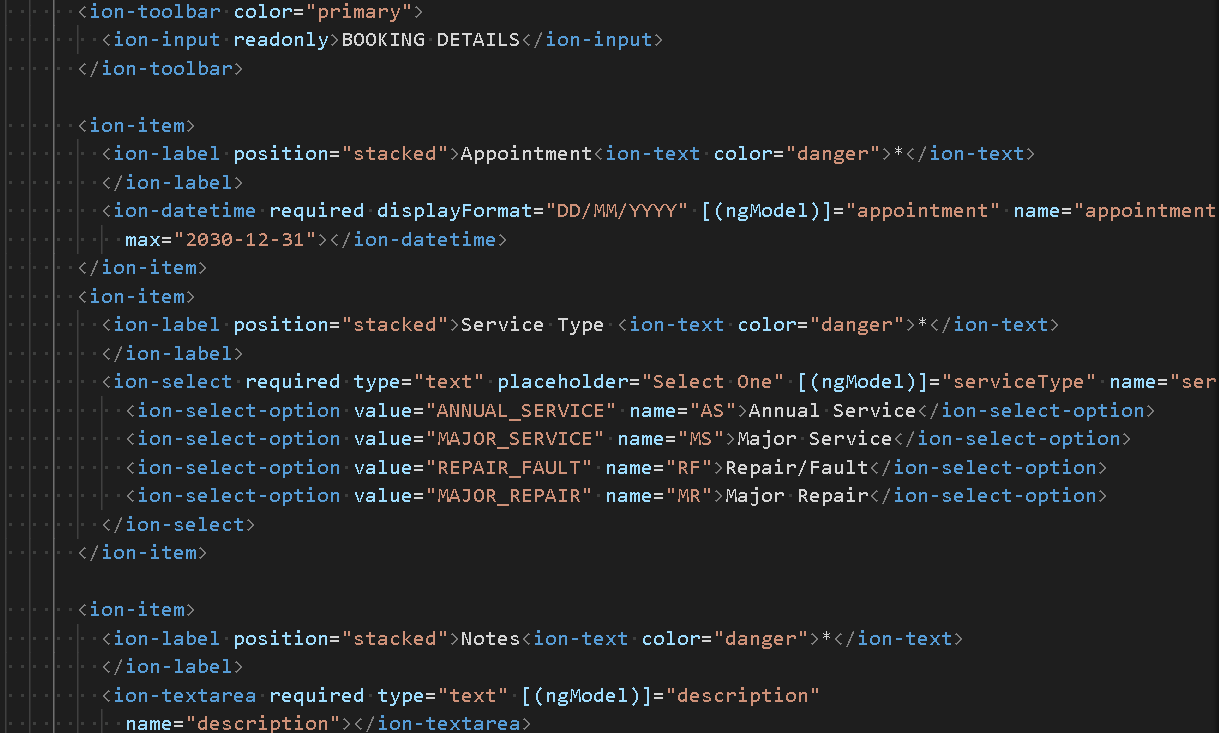
## Front-End

The front-end is the biggest challenge to build this web application. At the beginning, it took more time than estimated, leading to lack of time. However, after a considerable time spent to learn how to connect front and back-end and make the Ionic pages call the first endpoint, it became reasonably simple. Since all the core business is kept on the server side and exposed through WS resources, the front-end were designed and created only to send requests and get the responses.



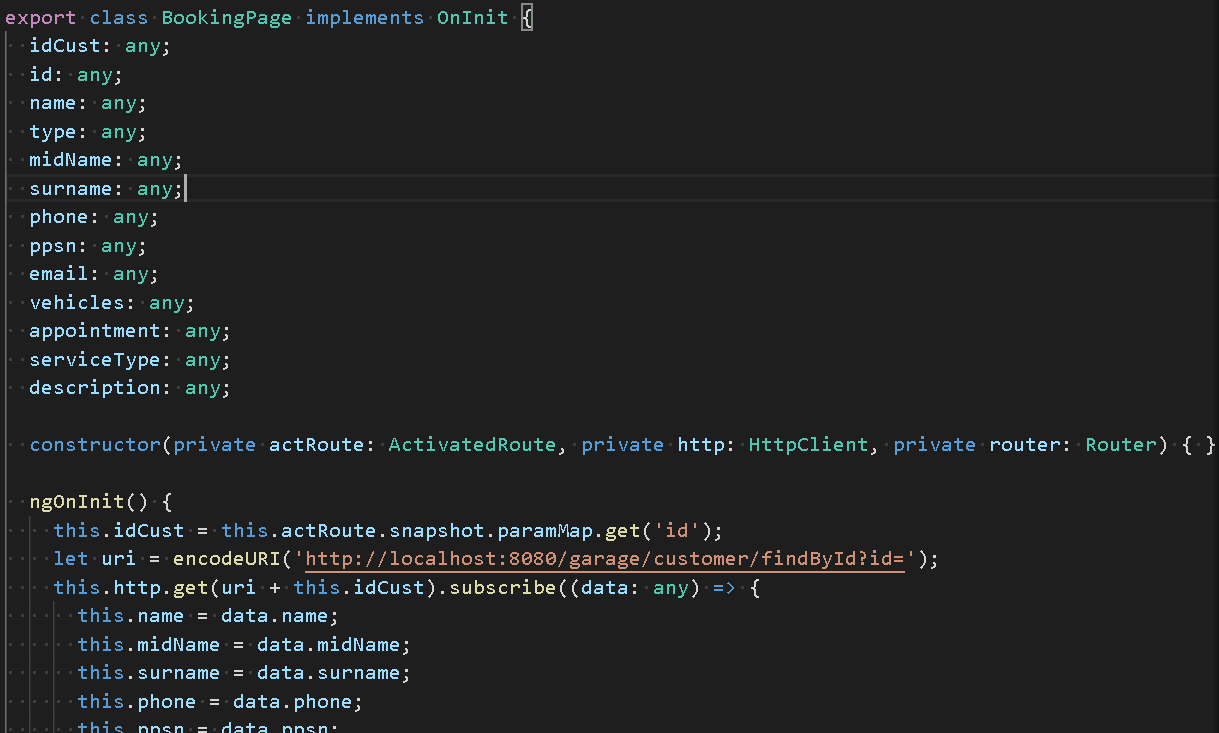
#### Visual Studio Code package structure

As planned, the front-end works apart of the back. See below a chunk of booking page.



#### Booking.page.html

The previous code represents what is loaded on the user browser. On the other hand, ionic use its own back-end side. Following the image below, the frontend is calling the customer resource.

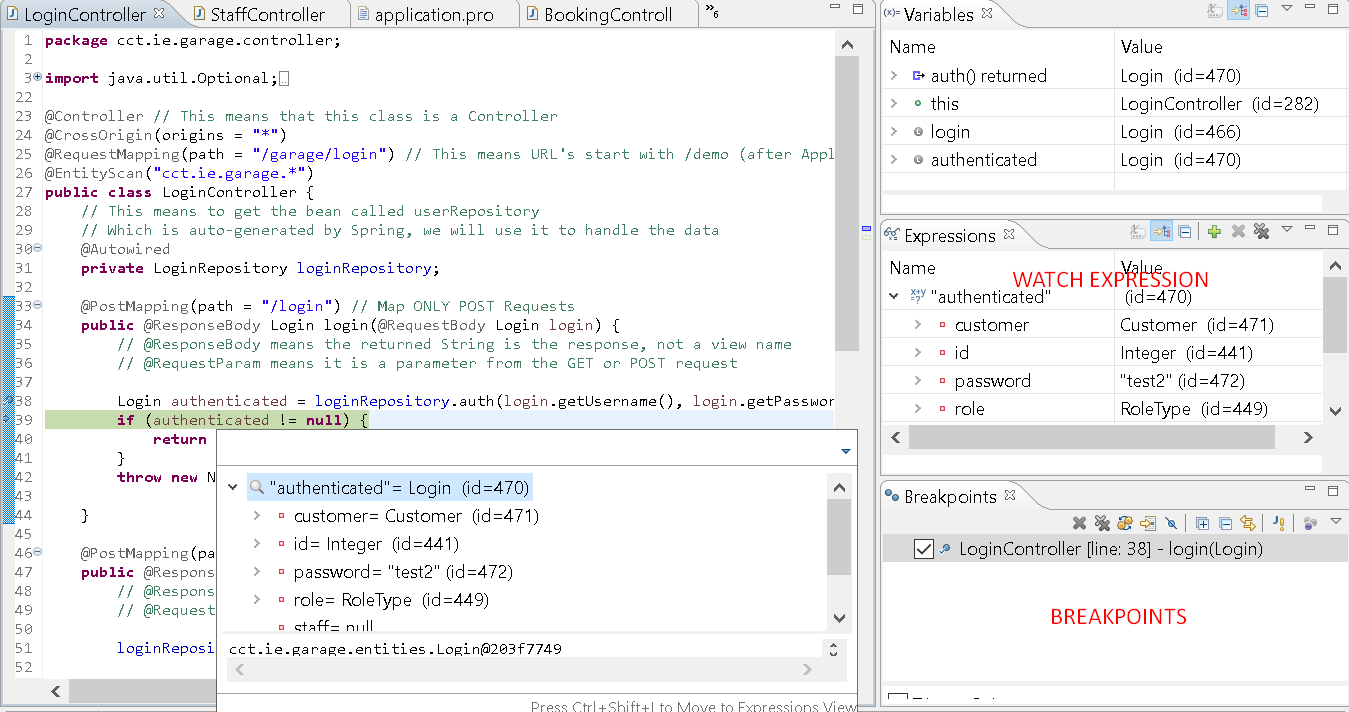


#### Booking.page.ts

# CHAPTER V: TESTING & EVALUATION

This section aim to describe the Ger’s Garage Web Application testing and evaluation process. It is very important to highlight that on the entire project life cycle, runtime tests were made. On the back-end and database, Postman and Eclipse debugging have helped on the development.

Using the Debug mode on Eclipse, this powerful tool helps the developer to look into and identify the error at run time. See below some functionalities.



#### Eclipse Debug

See on the illustration above few functionalities given on this mode. On the breakpoint tab is possible to identify: Class (LoginController), line where the interruption point has been placed (Line 38).

On the Expressions Tab is been shown an expression to be watched. Due to it, every time this expression is updated will be possible visualise. And at last, on the main tab, in this case, “LoginController”, when placing the cursor on the variable or object an pop up shows up with the values, type, quantity and others relevant details.

## ACCESS TO THE SYSTEM TEST

## Register form

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST INPUT** | **EXPECTED RESULT** | **CURRENT RESULT** | **COMMENTS** |
| New Customer Register | Save information on Customer Table and its relations, redirect to customer page and populate fields with saved details | Complete |  |
| New Customer Register form vaidation | Save information on Customer Table and its relations, redirect to customer page and populate fields with saved details | Customer stays at the same page. | There is validation but not message error. |
| New Staff Register | Admin inputs staff details and redirected to admin page | Complete |  |
| New Customer Register form vaidation | Save information on Customer Table and its relations, redirect to customer page and populate fields with saved details | Admin stays at the same page. | There is validation but not message error. |

## Login Form

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST INPUT** | **EXPECTED RESULT** | **CURRENT RESULT** | **COMMENTS** |
| New Customer / Staff / Admin Login | Chech account and redirect to the customer page | Complete |  |
| New Customer / Staff / Admin Login form vaidation | Message in case mandatory fields are not fill up | Customer stays at the same page. | There is validation but not message error. |

## Booking Form

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST INPUT** | **EXPECTED RESULT** | **CURRENT RESULT** | **COMMENTS** |
| Customer Booking Page | Save booking details on Booking Table and its relations, redirect to booking page and populate fields with saved details | Complete |  |
| Customer Booking Page form vaidation | After saving, rediretc to the booking view page | Customer stays at the same page. | There is validation but not message error. |
| Admin / Staff Booking Page | Go to booking view | Staff Complete | Admin booking view functionality not copleted. However, it is not a problem since staff workflow is working and he can do the same action of the admin role on this page. |

# CHAPTER VI: CONCLUSION & FURTHER WORK CONCLUSION

This project aim to achieve Ger’s garage needs. Even though I haven’t implemented all required functionalities, I feel relieved and comfortable with what I have achieved. On this project, I have been able to reach most of web application requirements and also overcome the challenges applied by myself.

It is clear, few functionalities are still missing and somehow it is not completed. However, as explained on the Chapter 1, it doesn’t affect much the application since it is micro service based. Due to it, this web application is still functional.

Moreover, I still intend to implement those missing functionalities, increase and improve database and core business. Plus, keep studying and researching about Ionic, Angular and others front-end tools, frameworks and languages because it is the area where I have spent more time and where I am still not felling completely comfortable.

Also, this project helped to catch up with old technologies which I have been away a while.

In my opinion, it has been very challenging. First of all, because of the project complexity and short time to achieve all requirements. It is important to highlight that, due to Corona Virus, made things a bit harder.

When I started the course I assumed it would be a bit challenging. However, after facing different errors, dealing with new technologies, running against time, I have realised it has been further overwhelming than supposed.

Although, I could find, analyse and solve all issues which has appeared. I have also overcome all challenges, caught up with technologies I knew already and learnt new ones, such as Ionic and Angular.

After one year, facing all problems, issues and challenges and going always beyond. Keeping my head up trying to find a solution and replanting when going towards obstacles and at the end being able to release this web application and the final report, I can say: I have one this battle.

# FURTHER WORK

Next Steps (Fixes):

* Complete missing functionalities: Admin booking view, messages error validation, print invoice;
* Host the application on cloud (AWS);

Next Steps (Improvements):

For the next version, new functionalities will be implemented:

* Payment Method;
* E-invoice;

# APPENDIX A: SOURCE CODE

All the source can be found on:

Documentation: <https://github.com/fampinho/garage/tree/master/Documentation>

Back-end: <https://github.com/fampinho/garage>

Front-end: <https://github.com/fampinho/garage-front>

## DATABASE

CREATE TABLE `booking` (

`id` int NOT NULL AUTO\_INCREMENT,

`type` varchar(14) NOT NULL,

`appointment` timestamp NOT NULL,

`description` varchar(50) NOT NULL,

`status` varchar(21) NOT NULL,

`staff\_id` int NOT NULL,

`customer\_id` int NOT NULL,

`vehicle\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `FK\_staff\_booking\_idx` (`staff\_id`),

KEY `FK\_customer\_booking\_idx` (`customer\_id`),

KEY `FK\_vehicle\_booking\_idx` (`vehicle\_id`),

CONSTRAINT `FK\_customer\_booking` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`),

CONSTRAINT `FK\_vehicle\_booking` FOREIGN KEY (`vehicle\_id`) REFERENCES `vehicle` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=39 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `car\_licence` (

`vehicle\_id` int NOT NULL,

`licence\_number` varchar(10) NOT NULL,

`manufacture` varchar(4) NOT NULL,

PRIMARY KEY (`vehicle\_id`),

UNIQUE KEY `vehicle\_id\_UNIQUE` (`vehicle\_id`),

KEY `FK\_vehicle\_car\_licence\_idx` (`vehicle\_id`),

CONSTRAINT `FK\_vehicle\_car\_licence` FOREIGN KEY (`vehicle\_id`) REFERENCES `vehicle` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `customer` (

`id` int NOT NULL AUTO\_INCREMENT,

`ppsn` varchar(9) NOT NULL,

`phone` varchar(10) NOT NULL,

`email` varchar(50) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=96 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `customer\_name` (

`customer\_id` int NOT NULL,

`name` varchar(10) NOT NULL,

`mid\_name` varchar(20) DEFAULT NULL,

`surname` varchar(20) NOT NULL,

PRIMARY KEY (`customer\_id`),

KEY `FK\_customer\_customer\_name\_idx` (`customer\_id`),

CONSTRAINT `FK\_customer\_customer\_name` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `login` (

`id` int NOT NULL AUTO\_INCREMENT,

`username` varchar(20) NOT NULL,

`password` varchar(20) NOT NULL,

`role` varchar(8) NOT NULL,

`customer\_id` int DEFAULT NULL,

`staff\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `FK\_customer\_login\_idx` (`customer\_id`),

KEY `FK\_staff\_login\_idx` (`staff\_id`),

CONSTRAINT `FK\_customer\_login` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`),

CONSTRAINT `FK\_staff\_login` FOREIGN KEY (`staff\_id`) REFERENCES `staff` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=33 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `service` (

`id` int NOT NULL AUTO\_INCREMENT,

`type` varchar(30) NOT NULL,

`date\_in` timestamp NOT NULL,

`date\_out` timestamp NOT NULL,

`price` float NOT NULL,

`extra\_price` float DEFAULT NULL,

`extra\_service` varchar(200) DEFAULT NULL,

`booking\_id` int NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `FK\_booking\_service\_idx` (`booking\_id`),

CONSTRAINT `FK\_booking\_service` FOREIGN KEY (`booking\_id`) REFERENCES `booking` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `staff` (

`id` int NOT NULL AUTO\_INCREMENT,

`ppsn` varchar(9) NOT NULL,

`salary` float NOT NULL,

`is\_available` tinyint(1) NOT NULL,

`service\_counter` int DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

UNIQUE KEY `ppsn\_UNIQUE` (`ppsn`)

) ENGINE=InnoDB AUTO\_INCREMENT=27 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `staff\_name` (

`staff\_id` int NOT NULL,

`name` varchar(10) NOT NULL,

`mid\_name` varchar(20) DEFAULT NULL,

`surname` varchar(20) NOT NULL,

PRIMARY KEY (`staff\_id`),

UNIQUE KEY `id\_UNIQUE` (`staff\_id`) /\*!80000 INVISIBLE \*/,

KEY `FK\_staff\_staff\_name\_idx` (`staff\_id`),

CONSTRAINT `FK\_staff\_staff\_name` FOREIGN KEY (`staff\_id`) REFERENCES `staff` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `stock` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(30) NOT NULL,

`quantity` int NOT NULL,

`date\_in` timestamp NOT NULL,

`date\_out` timestamp NULL DEFAULT NULL,

`unit\_price` float NOT NULL,

`service\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `FK\_service\_stock\_idx` (`service\_id`),

CONSTRAINT `FK\_service\_stock` FOREIGN KEY (`service\_id`) REFERENCES `service` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

CREATE TABLE `vehicle` (

`id` int NOT NULL AUTO\_INCREMENT,

`model` varchar(16) NOT NULL,

`maker` varchar(16) NOT NULL,

`fuel` varchar(8) NOT NULL,

`type` varchar(9) NOT NULL,

`customer\_id` int NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `FK\_customer\_vehicle\_idx` (`customer\_id`),

CONSTRAINT `FK\_customer\_vehicle` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=75 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

---

INSERT INTO booking (id,type,appointment,description,status,staff\_id,customer\_id,vehicle\_id) VALUES (4,'ANNUAL\_SERVICE',{ts '2020-07-05 00:00:00.'},'description test1','IN\_SERVICE',14,51,null);

---

INSERT INTO booking (id,type,appointment,description,status,staff\_id,customer\_id,vehicle\_id) VALUES (5,'ANNUAL\_SERVICE',{ts '2020-07-05 00:00:00.'},'description test1','BOOKED',5,52,null);

---

INSERT INTO booking (id,type,appointment,description,status,staff\_id,customer\_id,vehicle\_id) VALUES (6,'MAJOR\_REPAIR',{ts '2020-08-07 00:00:00.'},'description test6','BOOKED',6,56,null);

---

INSERT INTO booking (id,type,appointment,description,status,staff\_id,customer\_id,vehicle\_id) VALUES (7,'REPAIR\_FAULT',{ts '2020-07-18 00:00:00.'},'description test6','CANCELED',7,51,null);

---

INSERT INTO booking (id,type,appointment,description,status,staff\_id,customer\_id,vehicle\_id) VALUES (8,'ANNUAL\_SERVICE',{ts '2020-08-07 00:00:00.'},'FIRST BOOKING','BOOKED',6,38,1);

---

INSERT INTO booking (id,type,appointment,description,status,staff\_id,customer\_id,vehicle\_id) VALUES (9,'MAJOR\_SERVICE',{ts '2020-08-07 00:00:00.'},'SECOND BOOKING','BOOKED',5,38,1);

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (1,'1-xyzIE01','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (24,'1-xyzIE24','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (25,'1-xyzIE25','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (28,'1-xyzIE28','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (29,'1-xyzIE29','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (30,'1-xyzIE30','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (36,'1-xyzIE36','2020');

---

INSERT INTO car\_licence (vehicle\_id,licence\_number,manufacture) VALUES (37,'1-xyzIE37','2020');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (38,'1045814BA','0873654123','1045814BA@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (39,'1104581BC','0873654155','1104581BC@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (40,'0945814ZA','0873670123','0945814ZA@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (46,'abc001','00000001','test@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (47,'abc002','00000002','test2@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (48,'abc003','00000003','test2@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (51,'abc003','00000003','test2@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (52,'abc003','00000003','test2@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (53,'abc0053','000000053','test53@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (56,'abc0056','00000003','abc0056@gmail.com');

---

INSERT INTO customer (id,ppsn,phone,email) VALUES (57,'abc0057','00000003','abc0057@gmail.com');

---

INSERT INTO customer\_name (customer\_id,name,mid\_name,surname) VALUES (38,'Fabio','Marques','Pimentel');

---

INSERT INTO customer\_name (customer\_id,name,mid\_name,surname) VALUES (39,'Albert',null,'Einstein');

---

INSERT INTO customer\_name (customer\_id,name,mid\_name,surname) VALUES (40,'Leonardo','','da Vinci');

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (1,'test','test2','CUSTOMER',38,null);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (10,'Albert','Einstein','CUSTOMER',39,null);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (14,'Leonardo','DaVinci','CUSTOMER',40,null);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (21,'staff1','staff1','STAFF',null,5);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (22,'admin','admin','ADMIN',null,null);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (23,'staff2','staff2','STAFF',null,6);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (27,'staff3','staff3','STAFF',null,7);

---

INSERT INTO login (id,username,password,role,customer\_id,staff\_id) VALUES (28,'staff4','staff4','STAFF',null,14);

---

INSERT INTO staff (id,ppsn,salary,is\_available,service\_counter) VALUES (5,'staff01',1800.0,0,10);

---

INSERT INTO staff (id,ppsn,salary,is\_available,service\_counter) VALUES (6,'abc006',1800.0,0,10);

---

INSERT INTO staff (id,ppsn,salary,is\_available,service\_counter) VALUES (7,'abc007',1800.0,0,9);

---

INSERT INTO staff (id,ppsn,salary,is\_available,service\_counter) VALUES (14,'ABC008',1800.0,0,9);

---

INSERT INTO staff\_name (staff\_id,name,mid\_name,surname) VALUES (5,'Staff1','','Staff Surname1');

---

INSERT INTO staff\_name (staff\_id,name,mid\_name,surname) VALUES (6,'Staff2','Mid Name 2','Staff Surname2');

---

INSERT INTO staff\_name (staff\_id,name,mid\_name,surname) VALUES (7,'Staff3',null,'Staff Surname3');

---

INSERT INTO staff\_name (staff\_id,name,mid\_name,surname) VALUES (14,'Staff4',null,'Staff Surname4');

---

INSERT INTO stock (id,name,quantity,date\_in,date\_out,unit\_price,service\_id) VALUES (4,'Break oil',5,{ts '2020-07-06 00:00:00.'},null,10.0,null);

---

INSERT INTO stock (id,name,quantity,date\_in,date\_out,unit\_price,service\_id) VALUES (5,'Front Torch',5,{ts '2020-07-06 00:00:00.'},null,50.0,null);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (1,'A4','Audi','ELETRIC','CAR',38);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (23,'4.0','Wolks','ELETRIC','CAR',39);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (24,'Uno','Fiat','ELETRIC','CAR',40);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (25,'Uno','Fiat','ELETRIC','CAR',48);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (28,'Uno','Fiat','ELETRIC','CAR',51);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (29,'Uno','Fiat','ELETRIC','CAR',52);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (30,'Uno','Fiat','ELETRIC','CAR',53);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (36,'Compressor','BMW','PETROL','CAR',56);

---

INSERT INTO vehicle (id,model,maker,fuel,type,customer\_id) VALUES (37,'Compressor','BMW','PETROL','CAR',57);

---

# REFERENCES

MySQL, (2020). MySQL Tutorial. Retrieved 2020, MySQL Data Types: <https://www.mysqltutorial.org/mysql-data-types.aspx/>

BeginnersBook, (2020). BeginnersBook. Retrieved 2020, Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database:

<https://beginnersbook.com/2015/05/normalization-in-dbms/>

Fadatare, (2018). DZone. Retrieved 2020, All JPA Annotations: Mapping Annotations - Let's take a look at ALL the JPA mapping annotations:

<https://dzone.com/articles/all-jpa-annotations-mapping-annotations>

Dupire, (2020). Stackabuse. Retrieved 2020, Guide to JPA with Hibernate: Basic Mapping:

<https://stackabuse.com/guide-to-jpa-with-hibernate-basic-mapping>

Fejer, 2020. Baeldung. Retrieved 2020. Mapping a Single Entity to Multiple Tables in JPA:

<https://www.baeldung.com/jpa-mapping-single-entity-to-multiple-tables>

Stackoverflow, 2020. Stackoverflow. Retrieved 2020, How can I remove a specific item from an array?

<https://stackoverflow.com/questions/5767325/how-can-i-remove-a-specific-item-from-an-array>

Schwaber and Sutherland, 2017. Scrumguides.com. Retrieved on 2020, The Scrum Guide:

<https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100>

Scrum Org, 2020. Scrum.org. Retrieved on 2020. Kanban Guide for Scrum Team:

<https://www.scrum.org/resources/kanban-guide-scrum-teams>

Oracle, 2013. Oracle.com. Retrieved on 2020. The Java EE 6 Tutorial

<https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>

Ugarte, 2020. Baeldung.com. Retrieved on 2020. Building a Web Application with Spring Boot and Angular:

<https://www.baeldung.com/spring-boot-angular-web>

Ionic, 2020. Ionicframework.com. Retrieved on 2020. Ionic Framework:

<https://ionicframework.com/docs>