

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Anton Nikiforov 179852IADB

Development of Vehicle Sales Web Application

Bachelor's thesis

Supervisor: Aleksei Talisainen
Master's Degree

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Anton Nikiforov 179852IADB

Automüügi veebirakenduse arendamine

Bakalaureusetöö

Juhendaja: Aleksei Talisainen
Magistrikraad

Tallinn 2023

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Anton Nikiforov

07.05.2022

Abstract

The goal of the thesis is to create a web application, which would allow to quickly put any vehicle registered in Estonia for sale by entering just numberplate. The rest of the data related to vehicle should be loaded directly from Estonian Transport Administration and saved into database.

Thesis includes analysis of data openly provided by Transport Administration to confirm existence of problem that vehicles do not get utilized in a proper and safe way and are not being used, thus staying somewhere and provoking vandalism and damage to environment.

An overview of used technology stack is provided as well as reasons for selection. Author describes functional requirements of application and continues with development using chosen technologies. Part of thesis is dedicated to deployment of application into the cloud service. All parts related to development include examples of program code and commands that were used to give overview of used patterns and techniques.

Final part of thesis summarizes the results and describes possible future development to make application better fit for live environment.

This thesis is written in English and is 54 pages long, including 9 chapters, 36 figures and 13 tables.

Annotatsioon

Käesoleva töö eesmärgiks on luua lihtsa veebiportaali, mille abiga saaks kiirelt panna müügile Eesti Maanteeametis arvel olevaid sõidukeid, sisestades vaid sõiduki registreerimisnumbri. Portaal suhtleb otse Maanteeameti süsteemiga ning kõik autoga seotud andmed laetakse sealt automaatselt ning neid salvestatakse andmebaasi.

Töös on teostatud ametlike andmete analüüs, mis kinnitab et umbes neljandik arvel olevaid autodest ei saa ametlikult liikluses osaleda, kuna nende registreerimine on peatatud aga samas neid ei ole lammutatud või täielikult registrist kustutatud. See tähendaks seda, et nemad seisavad mõnes kohas, kus võivad provotseerida vandalismi või looduse reostamist. Veebiportaali peamine eesmärk ja eelis konkurentide eest on pakkuda tasuta ja kiiret lahendust nendele kes ei soovi kaua aega tegeleda vana auto müügiga.

Töös on olemas ülevaade arendamiseks kasutatud tehnoloogiatest ning on välja toodud peamised põhjused miks nemad olid välja valitud. Lõputöö autor kirjeldab rakenduse kasutaja nõuded ning arendab lahendused välja valitud tehnoloogiate abiga. Iga tehnoloogia kohta on välja kirjutatud olulised näidised, mis näitavad kuidas veebirakendus oli koostatud.

Lahti kirjutatud on ka rakenduse taristu koosseis ning kuidas esi rakendus on ühendatud taga rakendusega. Eraldi peatükk on kirjutatud et näidata peamised käsud mis on seotud rakenduse veebi ülespanemisega.

Viimases osas kirjeldab autor võimalikud edasiarendused mida tuleb teha et saaks rakenduse live-keskkonnas kasutada.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 54 leheküljel, 9 peatükki, 36 joonist, 13 tabelit.

Acknowledgements

I would like to express gratitude to my family for their unconditional, endless love and support in everything that I am doing, without it I would not have gone that far.

I am thankful to my motivational colleagues at work for being patient with my studies and allowing me concentrate on university when it was needed.

Huge thank you goes to Aliis Udras and Joel Jesse from Estonian Transport Department for making this thesis possible from the beginning by confirming and coordinating usage of department's system.

I express my acknowledgement to Riina Tallo and Dagmar Tamme, who endlessly assisted me in organizational questions and guided me throughout this journey.

Last but not least, I say thanks to my supervisor, Aleksei Talisainen, for giving invaluable advice required for completing this work in first place, and to pre-defence commission for their great feedback to polish it into final version.

List of abbreviations and terms

ACID	<i>Atomicity, Consistency, Isolation, Durability</i>
API	<i>Application Programming Interface</i>
AVP	Andmevahetusplatvorm
CI/CD	<i>Continuous Integration / Continuous Delivery</i>
CRUD	<i>Create, Read, Update, Delete</i>
CSV	<i>Comma-separated values</i>
DOM	<i>Document Object Model</i>
DTO	<i>Data Transfer Object</i>
ER	<i>Entity Relationship</i>
ETA	<i>Estonian Transport Administration</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IDE	<i>Integrated Development Environment</i>
JPA	<i>Jakarta Persistence API</i>
JSON	<i>JavaScript Object Notation</i>
JWT	<i>JSON Web Token</i>
kW	<i>Kilowatt</i>
LKF	Liikluskindlustusfond
MTR	Majandustegevuse register
ORM	<i>Object Relational Mapping</i>
OS	<i>Operational System</i>
POJO	<i>Plain Old Java Object</i>
SaaS	<i>Software as a Service</i>
SQL	<i>Structured Query Language</i>
SSH	<i>Secure Shell Protocol</i>
SUV	<i>Sport Utility Vehicle</i>
UI	<i>User Interface</i>
URL	<i>Uniform Resource Locator</i>
VIN	<i>Vehicle Identification Number</i>
VSP	<i>Vehicle Sales Platform</i>
XML	<i>Extensible Markup Language</i>

Table of contents

1 Introduction	14
2 Problem background and analysis.....	16
2.1 Goal	16
2.2 Data source analysis	17
2.3 Market analysis	23
2.4 Practical scope.....	24
3 Analysis of chosen technology stack	25
3.1 Backend - Spring Boot.....	25
3.2 Frontend – React, Bootstrap.....	26
3.3 Database – PostgreSQL.....	26
3.4 Infrastructure	27
3.4.1 Shared web hosting	27
3.4.2 Cloud platform	28
3.5 Data Exchange Platform (AVP).....	28
4 Application requirements	32
4.1.1 Advertisement creation process	34
4.1.2 Limitations of administrator role	35
4.1.3 Assigning badges to vehicles	36
5 Development of application and features.....	37
5.1 Design of database	37
5.2 Implementation of application domain	40
5.2.1 One-To-Many relationship set-up	41
5.2.2 Many-To-Many relationship set-up	42
5.3 Implementation of database connectivity.....	43
5.3.1 Repository pattern	43
5.3.2 Implementation of service.....	44
5.4 Implementation of API controllers.....	44
5.4.1 List of endpoints.....	45
5.4.2 Data Transfer Objects	46

5.4.3 Security	48
5.5 Implementation of Frontend using Bootstrap and ReactJS	49
5.5.1 Bootstrap design.....	50
5.5.2 Implementation of React functional components	50
5.5.3 Usage of ReactJS hooks	50
5.5.4 Usage of React Router for URL mapping.....	51
5.5.5 Usage of Axios to perform calls against API.....	51
5.5.6 Application structure	52
6 Infrastructure and deployment of application	53
6.1 Google Cloud SQL.....	53
6.2 Google Cloud Run.....	53
6.3 Google Cloud Storage	55
6.4 Cloud services-related setup in Spring Boot “application.properties” file	56
6.5 Frontend deployment.....	57
7 Testing of application.....	58
7.1 API testing during development using Postman	58
7.2 System testing of ready product	60
8 Results and possible future improvements.....	62
8.1 Future improvements to be considered	62
8.1.1 Captcha.....	62
8.1.2 Protection of customer data.....	63
8.1.3 Multilingual support.....	63
8.1.4 Login via existing services.....	63
8.1.5 Messaging between parties	63
8.1.6 Support for paid advertisements by 3 rd parties.....	63
8.1.7 Mobile app for Android and iOS	63
8.1.8 More details for advertisement page	64
8.1.9 Uptime monitoring	64
8.1.10 Automatic tests.....	64
8.1.11 Code refactoring.....	65
9 Summary	66
10 References	68
Appendix 1 – Screenshots of VSP interface	73

Appendix 2 – Non-exclusive licence for reproduction and publication of a graduation
thesis.....77

List of figures

Figure 1. Data example from dataset "Vehicle statuses in Estonia".	19
Figure 2. SQL script to retrieve total amount of vehicles from dataset.	20
Figure 3. SQL script to retrieve number of registered vehicles from dataset.	20
Figure 4. SQL script to retrieve number of vehicles with problematic registration.	21
Figure 5. Distribution of vehicles per status.	22
Figure 6. Average age of cars in Estonian registry by their status.	22
Figure 7. Distribution of cars on sale on auto24.ee by price span.	24
Figure 8. Flowchart of advertisement creation process.	35
Figure 9. Entity Relationship Diagram for VSP database.	38
Figure 10. Example of OneToMany mapping in Spring Boot.	41
Figure 11. Example of ManyToOne mapping in Spring Boot.	42
Figure 12. Example of ManyToMany mapping in Spring Boot.	42
Figure 13. Example of connecting table between Vehicle and Badge entities.	43
Figure 14. Example of Repository implementation in SpringBoot.	43
Figure 15. Example of service implementation in Spring Boot.	44
Figure 16. Visual example of data transfer object composition.	47
Figure 17. Visual representation of data transfer object creation.	48
Figure 18. Example of JWT communication between client and server.	49
Figure 19. Example of "useState" hook usage. Variable is created, setter method attached, initial value (empty array) is set.	50
Figure 20. Example of "useEffect" hook usage. Service to get user advertisement is called only once when page is loaded, response is being set to state variable "advertisements". In case of possible error, error details are also saved to variable "content".	51
Figure 21. Example of React route setup. Router maps URL to specific React component.	51
Figure 22. Example of method that utilizes Axios HTTP client to return data from API endpoint.	52
Figure 23. Example of Jib setup in "build.gradle" file.	54

Figure 24. Selection of container image to be deployed in Google Cloud Run.....	55
Figure 25. Example of application image storing in Spring Boot using Google Cloud Storage.....	56
Figure 26. Images uploaded from VSP as seen in bucket (directory) hosted in Google Cloud Storage.....	56
Figure 27. Postman response received from /api/advertisement/:id endpoint with detailed vehicle information.....	59
Figure 28. Example of Postman collections with sample saved requests to various endpoints.	60
Figure 29. Example of VSP interface - Home page with latest advertisements.	73
Figure 30. VSP interface - Top selection tab which displays badges assigned to vehicles.	73
Figure 31. VSP interface - Search page.	74
Figure 32. VSP interface - Open advertisement with photos and vehicle details.	74
Figure 33. VSP interface - Second row of open advertisement. Shows price, vehicle details, owner contacts and technical data on the right.	75
Figure 34. VSP interface - Third row of open advertisement. Shows details about technical inspections, registry actions and restrictions.	75
Figure 35. VSP interface - Administrator profile. Administrator sees every advertisement and is able to edit or remove it.....	76
Figure 36. VSP interface - edit screen of existing advertisement.	76

List of tables

Table 1. Dataset columns with explanation.	17
Table 2. Explanation of vehicle category abbreviations.	18
Table 3. Explanation of vehicle statuses.	19
Table 4. Number of cars on sale per price range.....	23
Table 5. Overview of packages provided by AVP.....	29
Table 6. Description of groups provided by AVP with list of returned parameters.	29
Table 7. List of required functionality to be developed.	32
Table 8. Description of badges that could be assigned to vehicle.....	36
Table 9. List of database tables with descriptions and relationship details.....	38
Table 10. List of supported API endpoints.	45
Table 11. ReactJS application directory list.....	52
Table 12. Contents of “src” directory in ReactJS application.....	52
Table 13. Google Cloud-specific setup in Spring Boot application.properties file.	57

1 Introduction

More than 135 years have passed since Karl Benz's Benz Patent-Motorwagen was unveiled for the public. Despite not being the first attempt at automobile creation, the invention of this vehicle was patented in 1886 and it is regarded as world's first car with internal combustion engine that was put into series production [1].

However, cars were not widely available until Henry Ford brought Ford Model T into the spotlight of mass attention back in 1908. The car was affordable not only for the rich but for middle class as well, and was being assembled using the production line, which made it possible to produce cars on massive scale, and at the same time to decrease their price [2]. Since then, number of cars in the world have started to grow significantly. In the Unites States alone, number of registered vehicles was 8000 in 1900 which grew into 458,377 in 1910 [3]. This is an incredible 5629,71% increase for just 10 years.

The general number of vehicles in the world keeps growing nowadays. Already in 2015 there were almost 1 billion personal vehicles in use [4]. On top of that, vehicles are being produced massively every year. There was a continuous growth of number of manufactured vehicles from 2010 up until Q4 2018, where a slight decline happened, but still year ended with 95 706 293 new vehicles produced [5].

With growing number of existing vehicles and new ones being produced every year, a question of utilization and recycling comes into play, as with any other type of consumer goods. Owners use their vehicles every day, sometimes during harsh winter conditions, sometimes during hot summer. The behavior of typical modern busy person also does not help to preserve vehicle in proper condition through the years, as people tend to neglect owner's manual, skip maintenance, use car in a non-intended way, which all speeds up the decline of mechanisms.

After the years, oil starts to leak, gearbox does not change gears as smooth as in year the car left dealership, additional equipment stops working and gives more problems than

assistance. The rust is making its way through the metal, and it gets more difficult every year to pass the inspection and keep car on the road.

It's not easy to understand, when is the right moment to utilize the car. It may have a small issue that can be fixed in the yard in spare time, like a burned-out bulb or broken door handle. It might be some issue that requires professional knowledge to be fixed, but still be affordable. But sometimes it happens that car is not anymore a safe place to be, for example when rust has destroyed the chassis. It may also happen that spending on old car exceeds forecasted spending on new one.

Some car owners do not have enough knowledge to see the whole picture, or do not have time or interest in this matter, and just buy another vehicle, sometimes letting the old one sit for months and years.

The author of this thesis strongly believes that life is movement and car should drive, and mechanisms work. Every vehicle should be used until logical end of its lifecycle, until full exhaustion of planned resource. If one does not need the vehicle anymore – it's time to think about giving it away to next owner. There are a lot of people out there who fancy old cars as a hobby, fixing them to be able to drive an old, electronic-free vehicle on sunny weekends, enjoying this very special experience, especially during modern times, when cars tend to look like gadgets – keeping driver in his lane, beeping to warn about not-always-very-important information. The car might be worthy of restoration in the hands of enthusiast, who will give it a new life as a garage relic. There are also people, who just want to have a cheap mean of transport for various reasons, be it for learning how to drive, for work appliance or as a temporary vehicle.

The problem is that it's not easy to connect these people - for owners, it's not easy to understand the real value of their vehicle, find time to analyze this and put advertisement online. Sometimes they are convinced that their car cost nothing at all and nobody will come for it. For potential buyers, searching for a proper low-budget vehicle in Internet is like searching for a needle in the stack - there are not much to choose from, most offers are from wholesalers and not owners, who do not know anything about car, the price is often too high, and history is shady. The problem exists in any country of the world, however, in the scope of this thesis we will focus on Estonia.

2 Problem background and analysis

According to Estonian Transport Administration statistics, as of 01.03.2022 there are approximately 1 227 000 vehicles registered in Estonia [6].

Out of this number, around 283 500 have either temporarily or permanently suspended registration, thus rendering them road illegal. Average age of these vehicles is 28,3 years. These vehicles are either illegally dismantled or are sitting in yards and fields, provoking vandalism and damage to environment. A more detailed analysis of official dataset provided by Estonian Transport Administration is conducted in sub-chapter “Data source analysis”.

As stated in introductory part, some of these vehicles can be historically valued and a true finding for restoration enthusiasts, others can be fixed with minor investments and used further until complete utilization.

The real objective is to find right hands which will take vehicle over. As recycling and reusage is very popular, an attempt must be made to use vehicle to full extent until end of lifecycle, thus ways to keep older cars on the road as much as possible should be promoted.

2.1 Goal

The goal of thesis is to create a simple, yet productive web application – VSP (*Vehicle Sales Platform*), which will allow owner of vehicle to quickly put it on sale by just providing numberplate and some photos. This functionality is not available in any other existing platform in Estonia. The application will be connected to Transport Administration API (*Application Programming Interface*) and will fetch all vehicle details just by supplying numberplate.

Buying side will also receive benefits in form of transparent and correct vehicle details in one place. The full overview of vehicle provided by the app will help to mitigate unfair vehicle sale practices, as some owners or resellers tend to hide real history of car.

Web application will focus on older and cheaper vehicles and will be free to use, in contrast to leaders of Estonian market [7] [8].

To reach the goal, an analysis will be performed based on data from official dataset, to understand how many vehicles are out there which need to be either re-used or recycled. By analyzing the data, it is planned to confirm the presence of problem and strengthen reason for implementation of practical part.

2.2 Data source analysis

This section describes the official dataset about vehicle statuses in Estonia and how the data was analyzed.

Author of thesis uses full dataset “Vehicle statuses in Estonia” that is issued every month by Estonian Transport Administration, which contains data about all vehicles registered in Estonia [6]. Some basic visual data is available on Transport Administration website [9] however this data is not sufficient for deeper analysis.

All data is actual as of 23.03.2022, while used dataset was created on 02.03.2022. Due to large size of dataset (158,73 MB) CSV (*Comma-separated values*) file was imported into MySQL database using application Sequel Pro into table “data”.

The available columns (names are in Estonian language) are described in Table 1.

Table 1. Dataset columns with explanation.

Name	Translation / Description
Soiduki_vanus	Age of vehicle
Maakond	County
Andmed_seisuga	Date when dataset information was acquired
ESMANE_REG_KP	Date of first registration
KYTUSEKOMBINATSIOON	Fuel combination
Kategooria	Category of vehicle ¹

¹ Explanation of possible abbreviations is available in Table 2. Data is based on Estonian Transport Administration official website [70].

Name	Translation / Description
Keretyyp	Style of car body
Kytuse_tyyp	Type of fuel used
MOOTORI_TYYP	Type of engine
Mark	Manufacturer
Mudel	Model
SOIDUK_ID_ID	Unique vehicle identifier
Staatus	Status of vehicle ¹
VARV	Colour

Other point of interest that would help our analysis are value in column ‘Kategoria’ – it helps to understand vehicle type. Table 2 provides description of abbreviations below.

Table 2. Explanation of vehicle category abbreviations.

Category abbreviation	Description
M	vehicles intended for carriage of passengers
N	vehicles intended for carriage of goods
L	two-, three- and four-wheel vehicles
T, R, C	agricultural and forestry vehicles
O	trailers

Last and main interest is column ‘Staatus’ – it provides overview of possible statuses vehicle can own in Transport Administration database. As of 01.03.2022 they are 9 unique statuses. Unfortunately, there is no official description available for each of them, however statuses were analyzed by author of the thesis and comments added as part of this analysis. It was not possible to verify meaning of 5 out of 9 statuses, as they do not have clear explanation based on their name. An attempt was made to contact author of dataset to receive official confirmation, but it was unsuccessful.

¹ Explanation of possible vehicle statuses is available in Table 3.

Table 3. Explanation of vehicle statuses.

Status	Description
REGISTRIS_OLEV_SOIDUK	Registered vehicle (Active registration)
VOORANDATUD	Expropriated but not registered again
KANNE_PEATATUD	Stopped registration
AJUTISELT_REGISTRIST_KUSTUT	Suspended registration
REG_YLEVAATUSE_LABINUD_REGISTR	Unverified status, assigned to 1988 vehicles
AJUTISELT_REGISTREERITUD	Unverified status, assigned to 88 vehicles (temporarily registered)
EKSPERTIISI_KONTROLLIMINE	Unverified status, assigned to 91 vehicles
REGMUUDATUS_TYYBIKINNITUS	Unverified status, assigned to 2 vehicles
VOORANDATUD_RM_LABINUD	Unverified status, assigned to 1 vehicle

Figure 1 shows example of how data looks like in dataset after being imported into MySQL database.

Keretyyp	Kytuse_tyyp	MOOTORI_TYYP	Mark	Mudel	SOIDUK_ID_ID	Staatus	VARV
UNIVERSAAL	DIISEL	DIISEL	VOLVO	V60	100870006	REGISTRIS_OLEV_SOIDUK	MUST
SEDAAN	BENSIIN	BENSIIN_KATALYSAATOR	VOLKSWAGEN	JETTA	100150629	REGISTRIS_OLEV_SOIDUK	VALGE
LUUKPÄRA	BENSIIN	BENSIIN_KATALYSAATOR	MAZDA	6	2000504	REGISTRIS_OLEV_SOIDUK	TUMEHALL
LUUKPÄRA	BENSIIN	BENSIIN_KATALYSAATOR	HONDA	CIVIC TOURER	100559758	REGISTRIS_OLEV_SOIDUK	PRUUN
UNIVERSAAL	DIISEL	DIISEL	FORD	FOCUS	2061177	VOORANDATUD	HOBEDANE
MAHTUNIVERSAAL	DIISEL	DIISEL	TOYOTA	YARIS	2550827	REGISTRIS_OLEV_SOIDUK	VALGE
UNIVERSAAL	DIISEL	DIISEL	FORD	S-MAX	100565027	REGISTRIS_OLEV_SOIDUK	MUST
SEDAAN	BENSIIN	BENSIIN	BMW	520I	97584	KANNE_PEATATUD	HALL
SEDAAN	BENSIIN	BENSIIN_KATALYSAATOR	OPEL	OMEGA	467545	KANNE_PEATATUD	TUMEHALL
SEDAAN	BENSIIN	BENSIIN_KATALYSAATOR	TOYOTA	CAMRY	472934	VOORANDATUD	PUNANE
KAUBIK	DIISEL	DIISEL	RENAULT	TRAFIC	100859125	REGISTRIS_OLEV_SOIDUK	HALL
UNIVERSAAL	DIISEL	DIISEL	TOYOTA	AVENSIS	100919652	REGISTRIS_OLEV_SOIDUK	SININE
LUUKPÄRA	BENSIIN	BENSIIN_KATALYSAATOR	CITROEN	C5	1201632	REGISTRIS_OLEV_SOIDUK	HOBEDANE
KAUBIK	BENSIIN	BENSIIN_KATALYSAATOR	SKODA	PRAKTIK	2807625	REGISTRIS_OLEV_SOIDUK	PUNANE
SEDAAN	BENSIIN	BENSIIN_KATALYSAATOR	OPEL	VECTRA	543071	REGISTRIS_OLEV_SOIDUK	PUNANE
UNIVERSAAL	BENSIIN	BENSIIN_KATALYSAATOR	VOLKSWAGEN	TIGUAN	100564988	REGISTRIS_OLEV_SOIDUK	HALL
SEDAAN	DIISEL	DIISEL	PEUGEOT	508	100187296	REGISTRIS_OLEV_SOIDUK	VALGE
UNIVERSAAL	BENSIIN	BENSIIN_KATALYSAATOR	OPEL	FRONTERA	306760	REGISTRIS_OLEV_SOIDUK	ROHELINE
MAHTUNIVERSAAL	BENSIIN	BENSIIN_HYBRIID	LEXUS	RX400H	100310604	REGISTRIS_OLEV_SOIDUK	TUMEHALL

Figure 1. Data example from dataset "Vehicle statuses in Estonia".

In the scope of this thesis, author focuses on four-wheel vehicles intended for carriage of passengers, therefore only entries with category = M% will be analyzed.

Statuses of vehicle like REG_YLEVAATUSE_LABINUD_REGISTR, AJUTISELT_REGISTREERITUD, EKSPERTIISI_KONTROLLIMINE, REGMUUDATUS_TYYBIKINNITUS, VOORANDATUD_RM_LABINUD are unverified and quite specific as described above and are applied only to marginal number of vehicles, thus will be excluded from the results.

To perform calculations, some simple queries have been created to be run against contents of dataset. These queries are available on figures below.

SQL (*Structured Query Language*) script to get total number of vehicles is shown on Figure 2.

```
select count(*)
FROM data
WHERE
1 = 1
AND Kategooria like 'M%'
AND staatus NOT IN ('REG_YLEVAATUSE_LABINUD_REGISTR',
'AJUTISELT_REGISTREERITUD', 'EKSPERTIISI_KONTROLLIMINE',
'REGMUUDATUS_TYYBIKINNITUS', 'VOORANDATUD_RM_LABINUD');
```

Figure 2. SQL script to retrieve total amount of vehicles from dataset.

Total amount of such vehicles is: 839 104

As next step, it was required to find number of registered vehicles, example script is provided on Figure 3.

```
select count (*)
FROM data
WHERE
1 = 1
AND Categories like 'M%'
AND staatus = 'REGISTRIS_OLEV_SOIDUK';
```

Figure 3. SQL script to retrieve number of registered vehicles from dataset.

Which results in 615 507.

This means, that there are 223 597 (839 104 – 615 507) vehicles with possibly problematic registration.

To recheck the numbers another query was used, which is displayed on Figure 4.

```
select count(*)
FROM data
WHERE
1 = 1
AND Kategoria like 'M%'
AND staatus IN ('VOORANDATUD', 'KANNE_PEATATUD',
'AJUTISELT_REGISTRIST_KUSTUT');
```

Figure 4. SQL script to retrieve number of vehicles with problematic registration.

Which has returned same number : 223 597.

This means that 26,65% (223 597) of all vehicles (839 104) have either stopped registration (168 531), are expropriated (30 742), or have suspended registration (24 324)¹. While there is some hope that minor part of them is still in road-worthy condition and they can be still brought easily back to registry, it shows the general number of vehicles that are not used and not recycled.

Figure 5 shows distribution of vehicles per status.

¹ Further queries are omitted for brevity.

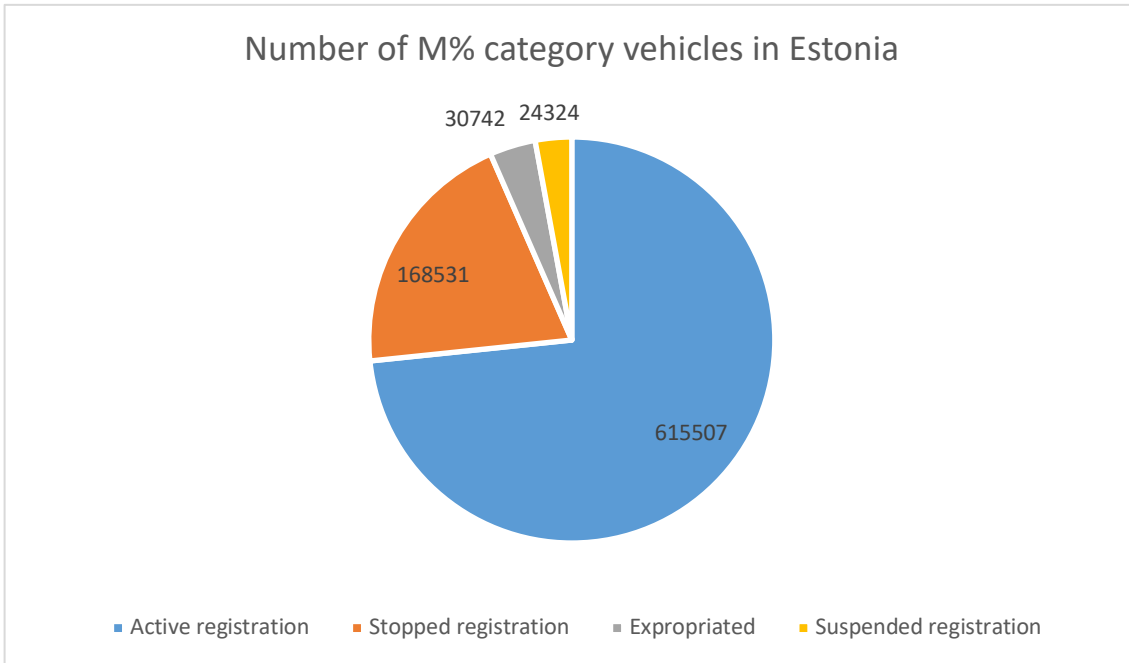


Figure 5. Distribution of vehicles per status.

The average age of these vehicles also cannot be compared to the ones in permanent usage. The average age of car with active registration is 13,14 years, while for cars with stopped, expropriated, and suspended registration the numbers are 29,64; 24; 21,09 respectively. Figure 6 below shows this difference graphically.

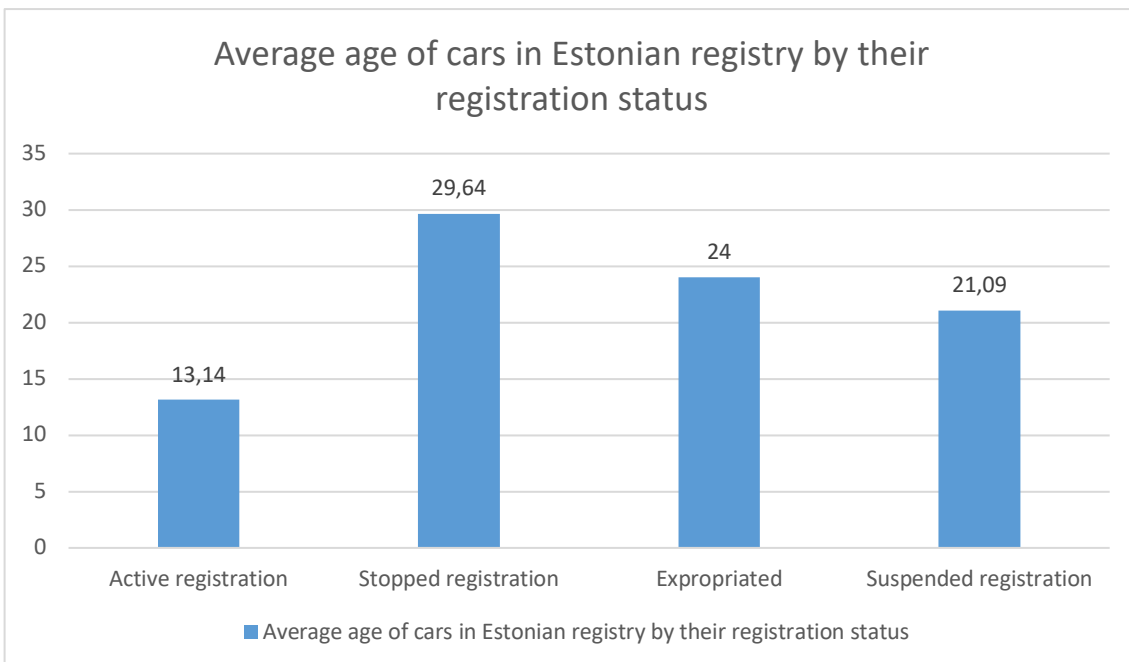


Figure 6. Average age of cars in Estonian registry by their status.

2.3 Market analysis

This chapter analyzes the current market state in Estonia using largest advertisement portal Auto24. Analysis helps to understand what pricing focus group is. Passenger and SUV (*Sport Utility Vehicle*) category as of 25.03.2022 contains 16 077 vehicles (number excludes advertisements without proper pricing, for example auctions or ads with missing price) [10]. Out of this amount the car advertisements are spread as shown in Table 4 below.

Table 4. Number of cars on sale per price range.

Price range	Number of cars on sale
Under €1000	161
€1000 – €1999	689
€2000 - €2999	944
€3000 - €3999	944
€4000 – €4999	967
€5000 – €9999	3632
€10000+	8740

Thus, total sum of cars priced up to €9999 is 7337, and total sum of cars costing over €10 000 is 8740. Graphically this data can be represented as shown on Figure 7 below.

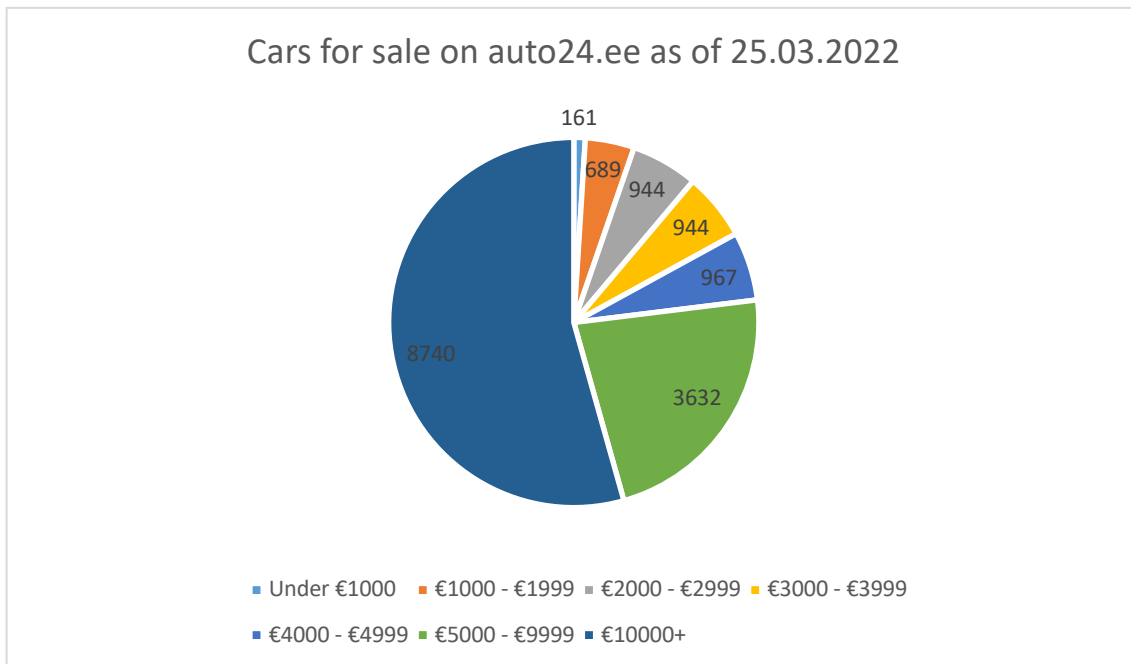


Figure 7. Distribution of cars on sale on auto24.ee by price span.

The graph shows that vehicles under €10 000 take 45,64% of market (7337 out of 16 077). Out of these 7337, 50,5% are in €1 - €4999 pricing range. (3705). This confirms the fact that number of cheaper cars is quite large and focus group of vehicles up to €10 000 can be taken.

2.4 Practical scope

Practical part of the thesis deals with development of basic version of application only. Practical outcome is not meant for immediate Production usage. Marketing, real-life usage and monetization of application fall out of current scope.

3 Analysis of chosen technology stack

Current chapter provides overview of technologies that were selected to create VSP. Main reason for their selection is general actuality at the moment of thesis writing. While comparison of frameworks and programming languages is not in scope of this thesis, some detailed reasons are provided along with short historical reference.

Chapter also includes short overview of AVP (Andmevahetusplatvorm) - Data Exchange Platform provided by Estonian Transport Administration – explaining data output and costs of usage.

3.1 Backend - Spring Boot

According to 2021 questionnaire by JetBrains, the developer behind IDE (*Integrated Development Environment*) IntelliJ IDEA, about 65% of developers are using Spring Boot to develop Java web applications [11].

Spring Boot is a back-end framework for rapid development - it provides base application and services that can be easily set up with minimum effort [12]. For the start it provides default configuration to be used out of the box, that can be customized as per specific application needs, thus allowing to write less repetitive code by just following known conventions [13].

The framework is flexible and easily extendable by using third-party libraries allowing to create apps with wide range of features. Security issues are usually dealt with quickly and patches provided once vulnerabilities are discovered. Last, but not least – there is huge community of developers around Spring Boot. This makes it easy to find tutorials, guides or help from open resources and fix any problem that arises when application is being developed.

All these reasons also contribute to fact that Spring is trusted by developers all over the world including the ones who work in big tech companies like Alibaba, Amazon, Google, and Microsoft [14].

Fast setup, availability of third-party libraries and extensive documentation are the main reasons behind framework selection for development of VSP – as development timeframe

is very limited, it is crucial to start development in fast pace and be able to find information on any possible problem that could arise.

3.2 Frontend – React, Bootstrap

Same questionnaire performed by JetBrains in 2021 for JavaScript shows that React is being number one most used framework to develop JavaScript applications, with second being Vue.js [15].

React allows to develop frontend applications quickly as it uses declarative views which make written code predictable and easy to debug. It's quite simple to build separate components that manage their state, then compose them to make a more complex user interface. State is kept out of the DOM (*Document Object Model*), allowing to pass data back and forth, thus updating UI (*User Interface*) effectively.

React also shares syntax with Node.js and React Native, which allow to use code on server-side and on mobile devices respectively [16].

Requirement for fast development of application meant that frameworks with steep learning curve could not be used on this occasion. React was selected mainly because of its high performance and component structure that will make code easier to maintain in longer run [17].

3.3 Database – PostgreSQL

PostgreSQL is a fully ACID (*Atomicity, Consistency, Isolation, Durability*) - compliant relational database that can run on every operating system, including Linux, Mac OS X and Microsoft Windows [18]. The initial project called POSTGRES (POST-Ingres) started back in 1986 but gained momentum in 1996 once initial QUEL query language was replaced with SQL and code began to be maintained by specific group of focused developers, who acquired a public server to host the code. The name was changed to PostgreSQL and since then the project is frequently updated and kept in good shape [19]. The project is solid and is being used for more than 25 years straight, during which a community has grown up, making it possible to find solutions to problems quickly.

According to recent survey of professional developers that was performed by StackOverflow, 44,08% of developers are using PostgreSQL for their projects, making it just a bit behind of MySQL (result 48,19%) [20]. The trend, however, is changing towards Postgre, as previous survey of 2020 showed numbers of 53,5% and 38,5% of MySQL and PostgreSQL usage respectively [21].

MySQL was considered as an option for database for VSP, but research showed that not only PostgreSQL uses less system resources, which is very important when running in the cloud, but also handles concurrent usage better and generally works faster in environments that require many heavy write operations [22]. As large data objects containing full vehicle history will be pushed into multiple tables at the same time and possibly by many users concurrently, it was decided to go for PostgreSQL. Moreover, development pace and bug fixing time has been slowed down since MySQL acquisition by Oracle in 2009, which slows issue reporting and bug fixing process, while PostgreSQL large and devoted community contribute often [23].

3.4 Infrastructure

This chapter briefly describes infrastructure around VSP – where frontend and backend applications are hosted, analysis of Cloud platform providers and selection reasoning.

3.4.1 Shared web hosting

Zone.ee is Estonian web hosting service provider since 1999 [24]. The pricing of shared web hosting service is competitive on local market and author of thesis uses Plan I for development which costs €6,55 monthly if billed annually. For this amount it provides SSH (*Secure Shell Protocol*) access and support for Node.js web applications [25].

As per comparison of web hosting providers, Zone.ee is best known provider in Estonia with great technical support and the only one which had an uptime of 100% during monthly test [26]. Also, author of thesis already had account there for hosting various other projects, so as a method to reduce costs of development and hosting, Zone.ee was selected to accommodate application frontend written in ReactJS.

3.4.2 Cloud platform

Backend of application in form of API and database will be stored in cloud platform.

Biggest players on cloud market are AWS, developed by Amazon, Azure, developed by Microsoft and Google Cloud, developed by Google. While Google Cloud is not market leader and has some drawbacks, it provides best prices for hosting small instances [27]. Moreover, it provides 300\$ in free credits for any signed-up account and has extensive documentation for setting up required services.

Google Cloud Run is one of Google Cloud services, that allows to deploy a containerized application written in Go, Python, Java, NodeJS or .NET and quickly make it available for incoming requests [28]. What is more important, it allows to build container from source Git¹ repository whenever updated code is pushed there.

Other Google services that will be used are Google Cloud SQL and Google Cloud Storage. More details are described in Chapter 6.

3.5 Data Exchange Platform (AVP)

The service is provided by Estonian Transport Administration. The purpose of it is to give access to public data by providing XML (*Extensible Markup Language*) response to queries.

The access to service must be requested beforehand and justified reason should be provided why data needs to be received and used. The service is paid.

ETA (*Estonian Transport Administration*) provides four options to choose from – Package 1 to 4. Packages 3 and 4 are for bulk queries to get details about various ETA operations with vehicles and are not suitable for development of application planned in the thesis, thus we will exclude them from overview and summarize possibilities only for packages 1 and 2. They mostly differ by number of allowed queries within specific timeframe. Summary of packages is provided in Table 5 below.

¹ Open-source distributed version control system for tracking changes in any set of files [69].

Table 5. Overview of packages provided by AVP.

Package 1	
Allowed queries in 1 minute	5
Allowed queries in 1 hour	15
Allowed queries in 24 hours	50
Monthly cost	€15
Package 2	
Allowed queries in 1 minute	10
Allowed queries in 1 hour	100
Allowed queries in 24 hours	300
Monthly cost	€80

As second package implies more intense usage, the price is higher (€80 monthly for package 2 and just €15 for package 1).

The query is created with one of 3 inputs (vehicle numberplate, vehicle VIN (*Vehicle Identification Number*) code, number of vehicle technical passport) and output can be constructed out of 5 possible groups (for brevity only most important details are provided in Table 6 below).

Table 6. Description of groups provided by AVP with list of returned parameters.

Group	Included parameters
1 - General details of vehicle	Manufacturer
	Model name
	Type of vehicle
	Modification
	Color
	Date of registration
	Date of registration in Estonia
	Country of initial registration
	Date of next technical inspection
2 – Technical details of vehicle	Engine type
	Engine capacity

Group	Included parameters
	Engine power
	Type of gearbox
	Number of doors
	Number of seats
	Weight
	Height
	Width
	Max load
	CO2 emissions
3 – Technical inspection details	Date of technical inspection
	Type of inspection
	Detailed list of faults discovered during inspection
	Odometer reading
	Performer of technical inspection (company name)
	Decision
	Date of next technical inspection
4 – Restriction data ¹	Type of restriction (loan, arrest)
	Setter of restriction
	Date of imposing
	Deposit sum (if vehicle used as deposit)
	Ranking (if restrictions are queued)
5 – Operation data ²	Type of operation
	Date

Each group is added to base package price at the cost of €15 per group [29].

¹ Section includes information about restrictions. Example of restriction is when a car is being used as deposit for loan or is arrested by law enforcement. If car is officially subject to any restrictions, it means that owner may not have rights to sell it. This information is used to provide more transparency.

² Section includes data about vehicle operations. Example of operation is change of ownership or issuing a duplicate numberplate.

For the development of application for the thesis, author have selected Package 1 and all 5 returnable groups. This would mean, that the service would cost €15 for package itself, plus $5 * €15$, totaling to €90 per month.

Before writing of the thesis, author has contacted Estonian Transport Administration and asked to provide free access for 3 months in order to write and defend thesis and received a positive reply. After signing an agreement between ETA and physical entity the access was provided.

4 Application requirements

Following part describes the basic functionality required to be developed during alpha stage. For simplicity, there are just 3 roles, buyer (visitor), seller and administrator. Buyer does not have to register an account to see active advertisements, so it's not required to be logged in to use website as visitor.

Table 7 shows list of requirements for each type of user and is used later-on for manual testing of ready product.

Table 7. List of required functionality to be developed.

Role	Functionality	Comment
Seller	Sign-up	In order to store advertisements, application needs to know, to whom they belong. Sign-up process is easy and requires just username, email, and a password.
	Login	Registered user should be able to login into system with his username and password.
	Create advertisement ¹	This is the most important and complex process in the whole application. It is divided into multiple steps to simplify and streamline the process.
	View list of his advertisements	Seller must be able to see all his active advertisements.
	Update details of his advertisements	Seller must be able to update details of his advertisements.

¹ Due to complexity of process, full overview is provided in sub-chapter "Advertisement creation process".

Role	Functionality	Comment
	Delete advertisement	Seller must be able to remove advertisements.
Buyer	View list of all active advertisements	Buyer should be able to see all advertisements.
	Visit unique advertisement to check its details	Buyer should be able to visit specific advertisement to see full vehicle details.
	Search for vehicles ¹	Search should be possible by generic vehicle characteristics – manufacturer, body type, fuel type, gearbox type, drive type, years of manufacture, engine power, mileage, price. It should be also possible to select only vehicles with valid technical inspection.
	Search for vehicles based on assigned badges ²	It should be possible to receive advertisements sorted by amount of issued badges to make search easier.
Administrator ³	View all advertisements by any user	Should have access to any advertisement.
	Edit any advertisement	It should be possible to impose corrective measures.
	Delete any advertisement	It should be possible to impose blocking measures.
	Create announcement	Announcements are visible in “Announcement” tab of website, used mainly for news and updates.
	Update announcement	It should be possible to make a correction.

¹ There is no pre-loaded information in search form, but rather search form is generated on the fly based on current advertisement list. For example, if only BMW and Toyota cars are saved in database, then only these selections are available in the form, or if weakest car engine power is 55 kW (*Kilowatt*) and toughest is 200 kW, there will be no options to select cars under 55 kW and over 200. This would help client to choose from what’s really available throughout the portal.

² Description of badging system of VSP is available in sub-chapter “Assigning badges to vehicles”.

³ In VSP, administrator can edit only user-entered data. Data coming from AVP is not available for changing. More details and reasoning are available in sub-chapter “Limitations of administrator role”.

4.1.1 Advertisement creation process

Registered user visits VSP and enters vehicle numberplate. Then, request is sent to AVP. Once reply with vehicle details is received, short summary of vehicle is displayed, and user is asked if correct vehicle details were loaded. Next step is to enter basic advertisement details, such as vehicle's odometer reading, price, contacts of owner and short description. Next and last step is to upload photo(s). Afterwards advertisement is saved and becomes available for visitors of website.

Saving an advertisement in backend is itself a complex process. Before saving, database is required to do pre-checks, for example it should check for availability of provided model and manufacturer data. It should be stressed out that VSP database by design should not initially have pre-loaded information on all manufacturers and models, and it is meant to be populated once an advertisement is created, based on data from AVP. If unknown model or manufacturer is included in incoming request, respective tables will need to be populated with new data.

The whole process can be graphically displayed using flowchart as shown on Figure 8.

Oval represents start or end point. Arrows represent relationship between entities, parallelogram represents input or output. Process is marked as a rectangle, while diamond shape is used to indicate decision [30].

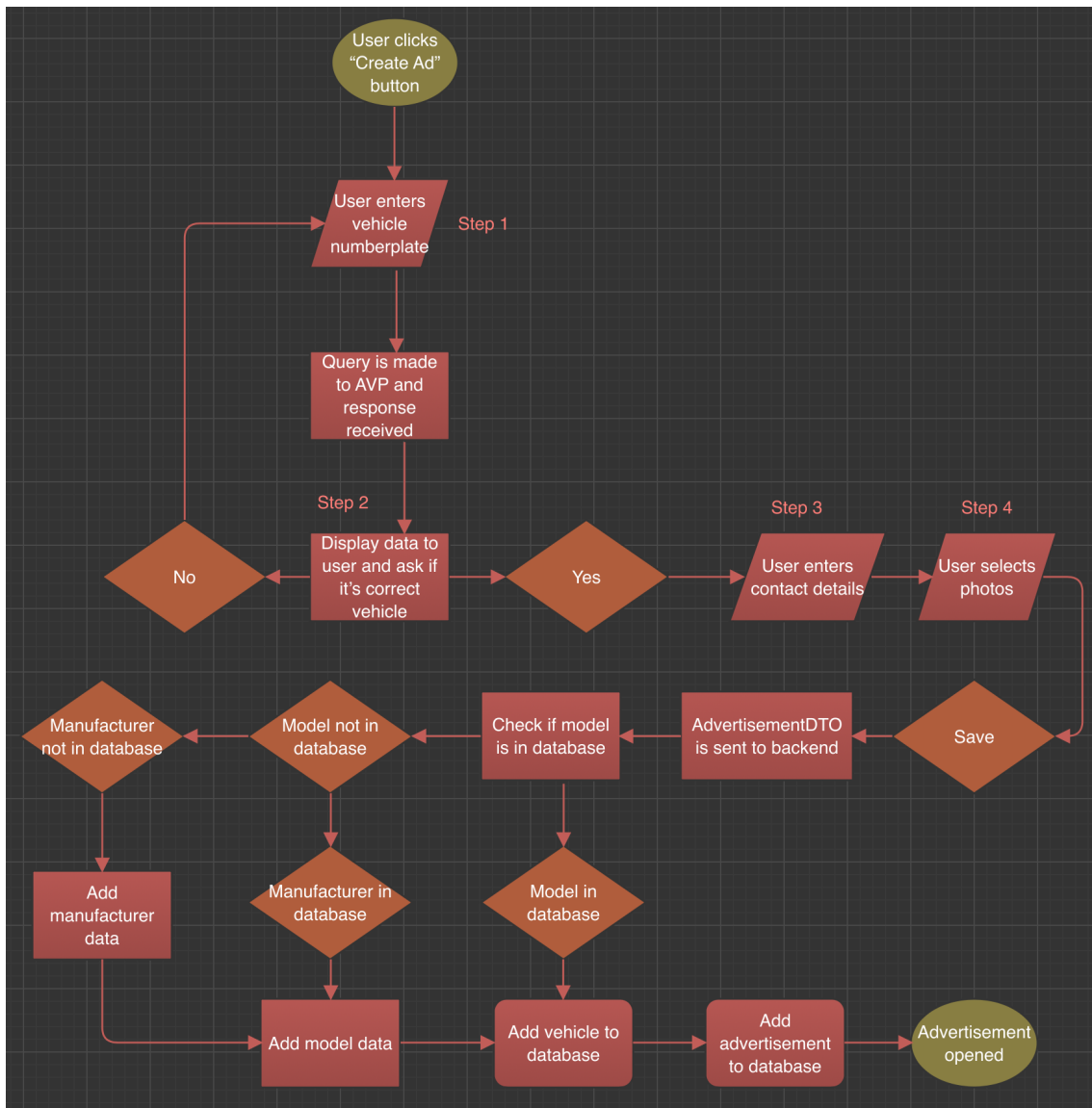


Figure 8. Flowchart of advertisement creation process.

4.1.2 Limitations of administrator role






Due to the nature of the application and its initial idea of receiving data from AVP, administrator has no means of updating list of manufacturers, models, vehicles, technical inspections, actions, restrictions. This data is originating from a trusted government source and is not meant to be changed or manipulated in any way.

Badges (this functionality is described in next chapter) and their links to vehicles are also not to be changed because they are assigned automatically by backend controller at the time of advertisement creation.

4.1.3 Assigning badges to vehicles

In order to help final users find best vehicle suitable for their needs, it is required to introduce automatic system of vehicle badging. Once information is fetched from AVP and entered by user it should be automatically analyzed and some of badges from Table 8 may be assigned.

Table 8. Description of badges that could be assigned to vehicle.

Shortcut	Full name	Description	Icon
LD	Long Description	Vehicle has description of over 200 symbols	
LTI	Long Technical Inspection	Vehicle has over 6 months until next technical inspection	
NR	No Restrictions	Vehicle has no restrictions (Not arrested by law enforcement and not used as loan deposit)	
LO	Long Ownership	Vehicle did not switch owners for over 3 years	
ATIS	All Technical Inspections Successful	Vehicle has passed all technical inspections successfully and never failed	

5 Development of application and features

In this chapter author of thesis describes all steps that were taken to develop the application. The codebase that was written as a result turned out to be relatively large, and below parts do not include full details of implementation, but rather short excerpts to display general concepts.

Moreover, some implementation details cannot be shared due to nature of agreement between the author of the thesis and Estonian Transport Administration.

5.1 Design of database

Despite using code-first approach for this project, which does not require real physical database structure to be present before writing application [31] it was decided to design database using web tool Lucidchart. It is a paid cloud-based service, which is considered one of best tools to create ER (*Entity Relationship*) diagrams [32]. It offers free package with limit of 3 created documents [33].

The database structure for this project is quite simple. It includes just 13¹ interconnected tables, as shown on Figure 9 below.

¹ Table “Announcement” is not included on ER diagram as it’s not related to anything and does not play significant role in structure.

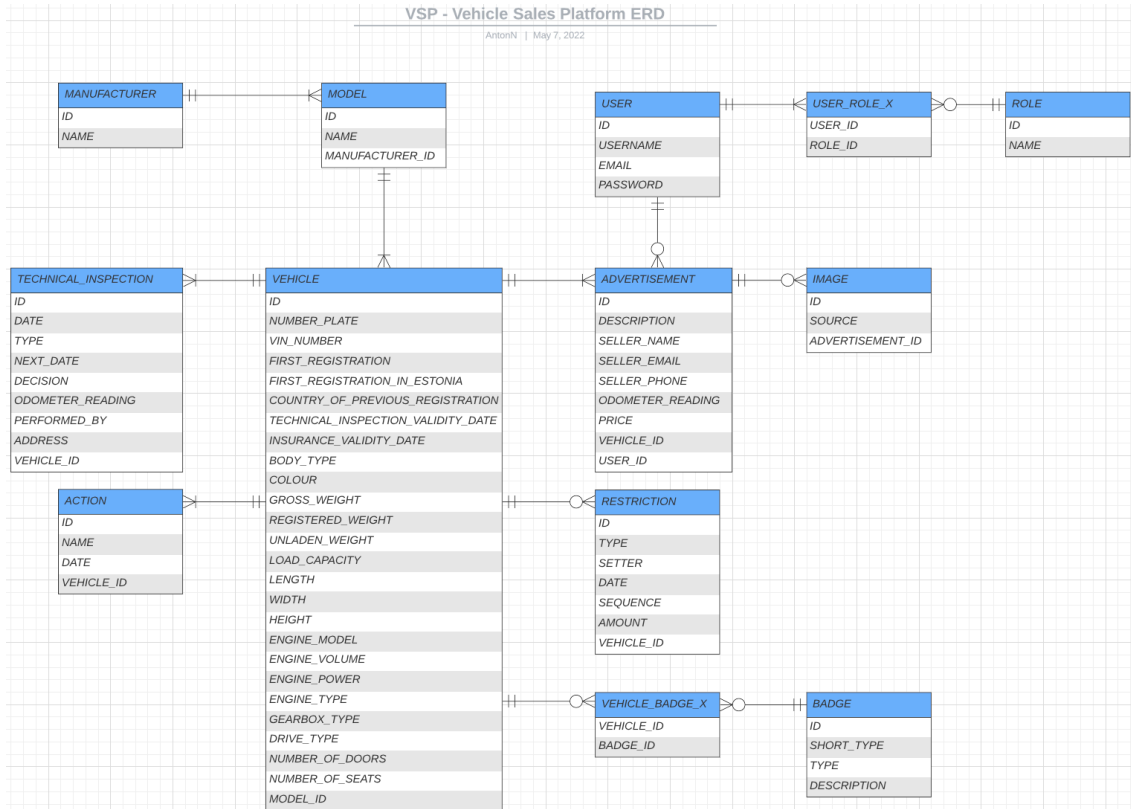


Figure 9. Entity Relationship Diagram for VSP database.

Database tables are described in more detail in Table 9.

Table 9. List of database tables with descriptions and relationship details.

Table name	Purpose	Relationship details
MANUFACTURER	Contains manufacturer data	One-To-Many relationship to Model – Manufacturer can have many models.
MODEL	Contains model data	Many-To-One relationship to Manufacturer – Model always belongs to specific manufacturer. One-To-Many relationship to Vehicle – Model can have many vehicles.
VEHICLE	Contains detailed vehicle data	Many-To-One relationship to Model – Vehicle always belongs to specific model. One-To-Many relationship to Technical Inspections, Actions, Restrictions. Vehicle can have many

Table name	Purpose	Relationship details
		<p>technical inspections, actions and restrictions.</p> <p>Many-To-Many relationship to Badges through connecting table. Vehicle can have many badges.</p> <p>One-To-Many relationship to Advertisement – Vehicle can have many advertisements.</p>
TECHNICAL_INSPECTION	Contains data about technical inspections	Many-To-One relationship to Vehicle. Technical inspection belongs to specific Vehicle.
ACTION	Contains data about vehicle-related registry actions	Many-To-One relationship to Vehicle. Action belongs to specific Vehicle.
RESTRICTION	Contains data about restrictions	Many-To-One relationship to Vehicle. Restriction belongs to specific Vehicle.
BADGE	Contains data about generic available badges	Many-To-Many relationship to Vehicle. Badge can belong to many vehicles.
VEHICLE_BADGE_X	Connecting table between Vehicle and Badge entities	
ADVERTISEMENT	Contains ad data	Many-To-One relationship to Vehicle – Advertisement always belongs to specific vehicle.
IMAGE	Contains locations of images	Many-To-One relationship to Advertisement – Image always belongs to specific Advertisement.
USER	Contains registered user data	<p>One-To-Many relationship to Advertisement – User can have many advertisements.</p> <p>Many-To-Many relationship to Role through connecting table USER_ROLE_X. User can have many roles.</p>
ROLE	Contains possible roles	Many-To-Many relationship to User through connecting table USER_ROLE_X. Role can belong to many users.

Table name	Purpose	Relationship details
USER_ROLE_X	Connecting table between User and Role entities	
ANNOUNCEMENT	Contains data about announcements	Has no relationship and does not play significant role in application inner structure.

5.2 Implementation of application domain

As next step, it was required to set up models in Java application. The number of models is almost¹ same as number of tables in above chapter. All of models extend Base class, which automatically provides time of entity creation and time of entity update.

In order to allow rapid development of application, author of thesis heavily uses annotations, especially Lombok annotation processor. Lombok is a special Java library, which allows to minimize amount of boilerplate and repetitive code by annotating classes or variables [34].

All domain classes are annotated with `@Data` annotation. This annotation automatically generates all getters and setters for non-final variables, as well as readable toString method [35] [36].

In addition to that, models are also annotated by `@Entity` and `@Table`, which are annotations provided by JPA (*Jakarta Persistence API*), previously known as Java Persistence API. As website will save user data as well as vehicle data, JPA implementation will provide ORM (*Object Relational Mapping*) layer to application, which will manage conversion of Java objects with tables and columns in relational database [37].

¹ With exception of connecting tables because they are generated automatically and setup of separate entity is not required.

`@Entity` annotation instructs that given POJO (*Plain Old Java Object*) needs to be persisted in database, which means that a table for it must be created. `@Table` annotation together with required name parameter instructs what should be the name of table [38].

All of models have unique identifier, which is saved in field “id”, with annotations `@Id` and `@GeneratedValue` (`strategy = GenerationType.IDENTITY`). These annotations advise JPA to use given field as unique identifier, and what should be its generation strategy. `IDENTITY` is one of easiest options which will make column use auto-incremented values [39].

The rest of primitive one-value fields are annotated with `@Column` – this annotation is used to map POJO field to database column. It can be created with parameters, such as name, nullability, maximum length [40].

Most importantly, it is required to map relationships between entities. Application mostly uses One-To-Many relationships, and also two Many-To-Many relationships. An example of each relationship will be provided below, other ones are omitted for brevity.

5.2.1 One-To-Many relationship set-up

As an example, below is a relationship between Vehicle and Technical Inspection. Each vehicle can have data regarding multiple Technical Inspections throughout years. Each separate technical inspection entity always belongs to some specific vehicle. Because of that, in Vehicle model relationship is set up as shown on Figure 10.

```
@OneToMany(mappedBy = "vehicle", cascade = CascadeType.ALL)
@JsonManagedReference
private List<TechnicalInspection> technicalInspections = new
ArrayList<>();
```

Figure 10. Example of OneToMany mapping in Spring Boot.

`@OneToMany` annotation explains cardinality of relationship. Parameter `mappedBy` is used to show which variable is used to represent parent class in child class, as mapping in Technical Inspection class is also created to make entity relationship bidirectional. This means that information about connected entities is available from both sides of relationship (Technical Inspection data available from Vehicle side, and Vehicle data is

available from Technical inspection side) [41]. Parameter cascade makes sure that all operations on parent entity Vehicle are propagated on child entity [42].

`@JsonManagedReference` is an annotation provided by Jackson library, which is used to serialize and deserialize POJOs. Such annotation is placed on parent, while `@JsonBackReference` is placed on child entity in order for Jackson to be able to understand relationship and not allow circular referencing and endless loop as a result during serialization process [43].

The other side of relationship, Technical Inspection entity has linkage set up as shown below on Figure 11.

```
@ManyToOne
@JoinColumn(name = "vehicle_id")
@JsonBackReference
private Vehicle vehicle;
```

Figure 11. Example of ManyToOne mapping in Spring Boot.

Here `JoinColumn` is set up in order to specify which column contains reference (foreign key) to parent object [44].

5.2.2 Many-To-Many relationship set-up

Example Many-To-Many reference is between vehicle and badges.

Each vehicle could have many badges, and badges may belong to many vehicles, as they do not hold any unique values, and are used rather to show similarities between vehicles.

From Vehicle side set up is made as shown on Figure 12.

```
@ManyToMany
@JoinTable(
    name = "vehicle_badge_x",
    joinColumns = @JoinColumn(name = "vehicle_id"),
    inverseJoinColumns = @JoinColumn(name = "badge_id"))
private List<Badge> badges;
```

Figure 12. Example of ManyToMany mapping in Spring Boot.

`@JoinTable` provides information how connecting table should be named, what is name of column that links to Vehicle entity and what is name of column that links to Badge entity [45]. The result of such setup with example data looks on database level as shown on Figure 13 below.

```
SELECT * FROM VEHICLE_BADGE_X;
```

VEHICLE_ID	BADGE_ID
1	5
1	4

(2 rows, 50 ms)

Figure 13. Example of connecting table between Vehicle and Badge entities.

5.3 Implementation of database connectivity

Following chapter describes how database connectivity is implemented and what techniques were used.

5.3.1 Repository pattern

Most of models require to be saved in database, and process of synchronization needs to be simple and straightforward. One of the options to make it possible is to implement repository pattern. This design pattern is useful where there are high number of domain classes and a lot of querying is utilized. Thus, repository layer was added between the domain entities and data mapping layers to isolate domain objects from details of the database implementation and to minimize duplication of query code [46].

Spring Boot and JPA provide a convenient way to do this out of the box – even an empty interface that extends `JpaRepository` can be created and basic CRUD (*Create, Read, Update, Delete*) functionality will become available [47]. Example of this functionality is provided on Figure 14.

```
@Repository
public interface BadgeRepository extends JpaRepository<Badge,
Long> {}
```

Figure 14. Example of Repository implementation in SpringBoot.

As interfaces cannot implement real logic, application should access data through several services, which are created for all repositories.

5.3.2 Implementation of service

The service provides a connectivity bridge to database through repository to be utilized by controller. Repository is injected into service upon loading automatically by Spring Boot and implements methods that our controller uses. For Manufacturer, there are 3 methods in use - save(), findByName() and findAll(), as shown on Figure 15.

```
@Service
public class ManufacturerService {

    private ManufacturerRepository manufacturerRepository;

    public ManufacturerService(ManufacturerRepository
manufacturerRepository) {
        this.manufacturerRepository = manufacturerRepository;
    }

    public Manufacturer createManufacturer(Manufacturer
manufacturer) {
        return manufacturerRepository.save(manufacturer);
    }

    public Manufacturer findByName(String name) {
        return manufacturerRepository.findByName(name);
    }

    public List<Manufacturer> findAll() {
        return manufacturerRepository.findAll();
    }
}
```

Figure 15. Example of service implementation in Spring Boot.

Service is directly called in controllers when communication with database is required.

5.4 Implementation of API controllers

As database connectivity was set up with the use of repositories and services, next step was to create controllers that would process client requests and utilize services. This chapter provides a list of endpoints that backend provides and explains how Data Transfer Objects are used to reduce data complexity and data quantity between client and REST service. Due to the nature of application, some routes should be protected from unauthorized requests and security topic is also covered in last sub-chapter.

5.4.1 List of endpoints

Application supports requests to several API endpoints. List of endpoints is shown in Table 10.

Table 10. List of supported API endpoints.

HTTP Method	API endpoint	Secured	Usage
POST	/api/auth/signin	No	Authorization in system
POST	/api/auth/signup	No	Registration in system
GET	/api/advertisements	No	Viewing list of all advertisements
GET	/api/advertisements/:id	No	Viewing one specific advertisement
GET	/api/advertisements/top	No	Viewing best offers (sorted by number of assigned badges)
GET	/api/advertisements/search	No	Viewing offers by specified GET parameters
GET	/api/advertisements/search-fields	No	Used internally to calculate possible selections on 'Search' page
POST	/api/advertisements	Yes	Creation of advertisement
POST	/api/advertisements/:id/images	Yes	Addition of images to advertisement
GET	/api/user-advertisements/	Yes	Viewing of advertisements created by current user
GET	/api/user-advertisements/:id	Yes	Viewing of single advertisement created by current user (for editing)
PATCH	/api/user-advertisements/:id	Yes	Allows partial update of advertisement
DELETE	/api/user-advertisements/:id	Yes	Allows deletion

5.4.2 Data Transfer Objects

Because application contains a large amount of logic and often exchanges information with frontend, it is relying heavily on usage of DTO (*Data Transfer Object*) pattern. DTO is an object that carries data between processes to reduce the number of method calls [48]. The idea behind Data Transfer Object was first introduced by Martin Fowler, who explained that the pattern's main purpose is to reduce roundtrips to the server by batching up multiple parameters in a single call [49]. This reduces the network overhead in such remote operations, moreover it can be used to hide domain implementation details that are not required in any way by sending or receiving party.

One of the largest DTOs used by VSP is AdvertisementGetDTO, which is returned by endpoint `/advertisements/:id` and consumed by React frontend to display full details about single advertisement. The schema of this DTO is visualized on Figure 16.

AdvertisementGetDTO		
	Type	Name
Ad	String	description seller_name seller_phone seller_email
Ad	Long	odo_reading price
Vehicle	VehicleGetDTO	vehicle_data
Image	List<Image GetDTO>	images

Figure 16. Visual example of data transfer object composition.

As DTOs are flat data structures that contain no business logic, they are constructed using a specially written mapper that assigns all required values. Because inside of AdvertisementGetDTO are more DTOs – VehicleGetDTO which sums up all important vehicle data and ImageGetDTO which contains URLs (*Uniform Resource Locator*) of vehicle uploaded images, the data is being processed via other mappers before assignment. The whole procedure schematically is represented on Figure 17.

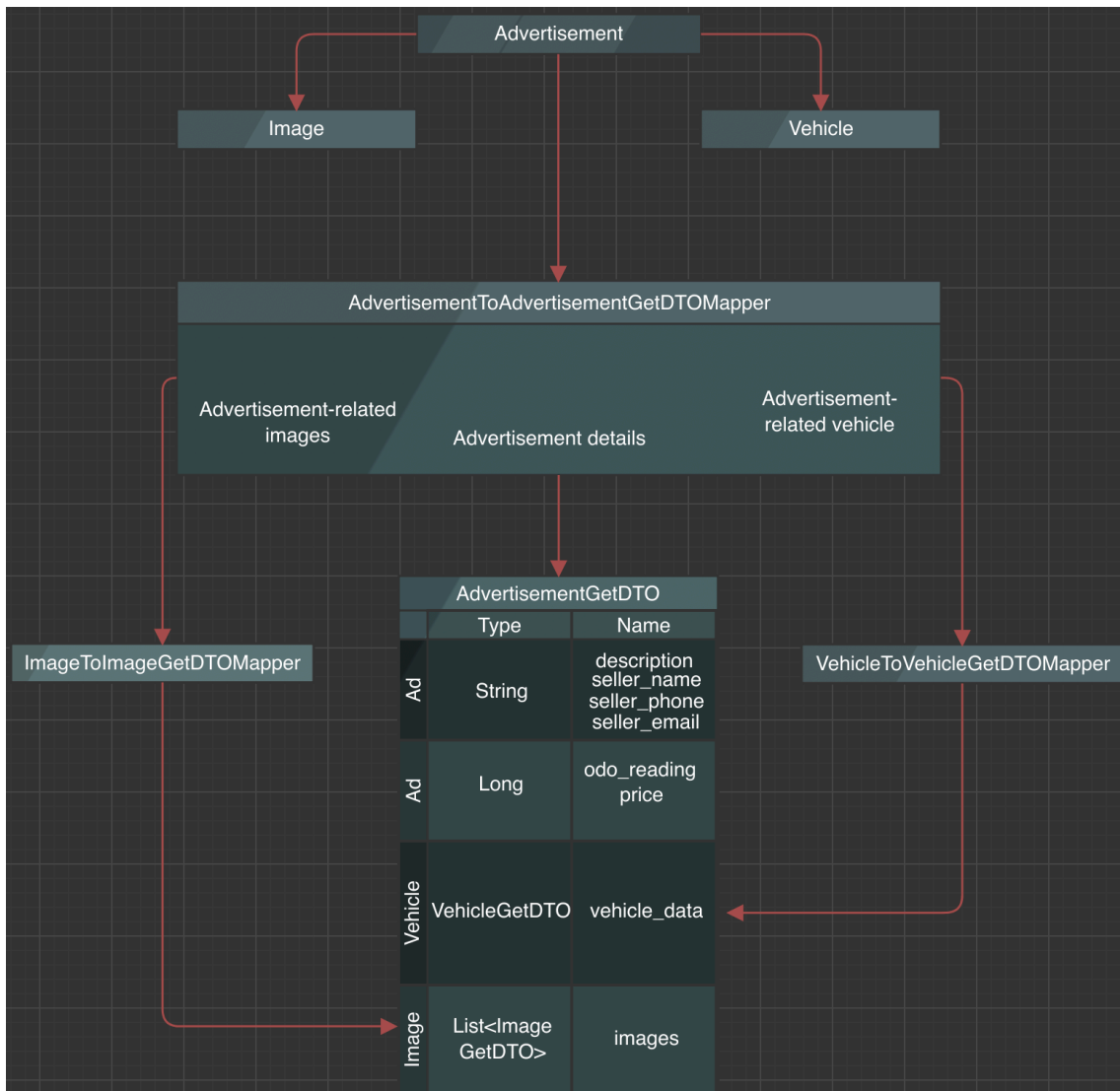


Figure 17. Visual representation of data transfer object creation.

5.4.3 Security

As described in endpoint table, access to most of API endpoints is permitted for all users, which makes it possible for anyone without account to view advertisements, respective vehicles, and all related data.

However, for those endpoints that deal with advertisement saving, it's crucial to understand who is creating the ad and who is going to have control over it in future.

To allow this functionality, app utilizes JWT-based (*JSON Web Token*) authorization. JWT is currently an industry standard method of representing claims security between two parties, in our case client and server [50].

Once user successfully authenticates in website, he receives JWT authorization token. This token can be used to further access resources he has rights to if authorization token is sent along in request header. In short, the schema of JWT communication looks as displayed on Figure 18 below.

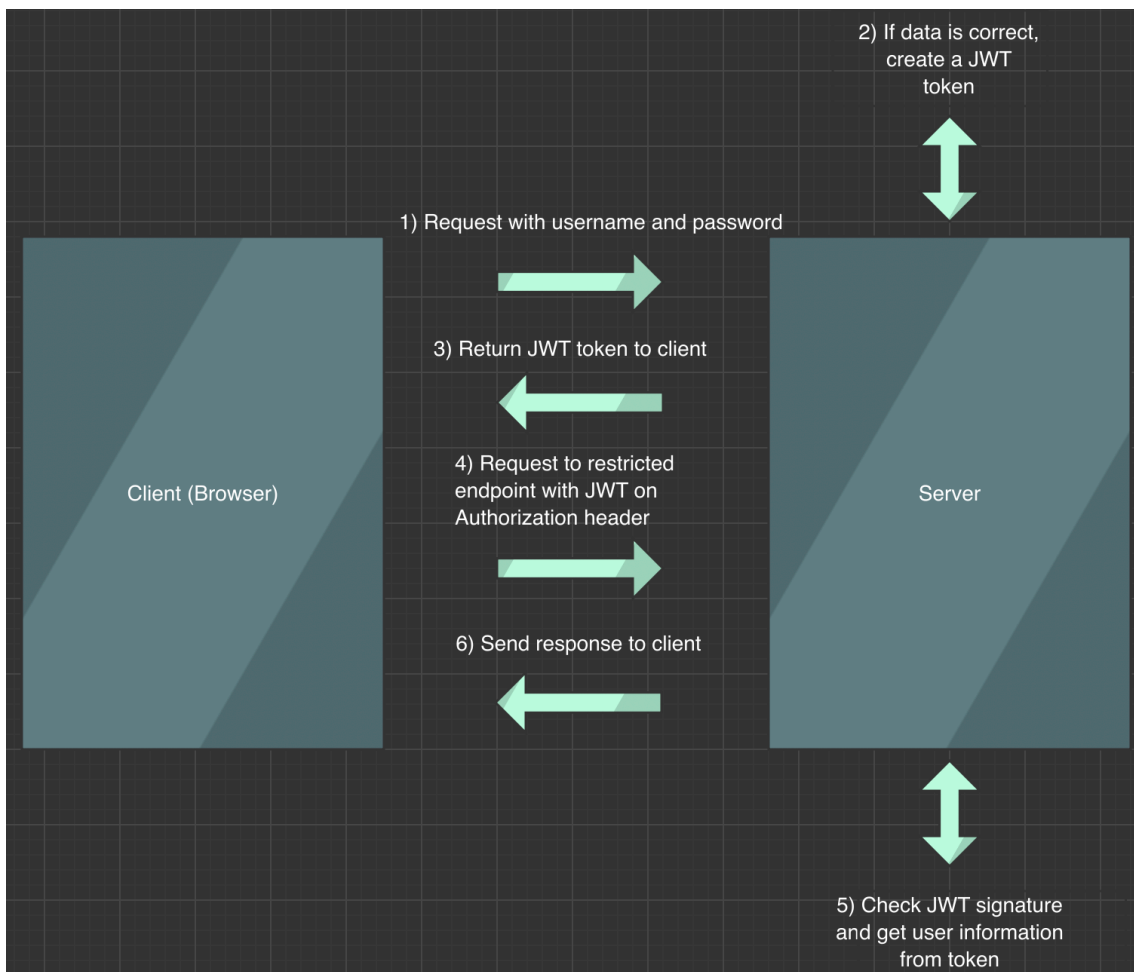


Figure 18. Example of JWT communication between client and server.

5.5 Implementation of Frontend using Bootstrap and ReactJS

Following chapter briefly describes how frontend application was constructed using Bootstrap and ReactJS and showcases structure of application and used techniques.

5.5.1 Bootstrap design

Bootstrap is world's most popular front-end open-source toolkit, which allows to develop fast and responsive web sites [51]. It provides pre-made styles for all frequently used components that are used in the web, such as form controls, buttons, navigation bars. Grid system allows to easily arrange elements, and pre-made classes allow to set up paddings and indents [52].

Using Bootstrap basic graphical user interface was assembled. Screenshots showcasing different pages are available in Appendix 1.

5.5.2 Implementation of React functional components

React provides two options to develop application – one is to use class components and another one to use functional components. Upon introduction of React Hooks in React 16.8, it was no longer required to write class components to use state [53]. Moreover, functional components are shorter and simpler to write, making code more readable [54]. As a result, functional components are more modern and easier approach. In VSP, most of self-written components are written as functions.

5.5.3 Usage of ReactJS hooks

Application has 35 components in total. To allow data flow between them, React hooks are heavily used to pass data around. One of utilized hooks is “useState”, which allows to assign state to function component to be preserved between re-renders. Another hook used is “useEffect”, which is usually used for actions such as data fetching upon page load [55]. Examples of hook usage are provided on Figure 19 and Figure 20 below.

```
const [advertisements, setAdvertisements] = useState([]);
```

Figure 19. Example of “useState” hook usage. Variable is created, setter method attached, initial value (empty array) is set.

```

useEffect(() => {
  UserService.getUserAdvertisements().then(
    response => {
      setAdvertisements(response.data.advertisements)
    },
    error => {
      this.setState({
        content:
          (error.response && error.response.data) ||
          error.message ||
          error.toString()
      });
    }
  );
}, []);

```

Figure 20. Example of “useEffect” hook usage. Service to get user advertisement is called only once when page is loaded, response is being set to state variable “advertisements”. In case of possible error, error details are also saved to variable “content”.

5.5.4 Usage of React Router for URL mapping

React Router is a very powerful tool that is used to match specific URL to function component. Using it makes possible to build full user’s interface [56]. Figure 21 demonstrates setup used in VSP.

```

<Routes>
  <Route exact path="/" element={<Home />} />
  <Route exact path="/login" element={<Login />} />
  <Route exact path="/register" element={<Register />} />
  <Route exact path="/profile" element={<Profile />} />
  <Route path="/about" element={<About />} />
  <Route path="/top" element={<AdTop />} />
  <Route path="/search" element={<AdSearch />} />
  <Route path="/announcements" element={<Announcements />} />
  <Route path="/advertisements" element={<Advertisements />} />
  <Route path="/advertisements/:id" element={<AdFull />}/>
  <Route path="/create" element={<CreateAd />}/>
  <Route path="/details/:id" element={<ChangeExistingDetails />}/>
</Routes>

```

Figure 21. Example of React route setup. Router maps URL to specific React component.

5.5.5 Usage of Axios to perform calls against API

As a modern distributed application, VSP is constantly sending requests to backend and receiving response to display. To handle vast number of requests, a client library is needed.

Axios is a promise-based HTTP (*Hypertext Transfer Protocol*) client which allows to make requests from browser and receive response, which is easy to set up and start to use [57]. Axios supports all possible types of requests and most importantly allows sending of headers, which is crucial as application needs to send JSON (*JavaScript Object Notation*). Backend, to understand that request is sent in JSON, needs to receive correct header. As parts of application require authorization, it's also very important that Axios supports sending authorization tokens. Figure 22 shows method that uses Axios functionality to perform a network request.

```

getAdvertisements() {
  return axios.get(API_URL + 'advertisements');
}

```

Figure 22. Example of method that utilizes Axios HTTP client to return data from API endpoint.

5.5.6 Application structure

Application structure of ReactJS frontend application is quite straightforward. Structure of main folder consist of 3 directories and is reflected in Table 11.

Table 11. ReactJS application directory list.

Directory name	Purpose
node_modules	Contains all required libraries to run application
public	Contains entry point of application (index.html)
src	Contains implementation data

Folder “src” has more important directories inside which mostly contain implementation details written in React and design implementation with Bootstrap. The contents of mentioned folder are reflected in Table 12 below.

Table 12. Contents of “src” directory in ReactJS application.

Directory name	Purpose
services	Contains generic service implementations
components	Contains functional components
css	Contains Cascading Style Sheet styles
img	Contains images used throughout the app

6 Infrastructure and deployment of application

This chapter provides a short overview of application infrastructure. Backend of VSP uses several services provided by Google Cloud, set up and shared under single project – Google Cloud Run for hosting Java API, Google Cloud SQL for hosting persisted PostgreSQL database and Google Cloud Storage for hosting uploaded images. All services are hosted in North Europe (Finland) to decrease load times from target market (Estonia). Frontend is set up on shared web hosting based in Estonia.

6.1 Google Cloud SQL

Google Cloud SQL instance is running PostgreSQL 9.6 and was set up according to official codelab [58]. Inside instance, database “vsp-prod” was created and IAM (*Identity and Access Management*) service account assigned to have access to it.

Google Cloud SQL and Spring Boot application can be conveniently interconnected by using dependency “spring-cloud-gcp-starter-sql-postgresql”. Once this is added to Spring Boot application, database credentials can be updated by setting three more variables in “application.properties” file. Full list of required application variables is shown in Table 12 in sub-chapter 6.4.

6.2 Google Cloud Run

To minimize amount of required setup and omit step with configuring and writing Dockerfile¹, it was decided to use Google Container Tool Jib. Jib allows to build optimized Docker image for Java applications without deep setup and is available as Maven or Gradle plugin [59].

¹ Setup file required to create Docker container out of application. Docker allows to package software into standardized units for deployment - containers. Container hosts all code with all dependencies, all settings and system tools that are required to run the code - it ensures that application will run in the same way on any machine running Docker Engine. Docker containers are lightweight and use machine’s OS (*Operational System*) kernel and do not require separate operating system for each application, making it a very efficient solution and reducing costs spent on servers and hosting. Docker is known to have created an industry standard for containers [68].

Its setup is quick as only few things need to be added to “build.gradle” file:

id 'com.google.cloud.tools.jib' version '3.2.1' into list of plugins and jib overall setup as shown on Figure 23.

```
jib {  
  to {  
    image 'eu.gcr.io/vsp-cloud-db/vsp'  
    credHelper = 'gcr'  
  }  
}
```

Figure 23. Example of Jib setup in “build.gradle” file.

Setup points to location of Cloud Image Repository where image should be pushed to and credential helper to use¹. After these changes command “gradle jib” becomes available and it automatically creates image out of Java application and pushes it to repository. Then in Google Cloud Run it is possible to deploy application from image easily, as shown on Figure 24.

¹ Credential helper needs to be set up separately to link to actual “credentials.json“ file and this setup is omitted for brevity.

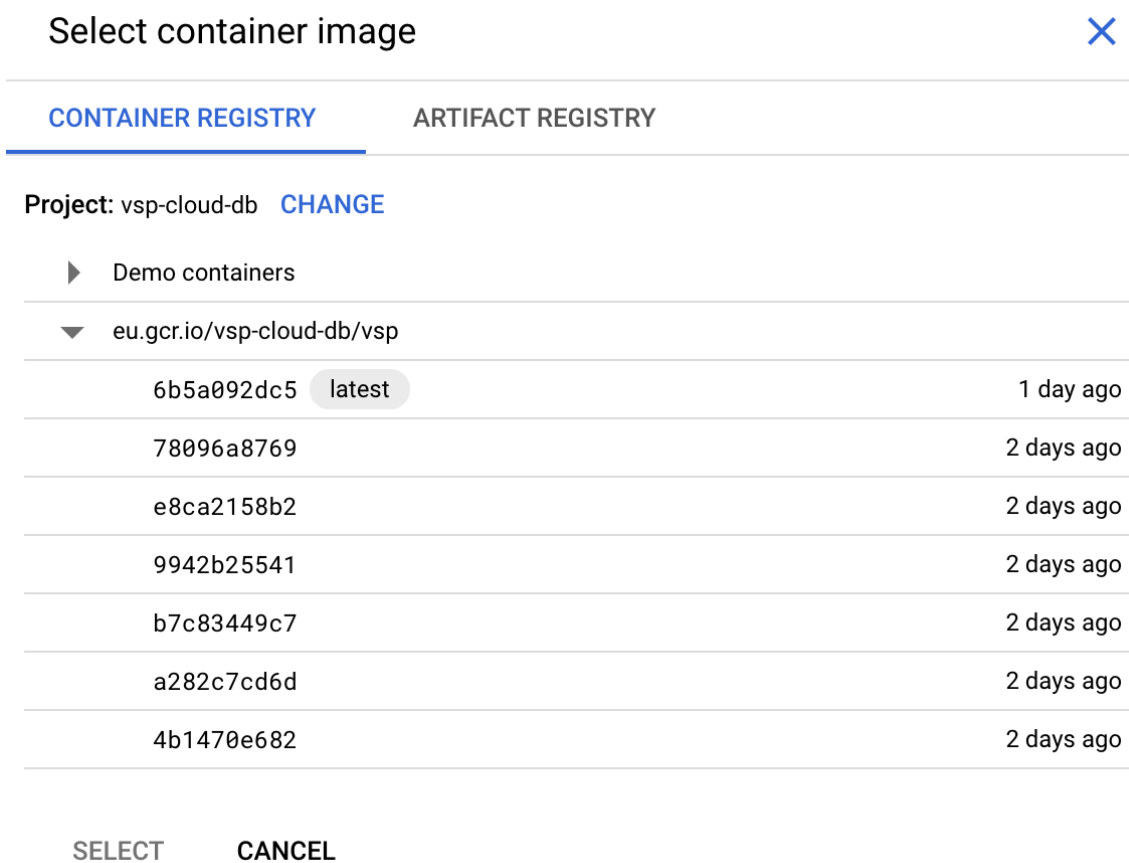


Figure 24. Selection of container image to be deployed in Google Cloud Run.

6.3 Google Cloud Storage

Google Cloud Run is a stateless container and cannot persist files that can be uploaded by user. Therefore, Google Cloud Storage is used for storing such data. A bucket¹ was created and then used in Java code to upload files to. In Spring Boot, setup requires just usage of “spring-cloud-gcp-storage” dependency, and after its addition, Cloud Storage functionality is ready to be used. Sample connection and image storing example is shown on Figure 25.

¹ Directory in context of Google Cloud Storage.

```

ClassPathResource resource = new
ClassPathResource("credentials.json");
InputStream inputStream = resource.getInputStream();

GoogleCredentials credentials =
GoogleCredentials.fromStream(inputStream);
Storage storage =
StorageOptions.newBuilder().setCredentials(credentials)
    .setProjectId("vsp-cloud-db").build().getService();

Bucket bucket = storage.get("vsp-images");
Timestamp timestamp = new Timestamp(System.currentTimeMillis());
fileName = timestamp + file.getOriginalFilename();

storage.create(
    BlobInfo.newBuilder(bucket, fileName).build(),
    file.getBytes());

```

Figure 25. Example of application image storing in Spring Boot using Google Cloud Storage.

After uploading, images can be seen in bucket as shown on Figure 26 and accessed via public internet¹.

Buckets > vsp-images

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter Filter objects and folders Show deleted data

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	
<input type="checkbox"/>	2022-05-02 19:40:25.847E391.jpeg	321.8 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 19:49:43.775B1.jpeg	2.5 MB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 19:49:44.242B2.jpeg	256.7 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 20:17:19.959E392.jpeg	129.9 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 20:17:20.307E391.jpeg	321.8 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 20:21:53.461P1.jpeg	178.9 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 20:21:53.461P2.jpeg	147.4 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL
<input type="checkbox"/>	2022-05-02 20:21:53.543P3.jpeg	169.6 KB	application/octet-stream	May 2, 20...	Standard	May 2, 202...	Public to internet	Copy URL

Figure 26. Images uploaded from VSP as seen in bucket (directory) hosted in Google Cloud Storage.

6.4 Cloud services-related setup in Spring Boot

“application.properties” file

Table 13 shows application variables that need to be set in Spring Boot to connect to all 3 used services in Google Cloud.

¹ Additional setup is required to make images from bucket visible publicly.

Table 13. Google Cloud-specific setup in Spring Boot application.properties file.

Variable	Value	Usage
Spring.cloud.gcp.project_id	Vsp-cloud-db	Name of the project in Google Cloud where all services are set up.
Spring.cloud.gcp.credentials.location	Classpath:credentials.json	File with credentials generated for IAM account [60].
Spring.cloud.gcp.sql.enableIamAuth	true	Enable connectivity to database via IAM service account.
Spring.cloud.gcp.sql.database-name	Vsp-prod	Name of Postgre database running in Cloud SQL instance.
Spring.cloud.gcp.sql.instance-connection-name	PROJECT_ID:LOCATION:INSTANCE_NAME ¹	Connection name consisting of project identifier, datacenter location and Cloud SQL instance name.

6.5 Frontend deployment

Frontend is hosted on Zone.ee hosting provider. After development with ReactJS is finished on local machine and changes need to be published, a command “npm run build” needs to be executed in application main directory to create build directory with production build of the application [61]. Contents of this directory need to be uploaded to hosting server and that step finishes deployment of frontend.

¹ Omitted here as security precaution as it exposes possibly sensitive data.

7 Testing of application

Testing of various parts of application was performed during development and product was also tested after deployment in live environment. This chapter provides more details about testing process and used tools.

7.1 API testing during development using Postman

Due to nature of VSP and distributed web applications in general, communication between client and server is happening via API calls. To test outcomes with different inputs, Postman tool was used.

Postman in general is an application used for API testing. It is an HTTP client which utilizes graphical user interface and allows to create collections and save groups of requests for future usage. Conveniently, it displays returned response code, type of response, returned content and time taken for request round-trip [62].

Figure 27 shows example output as seen in Postman.

```
Body Cookies (1) Headers (13) Test Results
Pretty Raw Preview Visualize JSON
width: 1902,
height: 1467,
engine_model: "B57D30A",
engine_power: 195,
engine_volume: 2993,
engine_type: "DIISEL",
gearbox_type: "AUTOMAAT",
drive_type: "AWD",
number_of_doors: 4,
number_of_seats: 5,
technical_inspections: [
  {
    date: "2020-02-11",
    type: "Korraline",
    decision: "Tehniliselt korras",
    next_date: "2022-03-31",
    odometer_reading: 64335,
    performed_by: "Center Services OÜ",
    address: "Pärnu mnt 552, 10916 Tallinn"
  },
  {
    date: "2017-10-03",
    type: "Korraline",
    decision: "Tehniliselt korras",
    next_date: "2020-03-31",
    odometer_reading: null,
    performed_by: "Maanteeamet",
    address: "-"
  }
]
```

Figure 27. Postman response received from /api/advertisement/:id endpoint with detailed vehicle information.

During testing 9 collections were created to verify processes such as authorization, advertisement creation and advertisement retrieval, some of them are demonstrated on Figure 28.

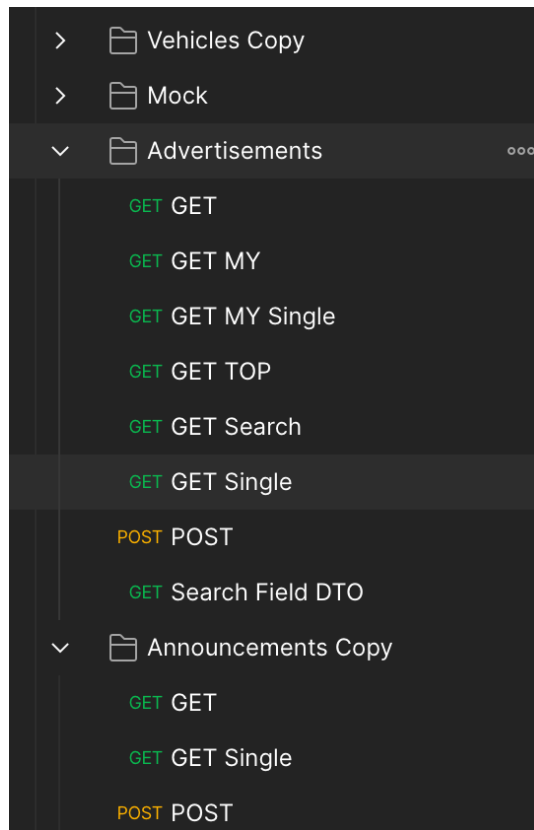


Figure 28. Example of Postman collections with sample saved requests to various endpoints.

7.2 System testing of ready product

Ready application was widely tested with manual testing techniques. Manual testing refers to testing process in which application is manually tested according to written test plan to identify bugs [63]. Due to an early development stage of application, a number of scenarios tested is limited and strongly tied to requirements described in “Application requirements” chapter.

Apart from that, it was required to analyze performance and usability of application from perspective of end user.

According to results of testing, there were no bugs found. Small issues that were found during system and UI testing cannot be considered as bugs because there was no concrete specification written how certain components should ideally behave or look like.

To optimize application for Production usage and allow further in-depth testing, application requirements will be refactored to be more detailed. Once this is done, more

detailed test plan will be written to cover all requirements. Then whole application will be tested again to detect any bugs.

As application is planned to expand and grow functionality, it is planned to introduce automatic tests as with every new feature it will be more difficult to test application manually¹.

¹ More details about automatic testing are available in next chapter, “Results and possible future improvements”.

8 Results and possible future improvements

During the initial analysis of dataset, it was found that there are over 26% of vehicles that cannot take part in traffic due to either stopped or suspended registration, or because they were expropriated but never properly registered again. The data analyzed was only for passenger vehicles, but it can be said that similar situation is present for two-wheel means of transport and for commercial transport. These facts altogether confirm the need for a solution. Proposed solution was a website, where it would be easy to put vehicle on sale by just supplying numberplate and photos. All further details are meant to be loaded directly from Transport Administration API, allowing to create a transparent and honest advertisement.

In second part of theoretical analysis, author described the technology used to implement such website. Short overview of technology stack was provided together with reasons why these choices were made.

In the practical part, author described most important points how the application was developed and why certain design decisions were made. All the initially proposed features were implemented and tested. The application was successfully deployed to planned infrastructure and can be accessed from public internet.

As with everything in life, no software product can be ideal. Therefore, already during development author found room for future improvements.

8.1 Future improvements to be considered

Author of thesis plans to proceed working on website and move it into Production stage at some point in future.

However, to make website more convenient, secure, and future proof, it would be great to add following features.

8.1.1 Captcha

As agreement with Transport Administration has some limitations on number of queries that can be performed daily, it is very important to protect the system from excessive

querying. Captcha should be introduced for any numberplate pre-checks to have excessive or machine-generated requests blocked.

8.1.2 Protection of customer data

To save customer's data from malicious web spiders that would steal phone number and email, additional measures to hide this sensitive data on advertisement page should be introduced.

8.1.3 Multilingual support

Currently website has only one language – English. However, as data is queried against Estonian Transport Administration, a lot of data is coming in Estonian language, making interface bilingual. A mechanism to translate generic strings that are coming into the system and saved should be introduced to bring interface to consistent state. Also, application should be slightly modernized to allow translation into other languages.

8.1.4 Login via existing services

Currently seller can only create a new account and use it to store his advertisements. It would be convenient if login would be possible by using Google account.

8.1.5 Messaging between parties

Currently potential buyer can contact seller only via contacts that seller supplied during creation of advertisement. It would be convenient if internal messaging system could be developed, making it possible to chat within website.

8.1.6 Support for paid advertisements by 3rd parties

At moment, there are no possible ways to monetize the application. A mechanism for controlling 3rd party advertisements should be introduced so that places and periods when they show up could be controlled via Admin panel.

8.1.7 Mobile app for Android and iOS

It would be even more convenient and quick to add sale advertisement directly from mobile phone. The mobile app can be easily developed against the existing API using shared codebase with React Native.

8.1.8 More details for advertisement page

The data coming from Road Administration is having some limitations as well. For example, there is no indication whether insurance is valid, or whether the car is being used or was used for taxi services. This data can be received from other, 3rd party resources like LKF (Liikluskindlustusfond) and MTR (Majandustegevuse register). To allow this behavior, analysis should be performed whether these resources provide API endpoints for fetching such data.

8.1.9 Uptime monitoring

Currently there is no website availability monitoring set up. In case application goes down for some reason, it will not be anyhow discovered.

For Production usage it is a must to set up uptime monitoring. Uptime is critical crux of any web experience that can break reputation and drive potential customers away [64].

As website is intended for local usage in Estonia at this point, it should be constantly monitored from various Estonian locations for availability. Such monitoring could be set up through services such as Site24x7. It is a full-stack performance monitoring tool provided in a form of SaaS (*Software as a Service*). Requests may be setup to check availability of different endpoints, provide response times, responses themselves, all of which will help to react on faults early.

8.1.10 Automatic tests

As it was pointed out before, during manual testing it became evident that with every new feature it will be more and more complicated to test application's functionality. Even with short test plan, process of manual testing is extremely time-consuming and repetitive and requires physical time and effort.

At some point automatic tests should be introduced. Initial time investment into writing automatic tests will pay off as application grows and more time will become available that could be spent on developing new features and not testing old ones from scratch and fixing found bugs [65].

Currently automatic deployment for the app is set up but application is not tested before being deployed. In order to set up full CI/CD (*Continuous Integration / Continuous Delivery*) cycle, it is required to introduce testing step before deployment [66].

8.1.11 Code refactoring

Due to essence of alpha application with imposed time limitations, it is built quickly and more in style of bazaar than a cathedral [67]. Code needs to be refactored to remove possible duplications, structure should be optimized, overall coding style and practices should be made more consistent to make it easier to read.

9 Summary

To summarize the result, the initial goal of thesis is fulfilled. Author has developed a working web application that allows to put any vehicle that is registered in Estonian Transport Administration quickly for sale. The seller just needs to provide numberplate and photos, the rest of information is fetched from AVP. This makes selling an easy and headache-free process which does not take more than 1 minute. The portal is making buying process easy and convenient for interested parties also, as all possible information about vehicle is available in one place, which adds transparency to process and helps to mitigate unfair vehicle sale practices.

However, current version is not yet fully suitable for usage in Production environment. Big list of tasks lies in front of author of the thesis waiting to be done. Before handing application down to real users, overall usability should be improved. Introducing captcha, possibility to hide sensitive user data in advertisements for web crawlers, adding convenience features like multilingual support and messaging are an absolute must to increase user confidence and trust in the website. To assure maintainability of product in the long run, code needs to be refactored as well.

When it is up and running and available for many users, it's crucial to have full overview of situation. Even minor outages or performance issues might quickly drive users away to competition. To prevent bugs from reaching Production environment, automatic tests must be introduced to ensure quality of current build – author believes that it is always easier to double check instead of finding issues later and reverting everything back. To react quickly on possible issues, it's important to set up proper monitoring that would notify author if anything goes wrong, for example if main page becomes inaccessible or response times have lingered.

Taking everything written into account, it can be said that product is usable, but far from ideal. Author plans to proceed working on VSP in his free time, making small steps further towards going to Production. As service provided by ETA is not free and costs considerable amount of money per month while still imposing heavy usage restrictions,

everything must be precisely analyzed and planned to ensure smooth operation. Moreover, it would be more convenient to operate through a legal entity, thus a company should be set up before stepping up to a new level.

Described flaws were expected before development and writing of the thesis and it was known that it would not be possible to cover everything in scope of provided work. Initial goals were fulfilled, and author has gained incredible experience, not only in writing code but in communication as well as making contracts with government entities was not something he had previous practice with.

As author has high motivation to develop his programming skills and deep personal interest in topic, work on the project will be continued until it's in shape for Production usage.

10 References

- [1] "Deutsches Patent- und Markenamt (German Patent and Trademark Office)," 2 Jan 2017. [Online]. Available: <https://web.archive.org/web/20170102082130/https://www.dpma.de/service/klassifikationen/ipc/ipcprojekt/einekurzgeschichtedesautomobils/geburtstagdesautos/index.html>. [Accessed 22 March 2022].
- [2] R. Price, "Division of Labor, Assembly Line Thought - The Paradox of Democratic Capitalism," 29 January 2004. [Online]. Available: http://www.rationalrevolution.net/articles/division_of_labor.htm. [Accessed 22 March 2022].
- [3] "U.S. Department of Transportation - Federal Highway Administration - State motor vehicle registration by years 1900-1995," April 1997. [Online]. Available: <https://www.fhwa.dot.gov/ohim/summary95/mv200.pdf>. [Accessed 22 March 2022].
- [4] OICA - International Organization of Motor Vehicle Manufacturers, "Personal World Vehicles in Use," 2015. [Online]. Available: https://www.oica.net/wp-content/uploads/PC_Vehicles-in-use.pdf. [Accessed 22 March 2022].
- [5] OICA - International Organization of Motor Vehicle Manufacturers, "World Motor Vehicle Production by Country and Type," 2018. [Online]. Available: <https://www.oica.net/wp-content/uploads/By-country-2018.pdf>. [Accessed 22 March 2022].
- [6] "Eesti avaandmed - Sõidukite staatused Eestis," Transpordiamet, 1 March 2022. [Online]. Available: <https://avaandmed.eesti.ee/datasets/soidukite-staatused-eesis>. [Accessed 23 March 2022].
- [7] "Olulisemad välismaised kasutatud autode müügiportaalid, kust oma unistuste autot otsida," Geenius, 18 May 2018. [Online]. Available: <https://auto.geenius.ee/blogi/inbanki-blogi/olulisemad-valismaised-kasutatud-autode-muugiportaalid-kust-oma-unistuste-autot-otsida/>. [Accessed 23 March 2022].
- [8] "Auto24.ee Pricelist," 2022. [Online]. Available: <https://eng.auto24.ee/main/pricelist.php>. [Accessed 23 March 2022].
- [9] "Sõidukite statistika," Estonian Road Administration, 2022. [Online]. Available: <https://www.mnt.ee/et/ametist/statistika/soidukite-statistika>. [Accessed 23 March 2022].
- [10] "Auto24 Search," 25 March 2022. [Online]. Available: <https://eng.auto24.ee/kasutatud/nimekiri.php?bn=2&a=101102&aj=&ssid=51158028&g1=1&ae=8&af=50&by=2&otsi=search>. [Accessed 25 March 2022].
- [11] "JetBrains Dev EcoSystem Java," 2021. [Online]. Available: <https://www.jetbrains.com/lp/devecosystem-2021/java/>.
- [12] R. K. Soni and N. Soni, "Deploy a Spring Boot Application as a REST API in AWS," in *Spring Boot with React and AWS: Learn to Deploy a Full Stack Spring Boot React Application to AWS*, Apress, 2021.
- [13] L. Spilca, "Projects from the Spring ecosystem," in *Spring Start Here*, Manning Publications, 2021.
- [14] "Spring.IO official documentation," [Online]. Available: <https://spring.io/why-spring>. [Accessed 22 03 2022].

- [15] "JetBrains Dev Ecosystem JavaScript," 2021. [Online]. Available: <https://www.jetbrains.com/lp/devecosystem-2021/javascript/>.
- [16] "ReactJS Official Website," 2022. [Online]. Available: <https://reactjs.org>. [Accessed 22 March 2022].
- [17] B. Williamson, "Which JavaScript Frameworks Should I Learn," 15 June 2021. [Online]. Available: <https://flatironschool.com/blog/which-javascript-frameworks-should-i-learn/>. [Accessed 5 May 2022].
- [18] L. Ferrari and E. Pirozzi, "PostgreSQL at a glance," in *Learn PostgreSQL*, Packt Publishing, 2020.
- [19] L. Ferrari and E. Pirozzi, "A brief history of PostgreSQL," in *Learn PostgreSQL*, Packt Publishing, 2020.
- [20] "StackOverflow Professional Developer Survey 2021," 2021. [Online]. Available: <https://insights.stackoverflow.com/survey/2021/#most-popular-technologies-database-prof>. [Accessed 22 March 2022].
- [21] "StackOverflow Professional Developer Survey 2020," 2020. [Online]. Available: <https://insights.stackoverflow.com/survey/2020/#technology-databases-professional-developers4>. [Accessed 22 March 2022].
- [22] P. Jahoda, "Benchmark databases in Docker: MySQL, PostgreSQL, SQL Server," 7 January 2021. [Online]. Available: <https://itnext.io/benchmark-databases-in-docker-mysql-postgresql-sql-server-7b129368eed7>. [Accessed 5 May 2022].
- [23] "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems," DigitalOcean, 9 March 2022. [Online]. Available: <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>. [Accessed 5 May 2022].
- [24] "About," [Online]. Available: <https://www.zone.ee/en/about/>. [Accessed 22 April 2022].
- [25] "Web Hosting Prices," [Online]. Available: <https://www.zone.ee/en/web-hosting/prices/>. [Accessed 22 April 2022].
- [26] I. Saar, "KIIRUSTEST: milline veebimajutus on Eesti parim?," Wixter, 31 August 2019. [Online]. Available: <https://wixter.ee/parim-veebimajutus/>. [Accessed 4 May 2022].
- [27] "AWS vs Azure vs Google Cloud: Choosing the Right Cloud Platform," IntelliPaat, 15 December 2021. [Online]. Available: <https://intellipaat.com/blog/aws-vs-azure-vs-google-cloud/>. [Accessed 4 May 2022].
- [28] "Cloud Run," Google, [Online]. Available: <https://cloud.google.com/run/>. [Accessed 22 April 2022].
- [29] "Estonian Transport Administration Data Exchange Platform," 2022. [Online]. Available: <https://www.mnt.ee/et/soiduk/liiklusregistri-andmetele-juurdepaasu-andmise-protsessi-kirjeldus>. [Accessed 22 March 2022].
- [30] "Flowchart Symbols," SmartDraw, [Online]. Available: <https://www.smartdraw.com/flowchart/flowchart-symbols.htm>. [Accessed 23 March 2022].
- [31] "Entity Framework Tutorial - Code First Approach," [Online]. Available: <https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx>. [Accessed 23 April 2022].

- [32] S. Cooper, "7 Best ER Diagram Tools," 30 January 2021. [Online]. Available: <https://www.comparitech.com/net-admin/er-diagram-tools/>. [Accessed 25 March 2022].
- [33] "Lucidchart Pricing," [Online]. Available: <https://lucid.app/pricing/lucidchart#/pricing>. [Accessed 25 March 2022].
- [34] "Project Lombok official website," [Online]. Available: <https://projectlombok.org/>. [Accessed 21 April 2022].
- [35] "Lombok documentation," [Online]. Available: <https://projectlombok.org/api/lombok/Data.html>. [Accessed 21 April 2022].
- [36] "Lombok annotation 'Data'," [Online]. Available: <https://projectlombok.org/features/Data>. [Accessed 21 April 2022].
- [37] M. Tyson, "What is JPA? Introduction to the Java Persistence API," 2 April 2019. [Online]. Available: <https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>. [Accessed 21 April 2022].
- [38] "Accessing Data JPA," [Online]. Available: <https://spring.io/guides/gs/accessing-data-jpa/>. [Accessed 21 April 2022].
- [39] T. Janssen, "JPA Generate Primary Keys," [Online]. Available: <https://thorbenjanssen.com/jpa-generate-primary-keys/>. [Accessed 21 April 2022].
- [40] "JPA Annotation - Column," [Online]. Available: <https://www.objectdb.com/api/java/jpa/Column>. [Accessed 21 April 2022].
- [41] "Hibernate One to Many Annotation Tutorial," 15 April 2022. [Online]. Available: <https://www.baeldung.com/hibernate-one-to-many>. [Accessed 21 April 2022].
- [42] "Overview of JPA/Hibernate Cascade Types," 11 May 2021. [Online]. Available: <https://www.baeldung.com/jpa-cascade-types>. [Accessed 21 April 2022].
- [43] "Jackson JSON - Using @JsonManagedReference and @JsonBackReference for circular references," 11 August 2020. [Online]. Available: <https://www.logicbig.com/tutorials/misc/jackson/json-managed-reference.html>. [Accessed 21 April 2022].
- [44] "@JoinColumn Annotation Explained," 24 November 2021. [Online]. Available: <https://www.baeldung.com/jpa-join-column>. [Accessed 21 April 2022].
- [45] V. Mihalcea, "Best way to map the JPA and Hibernate ManyToMany relationship," 19 November 2020. [Online]. Available: <https://vladmihalcea.com/the-best-way-to-use-the-manytomany-annotation-with-jpa-and-hibernate/>. [Accessed 21 April 2022].
- [46] "Java Design Patterns - Repository," [Online]. Available: <https://java-design-patterns.com/patterns/repository/>. [Accessed 21 April 2022].
- [47] "Spring.io JPA documentation," [Online]. Available: <https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html>. [Accessed 23 April 2022].
- [48] "The DTO Pattern (Data Transfer Object)," 31 March 2022. [Online]. Available: <https://www.baeldung.com/java-dto-pattern>. [Accessed 21 April 2022].
- [49] M. Fowler, "Data Transfer Object," [Online]. Available: <https://martinfowler.com/eaaCatalog/dataTransferObject.html>. [Accessed 21 April 2022].
- [50] "JSON Web Tokens official website," [Online]. Available: <https://jwt.io/>. [Accessed 21 April 2022].

- [51] "Bootstrap Official Page," [Online]. Available: <https://getbootstrap.com/>. [Accessed 22 April 2022].
- [52] "Bootstrap Documents," [Online]. Available: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>. [Accessed 22 April 2022].
- [53] "Introducing Hooks," [Online]. Available: <https://reactjs.org/docs/hooks-intro.html>. [Accessed 22 April 2022].
- [54] S. Yamazaki, 18 August 2020. [Online]. Available: <https://www.twilio.com/blog/react-choose-functional-components>. [Accessed 22 April 2022].
- [55] "Hooks at a Glance," [Online]. Available: <https://reactjs.org/docs/hooks-overview.html>. [Accessed 22 April 2022].
- [56] "React Router Main Concepts," [Online]. Available: <https://reactrouter.com/docs/en/v6/getting-started/concepts>. [Accessed 22 April 2022].
- [57] "What is Axios?," [Online]. Available: <https://axios-http.com/docs/intro>. [Accessed 22 April 2022].
- [58] "Connect a Spring Boot app to Cloud SQL," Google, 19 July 2021. [Online]. Available: <https://codelabs.developers.google.com/codelabs/cloud-spring-petclinic-cloudsql#0>. [Accessed 4 May 2022].
- [59] "Jib GitHub page," [Online]. Available: <https://github.com/GoogleContainerTools/jib>. [Accessed 4 May 2022].
- [60] "Getting started with authentication," [Online]. Available: <https://cloud.google.com/docs/authentication/getting-started>. [Accessed 4 May 2022].
- [61] A. Rai, "How To deploy React App on Shared Hosting(Cpanel)," 12 June 2019. [Online]. Available: <https://medium.com/@aforamitrai/how-to-deploy-react-app-on-shared-hosting-cpanel-d682b0342424>. [Accessed 4 May 2022].
- [62] G. Romero, "What is Postman API Test," Encora, 29 June 2021. [Online]. Available: <https://www.encora.com/insights/what-is-postman-api-test>. [Accessed 22 April 2022].
- [63] J. Unadkat, "Manual Testing for Beginners," BrowserStack, 11 December 2021. [Online]. Available: <https://www.browserstack.com/guide/manual-testing-tutorial>. [Accessed 22 April 2022].
- [64] "Site24x7 Website availability," [Online]. Available: <https://www.site24x7.com/web-site-availability.html>. [Accessed 22 April 2022].
- [65] "What is Automated Testing?," SmartBear, [Online]. Available: <https://smartbear.com/learn/automated-testing/what-is-automated-testing/>. [Accessed 22 April 2022].
- [66] "What is CI/CD?," 31 Jan 2018. [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-ci-cd>. [Accessed 22 April 2022].
- [67] E. S. Raymod, "The Cathedral and the Bazaar," 2000. [Online]. Available: <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>. [Accessed 23 April 2022].
- [68] "Docker Documentation - What is Container," [Online]. Available: <https://www.docker.com/resources/what-container/>. [Accessed 22 April 2022].

- [69] "Git Official Website," [Online]. Available: <https://git-scm.com/>. [Accessed 23 April 2022].
- [70] "Estonian Transport Administration," 2022. [Online]. Available: <https://www.mnt.ee/et/soiduk/soidukite-ja-masinate-kategooriad>. [Accessed 23 March 2022].

Appendix 1 – Screenshots of VSP interface

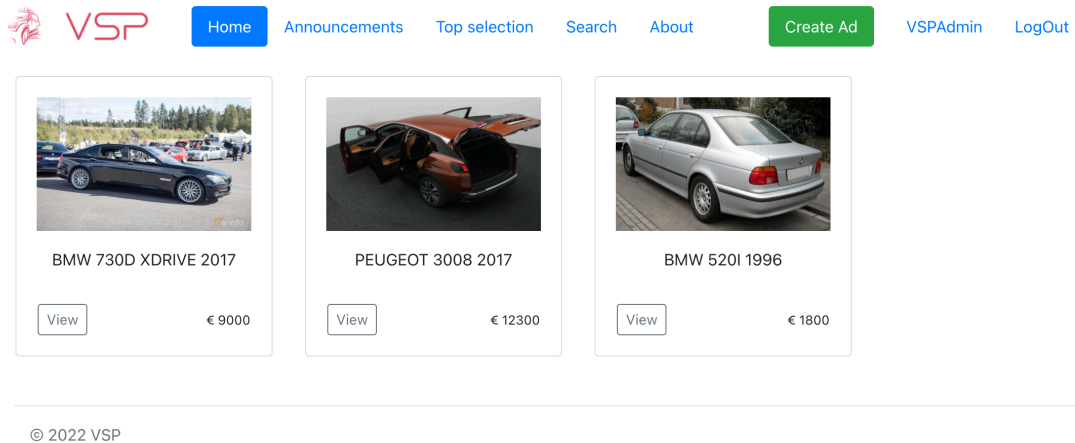


Figure 29. Example of VSP interface - Home page with latest advertisements.

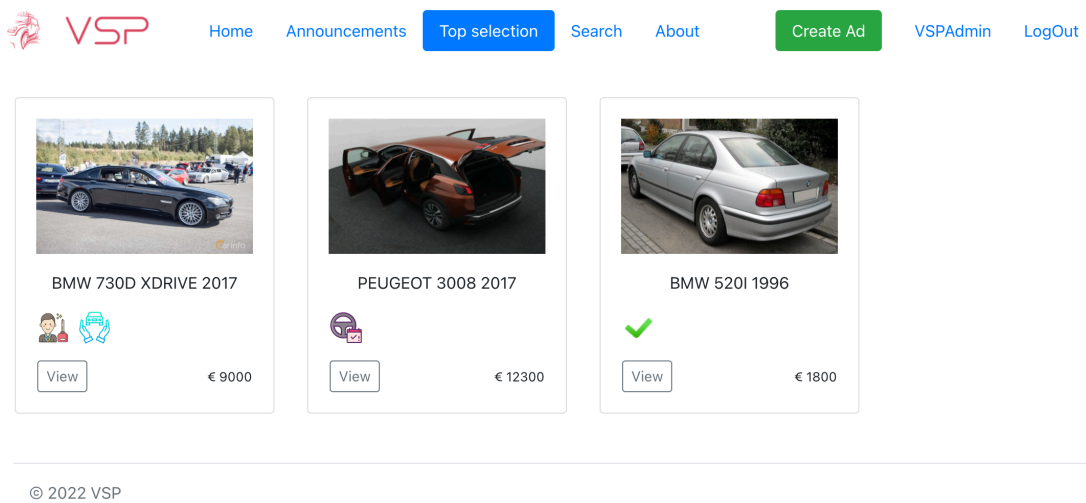


Figure 30. VSP interface - Top selection tab which displays badges assigned to vehicles.

Figure 31. VSP interface - Search page.

Engine	2993 DIISEL 195 kW B57D30A
Body	SEDAAN
Colour	MUST
Gearbox	AUTOMAAT
Drive	AWD
TUV valid	2022-03-31
Numberplate	999BMW
Registration	2018-02-14
Registration in Estonia	2018-06-03

Figure 32. VSP interface - Open advertisement with photos and vehicle details.

Price : € 9000

About

Odometer reading : 170000 km

Arestitud sõiduk

Contacts

Anton

58391273

Technical Data

Length	5098
Width	1902
Height	1467
Registered weight	2540
Unladen weight	1900

Figure 33. VSP interface - Second row of open advertisement. Shows price, vehicle details, owner contacts and technical data on the right.




2 technical inspections		8 registry actions		1 restrictions	
2020-02-11		2022-01-07		2019-03-01	
Type	Korraline	Action	Ajutine kustutamine e-teeninduses	Type	KEELUMARGE
Odometer	64335			Setter	HARJU MAAKOHUS
Decision	Tehniliselt korras	2018-01-11			
Performed	Center Services OÜ	Action	Reg.märk tagastatud		
2017-10-03		2018-01-11			
Type	Korraline	Action	Reg.märgi hoiule jätmine		
Decision	Tehniliselt korras	2018-01-11			
Performed	Maanteeamet	Action	Numbrimärk (korduv)		

Figure 34. VSP interface - Third row of open advertisement. Shows details about technical inspections, registry actions and restrictions.

VSPAdmin Profile

Email: vsp@vsp.ee

My ads:

 <p>BMW 730D XDRIVE 2017</p> <p>View Edit Delete € 9000</p>	 <p>PEUGEOT 3008 2017</p> <p>View Edit Delete € 12300</p>	 <p>BMW 520I 1996</p> <p>View Edit Delete € 1800</p>
--	--	--

© 2022 VSP

Figure 35. VSP interface - Administrator profile. Administrator sees every advertisement and is able to edit or remove it.

Odometer

Details

Price

Contact name

Contact email

Contact phone

Save details

© 2022 VSP

Figure 36. VSP interface - edit screen of existing advertisement.

Appendix 2 – Non-exclusive licence for reproduction and publication of a graduation thesis¹

I Anton Nikiforov

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Development of Vehicle Sales Web Application” supervised by Aleksei Talisainen:
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

04.12.2022

¹ The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

