OULU UNIVERSITY OF APPLIED SCIENCES

Abdishakur Hassan

# WEB DEVELOPER’S DIARY

# WEB DEVELOPER'S DIARY

Abdishakur Hassan
Bachelor's Thesis
Spring 2020
Information Technology
Oulu University of Applied Sciences

## ABSTRACT

Oulu University of Applied Sciences
Information Technology, Option of Software Development

Author: Abdishakur Hassan
Title of Bachelor´s thesis: Web Developer's Diary
Supervisor: Jukka Jauhiainen
Term and year of completion: Spring 2020 Number of pages: 57

This thesis has been conducted in Comille Oy customer's projects. Comille Oy provides Software development services, digital services, IT services and solutions for internal and external customers. This thesis is a result of daily development work at Comille Oy.

The aim of the thesis was to learn and develop Laravel development environment skills and acquire new skills regarding the Ruby on Rails web-development framework while developing customer projects.

The entries of the thesis were made on a weekly basis, the entries describe daily development work, tasks, and solutions that were provided to solve the tasks. The entries also describe some general web development tools, concepts, and terms that were encountered and used during the development process.

This thesis describes the daily work of a Junior Web developer's job and gives a clear reflection of a normal working day of a Web developer. As a developer, the work consists of designing and implementing a functional application according to the client's specifications, programming, integrating data from database and testing.

Keywords: Web development, Laravel, Ruby on Rails

## CONTENTS

## VOCUBULARY

| | |
|---|---|
| **AJAX** | Asynchronous JavaScript And Xml |
| **API** | Application Programming Interface |
| **COC** | Convention Over Configuration |
| **CMS** | Content Management System |
| **CRUD** | Create Read Update Delete |
| **CSS** | Cascading Style Sheets |
| **DRY** | Do Not Repeat Yourself |
| **ERB** | Embedded Ruby |
| **ERP** | Enterprise Resource Planning |
| **HTML** | Hypertext Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **JSON** | JavaScript Object Notation |
| **MVC** | Model View Controller |
| **PHP** | Hypertext Preprocessor |
| **REST** | Representational State Transfer |
| **ROR** | Ruby on Rails |
| **SASS** | Syntactically Awesome Style Sheets |

| | |
|---|---|
| **SPA** | Single Page Application |
| **SQL** | Structured Query Language |
| **UI** | User Interface |
| **URL** | Uniform Resource Locator |
| **Slim** | Slim is a Ruby template language. |

# 1 INTRODUCTION

The thesis is a diary-based thesis which describes my daily development work as a junior developer at Comille Oy for eight weeks. During this period, I have documented my overall work, all the tasks that I was assigned to, their solutions and the steps that were taken to come up with a solution. The thesis describes daily tasks tools that have been used while working on the given tasks. I participated in different customer projects, and I was a member of a development team that consisted of three people. All the development work was done at the company's office which is in Kempele.

The thesis does not mention detailed information about the projects or the customers but there will be a brief description about which project I worked with at any given day and the tasks that I have done for that project. The diary is mainly about what I have done for each day.

The basis of the thesis was to acquire new skills about web development using different frameworks at the same time while applying the skills that had been already acquired from the school or by self-taught. Prior to the start of the thesis, the essential skills that are needed are basic web development skills, such as HTML, CSS, JavaScript, a version control system such as Git, and overall understanding of how a web application works. Other skills that might help as a developer are good communication skills, problem-solving and an ability to learn quickly.

The daily responsibilities varied according to the project. The main responsibilities were creating new products and maintenance work for existing products. The Maintenance work was mainly bug-fixing, implementing new functionalities on existing products, updating existing functionalities and adopting the product to the changing environment. The main aim of the maintenance work is to create a totally new feature or requirement, improve the overall performance of the application, and the customer experience and minimize technical errors. As a web developer, the maintenance work helps to get familiar with the tools and systems. It also gives an opportunity to learn from past development mistakes and solutions.

# 2 TECHNOLOGY

This chapter provides a short description and an overview of the technologies and architecture used during the development process in different projects.

## 2.1 Laravel

Laravel is a PHP based web framework (1). The Laravel framework makes it easier to develop a web application because it provides a ready setup, architecture, and dependency packages that are readily available for the programmer. Laravel uses the MVC model.

## 2.2 Rails

Rails is a web application framework which was written in Ruby (2). Rails has a lot of ready-made features that help the developer to focus on the actual application logic. Rails has the Ruby programming language which is nicely readable and easier to code than the traditional object-oriented programming languages. Rails is an MVC based framework and it has other paradigms, such as convention over configuration (COC) (3) and it does not repeat itself (DRY) (4).

## 2.3 MVC

MVC (5) is an acronym for “Model View Controller”. MVC is a software architecture that divides the application into three parts: the model, the view, and the controller.
The model manages the data and its fundamental behaviors. It responds to requests for information and instruction to modify the state of the information.
The view provides a user interface for the application and it is the representation layer of the information. The view is where the user of the application normally interacts with the application.
The controller receives a user input and converts the input into commands that make calls to model objects and the view to perform appropriate actions. The MVC architecture gives an application a better structure that makes it easier to work with the application.

## 2.4 REST API

REST (6) stands for Representational State Transfer. It is an architectural style for an API (the Application Programming Interface). An API is an interface that allows applications to communicate with each other (7). For an API to be RESTful, it must adhere to six guiding constraints which are:

- Client – Server separation
- Uniform interface
- Stateless
- Layered system
- Cacheable
- Code-on-demand (optional)

**Client – server separation**

The client and server act independently. The interaction between them is only in the form of requests. The client makes a request and the server returns a response to the client.

**Uniform interface**

A RESTful API must follow the same rules to communicate with each other. A Uniform interface result is the one that requests from different clients to look the same.

**Stateless**

A Server does not store a session state. The server does not remember which resource the user of the API requested or what kind of request the user made.

**Layered system**

Individual components cannot see beyond the immediate level they interact with. Between the client's request and server's response, there might be a middle layer, such as a security layer or other functionality.

**Cacheable**

This means that the data within a response to a request must be labelled as cacheable or non-cacheable.

**Code-on-demand (optional)**

The client can request code from the server and then execute that code.

Figure 1 shows how a REST API works. For example, a client wants to get a resource, such as a list of users, a photo or an article. The client sends a request to a server using a GET HTTP method, the server returns a response usually formatted in JSON which contains the requested data. In this case a list of users, a photo or an article is returned from the server. Other available HTTP methods are POST for creating new resource, the PUT method for updating and the DELETE method for deleting.

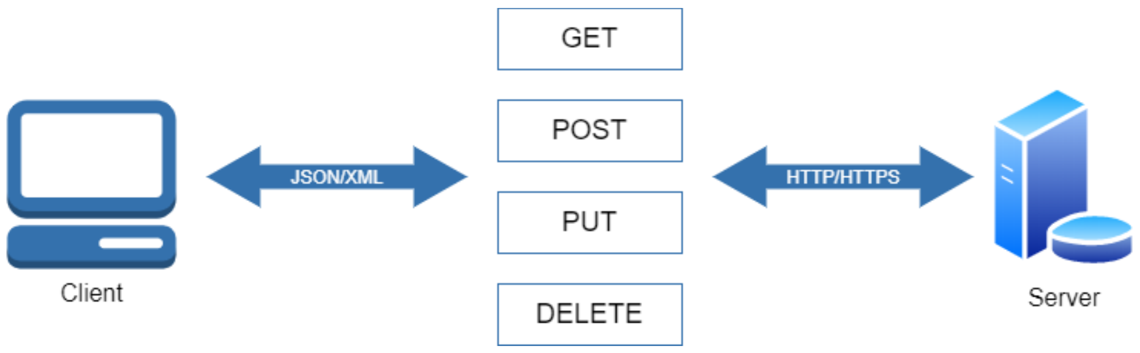


*FIGURE 1. REST API*

# 3 PROJECTS

This chapter provides a short description about the projects that I have worked with.

## 3.1 Autolle.com

Autolle.com is a car dealership website that sales new and used cars. Autolle.com is a project that was developed with the Laravel framework. The project uses Vue.js (8) in the front-end, PHP in the backend and the MySQL database (9).
Autolle.com has two sections, the main website section and the Admin section. The main website section is used by the end-user to search, buy, sell cars, make a payment for reservations and contact Autolle.com. There are also other services that are available the main website. The Admin section is mainly used by Autolle.com staff. The Admin section is used for updating the main website contents, processing customer orders and managing website users. Autolle.com Homepage is shown below.

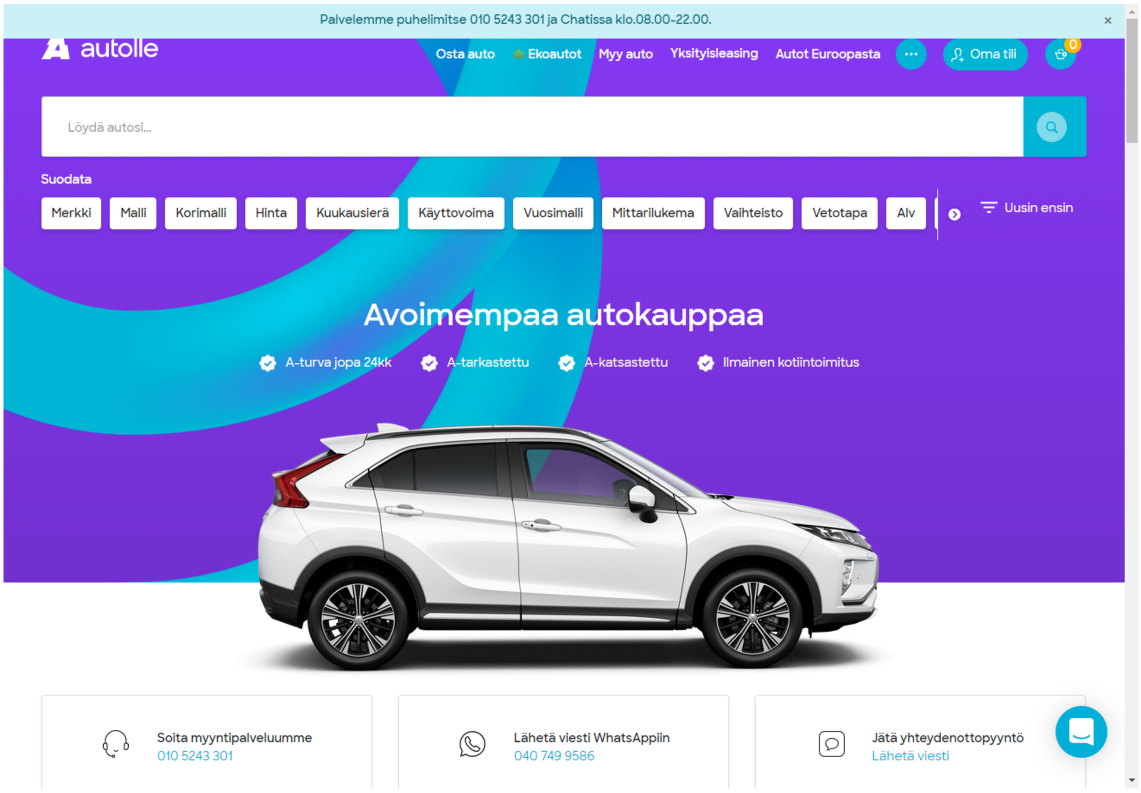
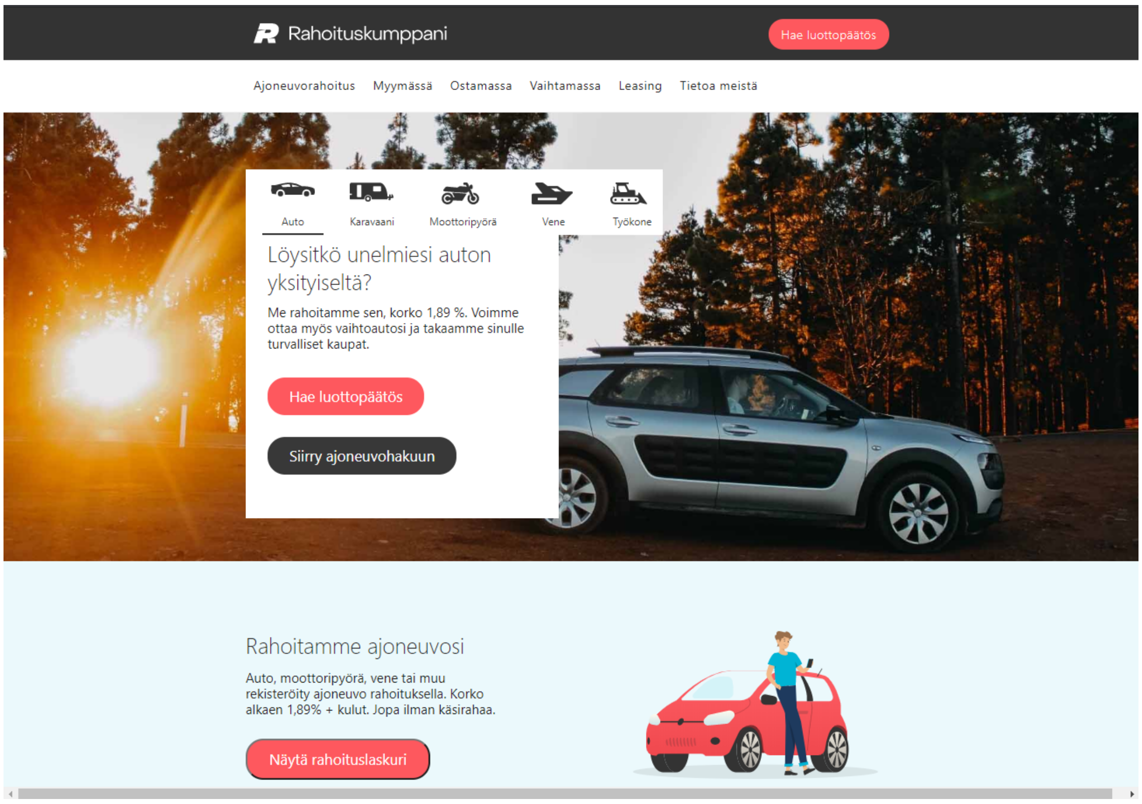


*FIGURE 2. Autolle.com Homepage.*

## 3.2 RK (Rahoituskumppani)

Rahoituskumppani is a website that provides financial services for vehicles. Rahoituskumppani is a project that was developed with the Ruby on Rails web development framework. The project uses the jQuery JavaScript library (10) and the PostgreSQL database (11).
Rahoituskumppani has an Admin side and a main website section. The Admin side is used by Rahoituskumppani staff for processing customer applications and managing the contents of the website. The main website section is used by the customers to make a funding application and to get more information about different available services. Figure 3 shows the Rahoituskumppani Homepage.



Löysitkö unelmiesi auton yksityiseltä?

Me rahoitamme sen, korko 1,89 %. Voimme ottaa myös vaihtoautosi ja takaamme sinulle turvalliset kaupat.

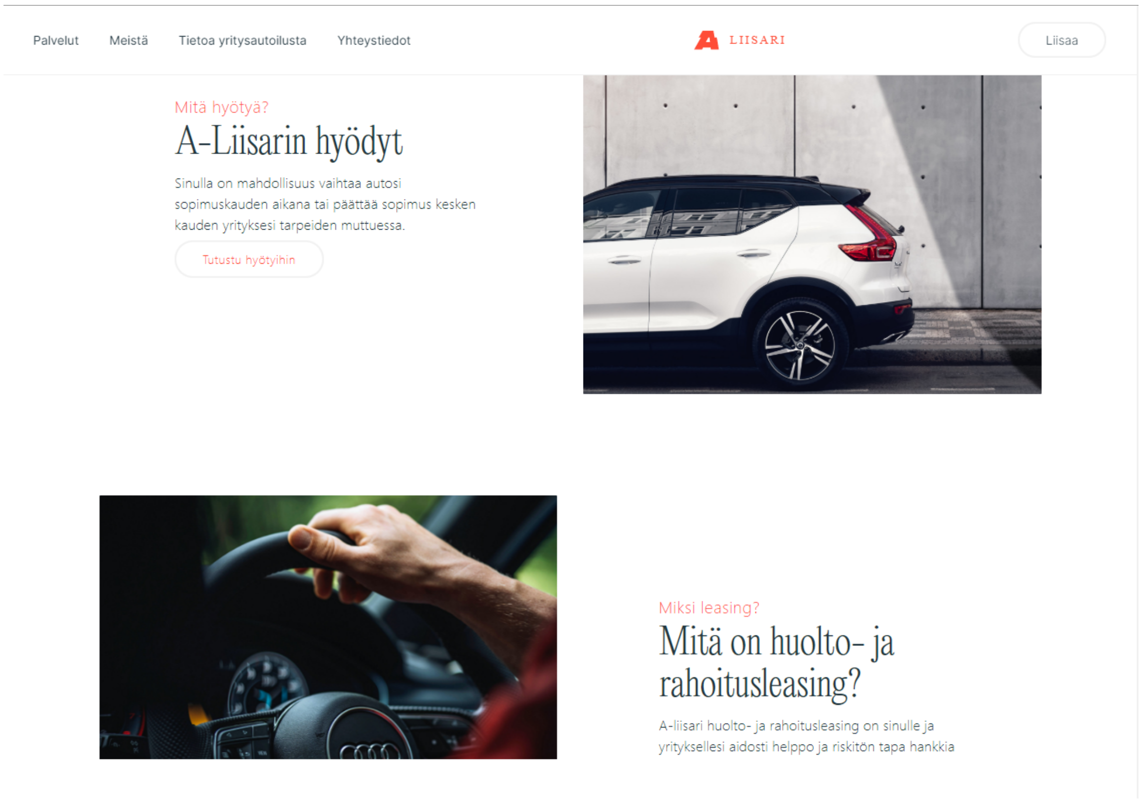
Rahoitamme ajoneuvosi

Auto, moottoripyörä, vene tai muu rekisteröity ajoneuvo rahoituksella. Korko alkaen 1,89% + kulut. Jopa ilman käsirahaa.

*FIGURE 3. Rahoituskumppani Homepage.*

## 3.3 Aliisari

Aliisari is a website that provides private and business leasing services. The Aliisari project was developed with Ruby on Rails framework. It uses jQuery JavaScript library and PostgreSQL database. The website has an Admin section that is used by Aliisari staff and a Main page section that is used by Aliisari customers. Aliisari Homepage is shown in figure 4.



Mitä hyötyä?

A-Liisarin hyödyt

Sinulla on mahdollisuus vaihtaa autosi sopimuskauden aikana tai päättää sopimus kesken kauden yrityksesi tarpeiden muttuessa.

Miksi leasing?

Mitä on huolto- ja rahoitusleasing?

A-liisari huolto- ja rahoitusleasing on sinulle ja yrityksellesi aidosti helppo ja riskitön tapa hankkia

*FIGURE 4. Aliisari Homepage.*

## 3.4 Nettix Pro REST API

Nettix Pro (12) is a REST API service that is used by Autolle.com and RK websites. The Nettix Pro provides an Enterprise resource planning (ERP) (13) system for car dealership companies. The Nettix Pro tool helps companies to manage inventory, quotation and reporting.

Autolle.com and RK send a GET request to the Nettix Pro REST API, for example, to get a list of cars that are available for sale. The Nettix pro REST API returns a JSON response that has the requested resource which is a list of all available cars for sale. After receiving the resource, Autolle.com and RK save or update the data to their local databases for future reuse because the API request is made four times per hour. The local database is used on the website.
Figure 5 shows how Autolle.com and RK websites use the Nettix Pro REST API. An example of a response from the Nettix Pro REST API is shown in figure 6 and 7.

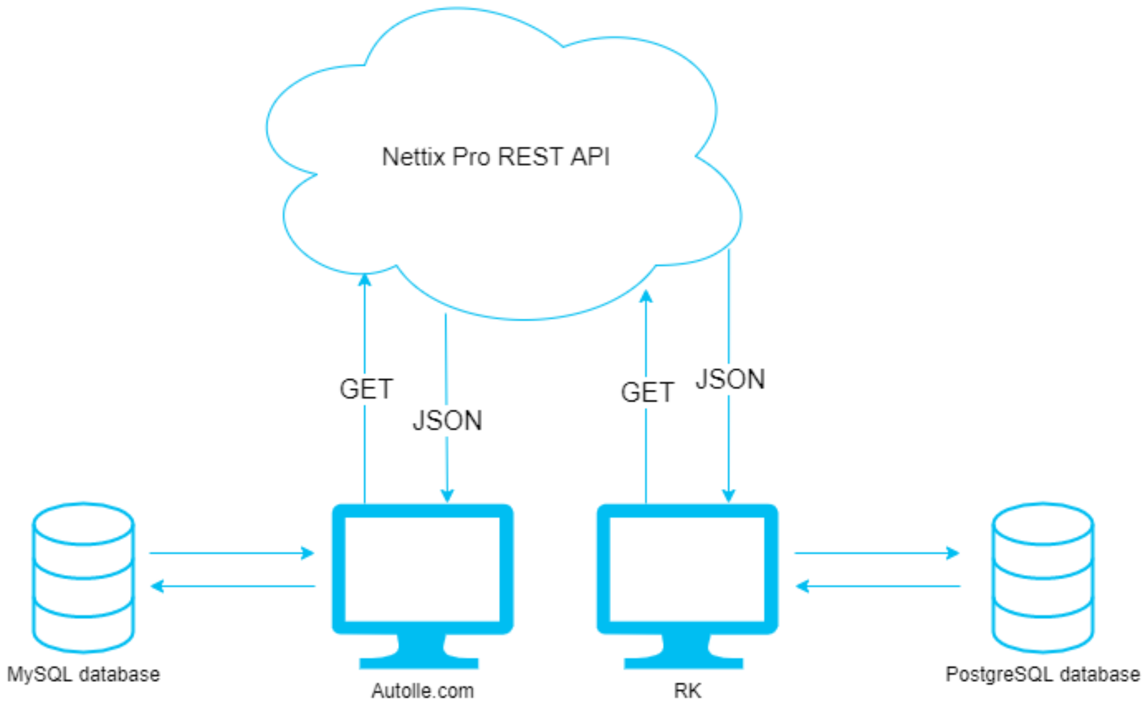
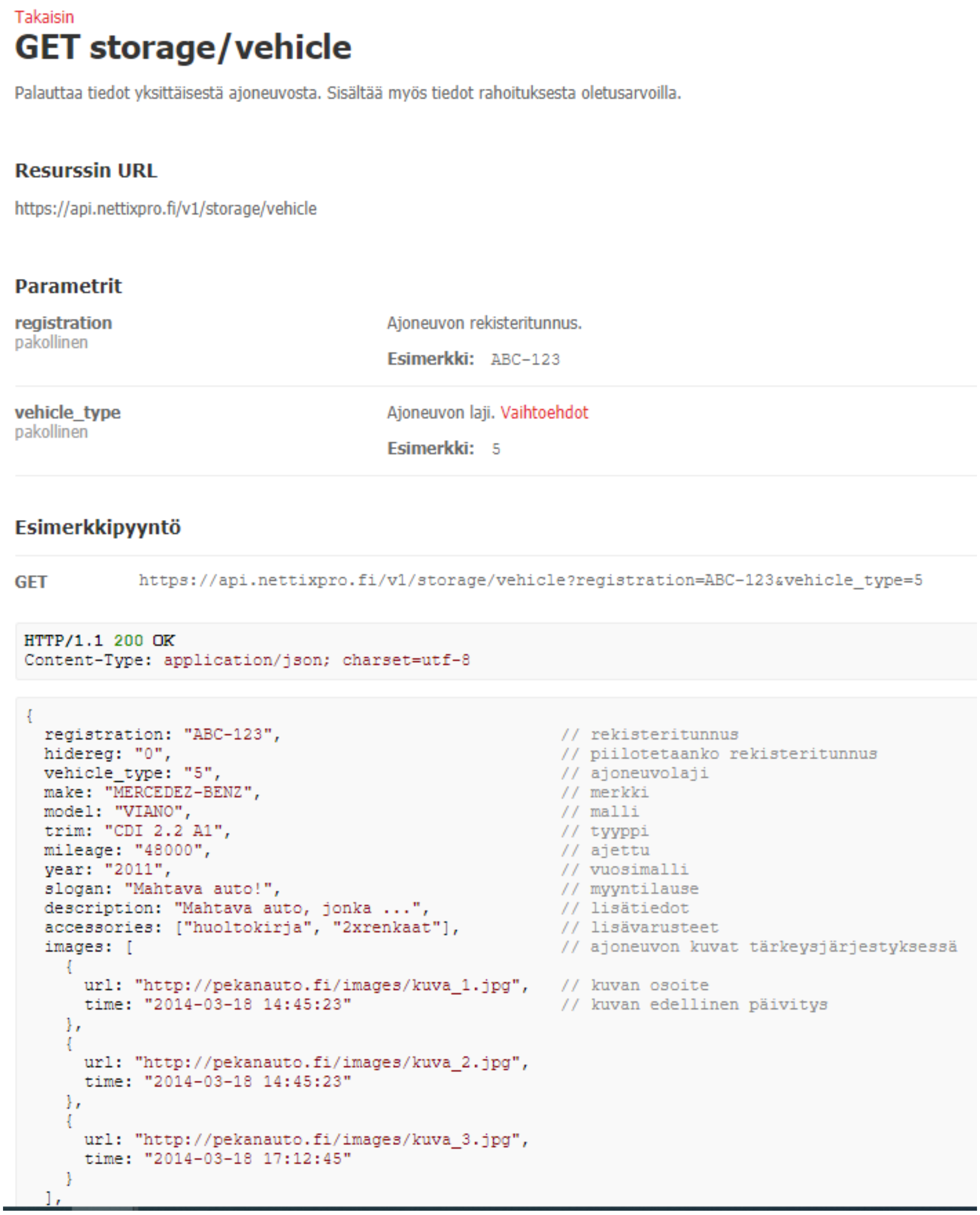


*FIGURE 5. How Autolle.com and RK websites use Nettix Pro REST API*



Takaisin

GET storage/vehicle

Palauttaa tiedot yksittäisestä ajoneuvosta. Sisältää myös tiedot rahoituksesta oletusarvoilla.

Resurssin URL

https://api.nettixpro.fi/v1/storage/vehicle

Parametrit

| | |
|---|---|
| **registration** pakollinen | Ajoneuvon rekisteritunnus. **Esimerkki:** `ABC-123` |
| **vehicle_type** pakollinen | Ajoneuvon laji. Vaihtoehdot **Esimerkki:** `5` |

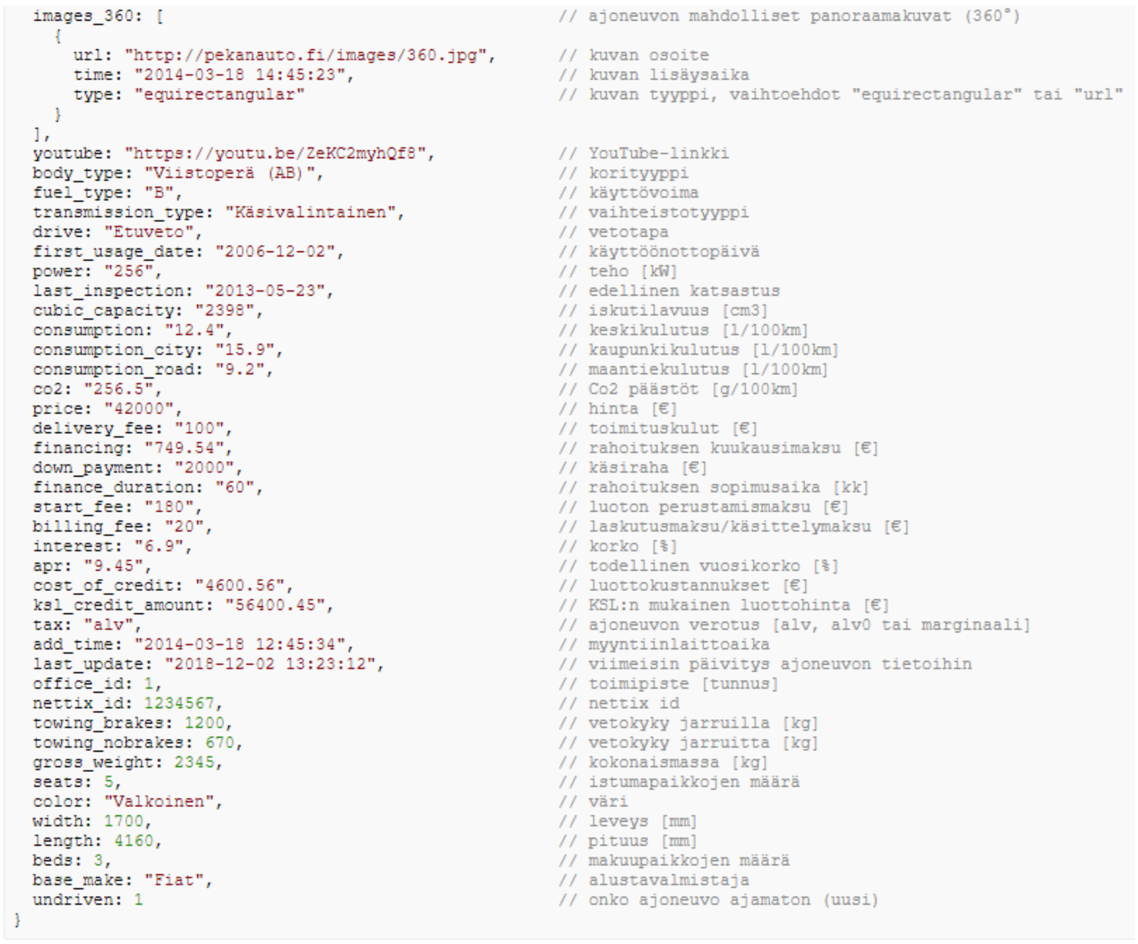
Esimerkkipyyntö

```
GET     https://api.nettixpro.fi/v1/storage/vehicle?registration=ABC-123&vehicle_type=5
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

```
{
  registration: "ABC-123",                              // rekisteritunnus
  hidereg: "0",                                         // piilotetaanko rekisteritunnus
  vehicle_type: "5",                                    // ajoneuvolaji
  make: "MERCEDEZ-BENZ",                                // merkki
  model: "VIANO",                                       // malli
  trim: "CDI 2.2 A1",                                   // tyyppi
  mileage: "48000",                                     // ajettu
  year: "2011",                                         // vuosimalli
  slogan: "Mahtava auto!",                              // myyntilause
  description: "Mahtava auto, jonka ...",               // lisätiedot
  accessories: ["huoltokirja", "2xrenkaat"],            // lisävarusteet
  images: [                                             // ajoneuvon kuvat tärkeysjärjestyksessä
    {
      url: "http://pekanauto.fi/images/kuva_1.jpg",     // kuvan osoite
      time: "2014-03-18 14:45:23"                       // kuvan edellinen päivitys
    },
    {
      url: "http://pekanauto.fi/images/kuva_2.jpg",
      time: "2014-03-18 14:45:23"
    },
    {
      url: "http://pekanauto.fi/images/kuva_3.jpg",
      time: "2014-03-18 17:12:45"
    }
  ],
```

*FIGURE 6. Example Nettix Pro REST API response (14)*



```
  images_360: [                                       // ajoneuvon mahdolliset panoraamakuvat (360°)
    {
      url: "http://pekanauto.fi/images/360.jpg",      // kuvan osoite
      time: "2014-03-18 14:45:23",                    // kuvan lisäysaika
      type: "equirectangular"                         // kuvan tyyppi, vaihtoehdot "equirectangular" tai "url"
    }
  ],
  youtube: "https://youtu.be/ZeKC2myhQf8",            // YouTube-linkki
  body_type: "Viistoperä (AB)",                       // korityyppi
  fuel_type: "B",                                     // käyttövoima
  transmission_type: "Käsivalintainen",               // vaihteistotyyppi
  drive: "Etuveto",                                   // vetotapa
  first_usage_date: "2006-12-02",                     // käyttöönottopäivä
  power: "256",                                       // teho [kW]
  last_inspection: "2013-05-23",                      // edellinen katsastus
  cubic_capacity: "2398",                             // iskutilavuus [cm3]
  consumption: "12.4",                                // keskikulutus [l/100km]
  consumption_city: "15.9",                           // kaupunkikulutus [l/100km]
  consumption_road: "9.2",                            // maantiekulutus [l/100km]
  co2: "256.5",                                       // Co2 päästöt [g/100km]
  price: "42000",                                     // hinta [€]
  delivery_fee: "100",                                // toimituskulut [€]
  financing: "749.54",                                // rahoituksen kuukausimaksu [€]
  down_payment: "2000",                               // käsiraha [€]
  finance_duration: "60",                             // rahoituksen sopimusaika [kk]
  start_fee: "180",                                   // luoton perustamismaksu [€]
  billing_fee: "20",                                  // laskutusmaksu/käsittelymaksu [€]
  interest: "6.9",                                    // korko [%]
  apr: "9.45",                                        // todellinen vuosikorko [%]
  cost_of_credit: "4600.56",                          // luottokustannukset [€]
  ksl_credit_amount: "56400.45",                      // KSL:n mukainen luottohinta [€]
  tax: "alv",                                         // ajoneuvon verotus [alv, alv0 tai marginaali]
  add_time: "2014-03-18 12:45:34",                    // myyntiinlaittoaika
  last_update: "2018-12-02 13:23:12",                 // viimeisin päivitys ajoneuvon tietoihin
  office_id: 1,                                       // toimipiste [tunnus]
  nettix_id: 1234567,                                 // nettix id
  towing_brakes: 1200,                                // vetokyky jarruilla [kg]
  towing_nobrakes: 670,                               // vetokyky jarruitta [kg]
  gross_weight: 2345,                                 // kokonaismassa [kg]
  seats: 5,                                           // istumapaikkojen määrä
  color: "Valkoinen",                                 // väri
  width: 1700,                                        // leveys [mm]
  length: 4160,                                       // pituus [mm]
  beds: 3,                                            // makuupaikkojen määrä
  base_make: "Fiat",                                  // alustavalmistaja
  undriven: 1                                         // onko ajoneuvo ajamaton (uusi)
}
```

*FIGURE 7. Example Nettix Pro REST API response*

# 4 DIARY ENTRIES

This section will hold all the diary entries for eight weeks about my daily work at Comille Oy. At the end of each week there will be an analysis about the whole week.

## 4.1 Week 1

### 4.1.1 Monday 23.3

I started to work on a new landing page for private leasing cars. The task was to make a fully functioning landing page that shows all the cars that are available for private leasing. The task included adding new columns to the database to control the details of the private leasing cars from the Admin side. In this task I used the Laravel framework and the task was done with Vue.js, which is JavaScript framework for building user interfaces, HTML, Sass and Bootstrap.

Laravel framework uses a Blade templating engine. Blade is a view that renders PHP code, Vue.js components, HTML and CSS styles. Blade outputs a nice representation of all these codes and it is what the user of the application sees. (15.)

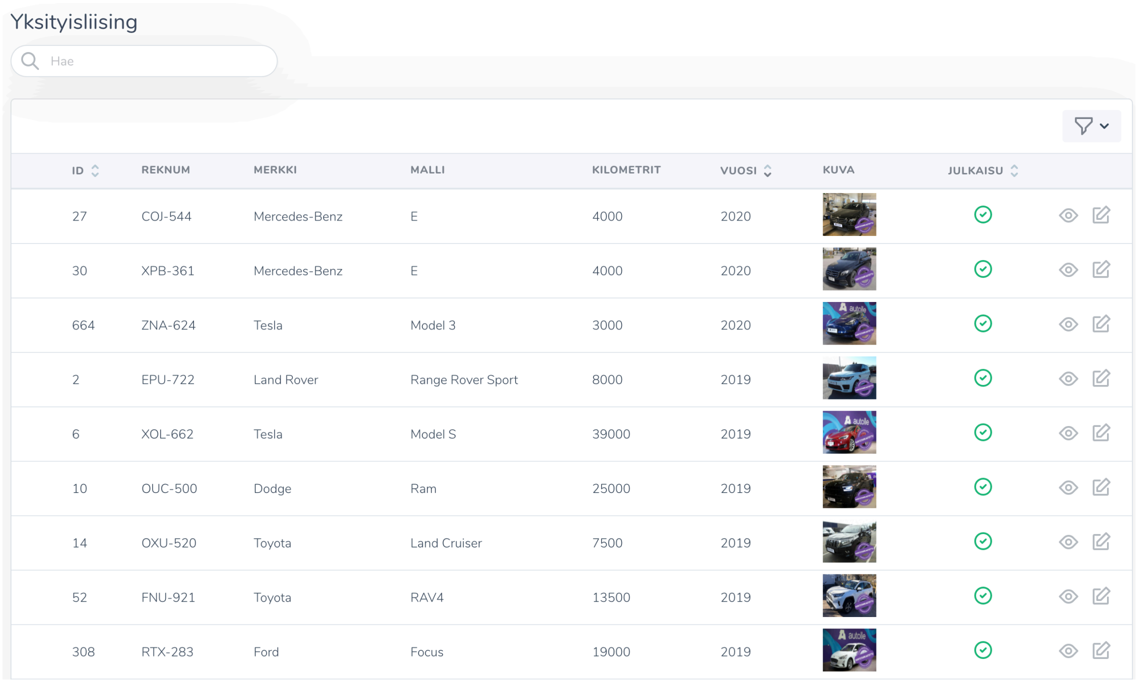
The landing page consists of different Vue.js components. Vue.js components are custom elements that consist of HTML elements, JavaScript code and CSS styles. A component is a reusable element that can be used in different places throughout the application. (16.) The landing page will have a search component, which is used for searching cars, a card component which shows an image and a detail summary of one car. The card is clickable and redirects to a single car page. To use these components together I have created another parent component that takes a search component and a card component as a children component. The parent can pass data to the children components through Props. Props makes the child component to accept data from the parent component (17).

I have added a column to the cars table in the database to hold the details for all the private leasing cars. These details are read from the database through a Controller and then the values are passed to the View. The Controller responds to user actions and requests. The action in this case is to fetch all private leasing cars and pass them to the Blade template, which is the View. The View also passes down this data to the Vue.js component which renders the data one by one.

I spent the day creating the placeholder's data in the database and displaying the placeholders on the landing page.

### 4.1.2 Tuesday 24.3

Today, I continued with the same task as yesterday and started working the Admin side. The Admin or the Admin panel is an administration section on the website that controls the creation of contents for the webpage. In this project I am using Laravel Nova (18). Laravel Nova is an administration panel tool for the Laravel framework, and it is mostly used with Laravel projects. I created a new section for controlling the private leasing cars. This section is responsible for displaying or hiding cars, editing, or removing details of every car. A car detail is a text or an attribute that describes more about a specific private leasing car. The Private leasing control panel is shown in figure 8.



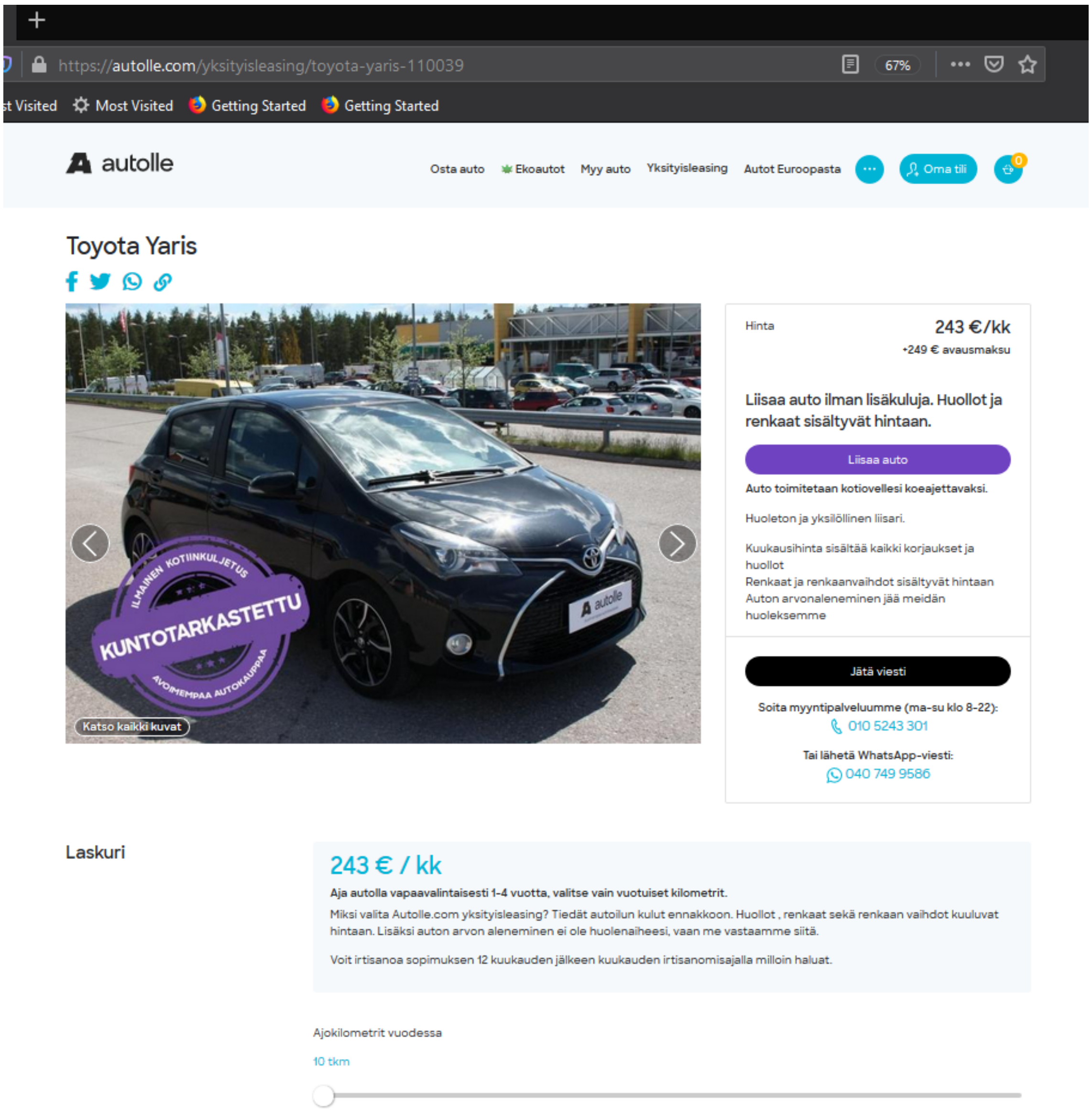
Yksityisliising

Hae

| ID | REKNUM | MERKKI | MALLI | KILOMETRIT | VUOSI | KUVA | JULKAISU |
|---|---|---|---|---|---|---|---|
| 27 | COJ-544 | Mercedes-Benz | E | 4000 | 2020 | | ✓ |
| 30 | XPB-361 | Mercedes-Benz | E | 4000 | 2020 | | ✓ |
| 664 | ZNA-624 | Tesla | Model 3 | 3000 | 2020 | | ✓ |
| 2 | EPU-722 | Land Rover | Range Rover Sport | 8000 | 2019 | | ✓ |
| 6 | XOL-662 | Tesla | Model S | 39000 | 2019 | | ✓ |
| 10 | OUC-500 | Dodge | Ram | 25000 | 2019 | | ✓ |
| 14 | OXU-520 | Toyota | Land Cruiser | 7500 | 2019 | | ✓ |
| 52 | FNU-921 | Toyota | RAV4 | 13500 | 2019 | | ✓ |
| 308 | RTX-283 | Ford | Focus | 19000 | 2019 | | ✓ |

*FIGURE 8. Private leasing control panel*

After I finished with the Admin side, I started outputting the data to the view and replaced all the placeholders with actual values that come from the database. These values from the database are controlled via the administration panel. Also, I added the ability to redirect to a single personal leasing car. The landing page just shows an image and a summary text for every car that is available, and the single car page shows more useful details about the car, such as different images or car equipment (figure 9).



Toyota Yaris

Hinta 243 €/kk
+249 € avausmaksu

Liisaa auto ilman lisäkuluja. Huollot ja renkaat sisältyvät hintaan.

Liisaa auto

Auto toimitetaan kotiovellesi koeajettavaksi.

Huoleton ja yksilöllinen liisari.

Kuukausihinta sisältää kaikki korjaukset ja huollot
Renkaat ja renkaanvaihdot sisältyvät hintaan
Auton arvonaleneminen jää meidän huoleksemme

Jätä viesti

Soita myyntipalveluumme (ma-su klo 8-22):
010 5243 301

Tai lähetä WhatsApp-viesti:
040 749 9586

Laskuri

243 € / kk

Aja autolla vapaavalintaisesti 1-4 vuotta, valitse vain vuotuiset kilometrit.

Miksi valita Autolle.com yksityisleasing? Tiedät autoilun kulut ennakkoon. Huollot , renkaat sekä renkaan vaihdot kuuluvat hintaan. Lisäksi auton arvon aleneminen ei ole huolenaiheesi, vaan me vastaamme siitä.

Voit irtisanoa sopimuksen 12 kuukauden jälkeen kuukauden irtisanomisajalla milloin haluat.

Ajokilometrit vuodessa
10 tkm

*FIGURE 9. Private leasing single car page*

I also implemented a search filter in the landing page. The search filter has only one input which takes one or more letters or word that is used as a keyword to filter out those cars that match the search criteria. The filter is simple JavaScript code that checks if the searched keyword is available in every attribute of the car. The search filter is responsive and shows the result of the search immediately as the user is typing.

A lot of debugging and testing was done especially in the admin panel side because the data fields in the admin side need to be controlled so that only allowed values can be inserted and may be modified. This implementation required frequent manual testing.

### 4.1.3 Wednesday 25.3

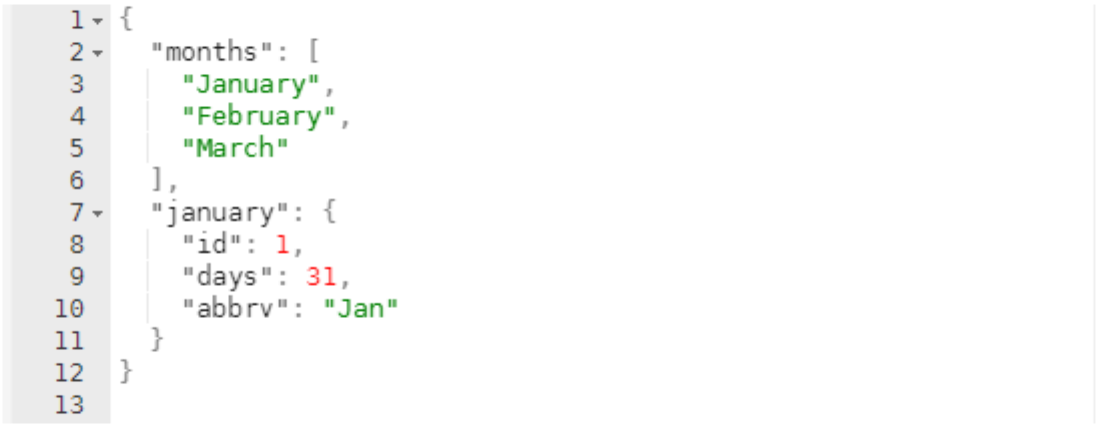
Today, I continued with the personal leasing landing page to implement more functionalities for the page. The task of the day was to implement a slider that calculates the monthly price approximation of every car. I created a Vue.js component which uses an NPM package module. The NPM module is JavaScript code that someone else has written and it is available in the NPM packages registry for download (19). Installed NPM packages are available in a “node_modules” folder in the current project environment. With the help of this external package, I have created a well-functioning slider that adjusts the prices according to the user’s requirements.

I have more attributes from private leasing cars to the single car page, some of the attributes that I added are leasing periods, more description of each leasing package and a precalculated price per month. Some of these were already available on the main landing page.

Today, I also had a meeting with the customer and presented a demo of already implemented functionalities for the project. I answered questions regarding some of the features and explained how the upcoming features will be integrated into the features that I had already achieved. After the meeting had ended, I immediately started implementing the recommendations of the customer.

### 4.1.4 Thursday 26.3

Today, I continued with the previous task and I added more database columns to the existing cars table. These columns are meant to hold different data sets which are meant to be used in the UI side. I also made an implementation of the business logic with PHP. The business logic is just a simple procedure that gets a value from a column in the database, performs an operation and then collects the outcome as JSON data and saves it back to the database. JSON is a format for storing data that uses a key value pair (20). A JSON example is shown below (Figure 10).



```
{
  "months": [
    "January",
    "February",
    "March"
  ],
  "january": {
    "id": 1,
    "days": 31,
    "abbrv": "Jan"
  }
}
```

*FIGURE 10. A JSON with two objects an array and an object*

After implementing the first part, the rest of the day went by debugging and reviewing the code. It all started after deploying changes to the server. These changes generated errors in the production environment. After spending few hours, I could not figure out what was the bug because the same code works in the development environment and not in the production environment. I sought help and found out that the errors were caused because of different package versions in the development and production environment. The production environment packages were updated using a composer and this solved the problem. A Composer is a tool that installs or updates dependency packages in the Laravel project. (21.)

### 4.1.5 Friday 27.3

Today, I started adding a header section to the private leasing landing page. The header consists of an image, icons, and texts of different styles. I also made the landing page to be responsive fitting different screen sizes, resized all the card components to the same size since all the images in the card are of different size thus affecting the card size and making some of them large and others small.

I redesigned the single car page, added more fields to show data from the database. Also, I reimplemented the slider that calculates the price approximation, reduced the slider to one from two and reduced the code that makes the calculation based on the selection of the slider.

### 4.1.6 Week Analysis

During this week, a full functioning landing page was achieved. An administration panel section for controlling the landing page was created. I fetched data from the database and presented it to the user. I implemented a Private leasing cars landing page and a Private leasing a single car page. I did a lot of refactoring, debugging and manual testing to make sure that things were working in the exact way as they were designed.

During this week, I have implemented create, read, update, and destroy (CRUD) operations in the administration side and implemented the user interface that helps the user to interact with the application.

## 4.2 Week 2

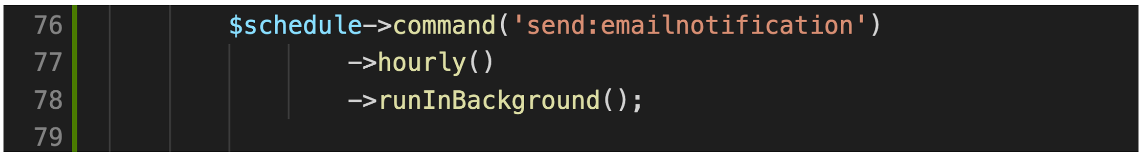
### 4.2.1 Monday 30.3

Today, I continued with the private leasing landing page implementation. The main work focused on the user interface and the overall page. I did some style changes, such as positioning of elements within the landing page so that makes it easy for the users to get information immediately as they visit the page. The style changes were modifying button sizes, applying different colors on different states, and aligning items. All these styles were implemented with CSS and Bootstrap.

I also added pagination to the landing page. Pagination is reducing the length of items that can appear on one page at any given time. I implemented the pagination with JavaScript using the Vue.js component. The component is passed to a set of items as props. Then the component calculates the total number of the items and decides how many pages are needed to be created. I also integrated a contact form to the page. The form already existed on other pages so I just linked it to the landing page.

### 4.2.2 Tuesday 31.3

The task today was to create a Task Scheduling Command which updates the details of private leasing cars. Task Scheduling in the Laravel framework is an automated task manager which executes a specific task or a command that performs a certain operation (22). To make use of the Task Scheduler I created a PHP custom class that implements the business logic. The class gets data from the database, performs business logic operations, and updates the database again with the newly modified values. For example, if the price of a car changes, this class performs an operation that updates other columns in the database that are linked to this price change. The Task Scheduler is assigned to run this task to immediately update the data when there is a change in the main source of the data. Figure 11 shows a Scheduling command that runs every hour for sending email.



```
        $schedule->command('send:emailnotification')
                ->hourly()
                ->runInBackground();
```

*FIGURE 11. Schedule command that runs email sending task every hour*

I also added two new filters to the Admin side. A filter is code that sets a certain condition that needs to be fulfilled. The first filter separates the private leasing car contacts from other contacts from the webpage. The second filter sets the suitability of a car for private leasing, checks whether the car is available for leasing.

I also made style changes, updated an existing page, and did some general bug fixes in different parts of the webpage.

### 4.2.3 Wednesday 1.4

Today, the task was to go through and understand the previous code base that integrates the Paytrail payment system to the website. Paytrail is an online payment service. (23.)

The payment process has five steps. Each step has its own form that collects specific details about the user, extra products, and services that the user is interested in, the delivery address and summary of the paid product and service. Figure 12 shows a simplified overview of the steps that are involved in the payment process. Each step has a lot of functionalities and options that the user can select according to their needs.

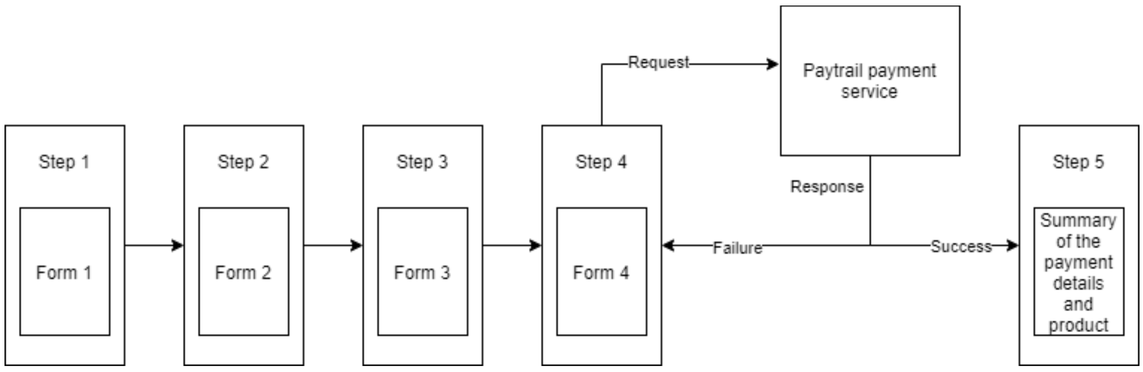


*FIGURE 12. Original payment process*

### 4.2.4 Thursday 2.4

I reimplemented the existing code base of the Paytrail payment system for private leasing cars. I reduced the steps of the payment process from the current steps of five to three steps. All the optional selections were removed from this reduced version. The options were not needed since the user has already selected a suitable car for leasing and the only important step that is remaining is the payment. Figure 13 shows the reduced steps of the payment process.

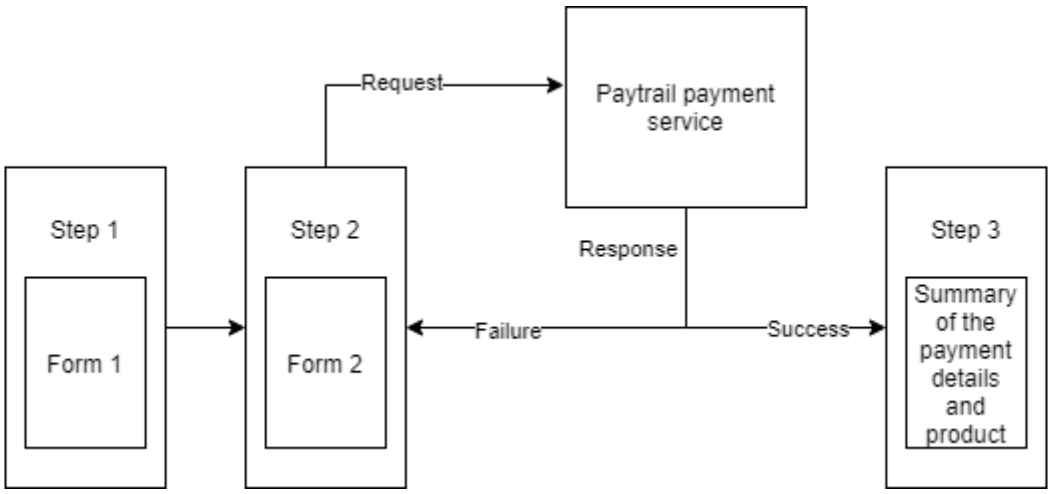


*FIGURE 13. Reduced payment process*

### 4.2.5 Friday 3.4

Today, I have reduced further the steps of the Paytrail payment system integration to the private leasing page, the steps were reduced further because the details could be combined to one step to minimize the number of steps. After the reduction, there was only one step that consisted of two different forms; one form collected user information and the other one delivery location. The user must fill the form one to proceed with the payment. The task of combining two different forms to a single action was challenging. I solved the problem by using the same validation method for both forms and a method that collects all input field values from the forms and submit all at once. The final version of minimized steps is shown in figure 14.

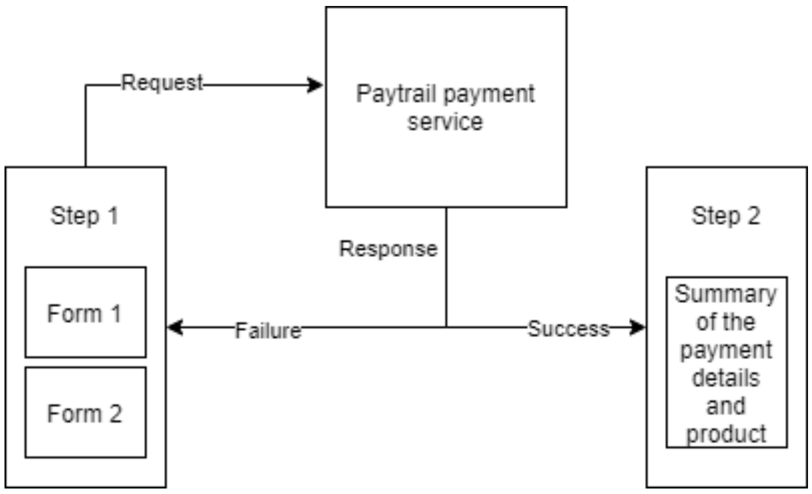
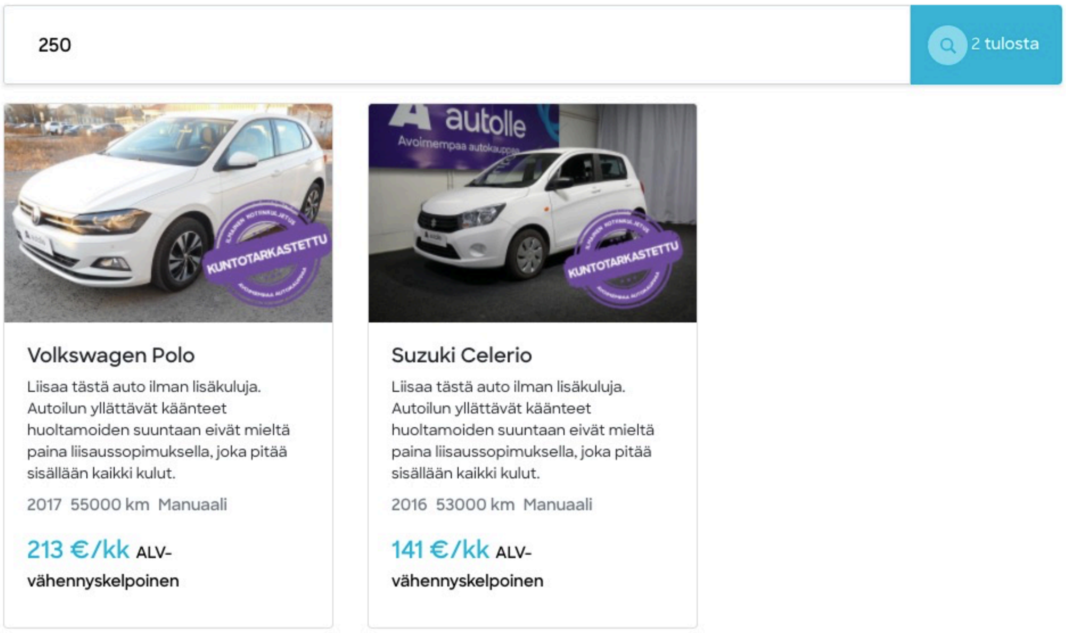


*FIGURE 14. Minimized payment process*

I also added a new search result feature to the leasing car search filter. The feature was that if a user puts the price to the search field, the filter displays all the prices that are less than or equal to the given price (Figure 15). For example, a 250-euro search term gives a result of cars that have the price of 250 euro and less. I have also changed some styles, mainly fonts and spacing. I also added a tax-deductible mark to those cars that fulfill tax-deduction criteria.


250

2 tulosta

Volkswagen Polo

Liisaa tästä auto ilman lisäkuluja. Autoilun yllättävät käänteet huoltamoiden suuntaan eivät mieltä paina liisaussopimuksella, joka pitää sisällään kaikki kulut.

2017 55000 km Manuaali

213 €/kk ALV-vähennyskelpoinen

Suzuki Celerio

Liisaa tästä auto ilman lisäkuluja. Autoilun yllättävät käänteet huoltamoiden suuntaan eivät mieltä paina liisaussopimuksella, joka pitää sisällään kaikki kulut.

2016 53000 km Manuaali

141 €/kk ALV-vähennyskelpoinen

*FIGURE 15. Search result*

### 4.2.6 Week Analysis

This week a lot was achieved. Most of the requirements for the landing page were implemented. The landing page was fully deployed successfully. Some of the main tasks of the week were overall style improvements: Task scheduling, understanding how the Paytrail payment system was integrated into the website, reducing payment steps from four to one and improving a private leasing car's search filter.

I have learnt a lot this week. Some of the tasks were challenging because the tasks involved creating database columns, saving user inputs, database queries, performing CRUD (Create, Read, Update, Destroy) operations, implementing business logic, testing and debugging.

## 4.3 Week 3

### 4.3.1 Monday 6.4

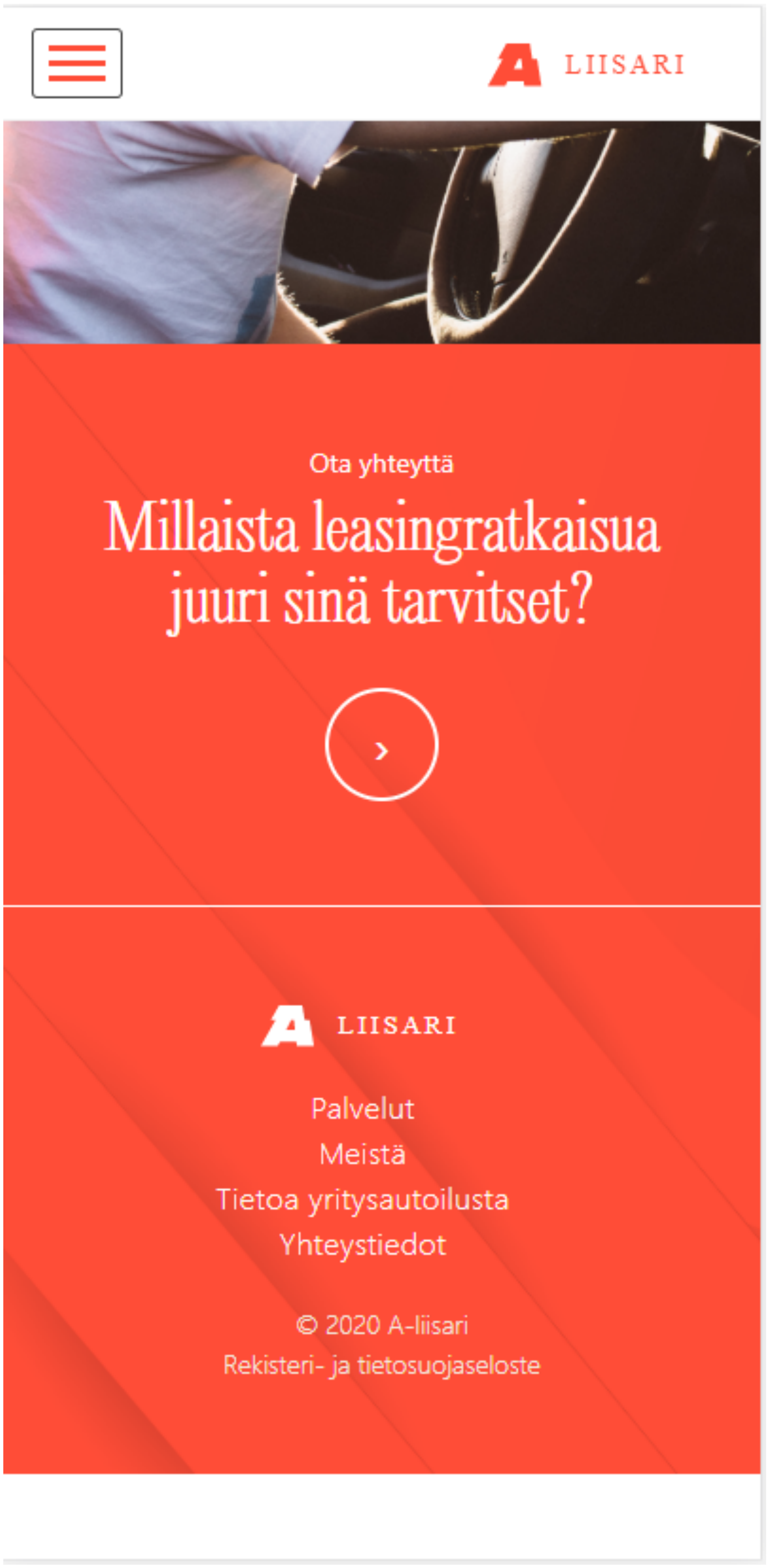
Today, I started working on a new project Aliisari. Most of the day went installing project boilerplates from GitLab, a version control management platform. I installed and imported packages and set up the development environment for this new project. The Aliisari project will be developed using the Ruby on Rails web development framework.

I fixed a bug on the private leasing landing page. The bug was a null value and it was caused by a situation when an element or elements could not find one or more car's attribute or car's price or any other value that is related to the car. This was caused when a new car had been just added to the database that had not been given a price or any other attribute that is used on the private leasing landing page. I fixed this bug by adding a condition that checks if the car has all the needed attributes or not. If there are one or more missing attributes, the operation returns an empty value instead of null otherwise it will return the value itself. In PHP, a variable of data type NULL is a variable that has no value assigned to it. (24.)

### 4.3.2 Tuesday 7.4

I continued with the Aliisari project. I started working on the Home page. In this project I will be following a design template and I will be implementing all the pages and functionality according to the provided design. I added Navbar, a Hero section, text and image columns and a Footer section to the home page. I also added responsiveness to the page fitting different screen resolutions. I used HTML, Sass and Bootstrap. Figure 16 shows a mobile view of the Footer section.

I updated the reservation fee for a private leasing car, and I also changed the content of the email notification on the private leasing landing page. Email notifications are normally sent when a user makes a reservation or a payment. Then the user is notified about these actions by email and the user is sent a short summary about the reservation or the payment.


LIISARI

Ota yhteyttä

Millaista leasingratkaisua juuri sinä tarvitset?

›

LIISARI

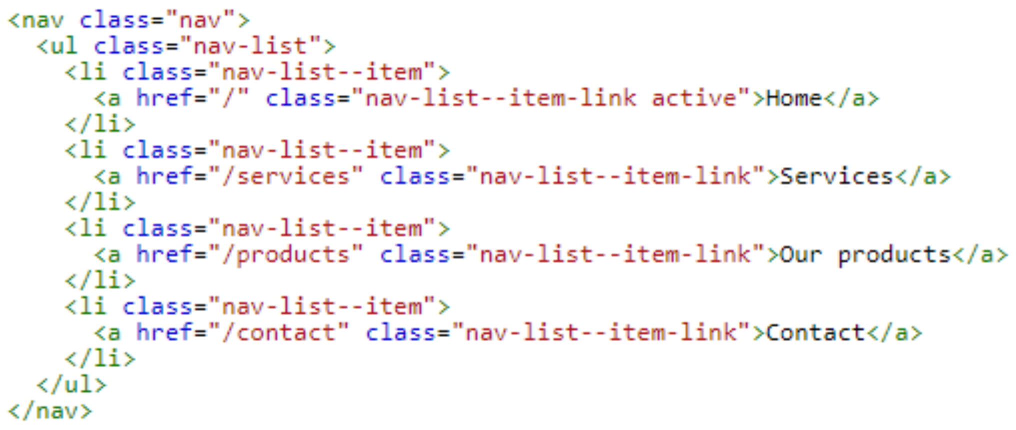
Palvelut
Meistä
Tietoa yritysautoilusta
Yhteystiedot

© 2020 A-liisari
Rekisteri- ja tietosuojaseloste

*FIGURE 16. Footer section*

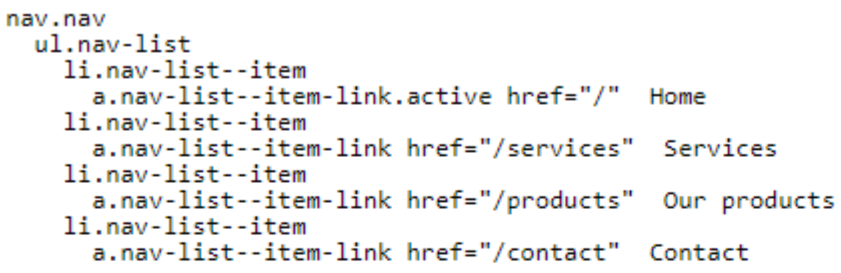
### 4.3.3 Wednesday 8.4

I continued with the Aliisari project. Today, I reimplemented the Home page using Slim instead of HTML. Slim is a templating language that has reduced the HTML markup and syntax. It is basically an HTML without an opening or a closing tag. The figures below show the difference between normal HTML (figure 17) and Slim HTML elements (figure 18), the HTML and Slim output the same result as shown in (figure 19).



```
<nav class="nav">
  <ul class="nav-list">
    <li class="nav-list--item">
      <a href="/" class="nav-list--item-link active">Home</a>
    </li>
    <li class="nav-list--item">
      <a href="/services" class="nav-list--item-link">Services</a>
    </li>
    <li class="nav-list--item">
      <a href="/products" class="nav-list--item-link">Our products</a>
    </li>
    <li class="nav-list--item">
      <a href="/contact" class="nav-list--item-link">Contact</a>
    </li>
  </ul>
</nav>
```

*FIGURE 17. HTML elements*



```
nav.nav
  ul.nav-list
    li.nav-list--item
      a.nav-list--item-link.active href="/"  Home
    li.nav-list--item
      a.nav-list--item-link href="/services"  Services
    li.nav-list--item
      a.nav-list--item-link href="/products"  Our products
    li.nav-list--item
      a.nav-list--item-link href="/contact"  Contact
```

*FIGURE 18. Slim elements*

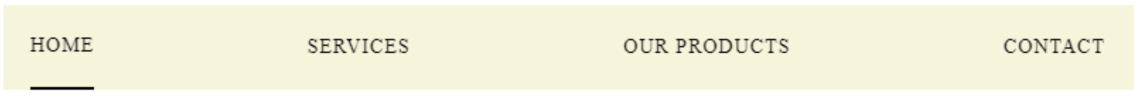


*FIGURE 19. Output of (figure 17) and (figure 18) code*

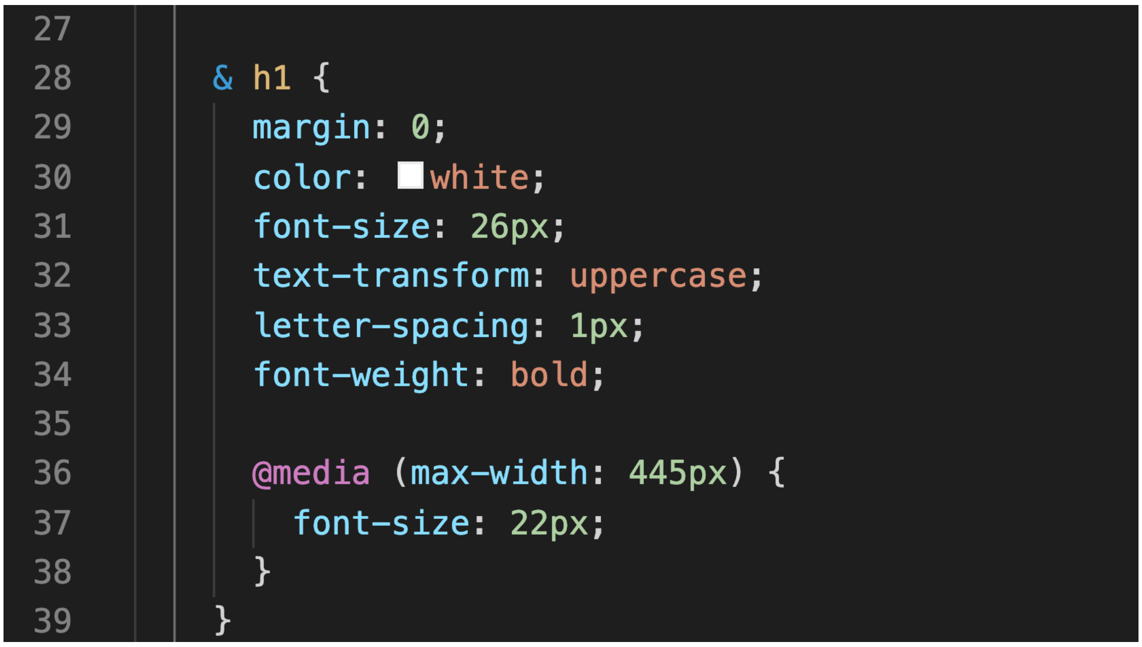
I positioned and cropped large images with CSS because this was according to the design. To fit the desired size, the images needed to be cropped to fit certain dimensions. The same cropping could be achieved with Image Manipulation Software, such as Gimp or Photoshop, and just using it with two lines of CSS code instead of ten lines of code.

I took part in a Ruby on Rails CMS meeting. The main discussion topic of the meeting was Alchemy CMS. Alchemy is an open source headless Content Management System engine which was written in Rails. (25.) CMS is normally used in the Administration side of a web application. The CMS helps to manage the content of the application, such as creating pre-styled custom web pages as well as administration and database management of the website. (26.) Although the topic was much more advanced than my level, I really understood the basic functionality about Alchemy CMS.

#### 4.3.4 Thursday 9.4

Today, I continued implementing the frontpage design of the Aliisari project. I have added media queries to make the landing page responsive. A Media query is a rule that is applied when certain conditions are fulfilled, for example, reducing the font size when the screen width is 445 pixels or less (figure 20).  Responsive is a term used to describe when a webpage fits and scales well in all screen sizes. I have added a navigation menu for all screen sizes using the media queries that I had defined. All the styles were implemented using Sass.

Today, I added a new Contact section to the private leasing project. The Contact section displays private leasing salespeople’s details. The details are their names, telephone numbers, department, and email addresses. The contact section is a Vue.js component that gets salespeople’s details from the database through a controller and the component renders these details inside the view.



```

  & h1 {
    margin: 0;
    color: ■white;
    font-size: 26px;
    text-transform: uppercase;
    letter-spacing: 1px;
    font-weight: bold;

    @media (max-width: 445px) {
      font-size: 22px;
    }
  }
```

*FIGURE 20. Media query example*

### 4.3.5 Week Analysis

This week it was all about Ruby on Rails. This was my first time I had developed a webpage with Ruby and Rails. It was a good experience and very interesting to be part of the Aliisari project. In the beginning I felt very uncomfortable with the Rails framework but as I introduced myself to the framework and got help from my workmates, I managed to understand the basics of the framework in a short period of time.

I mainly worked this week on the client side. The Client side is all about what the user sees on the application. This includes buttons, images and different styles that are visible on the webpage.

## 4.4 Week 4

### 4.4.1 Tuesday 14.4

I started working on an existing project RK. A senior developer was working on this project and due to the approaching deadline for the RK project, we needed to join forces and beat the deadline. Most of the features of the project are ready and have been already implemented. My work on the

RK project will be implementing new styles, modifying the existing styles and learning new approaches from the senior developer whom I am working with. In this project I will be also using Ruby on Rails, Sass and Slim HTML.

I started implementing a landing page for a single car in the RK project. This landing page will be a template for a single car detail and it is used by a car's list page to redirect the user if they want to get more information about a specific car. The landing page shows information about the car including the car's images and all other details that are related to the car, such as make, model and year. As usual, I used Slim for the HTML markup and Sass for styles of the page. The landing page will also be using available common templates, such as the navigation bar and the footer.

I made changes to the private leasing project's business logic to match with the newly provided instructions. I also did some debugging.

### 4.4.2 Wednesday 15.4

I continued with the RK project's single car detail page. I adjusted and improved overall styles, such as colors, fonts, and responsiveness of the page. Then I deployed the page to the production environment.

I also jumped back to the Aliisari project. I added a new info text "Our website is under construction" for the website visitors. I have also added an estimation timer that shows the number of days, hours, minutes and seconds that are remaining until the launch of the site and the webpage will be back to life. I did some debugging. The images were not visible on the webpage because the image links were broken. I fixed this error by rearranging the files and folders of the project.

### 4.4.3 Thursday 16.4

I continued with the RK project. My task today was to implement a sub navigation menu in the Hero section of the home page. The Hero section is an outstanding section that displays special images, texts, buttons, or general short information about the website. It is normally on the front page where the visitor lands first, immediately as they visit the website. I implemented the sub navigation section by creating the HTML markup with Slim and I styled those elements with Sass. The sub navigation is a horizontal list of tabs which contains an image and text that is describing what the section is about. A tab is a box area on the website which has a heading and a content. The tabs have a click event attached to them. A click event is an event that is triggered when an element has been clicked (27). When these tab elements are clicked, the content in the Hero section will change and display short information about the available services that the tab was indicating.

I also did debugging and a new component implementation with the RK project co-worker. We debugged the main landing page using the Chrome developer tool. We printed debugging messages to the console. A console is a tool which is built in the browser and which is used for logging messages. The debugging was fixing small features of the overall styles of the Homepage that were implemented with CSS and JavaScript.

I also improved the main navigation styles by adding a hover effect. It gives information to a user when the user's mouse hovers over the navigation elements.

#### 4.4.4 Friday 17.4

I changed and updated the private leasing business logic. There were not many changes to the previously implemented logic. I have added few extra steps.

Today, I also introduced myself to Alchemy CMS implementation in the RK project. The CMS had been already implemented and working in the project. I went through the codebase of the different files that had been used to implement Alchemy CMS. I made a PostgreSQL database installation for testing the RK project. As usual, we as well did debugging of CSS and HTML with the RK project's co-worker.

#### 4.4.5 Week Analysis

This week I took part in different projects' development work. I mainly worked on the front-end part of all the projects. There was only one project that included the server-side interaction. The front-end involves mostly what the user of the application sees. In front-end web development HTML, CSS and JavaScript are used. HTML are elements that make up the structure of the web page. It is the backbone of the structure of the web page. CSS gives a suitable style to the HTML elements that make up the web page. JavaScript makes the user to interact with the web application. Server side is the interaction of database and servers. The Server side gives the website dynamic contents that change over time because there is creating, retrieving, updating, and deleting of data. (28.)

## 4.5 Week 5

### 4.5.1 Monday 20.4

Today, our development team had a brief meeting about the situation of the RK project. We measured our achievement so far and we estimated the main features that are yet to be implemented. We discussed about our next steps and how we will proceed with the project.

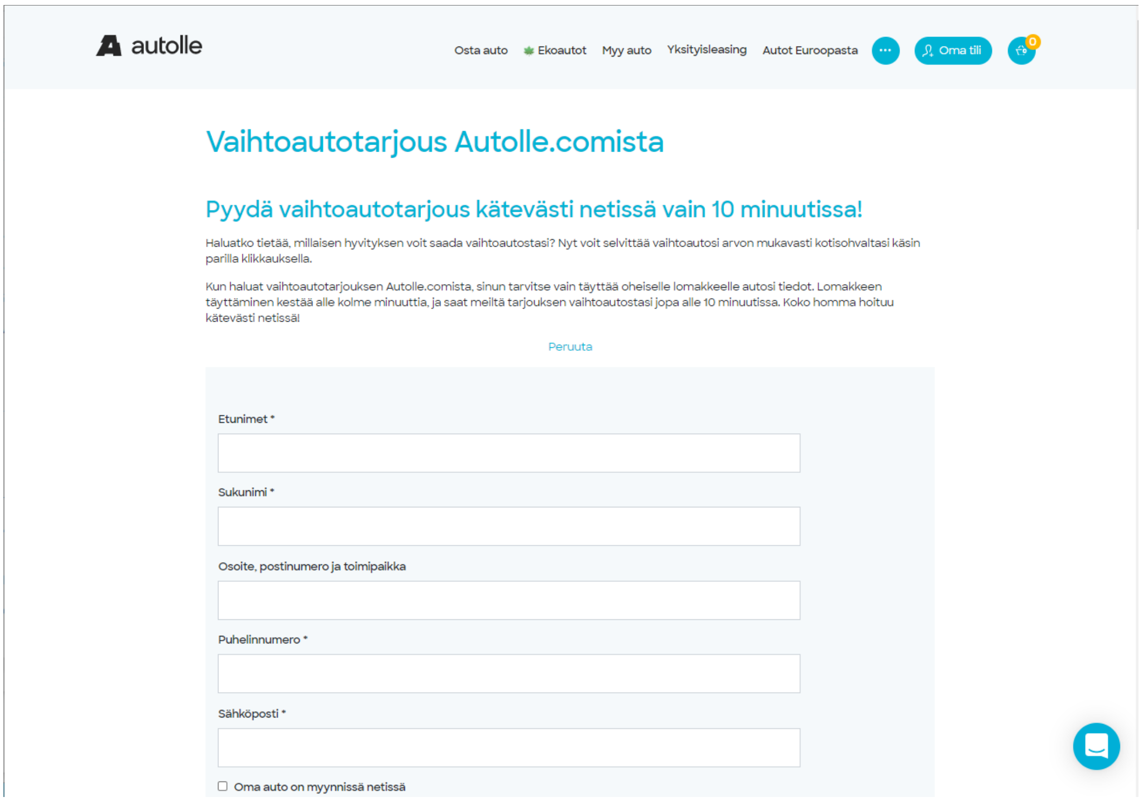
After the meeting, I continued with CMS side CSS styling and debugging the previous styles to fit the newly implemented styles.

At around noon I have switched from the RK project to the Autolle project. Autolle is the main project that I was working with at the start of my thesis. Private leasing landing page was part of the Autolle project.

The task was to create two landing pages for car exchange (figure 21) and car financing (figure 22). The landing pages would each have a description text about each service and would also have a form for the users to register for the service if they would like so.

To implement the task, I used HTML, CSS, Bootstrap and PHP. The steps were creating a route for each landing page and creating a controller that both landing pages will share, each with it is own method in the controller. Figure 23 shows controller methods that render car financing and car exchange views. HTML, CSS, and Bootstrap were used inside the Blade. Blade renders all the HTML elements and the styles that were applied with CSS alongside Bootstrap. Blade's output is what the user will see. PHP was used in the controller. The controller processes the requests from the user. For example, when the user submits a form, the controller validates the requests, checks if all the form fields are valid, and then it processes the form data and saves the data into the database using a Model. The controller also gives feedback if the process was successful or failed and notifies the user.

I had a problem when I was saving the form data into the database because there was a required field in the database that I forgot to include in the form fields. I managed to solve this problem by adding a new field to the form.



Vaihtoautotarjous Autolle.comista

Pyydä vaihtoautotarjous kätevästi netissä vain 10 minuutissa!

Haluatko tietää, millaisen hyvityksen voit saada vaihtoautostasi? Nyt voit selvittää vaihtoautosi arvon mukavasti kotisohvaltasi käsin parilla klikkauksella.

Kun haluat vaihtoautotarjouksen Autolle.comista, sinun tarvitse vain täyttää oheiselle lomakkeelle autosi tiedot. Lomakkeen täyttäminen kestää alle kolme minuuttia, ja saat meiltä tarjouksen vaihtoautostasi jopa alle 10 minuutissa. Koko homma hoituu kätevästi netissä!

Peruuta

Etunimet *

Sukunimi *

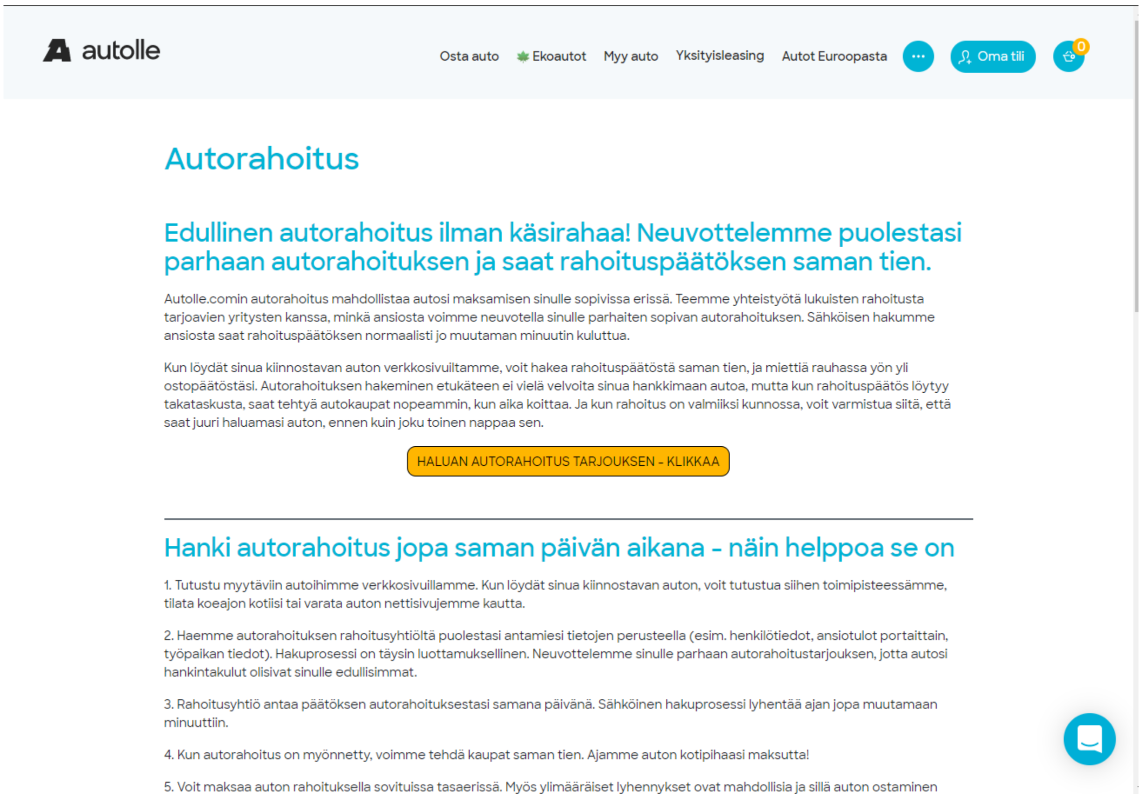
Osoite, postinumero ja toimipaikka

Puhelinnumero *

Sähköposti *

☐ Oma auto on myynnissä netissä

*FIGURE 21. Car exchange landing page*



Autorahoitus

Edullinen autorahoitus ilman käsirahaa! Neuvottelemme puolestasi parhaan autorahoituksen ja saat rahoituspäätöksen saman tien.

Autolle.comin autorahoitus mahdollistaa autosi maksamisen sinulle sopivissa erissä. Teemme yhteistyötä lukuisten rahoitusta tarjoavien yritysten kanssa, minkä ansiosta voimme neuvotella sinulle parhaiten sopivan autorahoituksen. Sähköisen hakumme ansiosta saat rahoituspäätöksen normaalisti jo muutaman minuutin kuluttua.

Kun löydät sinua kiinnostavan auton verkkosivuiltamme, voit hakea rahoituspäätöstä saman tien, ja miettiä rauhassa yön yli ostopäätöstäsi. Autorahoituksen hakeminen etukäteen ei vielä velvoita sinua hankkimaan autoa, mutta kun rahoituspäätös löytyy takataskusta, saat tehtyä autokaupat nopeammin, kun aika koittaa. Ja kun rahoitus on valmiiksi kunnossa, voit varmistua siitä, että saat juuri haluamasi auton, ennen kuin joku toinen nappaa sen.

HALUAN AUTORAHOITUS TARJOUKSEN - KLIKKAA

Hanki autorahoitus jopa saman päivän aikana - näin helppoa se on

1. Tutustu myytäviin autoihimme verkkosivuillamme. Kun löydät sinua kiinnostavan auton, voit tutustua siihen toimipisteessämme, tilata koeajon kotiisi tai varata auton nettisivujemme kautta.

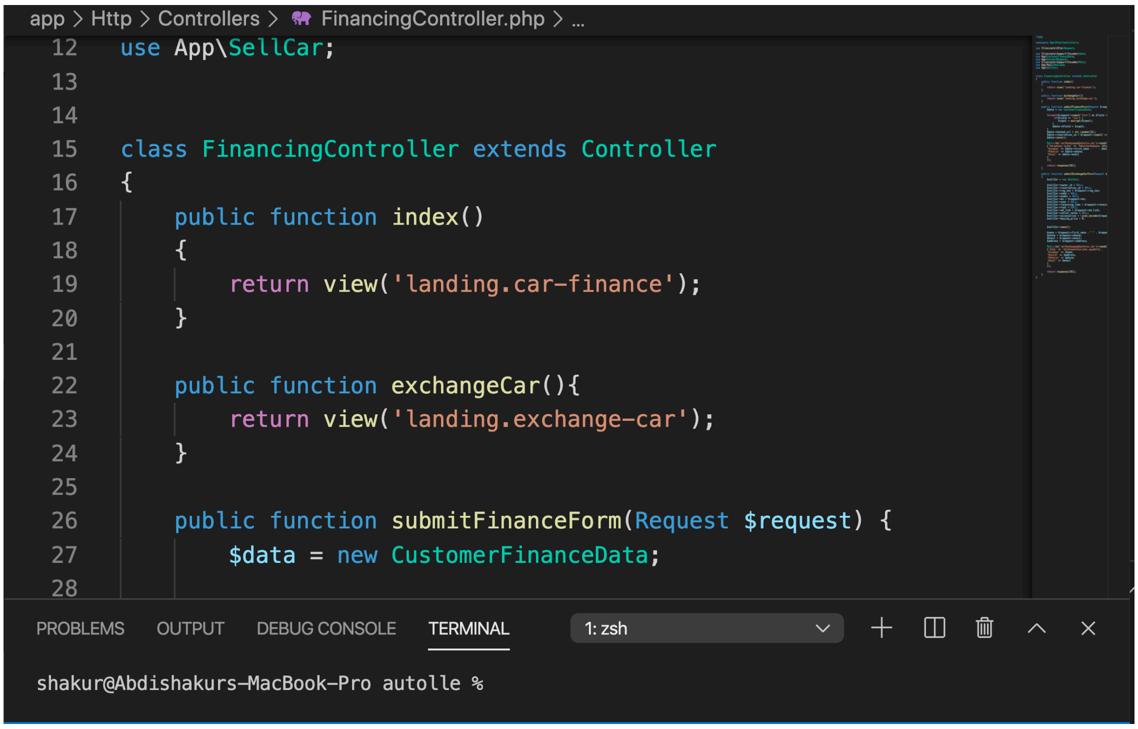
2. Haemme autorahoituksen rahoitusyhtiöltä puolestasi antamiesi tietojen perusteella (esim. henkilötiedot, ansiotulot portaittain, työpaikan tiedot). Hakuprosessi on täysin luottamuksellinen. Neuvottelemme sinulle parhaan autorahoitustarjouksen, jotta autosi hankintakulut olisivat sinulle edullisimmat.

3. Rahoitusyhtiö antaa päätöksen autorahoituksestasi samana päivänä. Sähköinen hakuprosessi lyhentää ajan jopa muutamaan minuuttiin.

4. Kun autorahoitus on myönnetty, voimme tehdä kaupat saman tien. Ajamme auton kotipihaasi maksutta!

5. Voit maksaa auton rahoituksella sovituissa tasaerissä. Myös ylimääräiset lyhennykset ovat mahdollisia ja sillä auton ostaminen

*FIGURE 22. Car financing landing page*



```php
use App\SellCar;


class FinancingController extends Controller
{
    public function index()
    {
        return view('landing.car-finance');
    }

    public function exchangeCar(){
        return view('landing.exchange-car');
    }

    public function submitFinanceForm(Request $request) {
        $data = new CustomerFinanceData;

```

```
shakur@Abdishakurs-MacBook-Pro autolle %
```

*FIGURE 23. Financing controller class that has public methods*

### 4.5.2 Tuesday 21.4

Today, I continued with the two landing pages of the Autolle project. I tested that both forms were sending data and saving that data to the database. I added a custom email notification message to the forms so that the salespeople will be notified that a new request of car exchange or car financing has been made. The email sending process had been already configured for the whole project.

The form data is saved into an existing table in the database which has a relationship to another table in the database. This caused a problem because the forms in the car exchange and car financing will use only one table and must be provided for an ID (foreign key) that points to the other table. During testing, I have assigned an ID which was not unique to the forms. Normally row IDs are created and autoincremented automatically in the database. I solved this problem by assigning the foreign ID of 0 to all car exchange and car financing data. This will not point to any row in the relationship table because the other table ID starts with index 1 so there will be no index 0.

The above solution created a problem because when fetching data from the database in the admin side, it generated an error that the item with the foreign key 0 was not found and it broke the whole process of fetching the data. I solved this problem by adding a condition before fetching the data that fetches data from the other table if the ID is greater than 0, otherwise do not fetch data from that table. Both the forms were implemented in the same way. Only the contents were different, but the functionalities were the same. The reason that a zero ID was given is to know that there is no registered user, or the user is not available. This was done because the earlier implementation forced users to register in order to apply for the available services.

### 4.5.3 Wednesday 22.4

Today, I jump back to the RK project and I added new styles to the Main articles page of RK project. I added a list of cards that shows heading and summary of the article. I also added more styles to the Single article page that displays the content of the article. All the other functionalities, such as getting articles from the database, had been already implemented. My task was to add styles to make then match the overall style of the webpage.

I added a new text content to car financing and car exchange landing pages.

### 4.5.4 Thursday 23.4

Today, I continued with the RK project. I recreated the top navigation bar in the CMS side. The previous navigation was using Bootstrap and custom Sass styles, which created a lot of overlapping styles and it was not the final version. To minimize this overlapping of styles, I redesigned the HTML elements and used Sass and JavaScript to style the element. I also added a custom animated hamburger menu that indicates different states, such as whether a menu is open or not. Figure 24 shows an open menu.

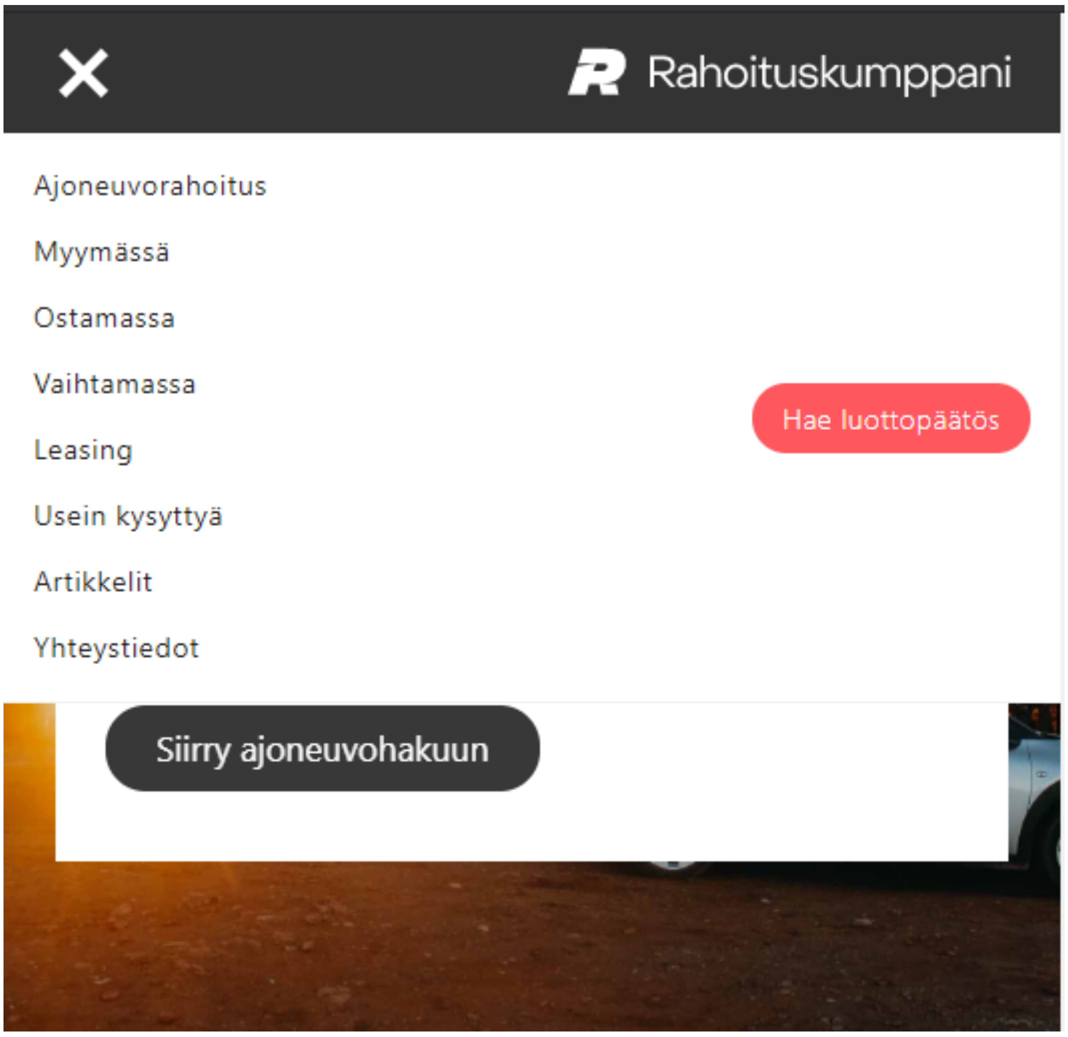


*FIGURE 24. Open menu*

I did more debugging with the RK project co-worker. We did together CSS and JavaScript debugging. We went through different files and started to review some sections of the code and did some refactoring. We also added new features, such as an on hover navigation. An on hover navigation is navigating the website by just hovering the mouse over elements without clicking on the menu. We also added media queries to make the webpage responsive.

### 4.5.5 Friday 24.4

I continued with the RK project. I changed some basic styles, such as the changing colors in different states and fixing images sizes.

Today, we solved together with my RK project co-worker a bug that was in the project. This bug was related to Turbolinks. Turbolinks is a JavaScript code that has been integrated with Ruby on Rails. Turbolinks makes the application behave like SPA (Single Page Application). It replaces the entire document body with a new one when the user switches between different pages in the application (29). The head of the document is not replaced. SPA is an application that works on a browser which does not need to reload a webpage when browsing different pages.

The problem was that during browsing the website, going back and forth in the application, some of the pages that have inline JavaScript code throw an error that the elements have been already declared and freezes the event listeners, such as clicks. This is because Turbolinks caches the current page and adds to the application history and rerenders again when the user comes back to the page through the back button, Turbolinks tries to declare a variable that has been already declared thus throwing the error. The error will not occur if the application browsing is done via navigation links because the current content is replaced with a new content (30).

We solved this problem by replacing all the variables that were declared with a `let` keyword to a `var`. One difference between let and var is that let is scoped inside the closing block {} which means that there will be a reference error if accessed outside the block. Variables declared with var are scoped to the immediate function body which means that they can be accessed outside the closed brackets (31).

### 4.5.6 Week Analysis

This week I did a lot of pair work with another developer, especially for debugging errors and refactoring of previous code. We solved together a lot of small annoying CSS and JavaScript errors.

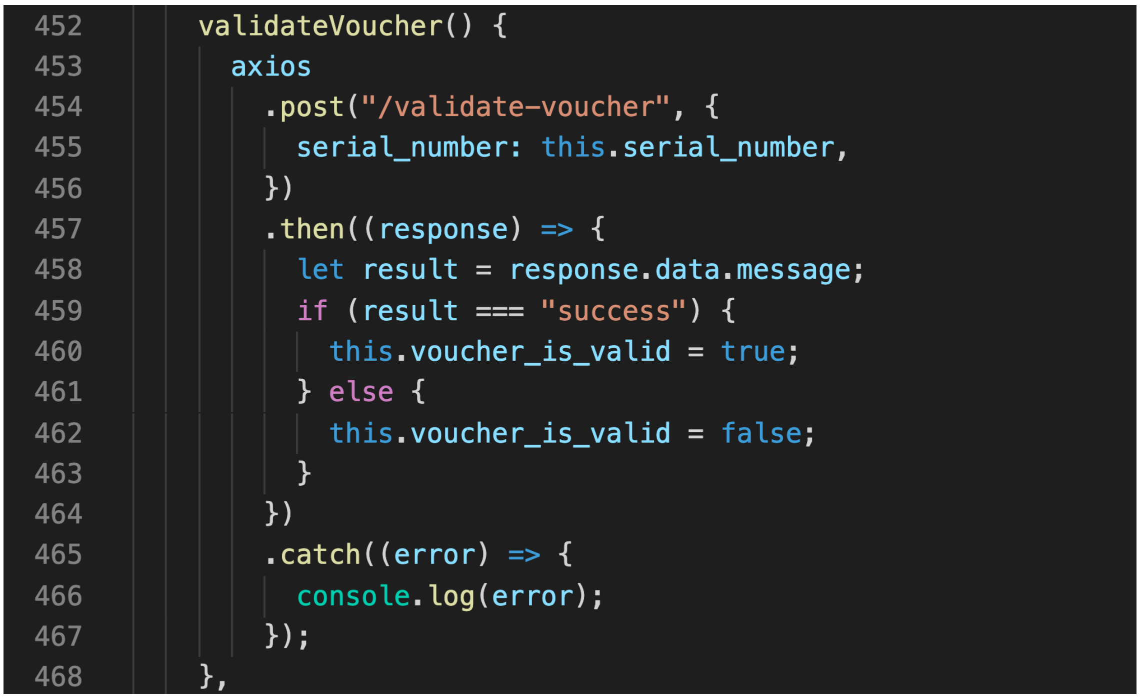
I also implemented new landing pages that collect user inputs, saves them into the MySQL database and notifies the admin about the newly saved information by email. I also did more styling, added new styles to the existing elements and modified the old styles.

## 4.6 Week 6

### 4.6.1 Monday 27.4

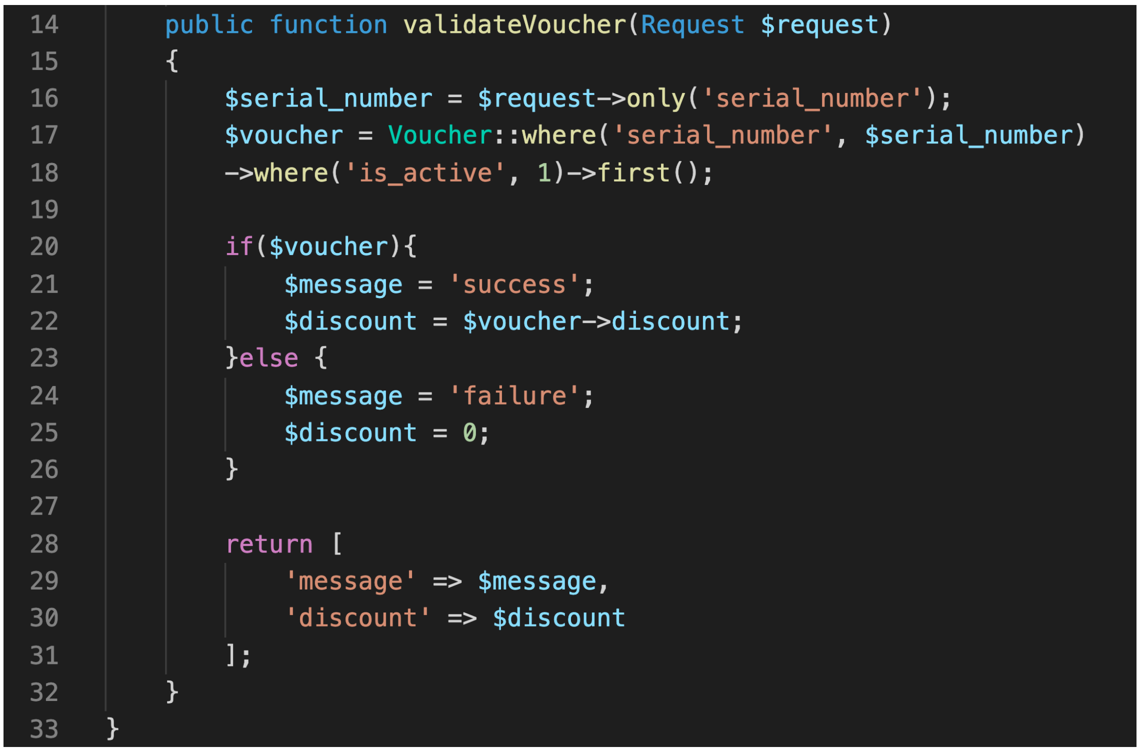
The task today was to add a discount voucher functionality to private leasing payments. The steps of the task are creating a new Voucher model, making database migration and making a controller to process and validate user requests. In the Laravel framework a model is a PHP class that has a functionality to interact with a database table. Each database table has a corresponding model. Models allow to query, insert and update data into the database (32). Database migration is a way of creating a database table or adding a column in the database from the application without creating the tables manually from a database manager.

The validation process starts with an AJAX POST request (figure 25). AJAX is sending an HTTP request to the server. The POST request is an HTTP method that is used to send data to the server. The POST request is used to send a user input to the server. In this case the voucher serial number is sent to the server for validation. When the request is submitted to the server, the server returns a response. If the response is successful the voucher is valid, otherwise it is not valid. Figure 26 shows a method that validates the voucher by checking if the voucher serial number is valid from the database.



```
  validateVoucher() {
    axios
      .post("/validate-voucher", {
        serial_number: this.serial_number,
      })
      .then((response) => {
        let result = response.data.message;
        if (result === "success") {
          this.voucher_is_valid = true;
        } else {
          this.voucher_is_valid = false;
        }
      })
      .catch((error) => {
        console.log(error);
      });
  },
```

*FIGURE 25. POST request with axios from client side*



```
    public function validateVoucher(Request $request)
    {
        $serial_number = $request->only('serial_number');
        $voucher = Voucher::where('serial_number', $serial_number)
        ->where('is_active', 1)->first();

        if($voucher){
            $message = 'success';
            $discount = $voucher->discount;
        }else {
            $message = 'failure';
            $discount = 0;
        }

        return [
            'message' => $message,
            'discount' => $discount
        ];
    }
}
```

*FIGURE 26. Validate voucher method in server side*

The user of private leasing service will get a discount equivalent to the amount in the voucher. If the voucher is valid, there is an indication whether the voucher is valid or not. Figure 27 shows an example of a valid voucher. The user will immediately see a new total price after the discount.

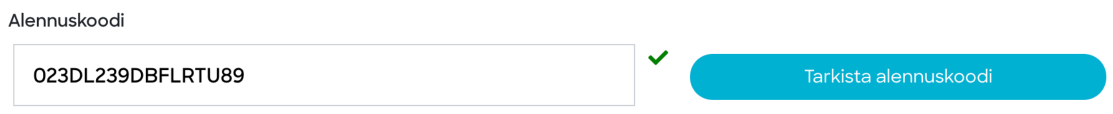


*FIGURE 27. A valid voucher numbers*

### 4.6.2 Tuesday 28.4

Today, I continued with discount voucher implementation. I added Voucher recourse management to the admin side. Here, the admin can create a new voucher or update an existing voucher, for example activate, deactivate, or change the amount of the discount voucher.

I also added voucher functionality to the buy-car page. I added a user input to the buy-car landing page. All other functionality of the voucher is shared across the application. During the testing, I discovered that the voucher cannot be activated or deactivated from the admin side because there was a typo in the code. It took a while to find which part of the code was generating the error.

In the afternoon I switched to the Aliisari project. I did not add any new elements. I refactored and changed the existing styles. I resized the images, changed colors, background, added sticky navigation bar, and added JavaScript events. I did a lot of small changes.

### 4.6.3 Wednesday 29.4

Today, I added “Services page”, “About us”, “More about our services” and “Contact pages” to the Aliisari project. The implementation of the pages followed a design that was already available. I used Slim HMTL, Sass and JavaScript to create all the above pages.

### 4.6.4 Thursday 30.4

I continued the Contact page of the Aliisari project. I added contact form, company's employees contact details and company details. I submitted the finished pages to my project co-worker to add form functionality, such as saving form data and creating database table.

During noon, I have switched to the RK project to fix a bug. The bug was that images in the slide show component were of different sizes and the thumb nails stretched to fit the whole space available. I solved the problem by adding a quick fix solution which is limiting the height of the images, so the images cannot be bigger than a predefined height. If the image is too big and if the image is small, it cannot be smaller than the predefined height, so the image should be between these two predefined heights.

At last I went back to the Aliisari project to fix broken styles of the contacts form. The styles broke because of the use of Ruby on Rails simple form component. I also added a confirmation page that notifies the users that their contact details have been submitted successfully. Finally, the first version of the Aliisari project was released.

### 4.6.5 Week Analysis

I worked with three different projects. I added voucher discount functionality to the Autolle project, refactored and added new landing pages to the Aliisari project. At the end of the week the Aliisari project was published ten hours before the deadline. I also fixed the slide shows bug in the RK project.

We did a lot of teamwork. Our team was very productive this week. We managed to fix bugs in different projects, we discussed different ways to solve problems and we shared tips to overcome certain issues in the application development process.

## 4.7 Week 7

### 4.7.1 Monday 4.5

Today, I re-implemented the Autolle private leasing landing page. The reimplementation was necessary because the current search filter that I had implemented in my first week of my thesis work is not quite as effective as the main search of the main page and it is also visually different from the main search. The search filter had only one input field where the user needs to put the search keyword. This made mandatory for the users to type because there were no other options for selecting a search keyword such as dropdown menus, checkboxes, sliders or any other input types.

The task was to use the main search component and integrate it into the private leasing landing page to replace the current search filter without modifying other functionalities. The main search consists of three Vue.js components. The first one has only one input field that accepts values, the second one has buttons for the different categories for example price and the third one is a modal that pops up when the buttons are clicked. The modal has other inputs, such as a slider and checkboxes for further selections.

The main search component works in a way that when the user types a keyword or selects the search keyword from the options, the component sends an AJAX request to the server with the search keyword as parameters of the request. URL parameters or query are the portion of the URL that follow the question mark. The parameters consists of key value pairs. Figure 28 shows the different sections of the query string.

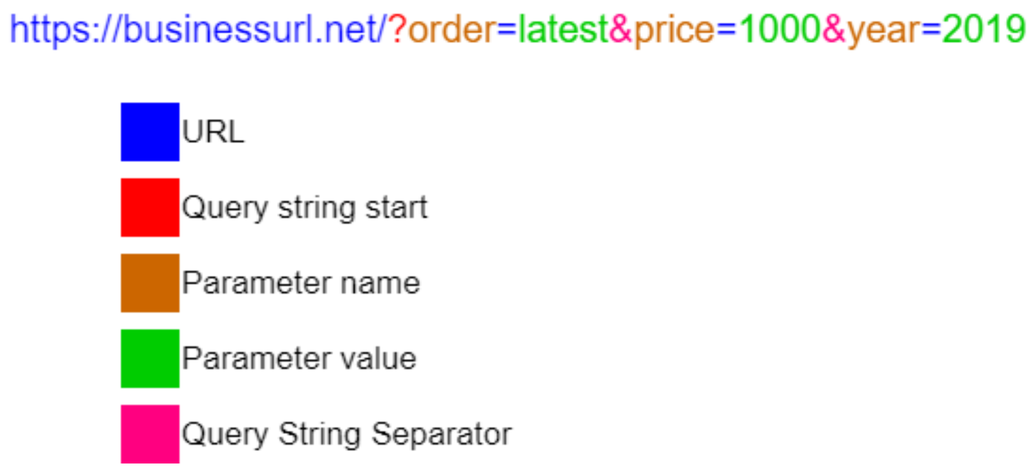


*FIGURE 28. Different sections of query string*

After sending the request, a response will be received which may have one or more values that correspond the search keyword. These values are what is presented to the user. If the response is empty, the user is presented with a message “no search result”. This is the overview of how the main search component works.

I replaced the previous search filter with the main search component. I tested that it and the landing page were working as before.

### 4.7.2 Tuesday 5.5

Today, I went through the form validation and submission process in the Aliisari project’s Contacts page. There was a bug in the form submission process. If the user fills some fields and leaves other fields unfilled and submits the form, the user gets a notification that all the required fields must be filled but it clears all the field values forcing the user to type again all the fields. The form was not keeping the details of already filled fields.

The problem was that the data validation process was rerendering the form if there was an error in the submitted value. This was resetting all the fields thus forcing the user to fill in the fields again. We came up with two solutions, One is a simple solution where the submitted values are stored in a variable and if there is an error, the stored values are sent back to the user along with the message “please, fill all the required fields”. The sent back values are used to prefill the existing fields that have been already filled. The second solution is to save the fields values to the session storage. If there is an error and the user is redirected to the form page, the values stored in the session are read and prefilled in the fields that have been already filled. Both solutions solved the problem, but we used the latter one because we thought it is better than the first solution.

I also did some refactoring of styles in the Aliisari project especially in the mobile view, I resized the font sizes and spaces between the elements. I also started the new landing page in RK project.

### 4.7.3 Wednesday 6.5

I continued with the RK project's landing page. The landing page consisted of headings, rows and columns that have texts, images and links that redirect to other pages on the website. I used Sass and Slim to implement the landing page.

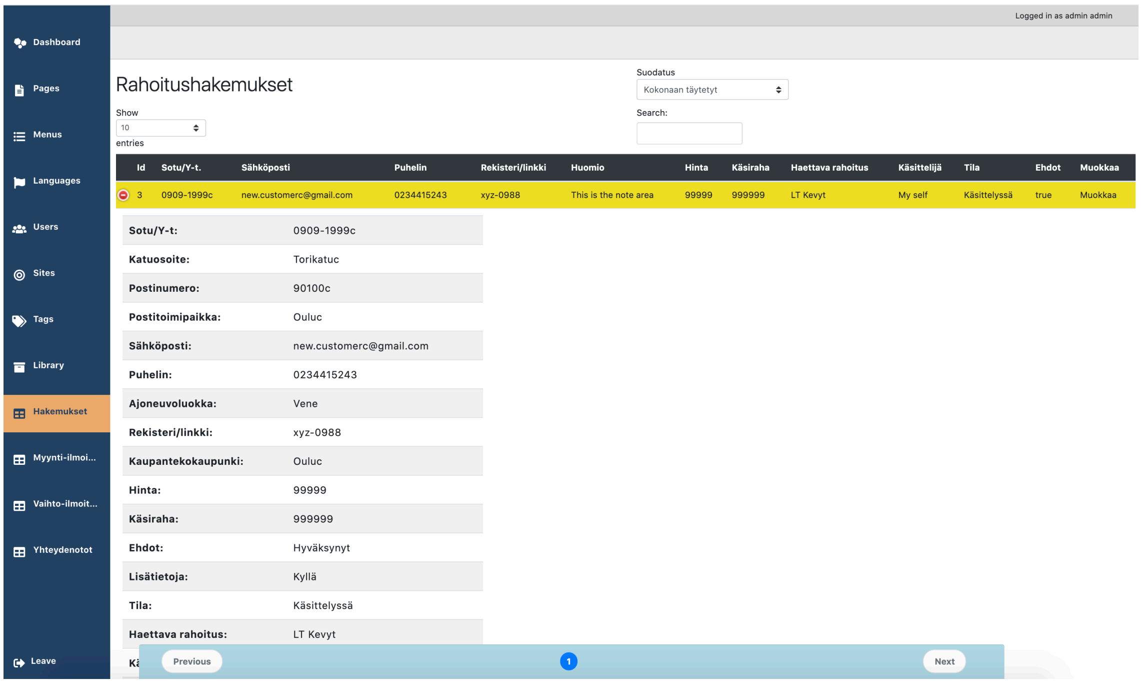
Today, I started learning how DataTables have been integrated into the RK project's Admin side which is using Alchemy CMS. DataTables is a plug-in for the jQuery JavaScript library that is used to display a huge amount of data information into a table of rows and columns which have searching, sorting, filtering, and pagination functionalities (33). The main task was to find a way to modify the default styles of Alchemy CMS and to make the DataTables responsive to fit all available screens.

Today, I also got a chance to learn more about Ruby and Rails from a Senior developer whom I was working with. We worked together mainly on RK project. I learnt about how the MVC design pattern works in the RK project, we went through the codebase of the project including the models, database schemas, views, and controller. I also learnt about best practices and conventions of coding.

### 4.7.4 Thursday 7.5

I continued testing the DataTables jQuery JavaScript library. To understand more about this library and how to modify it, I created a small project to test and try all DataTables functionalities. There are two main features that I was assigned to find. The first one is to able to have sub rows that expand when clicked on the open icon (figure 29). The idea is to put into these expandable rows extra information that is not part of the main rows. The other one is to assign different background colors to the rows according to a value that is coming from the database. For example, if a status value is "pending", the background color of that row should be yellow and vice versa.

As all the functionalities worked well in the small project, I switched to the main RK project to implement the tested features. The expanding feature worked well but the background colors feature did not work well. Some rows the background color was applied to, and others were not. I tried to figure out why this is happening, but I could not find a reason behind this strange action. I thought that maybe it is Alchemy styles that are overriding the styles that I had implemented to change the background colors; I could not find a solution.



Rahoitushakemukset

| Id | Sotu/Y-t. | Sähköposti | Puhelin | Rekisteri/linkki | Huomio | Hinta | Käsiraha | Haettava rahoitus | Käsittelijä | Tila | Ehdot | Muokkaa |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0909-1999c | new.customerc@gmail.com | 0234415243 | xyz-0988 | This is the note area | 99999 | 999999 | LT Kevyt | My self | Käsittelyssä | true | Muokkaa |

| | |
|---|---|
| **Sotu/Y-t:** | 0909-1999c |
| **Katuosoite:** | Torikatuc |
| **Postinumero:** | 90100c |
| **Postitoimipaikka:** | Ouluc |
| **Sähköposti:** | new.customerc@gmail.com |
| **Puhelin:** | 0234415243 |
| **Ajoneuvoluokka:** | Vene |
| **Rekisteri/linkki:** | xyz-0988 |
| **Kaupantekokaupunki:** | Ouluc |
| **Hinta:** | 99999 |
| **Käsiraha:** | 999999 |
| **Ehdot:** | Hyväksynyt |
| **Lisätietoja:** | Kyllä |
| **Tila:** | Käsittelyssä |
| **Haettava rahoitus:** | LT Kevyt |
| **Kä** | |

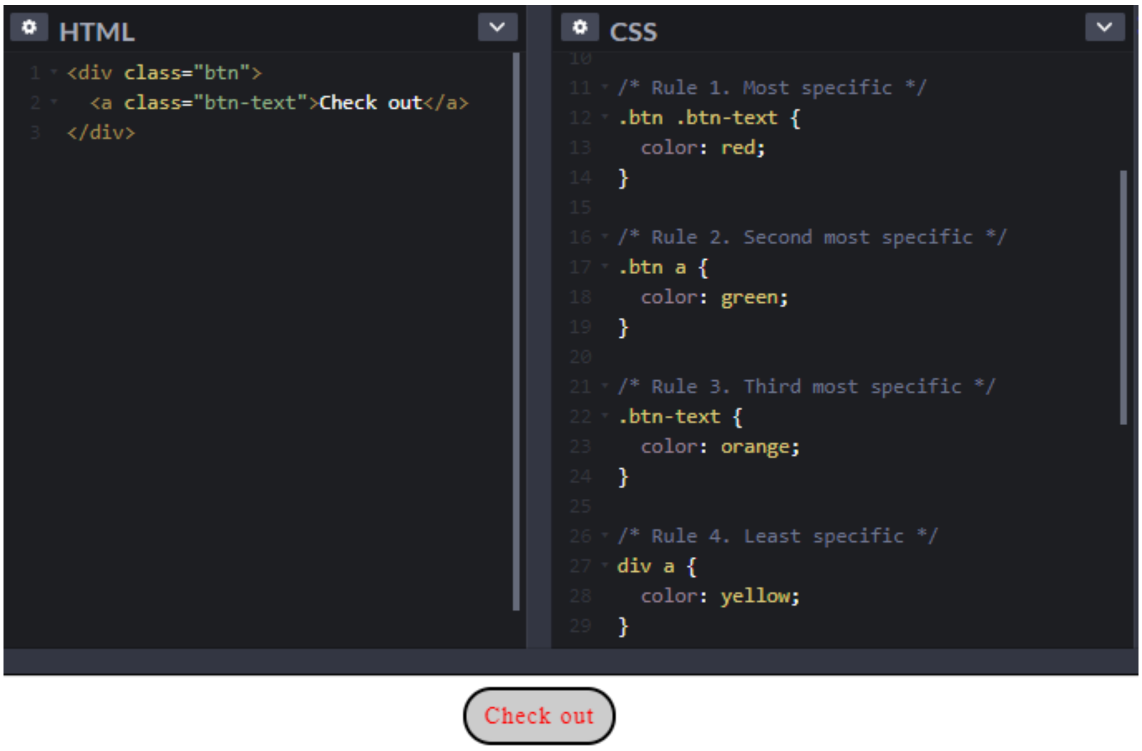
*FIGURE 29. Expanded sub row*

### 4.7.5 Friday 8.5

I continued to find a solution for why the background color styles were not applied at all or were applied partially. I started making serious debugging of all the styles that make up the page. The page had several different styles Bootstrap, DataTables default styles, Alchemy default styles and the local styles. I used the Chrome developer tool for debugging but any other browser could be used. The developer tool helps diagnose problems. I used the styles section of the DevTools to see how different styles have been applied.

After going through most of the elements styles, I discovered the problem. The problem was CSS Specificity (34). CSS Specificity is a weight that is applied to a given CSS declaration. The weight determines which style is applied to an element. The DataTables styles on the table were more specific than the styles that I had added, thus having no effect. I solved the problem by adding a transparent color to DataTables styles to make it easy for other background colors to be visible. For example, an element with an ID has more specificity than an element which has a class. In

Figure 30 CSS rule 1 is more specific than rule 2, 3 and 4. That is why the button's text color is red. No matter of the order, text will be red because rule 1 is more specific.



```html
<div class="btn">
  <a class="btn-text">Check out</a>
</div>
```

```css
/* Rule 1. Most specific */
.btn .btn-text {
  color: red;
}

/* Rule 2. Second most specific */
.btn a {
  color: green;
}

/* Rule 3. Third most specific */
.btn-text {
  color: orange;
}

/* Rule 4. Least specific */
div a {
  color: yellow;
}
```

*FIGURE 30. Specificity example*

After solving that problem, I added a color picker that the user can pick any color and apply to the background of the table's row. This makes it easy for the user to identify the rows by color.

### 4.7.6 Week Analysis

This week was mainly about fixing bugs in different projects to make the applications work better and to increase the usability of the applications. This week I learnt more about the Ruby on Rails framework and I did more interaction with the framework.

This week I learnt advanced features of the Ruby on Rails framework and had a chance to work closely with a senior develop. This gave me more experience towards the RoR framework and improved my RoR skills.

I replaced the private leasing search component to use the main page's search component. I created the new landing page for the RK project and tested the DataTables jQuery library integration and modification in the RK project.

## 4.8 Week 8

### 4.8.1 Monday 11.5

I continued working with the RK project's Admin side. The Admin side uses Alchemy CMS. Today's work was a continuation of last week's work. I mostly worked with the Index and Edit page of the Admin section. The Admin section is a section that shows the list of details that were submitted by the users. All the functionalities in this section had been already implemented. My task was to make these pages easy to use and get information easily by the users of Admin section. I added more fields with extra information to the main Index page and I have also added more form input fields to the Edit details page. I modified overall styles of the Index page such as adding different styles for table rows, font sizes and adding spaces between elements. Also, I modified the Edit page and added buttons and more styles and made it responsive to fit all screen sizes.

Today I mostly worked with Sass, ERB and Slim, JavaScript and Ruby.

### 4.8.2 Tuesday 12.5

Continuing with the RK project, today I did mostly debugging and modifying overall styles of the Admin side. I added more input fields to the Edit view page and removed extra fields from main Index page. Today I also started investigating Admin side localization files and how they are implemented since the localization files are generated automatically with the Alchemy CMS Gem. Today I started debugging the main landing page of the RK project. The debugging was mainly about HTML tags modification, replacement, and rearrangement, also there were CSS style changes.

### 4.8.3 Wednesday 13.5

I mostly did debugging of the RK project. The debugging was about the overall UI improvement of the RK project. I worked with HTML Slim and CSS styles.

### 4.8.4 Thursday 14.5

I started to work with the Autolle project. Today I mostly did Trello cards. Trello is a Web service that is used to organize projects (35). The tasks were mainly about fixing bugs and adding new features to the existing functionalities. The main task was to update the exchange car form. I included new fields that take user input and save it into the database. Today I mainly worked with Vuejs, HTML, Sass, PHP and the MySQL database.

### 4.8.5 Friday 15.5

I continued with the Autolle project. Today I worked with the reclamation form. The task was to add extra fields for the user to choose between different options. And according to the selected option, the user will give more details. The task consisted of working with the UI and the MySQL database.

I also investigated car inspection processes codebase that consisted of several different PHP classes, Vuejs components and several database tables which are linked to each other. The findings were the start of solving the task which is needed to modify existing fields and steps and add new fields and features to the inspection process.

### 4.8.6 Week Analysis

This week was about fixing bugs and improving the overall application user interfaces for both RK and Autolle projects. New features were also added to the Autolle project.

This week I worked with the wide range of tools such as Alchemy CMS, the Laravel Nova application panel, Ruby on Rails, HTML Slim, HTML, Sass, CSS, Bootstrap, jQuery, JavaScript, Vue.js, PHP and MySQL.

# 5 DISCUSSIONS AND CONCLUSION

The objective of the thesis was to learn and develop the skills for two webframeworks, PHP Laravel and Ruby on Rails at Comille Oy. The diary entries hold the development work for different customer projects for eight weeks. During these eight weeks, I have progressed in many ways as a web developer. At the start of my thesis some of the tools and frameworks used for the development at Comille Oy were totally new, some of them I had a basic understanding and others I was comfortable working with them.

My first learning and development process started with the PHP Laravel webframework developing project. I have not participated in any new Laravel projects during the thesis. I have mainly done maintenance work, adding new features, fixing bugs, and modifying existing codebases. I have learnt a lot of things from this project. This project has improved my skills in understanding and reading large project codebases. The most challenging thing was adding new features to an existing project that I was new to. This was challenging because first you have to find where to add the feature to from the codebases and that needs going through multiple files that link to each other. Sometimes I spent more time going through the project files and reading the code line by line to understand the output. This process of going through the project files gave me a strong understanding of how all the pieces that consist of the big project fit together. This made it easier to add new features, modify and replace the existing features.

The other framework that I have worked with is Ruby on Rails. This was the totally new framework. I did not have the previous experience before I started working with this framework. I have participated in two projects that were using the Ruby on Rails framework. I have learnt a lot of new skills from these two projects although my main area of work was client-side for both projects. Some of the things that I have learnt was the overall Rails project structure and architecture. I have also learnt working with new tools, working with databases, performing basic database operations, such as creating, updating, deleting and presenting the retrieved data to the view. After participating in these projects, I feel comfortable to take part in Ruby on Rails projects in the future. This was a good opportunity to learn new skills while applying them at the same time and developing a real application for a customer. These two projects have also improved my overall web development skills and I had a chance to experience the simplicity of the Ruby on Rails framework.

Apart from improving my programming and technical skills, I have also learnt the importance of teamwork in project development. Since I was a member of a development team, I had the chance to participate in team discussions involving different development related issues. Mostly we discussed possible ways to solve specific problems or ways to improve existing issues that are related to the current project, previous projects, or upcoming projects. In the discussion events, we asked ourselves many questions that are related to different scenarios that apply to the discussion topic and every member had a chance to give their argument. Team discussions simplified complex topics and made me understand them better.

Other skills that I have improved are problem-solving skills. All the projects that I have participated in had a specific issue that needed to be resolved, some of the issues were to find a better way to improve a feature, to find a bug or simply to find out why some features may not work as intended. To solve these issues, the source of the problem should be identified by researching or analyzing to find a proper and effective solution for the problem. Then the solution might be shared with the team for the further analysis or to be applied to solve the problem. I have used online searching to solve problems. There are a lot of recourses on the Internet which might help to simplify complex concepts that could be difficult to solve on your own.

The overall work went well, and it has been a good learning opportunity for me. The most valuable experience I had was to develop real-world applications for customers. This improved my web development skills a lot and gave me a chance to realize that as a developer, I need to improve my skills continuously since the technologies used in one project might not be used in another project, although the overall concepts remain the same for the most of web applications.

# REFERENCES

1. The PHP Framework for Web Artisans. 2020. Laravel. Available: https://laravel.com/. Accessed: 1.6.2020.
2. Imagine what you could build if you learned Ruby on Rails. 2020. Rails. Available: https://rubyonrails.org/. Accessed: 1.6.2020.
3. Heinemeier Hansson, David. The Rails Doctrine. Rails. January 2016. Available: https://rubyonrails.org/doctrine/#convention-over-configuration. Accessed: 1.6.2020.
4. Don’t repeat yourself. 2020. Wikipedia. Available: https://en.wikipedia.org/wiki/Don%27t_repeat_yourself. Accessed: 1.6.2020.
5. MVC: Model, View, Controller. 2020. codecademy. Available: https://www.codecademy.com/articles/mvc. Accessed: 1.6.2020.
6. What is REST? 2020. codecademy. Available: https://www.codecademy.com/articles/what-is-rest. Accessed: 1.6.2020.
7. Application programming interface. 2020. Wikipedia. Available: https://en.wikipedia.org/wiki/Application_programming_interface. Accessed: 1.6.2020.
8. The Progressive JavaScritp Framework. 2020. Vue.js. Available: https://vuejs.org/. Accessed: 2.6.2020.
9. MySQL Documentation. 2020. MySQL. Available: https://dev.mysql.com/doc/. Accessed: 2.6.2020.
10. What is jQuery. 2020. jQuery. Available: https://jquery.com/. Accessed: 2.6.2020.
11. About. What is PostgreSQL? 2020. PostgreSQL. Available: https://www.postgresql.org/about/. Accessed: 2.6.2020.
12. NETTIX PRO. 2020. NETTIX. Available: https://nettix.fi/yrityksille/kaikki-palvelut/nettix-pro/. Accessed: 1.6.2020.
13. Enterprise resource planning. 2020. Wikipedia. Available: https://en.wikipedia.org/wiki/Enterprise_resource_planning. Accessed: 2.6.2020.
14. Get storage/vehicle. 2020. Nettix Pro REST API. Available: https://api.nettixpro.fi/docs/get/storage/vehicle.html. Accessed: 2.6.2020.
15. Blade Templates. 2020. Laravel. Available: https://laravel.com/docs/7.x/blade. Accessed: 12.05.2020.
16. Components Basics. 2020. Vue.js. Available: https://vuejs.org/v2/guide/components.html. Accessed: 12.5.2020.

17. Props. 2020. Vue.js. Available: https://vuejs.org/v2/guide/components-props.html. Accessed: 12.5.2020.
18. Master Your Universe. 2020. Laravel Nova. Available: https://nova.laravel.com/. Accessed: 2.6.2020.
19. About packages and modules. 2020. npm. Available: https://docs.npmjs.com/about-packages-and-modules. Accessed: 12.5.2020.
20. Working with JSON. 2020. MDN web docs. Mozilla. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON. Accessed: 12.5.2020.
21. Composer (software). 2020. Wikipedia. Available: https://en.wikipedia.org/wiki/Composer_(software). Accessed: 12.5.2020.
22. Task Scheduling. 2020. Laravel. Available: https://laravel.com/docs/7.x/scheduling. Accessed: 12.5.2020.
23. Paytrail – Integration guide. v5.7. 2020. Paytrail. Available: https://docs.paytrail.com/en/index-all.html. Accessed: 16.5.2020.
24. PHP Data Types. 2020. w3schools.com. Available: https://www.w3schools.com/php/php_datatypes.asp Accessed: 16.5.2020
25. AlchemyCMS / alchemy_cms. 2020. GitHub. Available: https://github.com/AlchemyCMS/alchemy_cms. Accessed: 16.5.2020.
26. Content management system. 2020. Wikipedia. Available: https://en.wikipedia.org/wiki/Content_management_system. Accessed: 16.5.2020.
27. Element: click event. 2019. MDN web docs. Mozilla. May 9, 2019. Available: https://developer.mozilla.org/en-US/docs/Web/API/Element/click_event. Accessed: 17.5.2020.
28. What's the Difference Between the Front-End and Back-End? 2015. Pluralsight. January 28, 2015. Updated 10.9.2019. Available: https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end. Accessed: 17.5.2020.
29. Single-page application. 2020. Wikipedia. Available: https://en.wikipedia.org/wiki/Single-page_application. Accessed: 17.5.2020.
30. Seifer, Jason 2012. Rails 4: A Look at Turbolinks. treehouse. October 4, 2012. Available: https://blog.teamtreehouse.com/rails-4-a-look-at-turbolinks. Accessed: 17.5.2020.
31. What's the difference between using "let" and "var"? 2020. stackoverflow. Available: https://stackoverflow.com/questions/762011/whats-the-difference-between-using-let-and-var. Accessed: 17.5.2020.

32. Eloquent: Getting Started. 2020. Laravel. Available: https://laravel.com/docs/7.x/eloquent. Accessed: 17.5.2020.
33. Using DataTables. Manual. 2020. DataTables. Available: https://datatables.net/manual/index. Accessed: 17.5.2020.
34. Specificity. 2020. MDN web docs. Mozilla. Apr 20, 2020. Available: https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity. Accessed: 17.5.2020.
35. Trello lets you work more collaboratively and get more done. 2020. Trello. Available: https://trello.com/en-US. Accessed: 2.6.2020.