Software Documentation for Web application

**“TruckGistic”**

Produced By:

Piotr Myszkowski

6500 25th Ave NE

Seattle, WA 98115

Email: [myszon01@gmail.com](mailto:myszon01@gmail.com)

Contents

[1. Introduction 3](#_Toc516396408)

[2. Subject Area Description 4](#_Toc516396409)

[3. Description of Technologies Used 5](#_Toc516396410)

[*Back****-****end* 5](#_Toc516396411)

[*Front****-****end* 7](#_Toc516396412)

[*Host* 8](#_Toc516396413)

[4. Database Design 9](#_Toc516396414)

[5. Application Structure and UI Design 12](#_Toc516396415)

[6. Algorithms Design 13](#_Toc516396416)

[7. Testing 14](#_Toc516396417)

[8. Conclusion 14](#_Toc516396418)

[9. References 15](#_Toc516396419)

# Introduction

The purpose of this document is to describe the implementation of the TruckGistic web application. The TruckGistic web application software is designed to help freight management businesses focused in USA territory to track theirs loads, drivers and customers.

This document describes the implementation details of the TruckGistic Web Application. The software will consist of multiple functions. That include create/delete/update loads, customers, drivers, facilities. We will also outline and describe specific components of the project.

When designing this software it was critical to know and deeply understand logistics problems that freight companies have to face every day. This application was consultant with group of people that work in the industry to help understand the best design and solution for the problems.

This application is design for multiple companies/users. Because of the design and lightweight framework that were used it can handle thousands of users making requests with no downtime. Thanks to newest technologies and design techniques used in this project it will me much easier to maintain, troubleshoot, add new features and scale this web application.

All technologies, tools that were used are free besides server that will host database and/or web application. That is really important point since I want this application to be free for all user and deliver to them the best quality of the freight managing web application available on the market for free.

# Subject Area Description

**Freight Transpiration -** simply defined, freight transportation is the movement of goods from one area to another. Freight Transpiration allows production and consumption to occur at different locations. Transportation is necessary for economic specialization. Freight transportation allows companies to specialize in producing the products for which they are best suited and trade with other companies to obtain products that can be made more efficiently by others.

Our nation’s economic strength and competitiveness depends on a safe, efficient, sustainable and secure freight transportation system. Whether it is moving coal from the Powder River Basin in Wyoming to Atlanta, GA or tennis shoes from China to Rochester, New York, the system must provide for the reliable, flexible and economic movement of goods – bulk and consumer – from a diverse array of sources.

Customer relationship management (CRM) is a model for managing a company’s interactions with current and future customers. It involves using technology to organize, automate, and synchronize sales, marketing, customer service, and technical support.

Although for many enterprises it takes time for it to sink in, the result is inevitable. Marketing led the way with the concept of tailoring products and services to the requirements of specific groups of customers. It named its approach ‘segmentation’ and asked customers what they wanted, so that the company could produce different offerings that matched varying needs and wants.

**Fleet management** is the function that oversees, coordinates and facilitates various transport and transport related activities. For the purpose of this document it will cover vehicles involved in the movement of goods; the management of light vehicle fleets used in the transportation of people and light cargo; possibly motorbikes and other equipment such as generators and warehouse handling equipment. Fleet management underpins and supports transport related activities through the management of the assets that are used.

Effective fleet management aims at reducing and minimizing overall costs through maximum, cost effective utilization of resources such as vehicles, fuel, spare parts, etc.

The administration and financial management of fleet is very organizational specific. It largely depends on donor requirements and organizational policies. For example, in some organizations vehicles are restricted to specific projects and others utilize vehicle pools to serve multiple projects. Driving policies can vary from a strict reliance on a dedicated driver from the organization, to using staff to drive the vehicles. The administrative policies of the individual organization will dictate which approach will be utilized. This results in the custodian of the fleet management function to be very much dependent on organizational policies and structures.

# Description of Technologies Used

## **Back-end**

For the core language I choose **Java** programing language. The reason Java is core language for this web app is that Java has his own JVM and same code can be run on different machines. Another reason for Java is that it has very powerful Spring framework.

Every good application needs to be scalable and have lightweight technologies that work great with each other. That was the reason I choose **Spring Boot** **framework** as a backend. This framework of Java is great if it comes to large, enterprise applications and have many API, Documentation, support that helps implement different technologies to it. Spring Boot has no XML based configurations at all. Very much simplified properties. The beans are initialized, configured and wired automatically. Spring is also lightweight container, can deployed on tomcat or jetty, easy to implement new features if your application getting bigger and bigger.

For database connectivity I choose **Hibernate** **ORM** over JDBC because JDBC is complex when is used in large projects, it is harder to implement MVC concept, there is no encapsulation. Hibernate on other side has transparent persistence that ensures automatic connection between the application’s object with the database tables. This feature prevents developers from writing lines of connection code. Transparent persistence enables hibernate to reduce the development time and maintenance cost. Hibernate supports both first level and second level caching mechanism. The first level caching is associated with Session object which is used by default. The second level caching is associated with Session Factory Object. Through caching concept, Hibernate retains the objects in cache so as to reduce repeated hits to the database. This feature makes Hibernate highly scalable and optimizes the application’s performance. Hibernate supports optimistic locking through its version property feature. This functionality supports multiple transactions without affecting one another. For example, when two or more users try to alter a database entity at the same time, the version field avoids the conflict and gives preference to the user who commits the changes first. The other user will be prompted with an error message and will be asked to restart the process.

I integrate Hibernate with **Spring Data** so I will have repositories CRUD out of box, nice and easy ways to extend your repositories with query methods, to navigate between entities.

For authentication and authorization, I used **Spring Security.** It is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications.

As a database I choose **MySQL** because is globally renowned for being the most secure and reliable database management system used in popular web applications. MySQL offers unmatched scalability to facilitate the management of deeply embedded apps using a smaller footprint even in massive warehouses that stack terabytes of data. On-demand flexibility is the star feature of MySQL. This open source solution allows complete customization to eCommerce businesses with unique database server requirements. MySQL features a distinct storage-engine framework that facilitates system administrators to configure the MySQL database server for a flawless performance. Whether it is an eCommerce website that receives a million queries every single day or a high-speed transactional processing system, MySQL is designed to meet even the most demanding applications while ensuring optimum speed, full-text indexes and unique memory caches for enhanced performance. All the fears and worries that arise in an open source solution can be brought to an end with My SQL’s round-the-clock support and enterprise indemnification. The secure processing and trusted software of MySQL combine to provide effective transactions for large volume projects. It makes maintenance, debugging and upgrades fast and easy while enhancing the end-user experience. MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark tests (though such tests -- regardless of the source -- should be considered with a good dose of skepticism). MySQL is really easy to use with hibernate. There is nothing too complicated to have those two technologies configure and up and running in less than 1 hour. MySQL also provides with very powerful tool – MySQL Workbench – that makes your development of database super easy and enjoyable. On top of that this tool has many other useful tools like data migration, database backup, UML creator, schema generator etc.

**Webpack** is an open-source JavaScript module bundler. Webpack takes modules with dependencies and generates static assets representing those modules. It takes the dependencies and generates a dependency graph allowing web developers to use a modular approach for their web application development purposes.

**Node.js** is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts

## **Front-end**

**HTML** is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.

**JavaScript** is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

For front end part I have choose **ReactJS**. It is pretty new JavaScript framework developed and released by Facebook. With React, it is possible to develop large web-based applications that change data while reloading pages. You can also use React together with other forms of frameworks or JavaScript libraries like Angular in MVC. React is an open source JavaScript (JS) library used for developing user interfaces, basically for the single-page applications. When dealing with ReactJS, the basic programming knowledge that you need to possess is HTML and CSS. Angular, on the other hand, is more difficult to learn. If you have not familiarized yourself with typescript, then do so before learning Angular. Typescript on its own is not easy to learn, so be prepared for a long learning curve.One major advantage Facebook brought with ReactJS is the possibility of reusing code components. Webmasters can now use codes of different levels anytime, and this saves time. Usually, system upgrades are a headache to webmasters, as other components can be affected by any changes made.With ReactJS, it is easier to manage updates. The reason is that all the components of the system are completely isolated. As a result of this, changes initiated in any of the components will not affect the others. It makes it possible to reuse components that didn’t produce changes, and this makes programming more comfortable and precise.Another advantage with ReactJS is code stability. It allows programmers to work directly with the components and also make use of the [data-binding technique](https://www.techopedia.com/definition/15652/data-binding) to make sure changes that what took place in child structures doesn’t affect the parents.

**CSS** is used for presentation layer. CSS stands for Cascading Style Sheets.CSS describes how HTML elements are to be displayed on screen, paper, or in other media.CSS saves a lot of work. It can control the layout of multiple web pages all at once.

**Bootstrap** framework is responsible for presentation layer. Bootstrap is responsive, mobile-first, prevailing, and front-end framework, which is developed along with CSS, JavaScript, and HTML. It is extremely an easy and speedy procedure to begin with Bootstrap. Bootstrap is very adaptable too.One of the main benefits of utilizing Bootstrap happens to be the speed of the development. While driving out a new, fresh website or application swiftly, you should certainly reflect upon utilizing Bootstrap. Instead of coding from scrape, Bootstrap lets you to use ready-made coding blocks in order to assist you in setting up. You can blend that along with CSS-Less functionality and cross-browser compatibility that can give way to saving of ample hours of coding. You can even buy ready-made Bootstrap themes and alter them to fit your requirements, for gaining the quickest potential route. The Bootstrap can be customized as per the designs of your project. Bootstrap approaches with a pack of JavaScript components for including the functionality that crafts it in simple way for operating things, such as tooltips, modal windows, alerts, etc. You can even leave out the writing scripts completely.

## **Host**

This application is hosted online on **Apache Tomcat**. Apache Tomcat is an open source Web server tool developed by the Apache Software Foundation (ASF). It is one of many Apache-related open source products used by IT professionals for various tasks and objectives. Apache tomcat is an incredibly lightweight. If offers only the most basic functionality necessary to run a server, meaning it provides relatively quick load and redeploy times compared to many of its peers, which are bogged down with far too many bells and whistles. This lightweight nature also allows it to enjoy a significantly faster development cycle. It is open-source always counts as a win. Tomcat’s free, and the source code for the server is readily available to anyone who’d care to download it. Tomcat is an extremely stable platform to build on – and using it to run your applications will contribute to your server’s stability, as well. This is because Tomcat runs independently of your Apache installation – even if a significant failure in Tomcat caused it to stop working, the rest of your server would run just fine.

Apache Tomcat web server is hosted on **Centos**. Centos is a [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution) that provides a free, [enterprise-class](https://en.wikipedia.org/wiki/Enterprise_software), community-supported computing platform functionally compatible with its upstream source, Red Hat Enterprise Linux (RHEL). When CentOS 7 is properly configured, and running on quality hardware, it is a very stable server operating system, with very few (if any) problems. There is reduced risk of crashes and errors, as it runs only stable versions of packaged software.

For PaaS (**P**latfrom **a**s **a** **S**ervice) I used **DigitalOcean.** It is an [American](https://en.wikipedia.org/wiki/Americas) [cloud](https://en.wikipedia.org/wiki/Cloud_computing) infrastructure provider headquartered in [New York City](https://en.wikipedia.org/wiki/New_York_City) with data centers worldwide. DigitalOcean provides developers cloud services that help to deploy and scale applications that run simultaneously on multiple computers. As of January 2018, DigitalOcean was the third-largest [hosting](https://en.wikipedia.org/wiki/Web_hosting_service) company in the world in terms of web-facing computers.

# Database Design

The first step in the database design was to analyze the data that would be collected and determine the expected uses of the data.

Customer place an order through freight company representative. Then representative have to enter load information to the system, assign it to the driver and to customer. In this database we have total of 5 tables. They are:

1. **users** – this table hold information about users that logging in to the application. They are drivers and dispatchers (representative that agree with customer on rate and load). This is the most important table since it connects to rest 4 tables.

|  |  |  |
| --- | --- | --- |
| **Users** | | |
| **Column Name** | **Datatype** | **Description** |
| user\_ID | INT(11) | ID number |
| username | VARCHAR(45) | Username |
| password | CHAR(60) | Password |
| role | VARCHAR(45) | Role of the user |
| email | VARCHAR(255) | E-Mail address |
| confirmation\_token | VARCHAR(255) | Confirmation token for new password |
| enabled | BIT(1) | Enabled field says if user is active in the system |
| organization | VARCHAR(255) | Organization name |
| first\_name | VARCHAR(45) | First Name |
| last\_name | VARCHAR(45) | Last Name |
| address | VARCHAR(45) | Address |
| city | VARCHAR(45) | City |
| state | VARCHAR(2) | State |
| phone\_number | VARCHAR(45) | Phone Number |
| zipcode | VARCHAR(45) | Zip code |
| version | VARCHAR(45) | Version NR. of this user it is used for locking mechanism |

1. **customer** – this table is responsible for keeping information about customer.

|  |  |  |
| --- | --- | --- |
| **Customers** | | |
| **Column Name** | **Datatype** | **Description** |
| customer\_ID | INT(11) | ID number |
| name | VARCHAR(45) | name |
| mc\_number | INT(11) | MC number |
| dot\_number | VARCHAR(45) | DOT number |
| address | VARCHAR(45) | Address |
| city | VARCHAR(45) | City |
| state | VARCHAR(45) | State |
| zip\_code | VARCHAR(45) | Zip Code |
| phone\_number | VARCHAR(45) | Phone Number |
| email | VARCHAR(45) | E-Mail |
| fax | VARCHAR(45) | fax number |
| description | VARCHAR(45) | Description |
| version | INT(11) | Version NR. of this customer it is used for locking mechanism |
| users\_user\_ID | VARCHAR(45) | User ID to which this customer belongs |

1. **facilities** – this table contain information about facility that load is picking up from

|  |  |  |
| --- | --- | --- |
| **Facilities** | | |
| **Column Name** | **Datatype** | **Description** |
| facility\_ID | INT(11) | ID number |
| name | VARCHAR(45) | Name |
| address | VARCHAR(45) | Address |
| city | VARCHAR(45) | City |
| zip\_code | VARCHAR(10) | State |
| state | VARCHAR(2) | Zip Code |
| phone\_number | VARCHAR(45) | Phone Number |
| email | VARCHAR(45) | E-Mail |
| version | VARCHAR(45) | Version NR. of the facility it is used for locking mechanism |
| users\_user\_ID | INT(11) | User ID to which this facility belongs |

1. **loads** – this table contain information about driver that is assign for this load, rate, extra cost, instruction

|  |  |  |
| --- | --- | --- |
| **Loads** | | |
| **Column Name** | **Datatype** | **Description** |
| load\_ID | INT(11) | ID number |
| rate | VARCHAR(45) | Name |
| extra\_cost | VARCHAR(45) | Address |
| instruction | VARCHAR(45) | City |
| version | VARCHAR(45) | Version NR. of the facility it is used for locking mechanism |
| users\_user\_ID | INT(11) | User ID to which this load belongs |
| driver\_ID | VARCHAR(45) | user ID with role driver to which this load is assign to |
| customers\_customer\_ID | VARCHAR(45) | customer id to which this load belongs |

1. **pick\_drops** – this table connect load table to facilities table and contain extra information about each pickup or drop like time, date, type (pick or drop), and instruction for this specific pickup.

|  |  |  |
| --- | --- | --- |
| **picks\_drops** | | |
| **Column Name** | **Datatype** | **Description** |
| picks\_drops\_ID | INT(11) | ID number |
| facility\_ID | INT(11) | facility ID number to which this pickup/drop is assaign to |
| load\_ID | INT(11) | Load ID numbet to which this pickup/drop belongs |
| work\_type | VARCHAR(45) | City |
| time | TIME | time of pickup/drop |
| date | DATE | date of pickup/drop |
| instruction | VARCHAR(45) | instruction |
| version | VARCHAR(45) | Version NR. of the pickup/drop it is used for locking mechanism |
| users\_user\_ID | INT(11) | User ID to which this pickup/drop belongs |

Also, all of those table (besides table user) contain reference to table user because we have multiples users. When user creates account, he creates root account for his company. When user is creating driver it automatically creates new user in database. There are two roles of user available. One is role ROLE\_MANAGER which is responsible for creating/deleting/updating customers, loads, facilities, pick\_drops, users tables (for users table ROLE\_MANAGER can only manipulate users with role ROLE\_DRIVER.

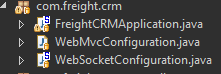
Second role is ROLE\_DRIVER and he can only view data in customers, loads, facilities, pick\_drops tables assign only for his manager account.

# Application Structure and UI Design

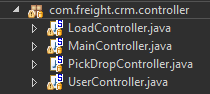
The TuckGistic application is broken up into three parts: client-side application, server side application and MySQL database (database is described in previous chapter).

The **server-side** is based on Java framework Spring and is separated into 6 different packages:

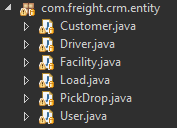
* *com.freight.crm* - contain three classes that are responsible for configuration of spring framework, WebSocket and main Spring class responsible for running application



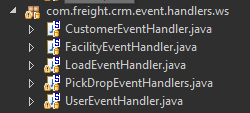
* *com.freight.crm.controller* – contain controllers responsible for mapping and process custom request from client-side application



* *com.freight.crm.entity* – contains classes that represents tables and relationship in MySQL database and contain setters and getters methods for each class



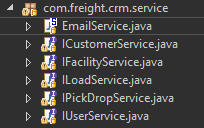
* *com.freight.crm.event.handlers.ws* – this package contains classes that are responsible for handling before/after create delete update operation on entity classes. Also it is responsible to send message to connected users in WebSocket endpoint to refresh their content in web.



* *com.freight.crm.security* – is responsible for Spring security context. Here are all configuration to what quest has access and registered users. It also contain class to search users by username.



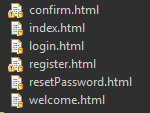
* *com.freight.crm.service* – this last package is responsible business logic. Most of the interfaces extands PagingAndSortingRepository interface which enables simple REST endpoint for clinet. It also contain class that is responsible for sending confirmation emails to new users.



The **client side** is based on JavaScript framework ReactJS. We use webpack to compile JavaScript components into a single, loadable bundle. The whole client side is hosted on node.js server. When user access application first time it first check with back-end engine to check for authentication and Spring returns view for gest with help of Thymeleaf framework. When user logging in and pass authentication part he is then logged in to ReactJS application.

Per above front end side is divided in two parts. One before user is logged in and all request are handled by Spring framework and second when user is already logged in and Spring will return page where ReactJS has his start point. There are 6 html pages that are return by Spring framework. It is:

* *confirm.html* – this page is displayed when user click on link with his confirmation token to confirm that his email is correct and to enter his new password,
* *index.html* – this page is displayed when user is already logged in and return start point for ReactJS application and then ReactJS makes REST call to back end engine Spring,
* *login.html* - this page is displayed when user is trying to logging in to the application
* *register.html* – this page is displayed whenever new user wants to register,
* *restPassword.html* – this page is displayed whenever user wants to reset his password,
* *welcome.html* – this page is displayed for quest users when they first time enter the website.



Each of this page contain two fragments:

* *footer.html* – contains footer part for web pages
* *header.html* – contains header part for web pages

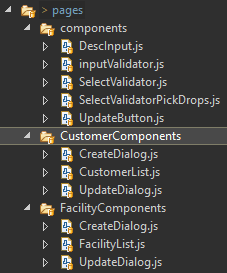
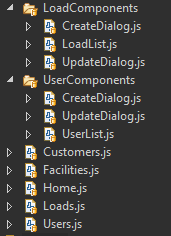


The second part of client side is ReactJS. This part is available only for authorized users after they pass authentication level. We use webpack to define entry point, sourcemaps, compile all JavaScripts bits and it hooks the babel engine using both es2016 and react presets in order to compile ES6 React code into format able to run in any standard browser. All this configuration are located in webpack.config.js file. ReactJS app contain:

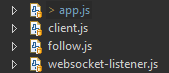
1. *api* Folder – it contains .js file responsible supports URI Templates



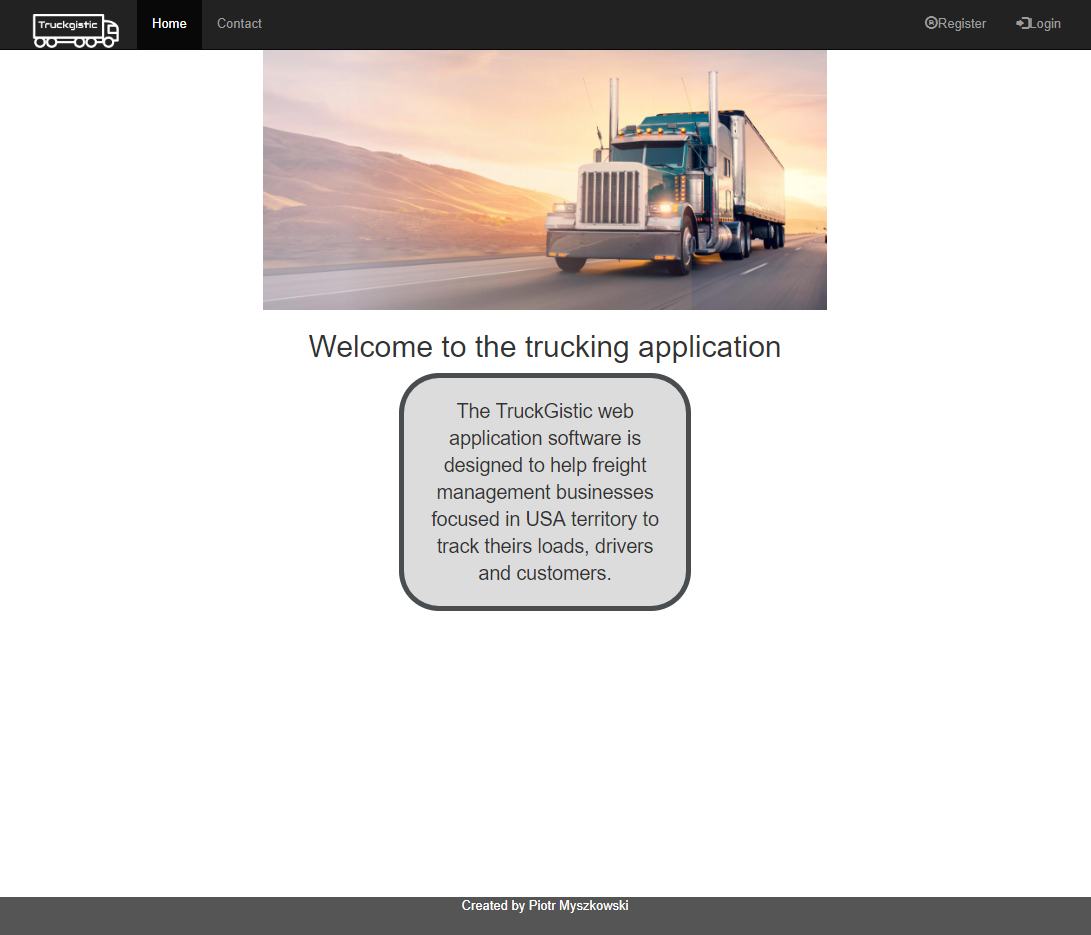
1. *pages* Folder – contains .js file for each page. This folder have .js file that generates web pages for facilities, customers, loads, users page. This folder has following structure:
   1. *components* Folder – contain global components that are used in forms for page in pages Folder.
   2. *CustomerComponents* Folder – contain .js files for create, updated and view customers
   3. *FacilityComponents* Folder - contain .js files for create, updated and view facilities
   4. *LoadComponents* Folder - contain .js files for create, updated and view loads
   5. *UserComponents* Folder - contain .js files for create, updated and view drivers assigned for Manager user
   6. *Customers.js* – is main .js file for customer page that is responsible to make REST call to spring, render view using components from *CustomerComponents* Folder by passing data to those components
   7. *Facility.js* - is main .js file for facility page that is responsible to make REST call to spring, render view using components from FacilityComponents folder by passing data to those components
   8. *Loads.js* - is main .js file for loads page that is responsible to make REST call to spring, render view using components from LoadComponents folder by passing data to those components
   9. *Users.js* - is main .js file for users page that is responsible to make REST call to spring, render view using components from UserComponents folder by passing data to those components

1. *app.js* – is entry point and it is responsible to route user to whichever component he is trying to access
2. *client.js* - is custom code that configures rest.js to include support for HAL, URI Templates, and other things. It also sets the default Accept request header to application/hal+json.
3. *follow.js* – contain follow function that is responsible to navigate between relationships return by JSON object from the root level
4. *websocket-listener.js* - is responsible for registering user to websocket endpoint in Spring framework server.

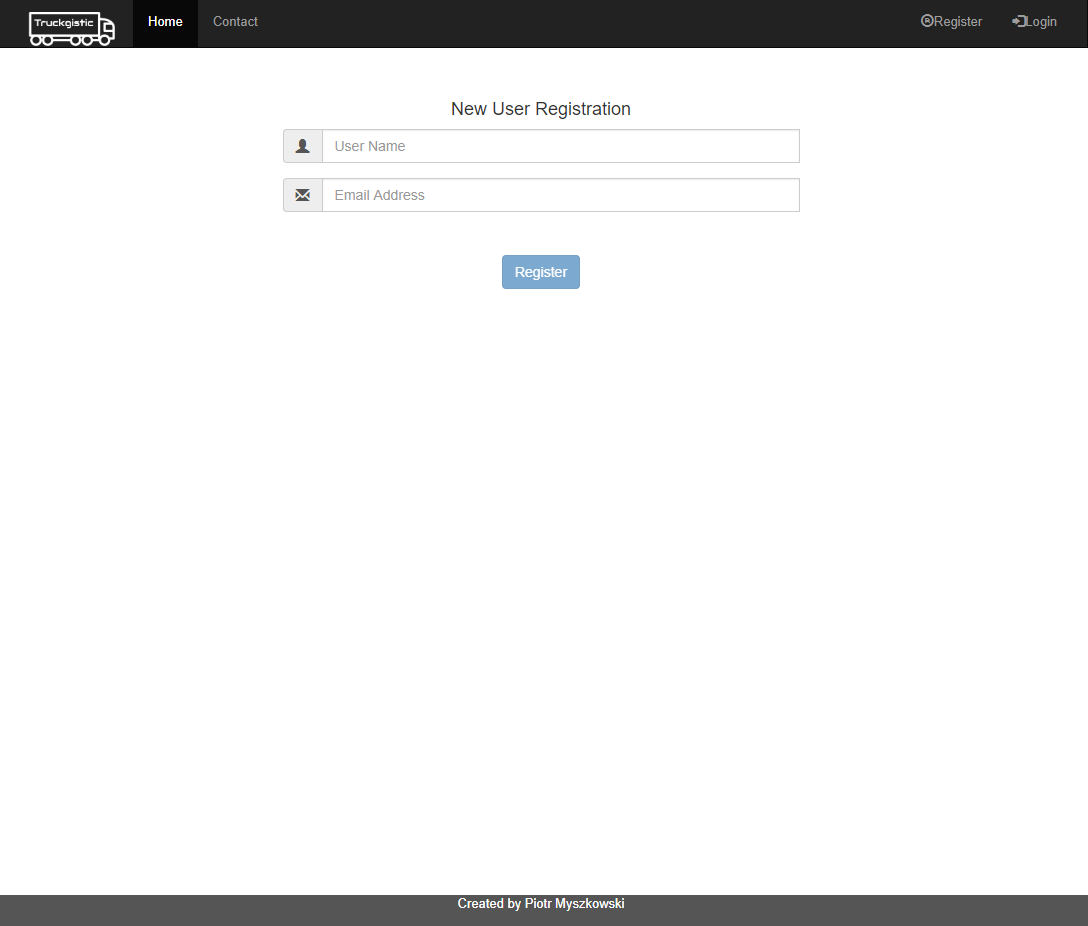


User Interface Design is simple and straight forward. When you enter the web application from you PC or mobile device you will see friendly home page.



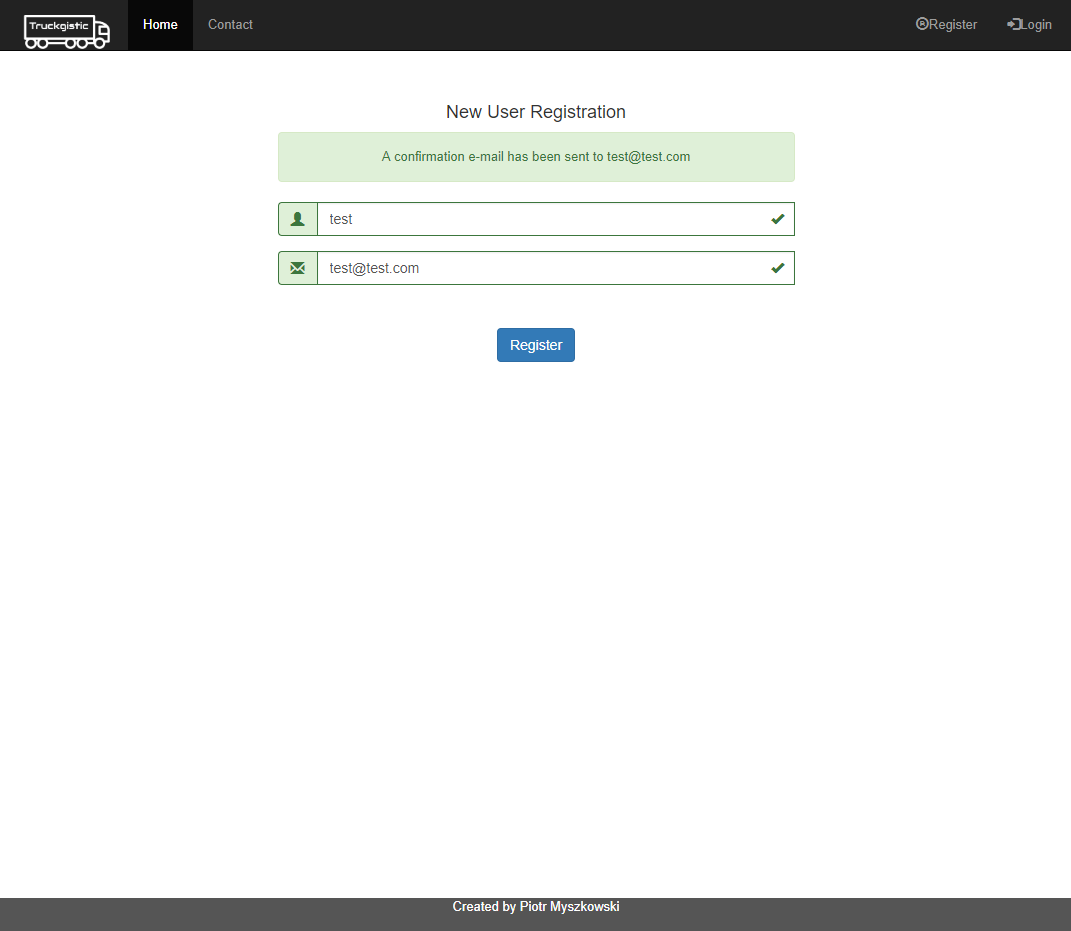
Home page

If the user is a new user then he first have to go through the register process by clicking register button located on top right corner.



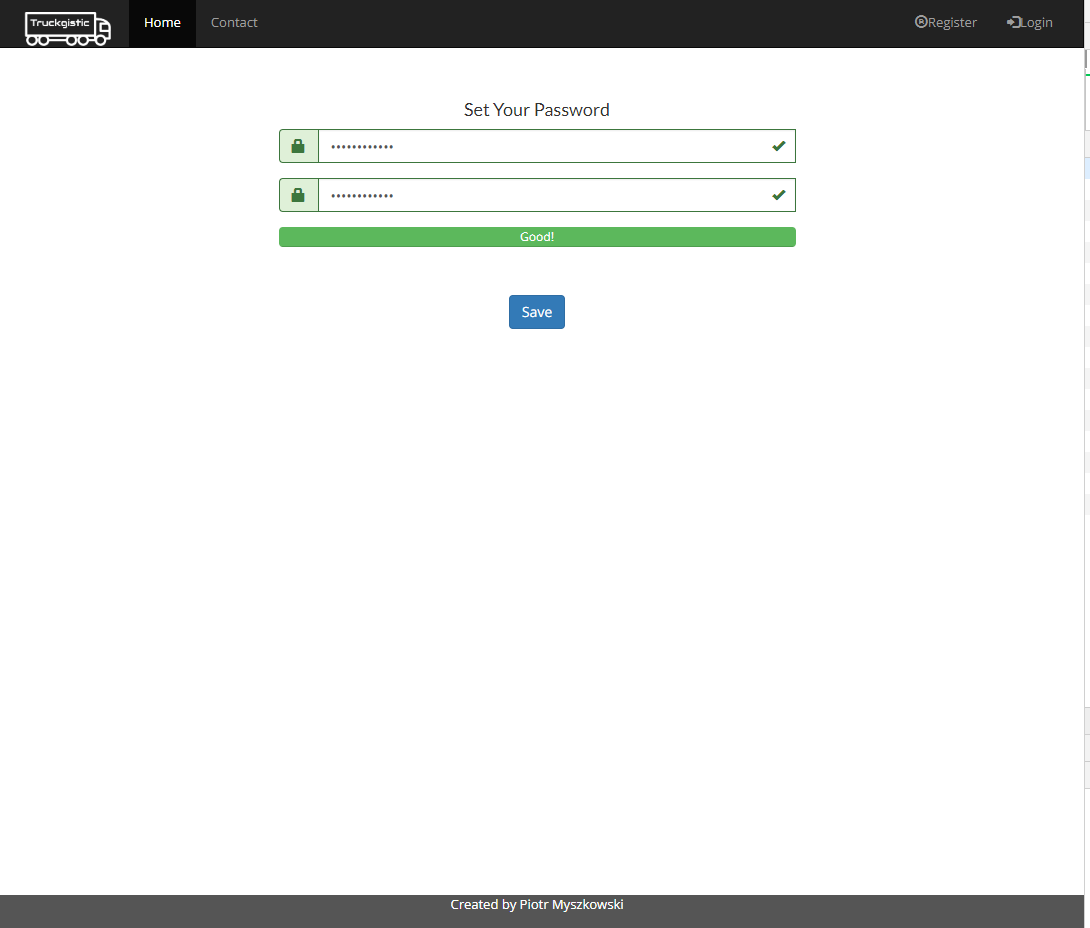
Register page

User has to provide unique email as well as username. Then the system sends the email that was provided by user with the link he has to go in order to create his password. This method make sure that the email is correct and user has access to this email.



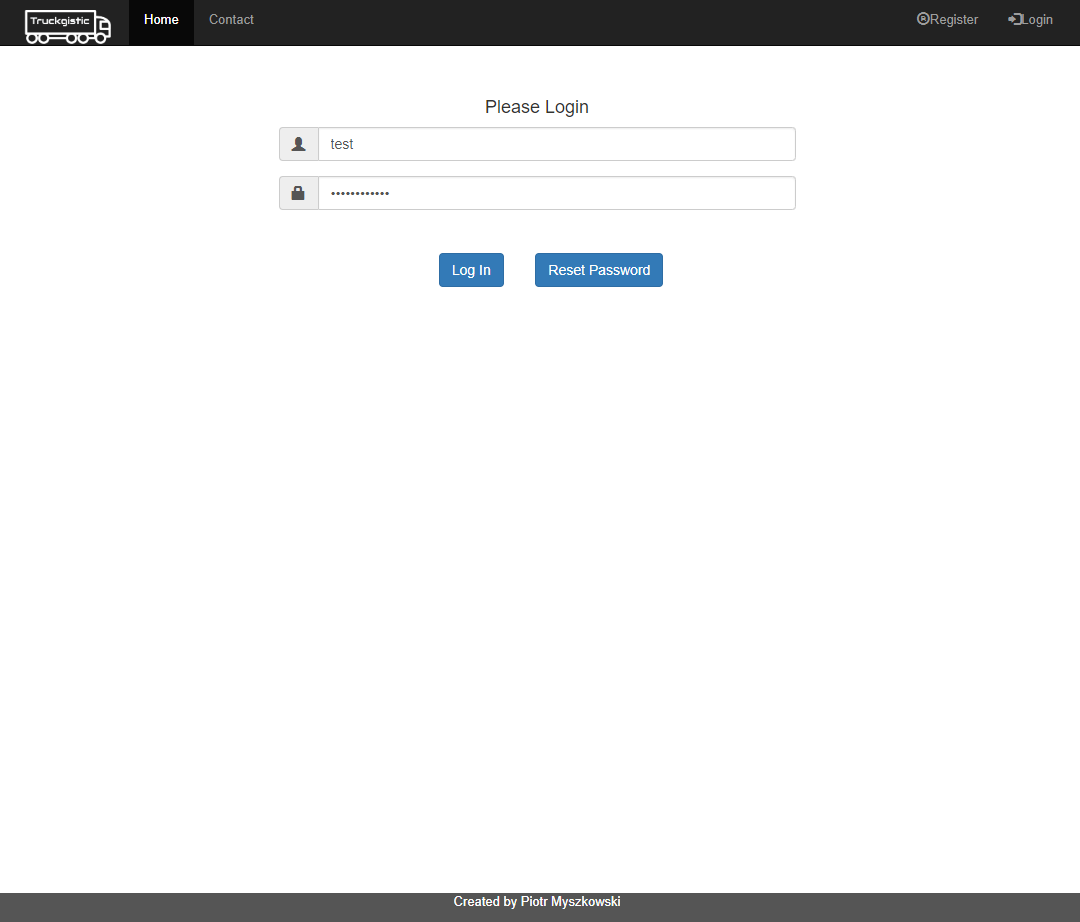
Confirmation page

When user click on the link provided in the email send by the web application, he will be redirected to the website to enter his password. Password has to have one upper letter, special character, number and minimum of 8 characters.



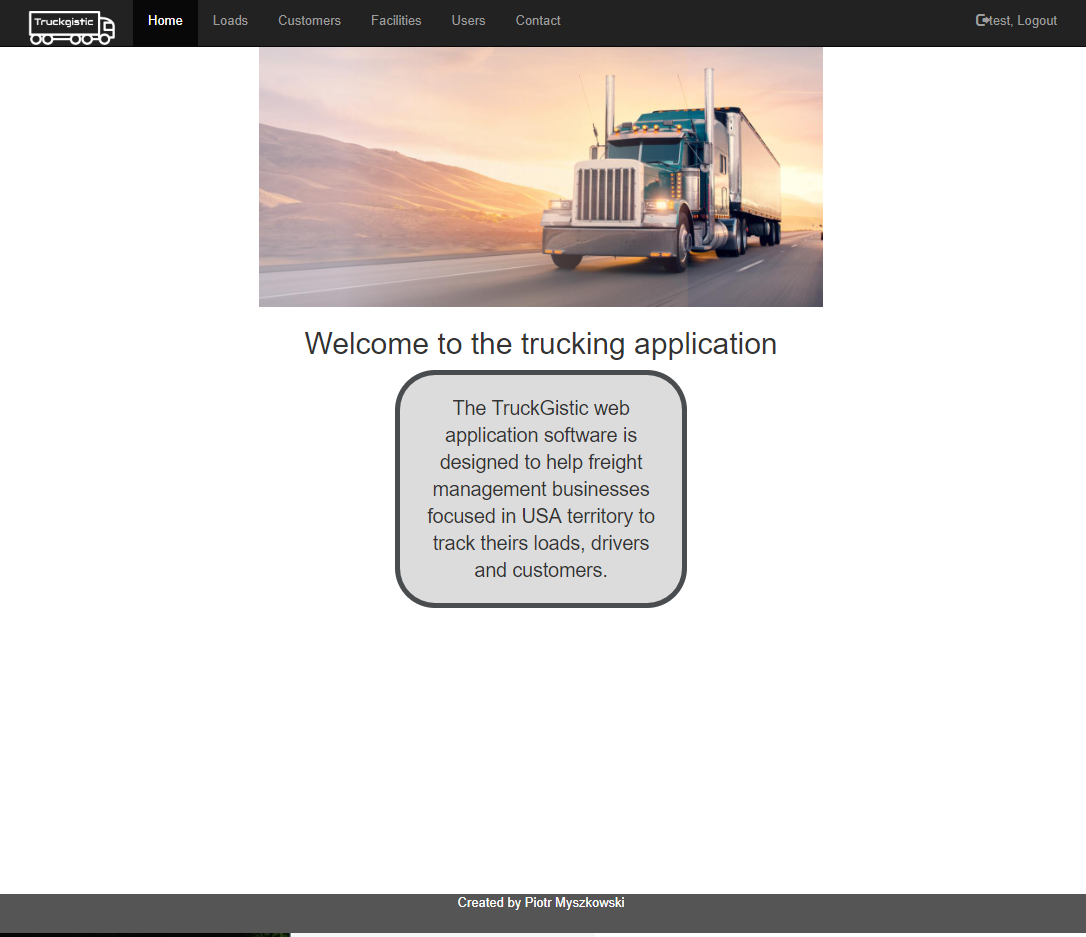
Set password page

After registration process is completed user can finally login to the website.



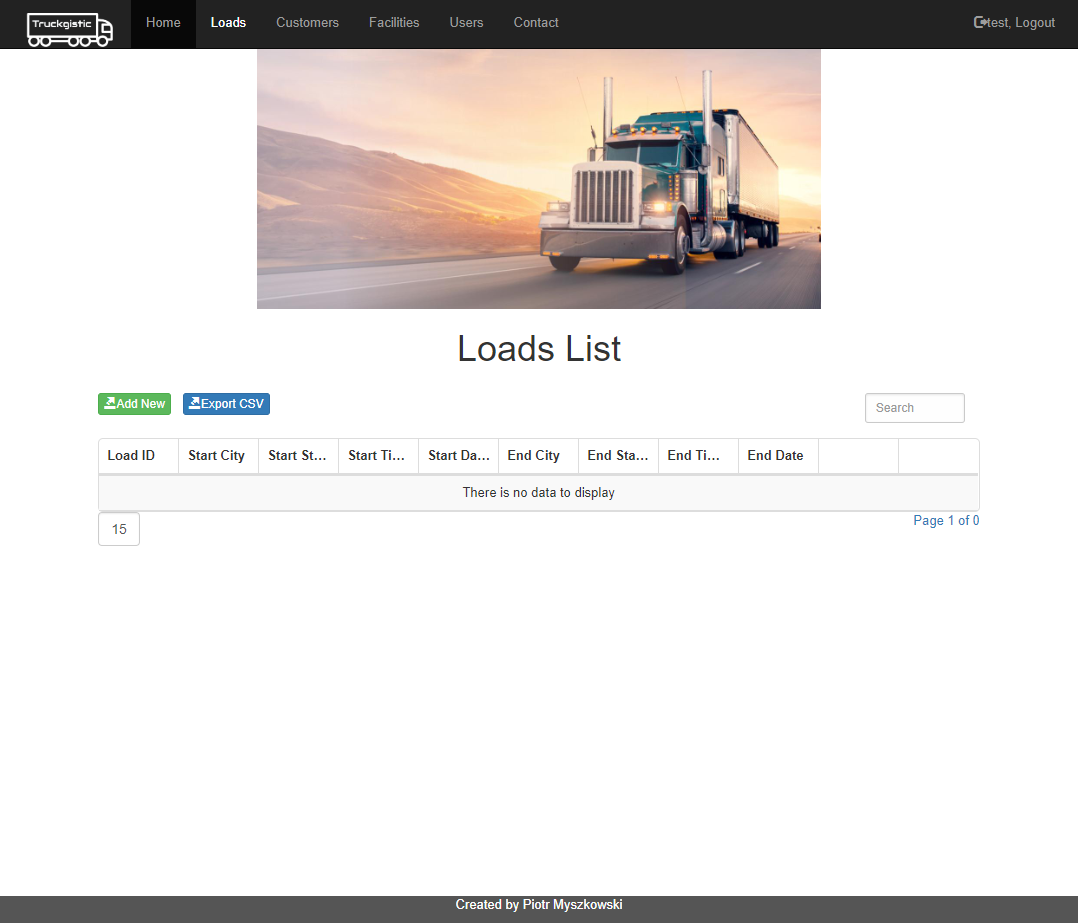
Login page

There are two types of user’s, manager and driver. Managers can add/remove/edit loads, customers, facilities, drivers. When manager creates new driver our web application creates new account with different permissions. After successful login user will see similar page to the one below. Since we are having manager type account we can see tabs for loads, customers, facilities, drivers.



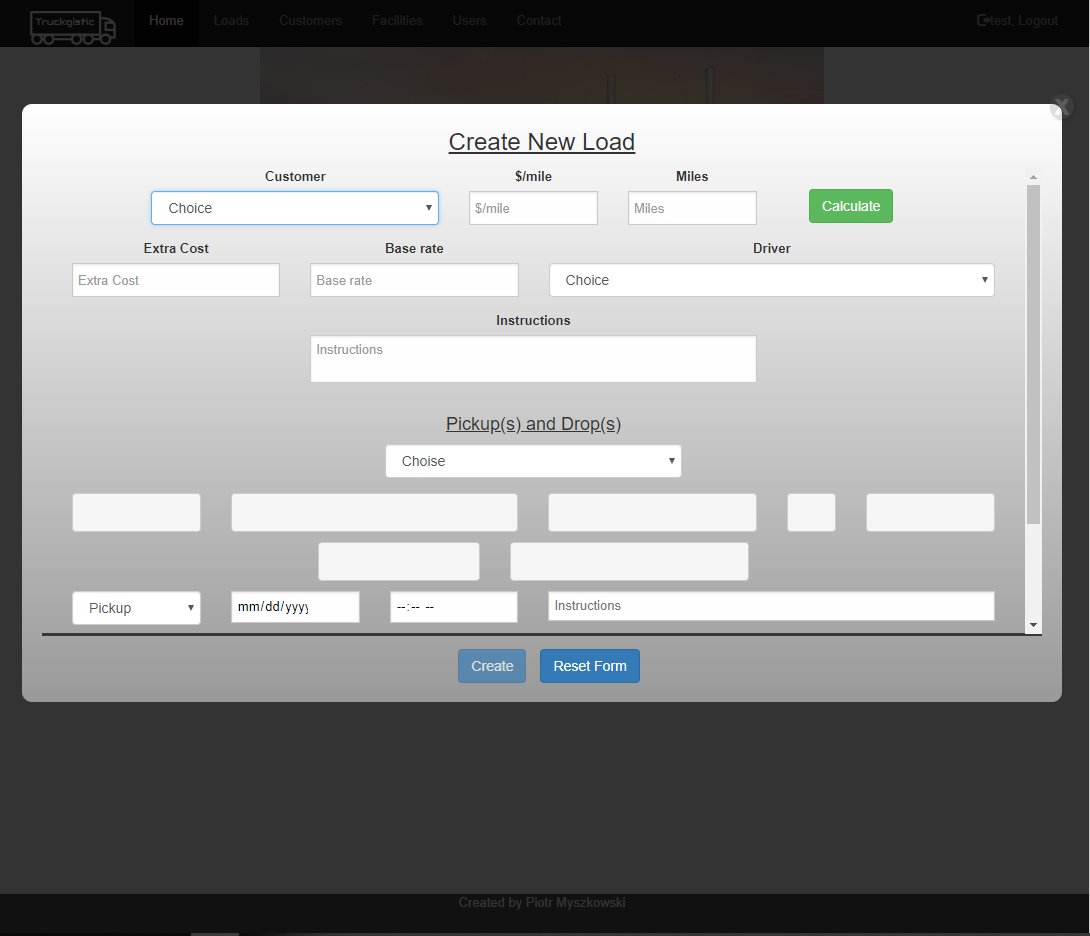
Home page after login

On the Loads tab user can add/remove/edit/search or export to csv existing loads. Whenever there is new load or update on the load from different user we would see this update right away because we are subscribed to the websocket broker of the server. Whenever there is an update server sends, to the all clients subscribed to it, a message to update the content.



Load list page

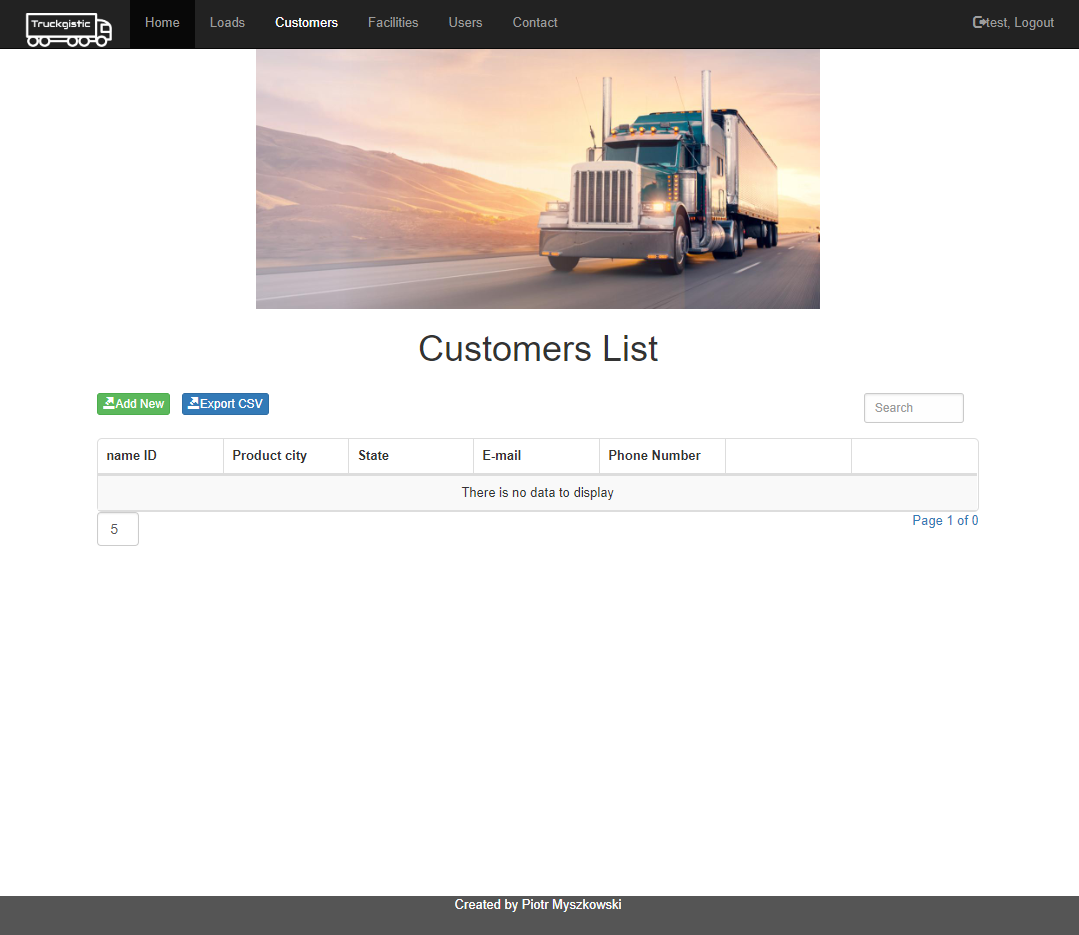
To create a new load, we simply click on Add New and a new pop up window appear. From Customer dropdown we select a customer that we are transporting this freight for. Application can calculate the cost of the service by giving it cost per mile and miles (if we don’t specify the miles calculator will evaluate the distance between two or more points by itself). Next, we have Driver that this load will be assign to. Another input is the global instructions/notes for this load.



Load create load

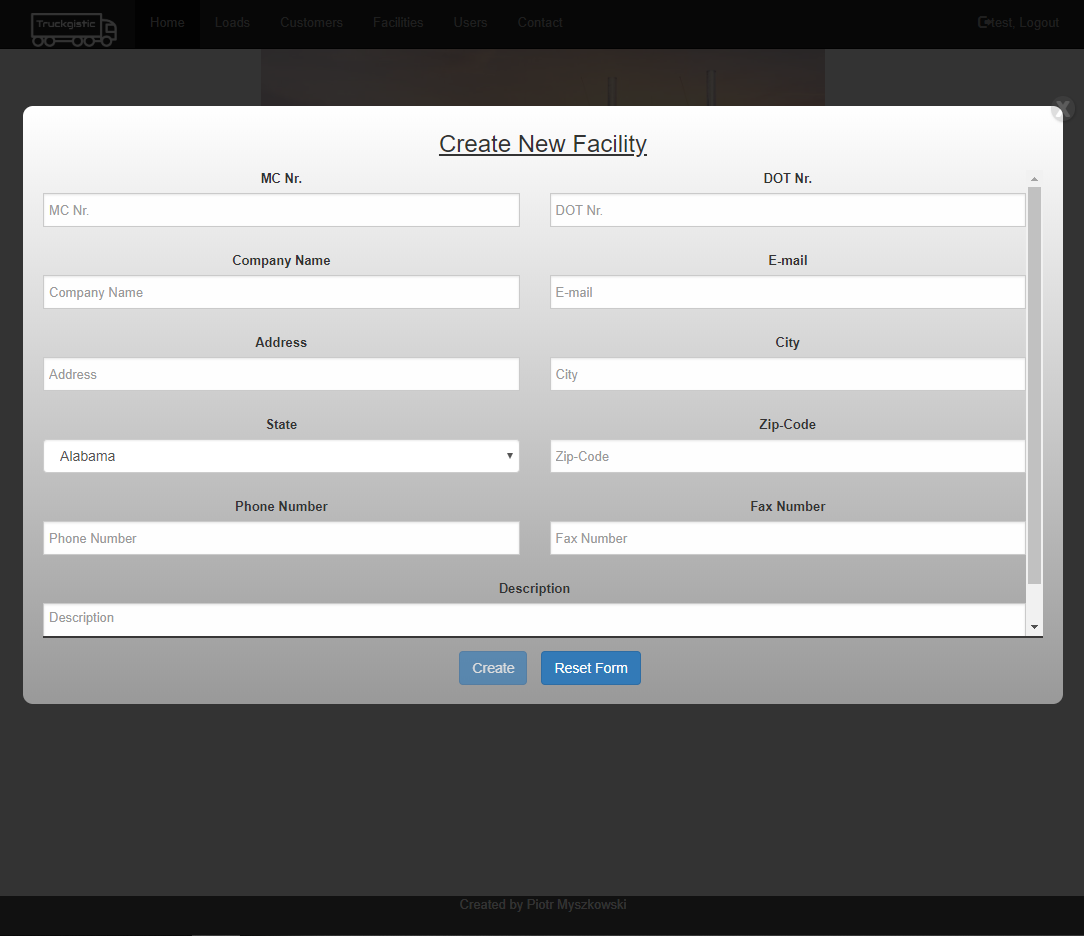
Second section of our form is Pick and Drop tool. Here we selecting from dropdown facilities we have in our database. These represent pickup or delivery location. Blank input you see on the picture below are information about the facility selected in the dropdown. When we are done with selecting the facility we have to choose the type of the stop (pickup/delivery), time and date. After we are done with filling out the form we can hit create button. The load will be created and server will notify all users that there is new load.

Customers page is responsible for managing customers. You can create, delete, edit, search customers or export all customers to CSV file and all users will be able to see the change right away on their UI without reloading the page. Customers are required field for creating a new load.



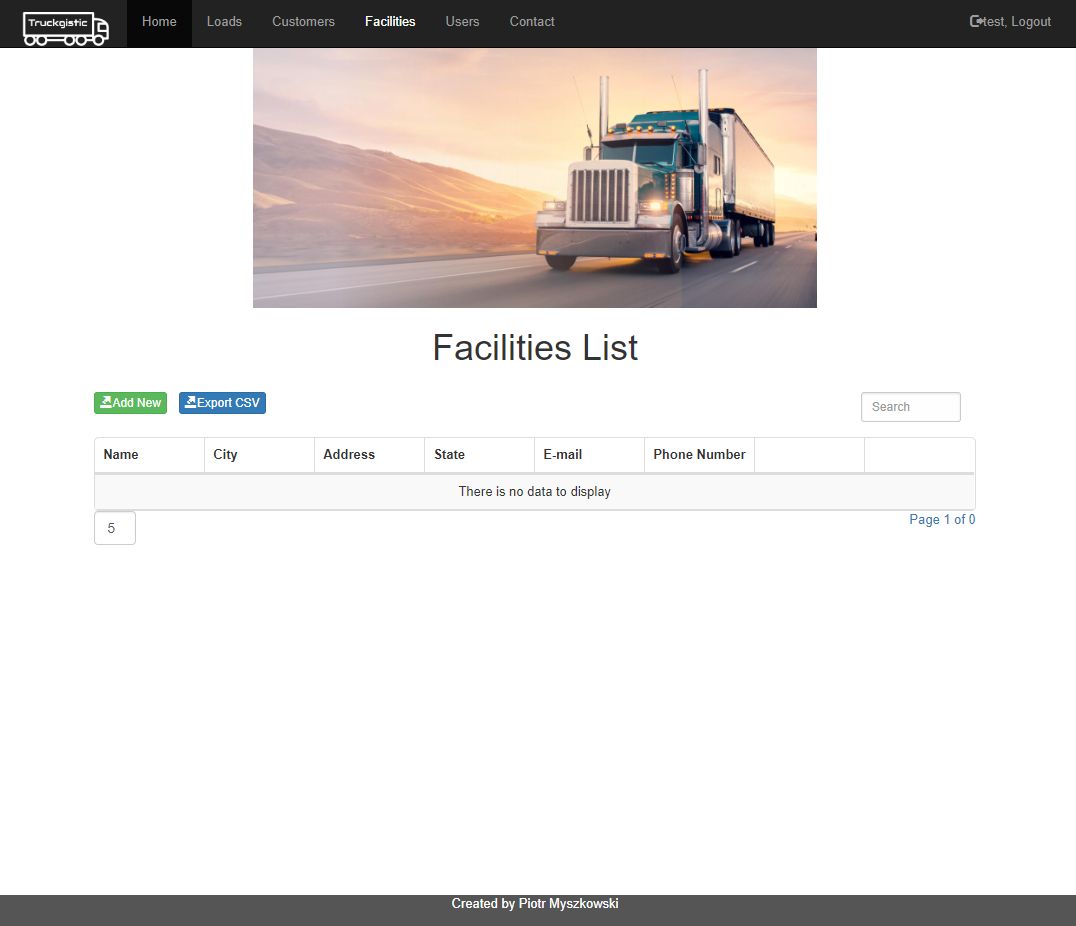
Customer’s page

When creating a new customer, we have to know their detailed information’s. Like MC Number, DOT Number, Company, address, etc. There is description field that let user add additional notes about the customer, it is optional.



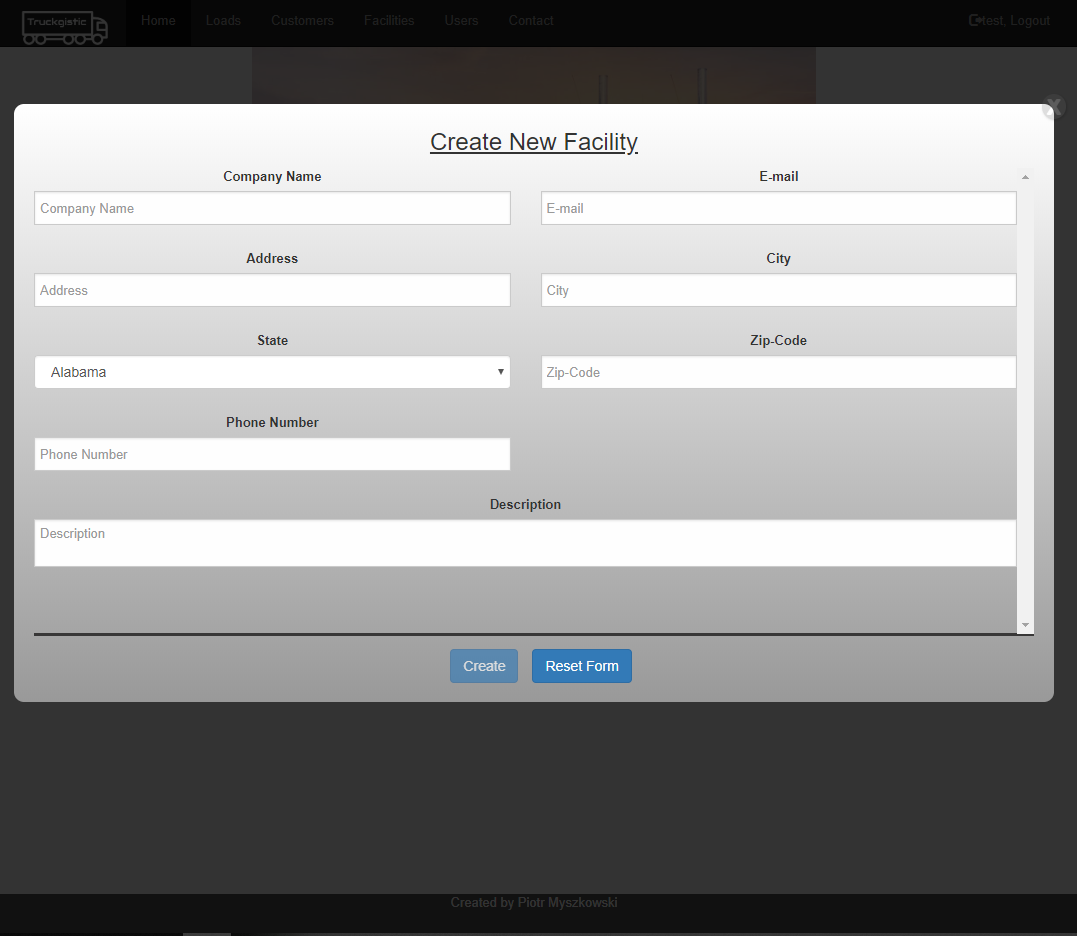
Create new customer

In Facility page it is really similar to Customer page. You can do exactly same task except here our entity is facility.



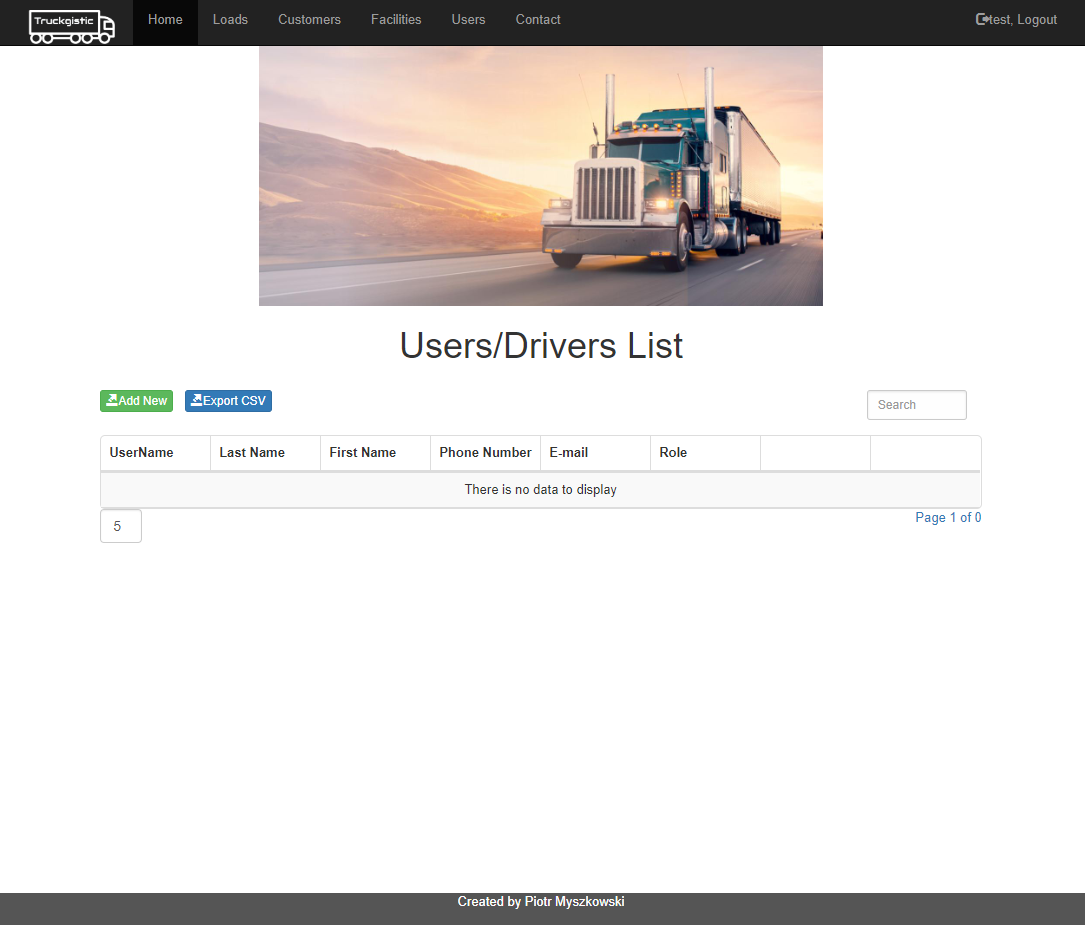
Facilities List

Create facility process is similar to create customer process as well. There are fewer inputs to populate since this part in logistics industry is as not critical as customer who will pay us for the service.



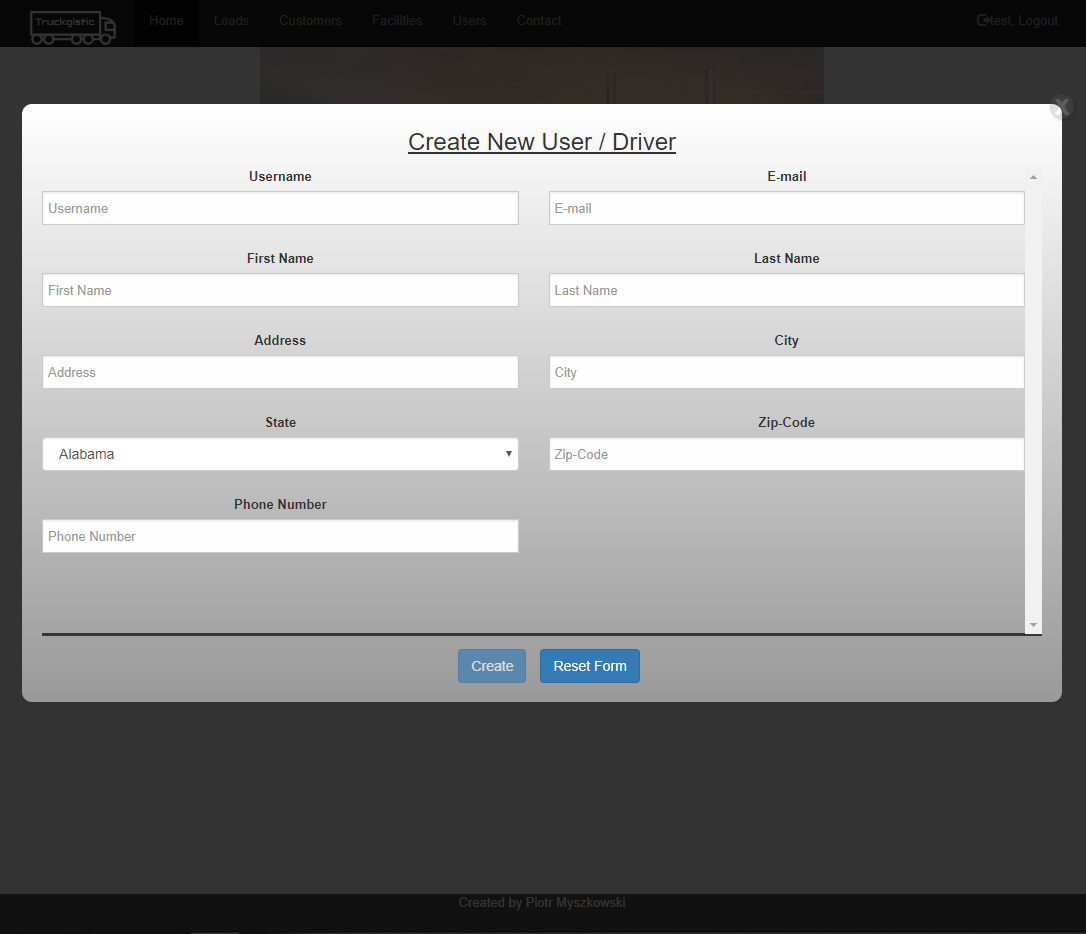
Create new facility

Our next page is users/driver page. Here we can manage our drivers. When user creates new driver our web application creates new account that this driver can access the application from any browser or device and check loads he is assign to. Most of the functionality on this page are inherit from Customers, Facilities and Loads pages. We can also delete, remove, edit drivers.



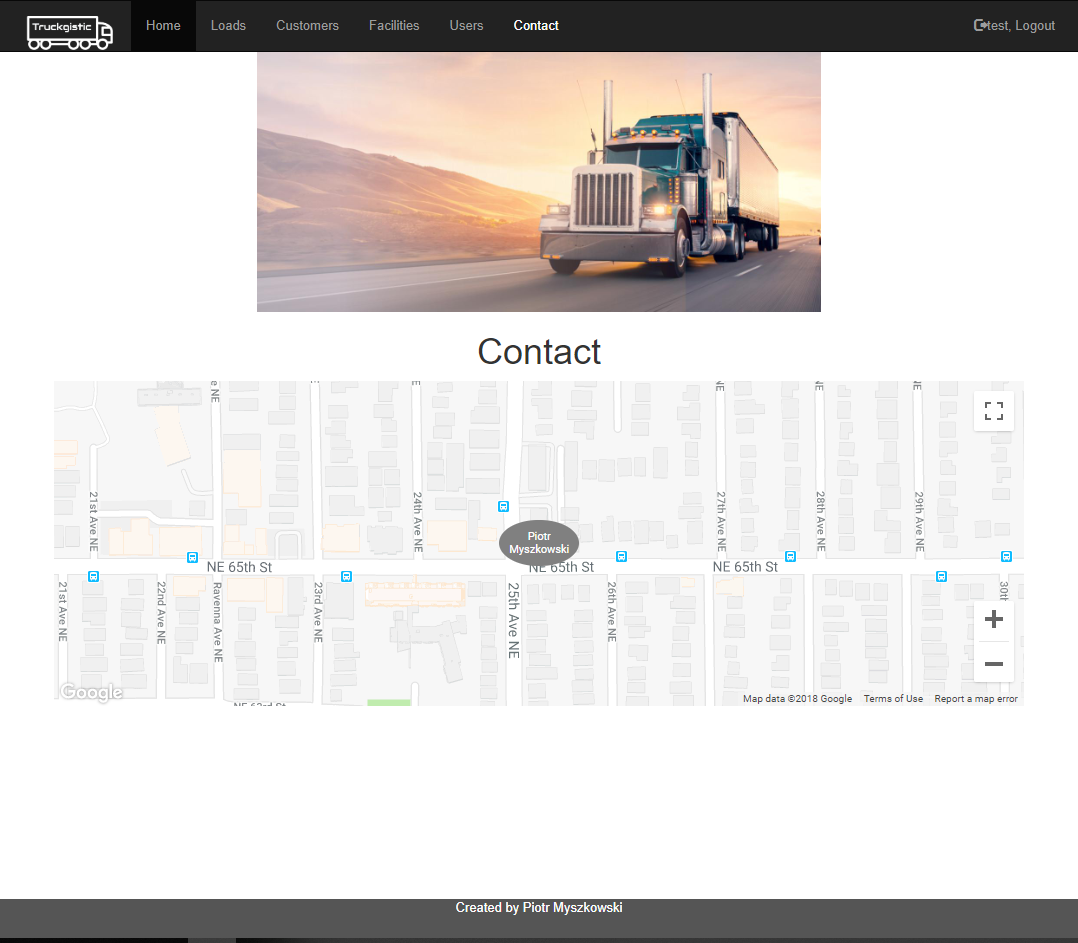
Users list

The creation form for new driver has basic information about the driver. Notice that there is no password specified when creating new driver. It is because driver will receive the link where he can create his own password.



Users create user

Last page Contact contain location and contact information where I can be found.



My contact information’s

# Testing

# Conclusion

# References