Technical Solution Description

LogiWeb Application

author: Evgenii Govorushkin
version 1.0

Saint Petersburg, 2021

Table of Contents

About system.....2
Used technologies and instruments.....3
Additional implemented features.....5
Database schema 6
Architecture 7
Layers of application.....8
User interface.....11
Unit tests.....19
Build and deploy applications.....20
Logging.....21
The statistic about application code.....22
Planned improvements.....23

About system

LogiWeb - web-application, a logistics system for some transportation company. It contains two types of users - Administrator and Driver. The administrator can view, add, modify and edit drivers, trucks, cargoes and orders.

Drivers can view their personal information, change their status and order status.

Used technologies and instruments

Instruments

Name	Version
IntelliJ IDEA 2020	2020.2.3 (Ultimate Edition)
MySQL Workbench 8.0	8.0.22.Build 107600 CE (64 bits) Community

Technologies

Name	Version
JDK	1.8.0_275
Apache Maven	3.6.3
Apache Tomcat	8.5.61
WildFly (JBoss)	22.0.0.Final
Spring Framework	5.3.2
Spring Security	5.3.6.RELEASE
Hibernate	5.4.26.Final
Hibernate Validator	6.1.7.Final
MySQL Connector	8.0.22
Servlet API	4.0.1
JSTL	1.2
C3P0	0.9.5.5
Model Mapper	2.3.9
Log4j	1.2.17
Sl4j	1.7.30
JUnit 5 (Jupiter API)	5.7.0
Mockito	3.7.7
Spring Test	5.3.3
Hamcrest	2.2
Jackson	2.12.1
Active MQ Spring	5.16.1
Spring JMS	5.3.3
GSON	2.8.6
JSON	20201115
SonarQube	8.5.1.38104
Google Maps API	0.17.0
JSF	2.2.20
JavaEE	8.0.1
Primefaces	10.0.0-RC1
Bootstrap	4.5.2

Datatables	1.10.23
JQuery	3.5.1
Feather Icons	4.28.0
Font Awesome Icons	5.15.2

Additional implemented features

To calculate the distance between two cities and travel time is used Google Maps API (Directions API).

Database schema

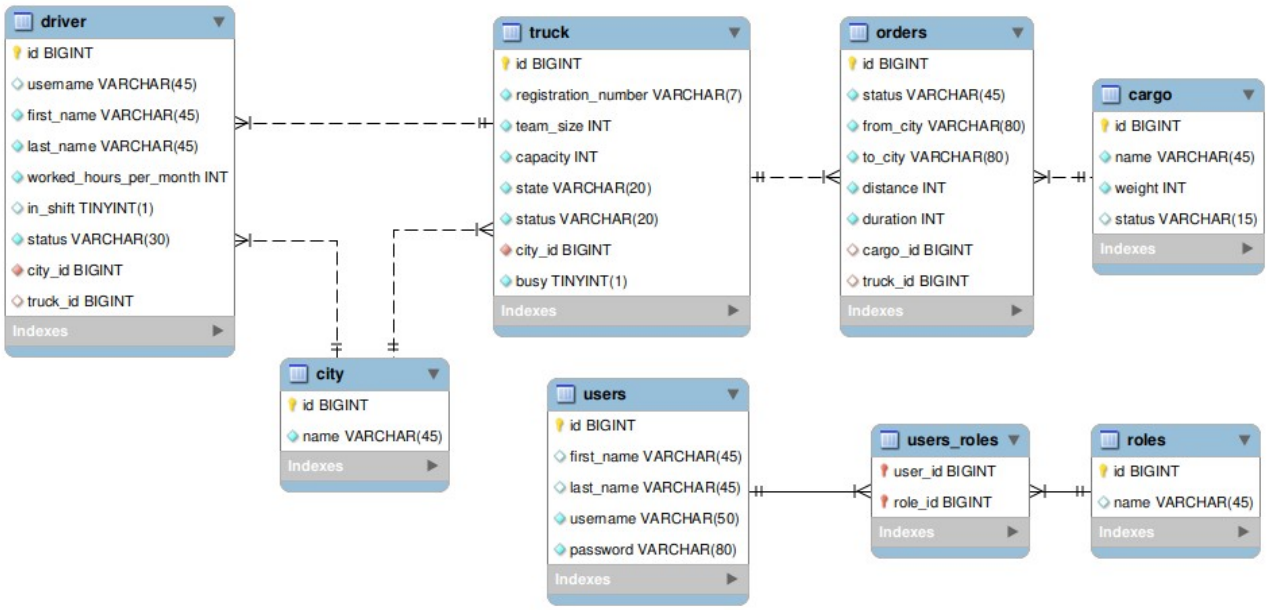

Figure 1 - Database schema

Architecture

The web application based on client-server architecture. On server side used Model-View-Controller (MVC) pattern. That pattern divides the information related program logic into three interconnected elements.

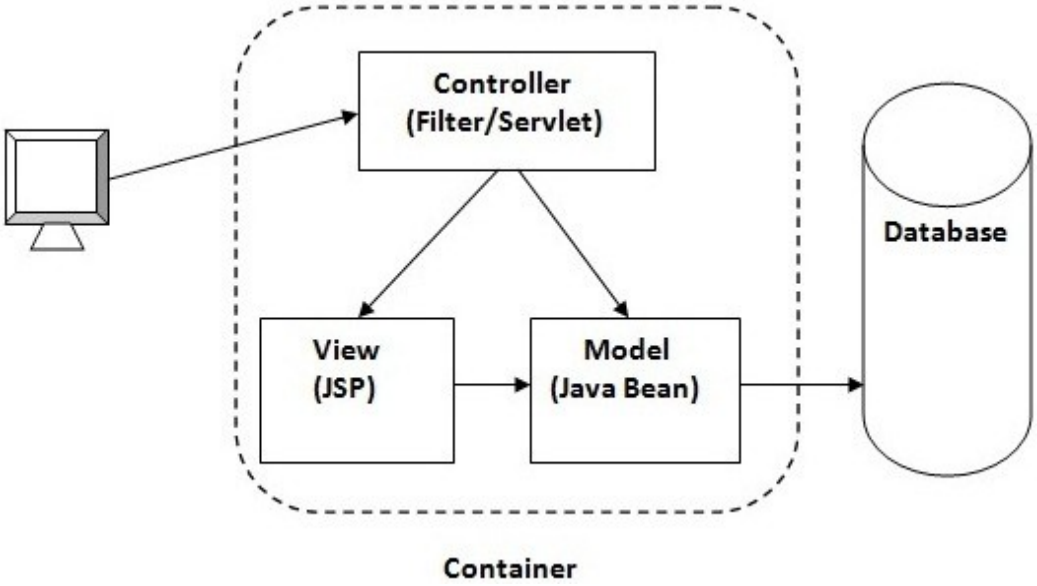

Figure 2 - MVC Pattern

To transfer data from view to controller and vice versa using the DTO pattern.

Layers of application

Model layer

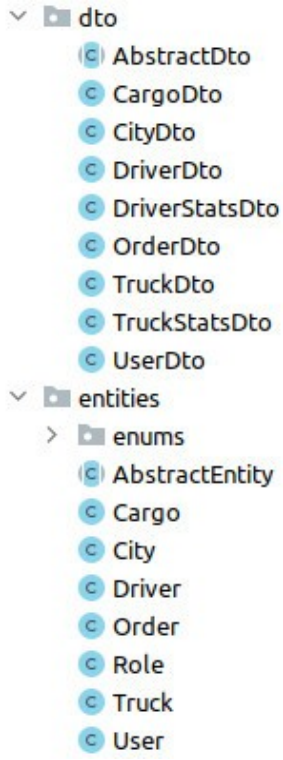

Figure 3 - Model layer

Controller layer

Figure 4 - Controller layer

Service layer

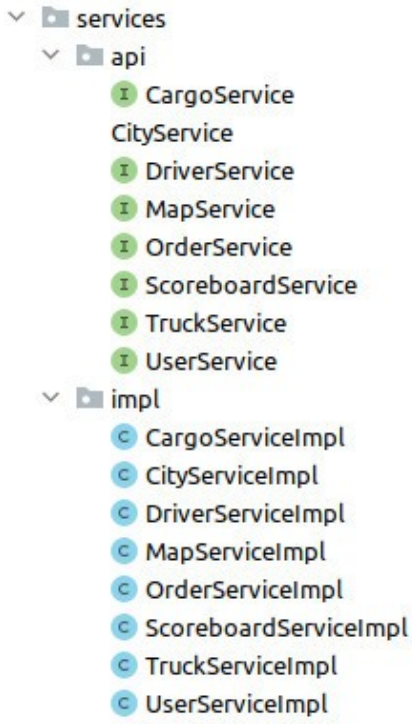

Figure 5 - Service layer

DAO layer

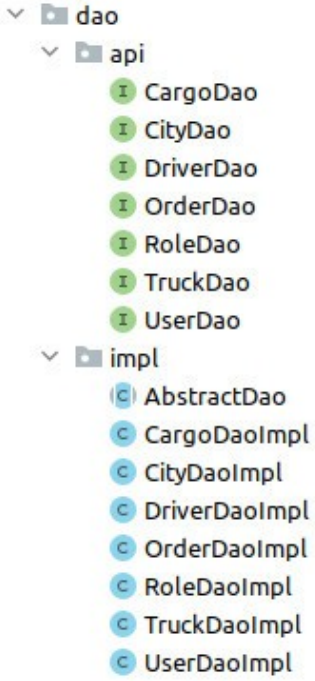

Figure 6 - DAO layer

View layer

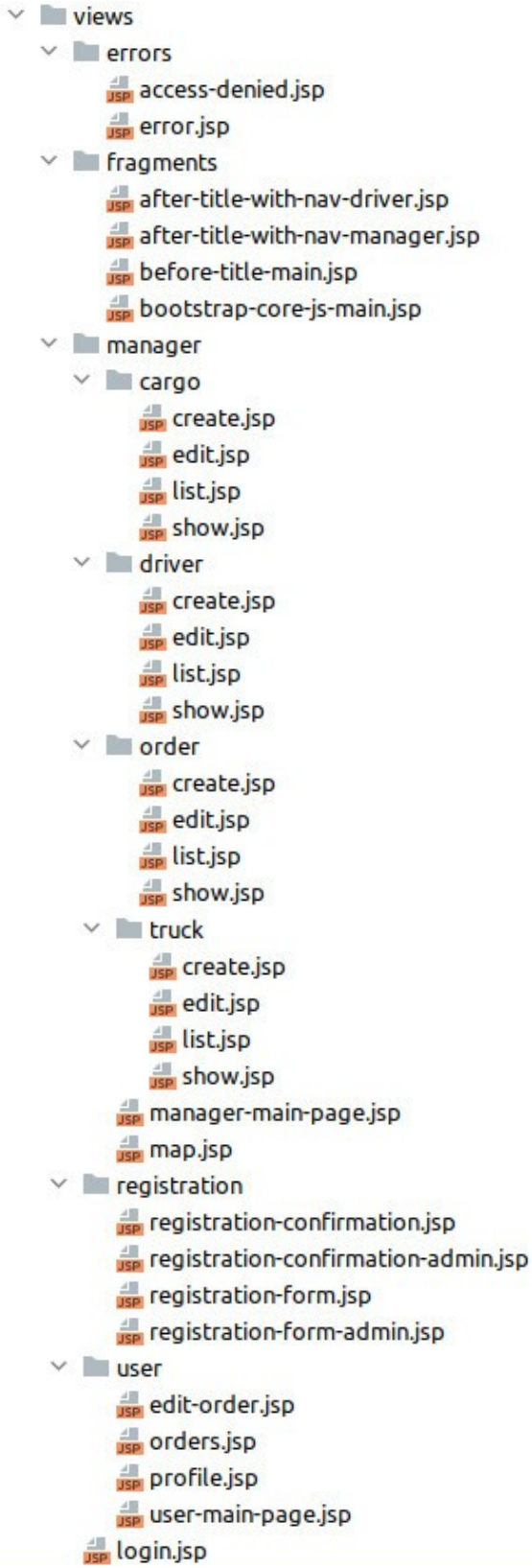

Figure 7 - View layer

User interface

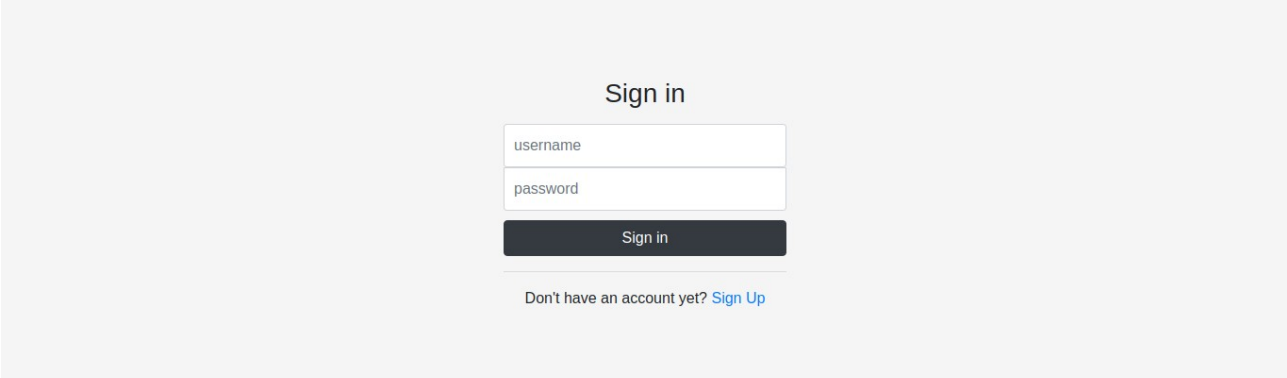

Figure 8 - Login page

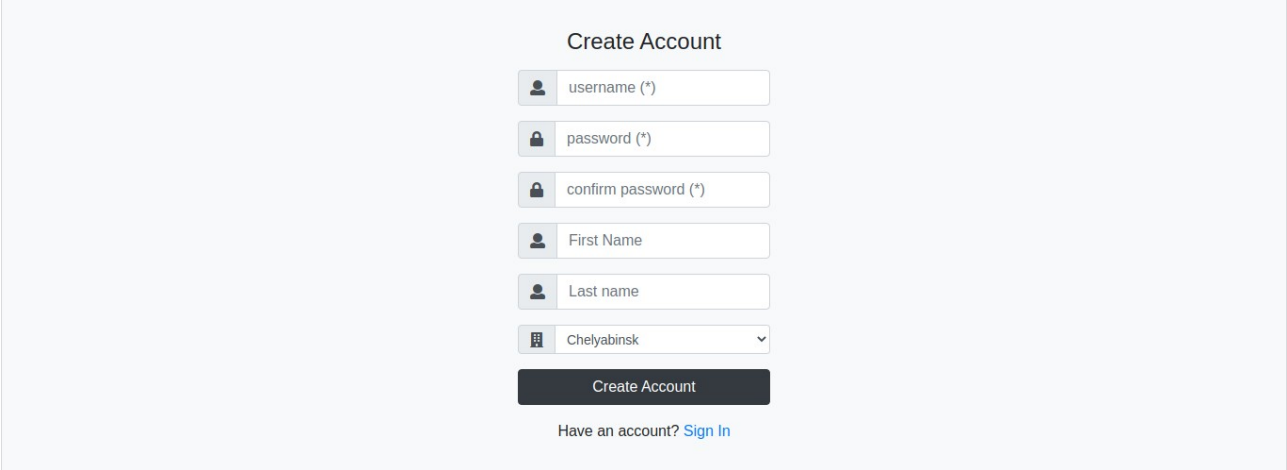

Figure 9 - Registration page

The next pages represent when the user logged with the Admin role:

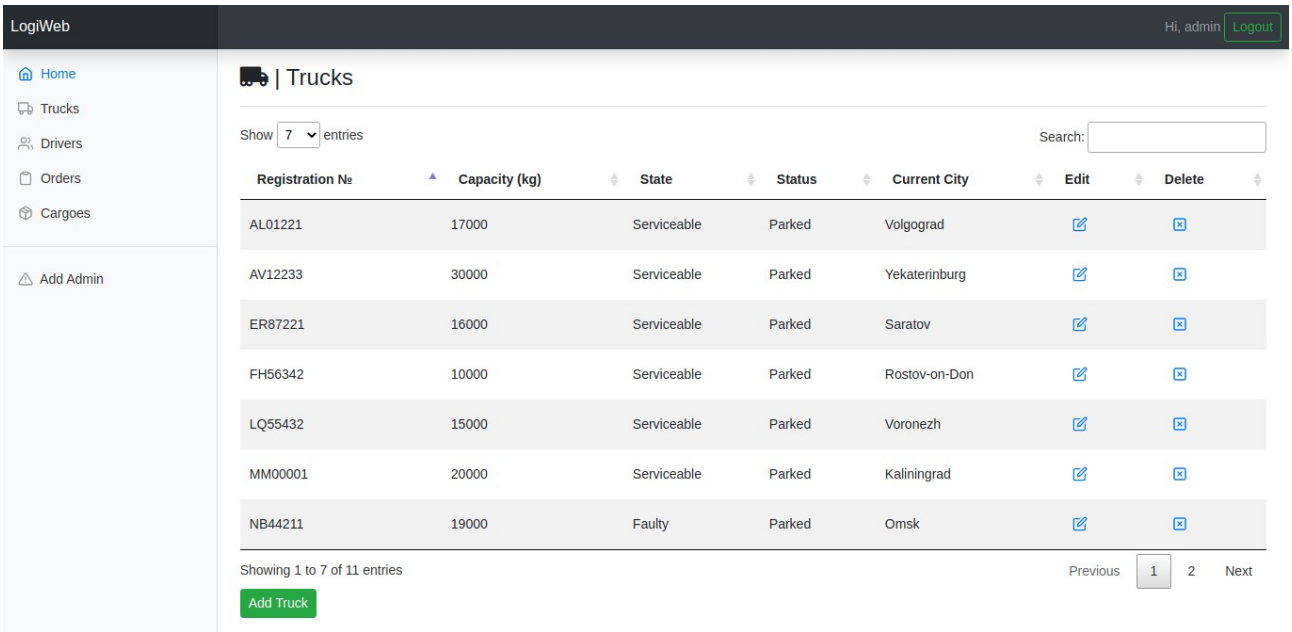

Figure 10 - All trucks page (only Admin)

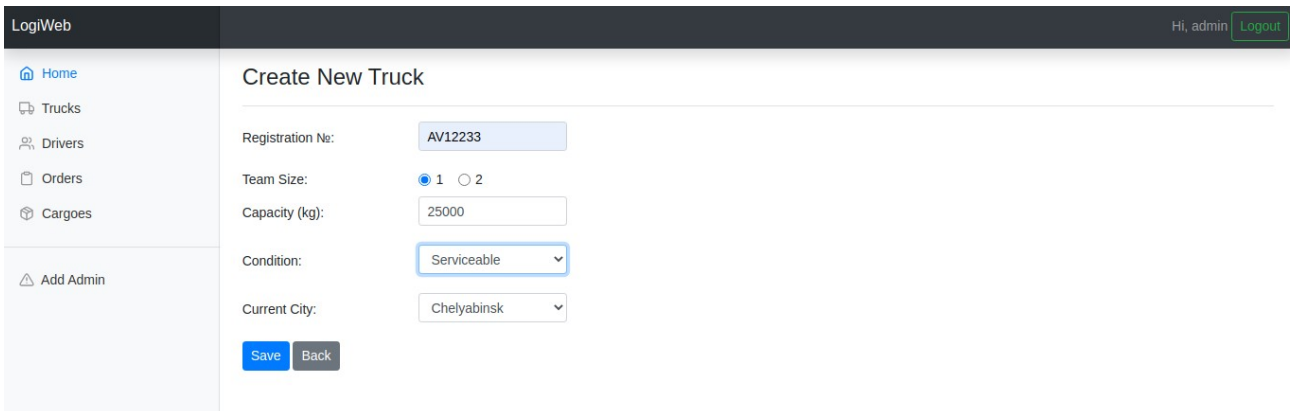

Figure 11 - Create new truck page (only Admin)

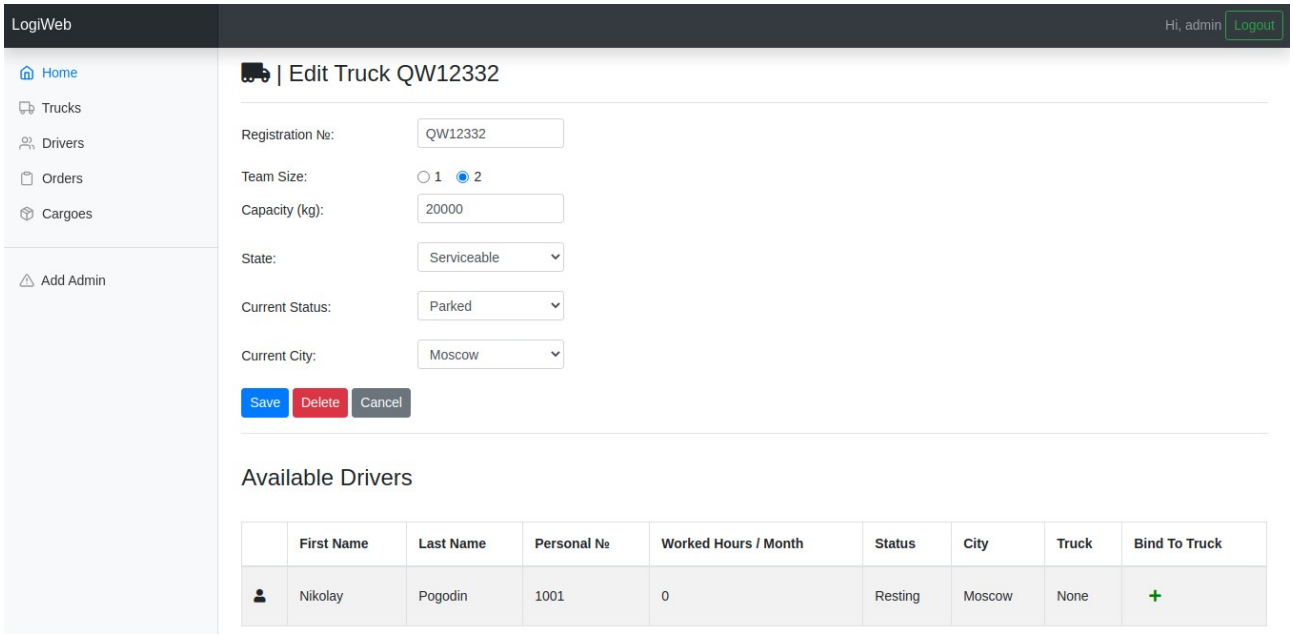

Figure 12 - Edit truck page (only Admin)

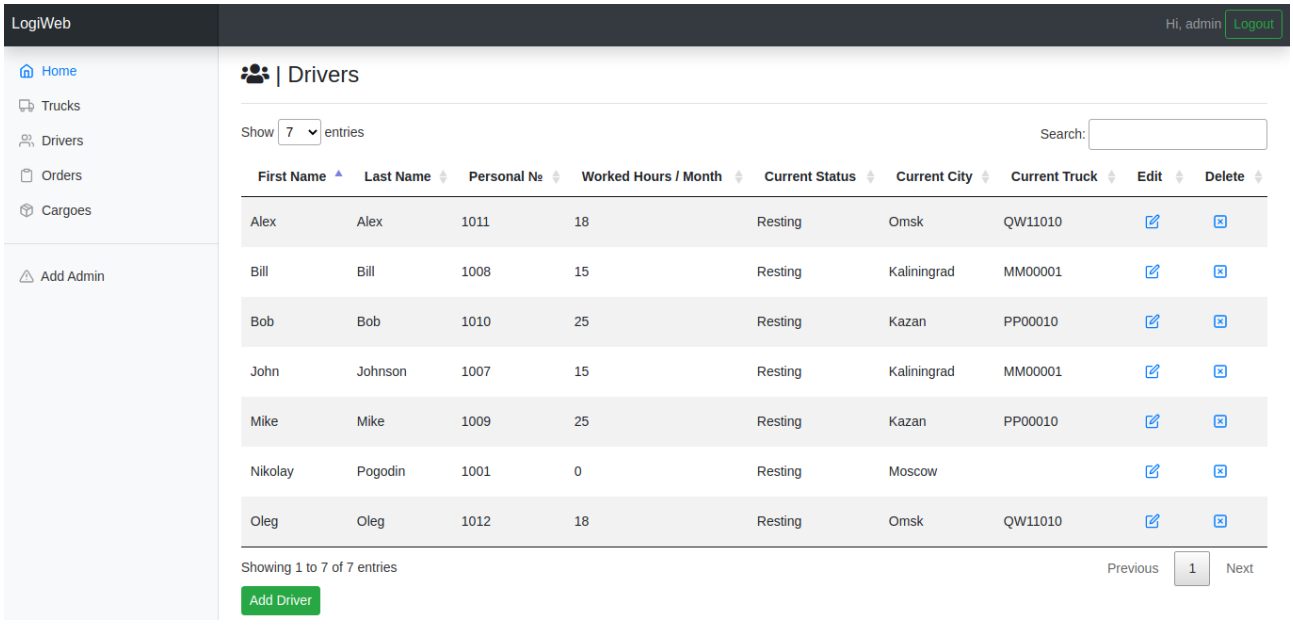

Figure 13 - All drivers page (only Admin)

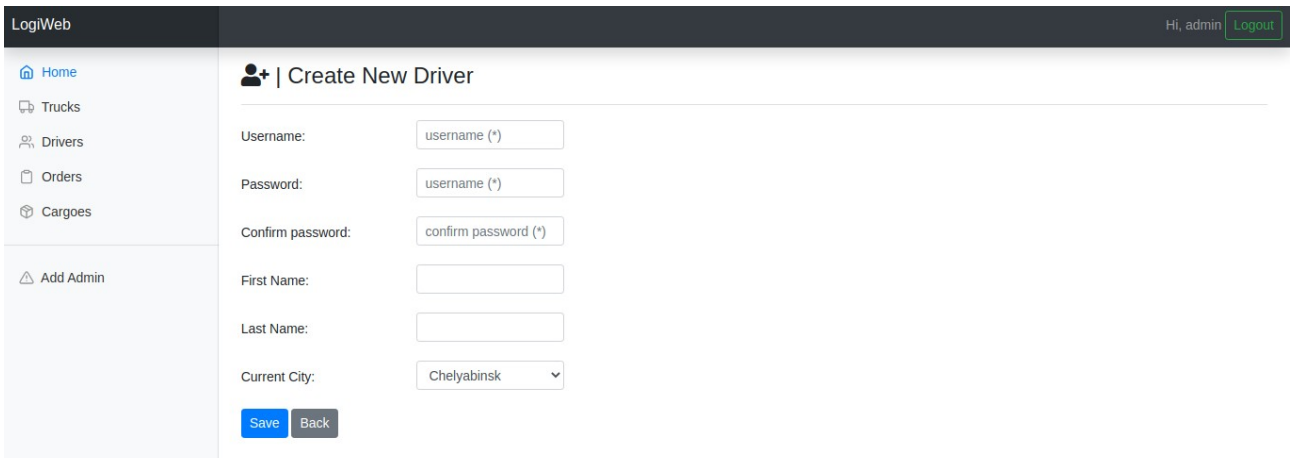

Figure 14 - Create new driver page (only Admin)

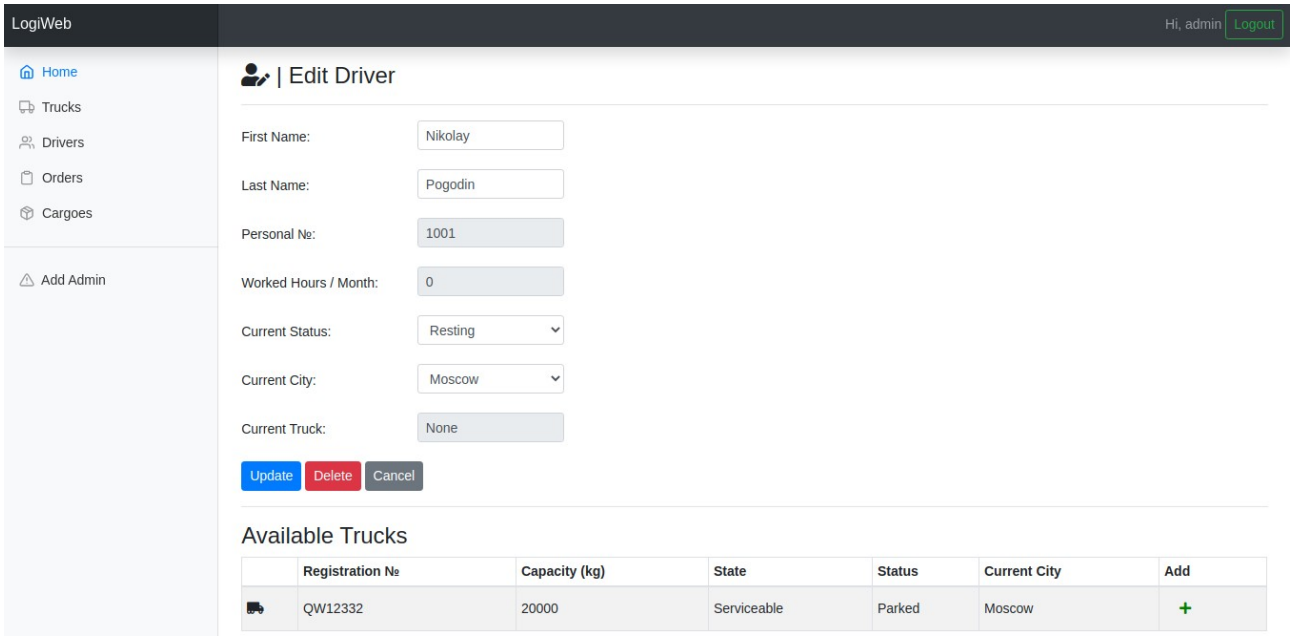

Figure 15 - Edit driver page (only Admin)

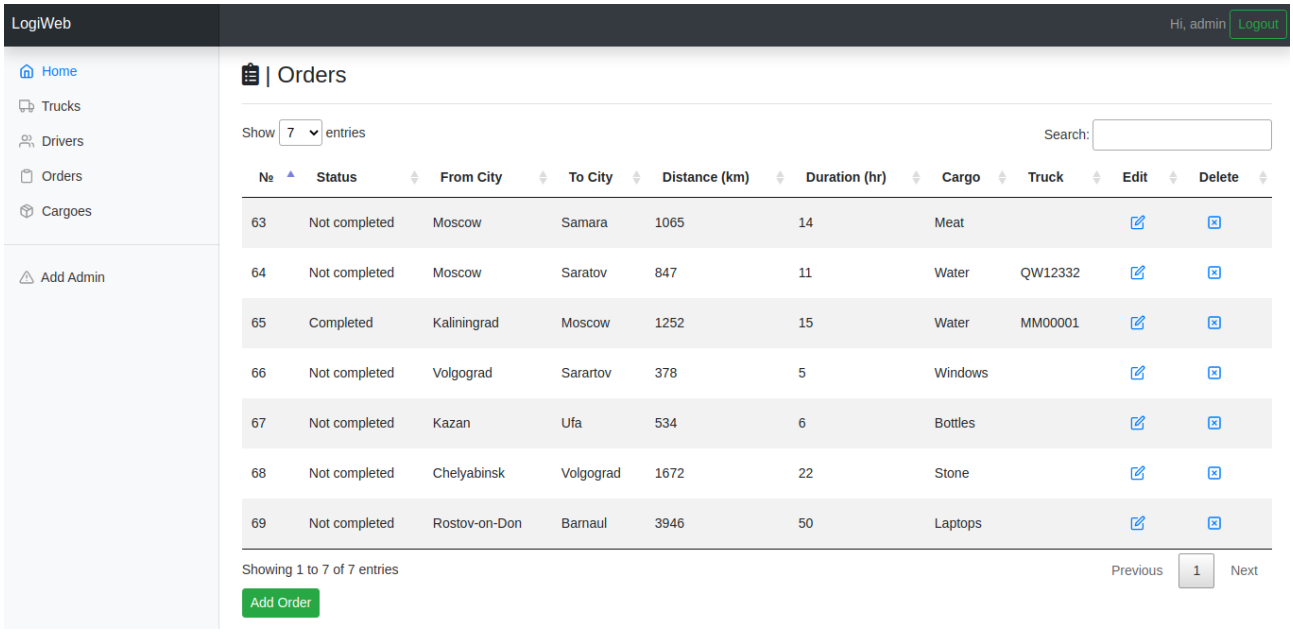

Figure 16 - All orders page (only Admin)

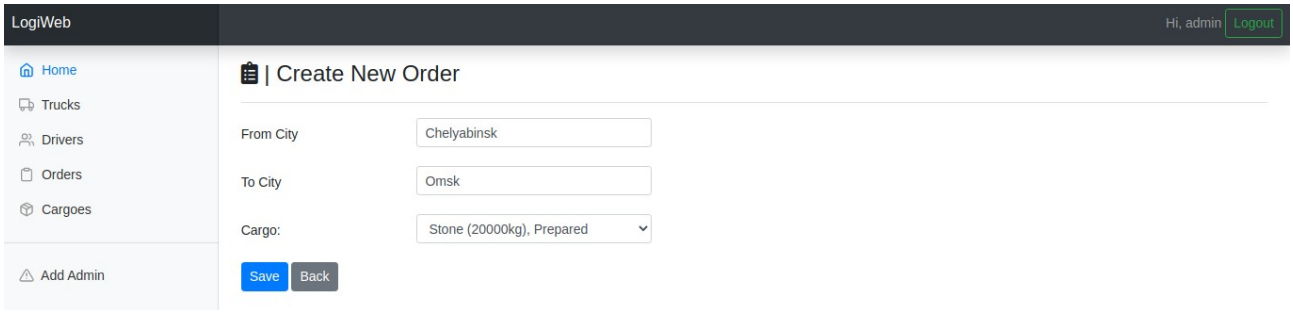

Figure 17 - Create new order page (only Admin)

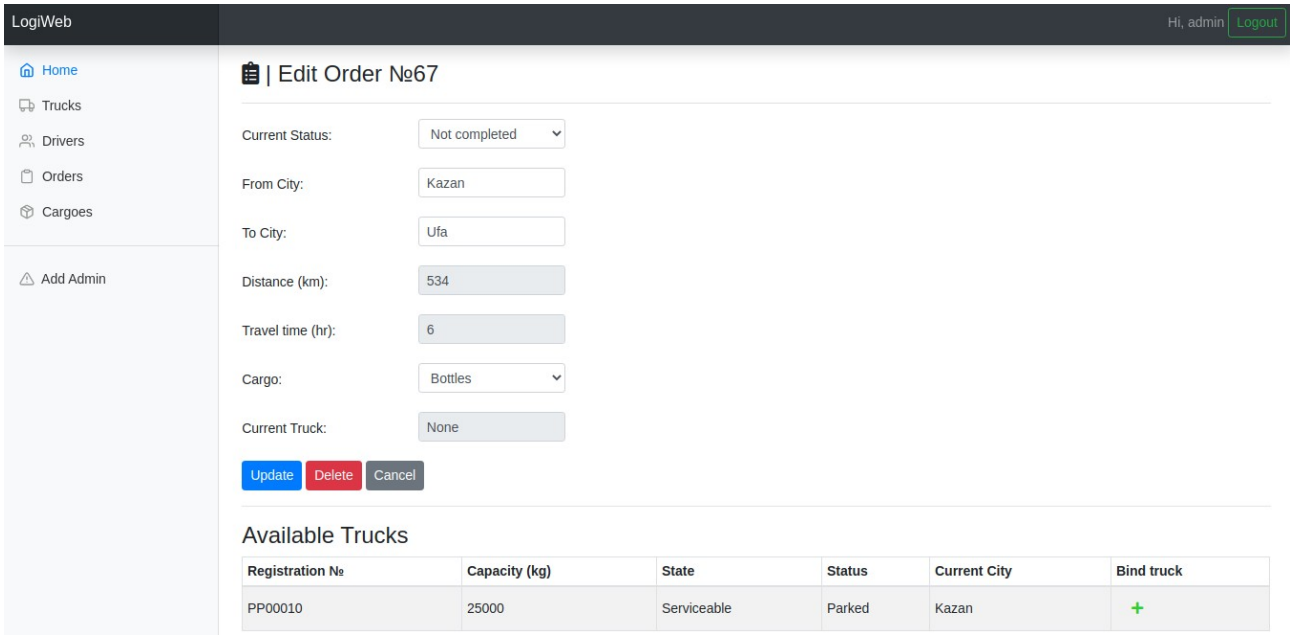

Figure 18 - Edit order page (only Admin)

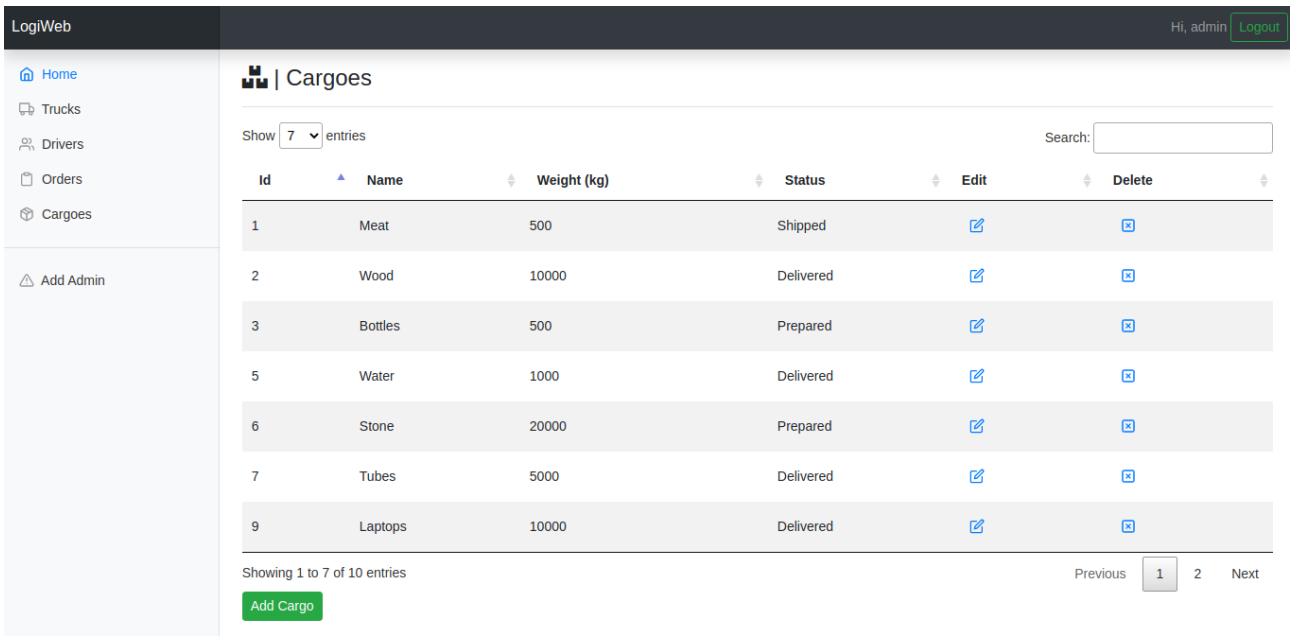

Figure 19 - All cargoes page (only Admin)

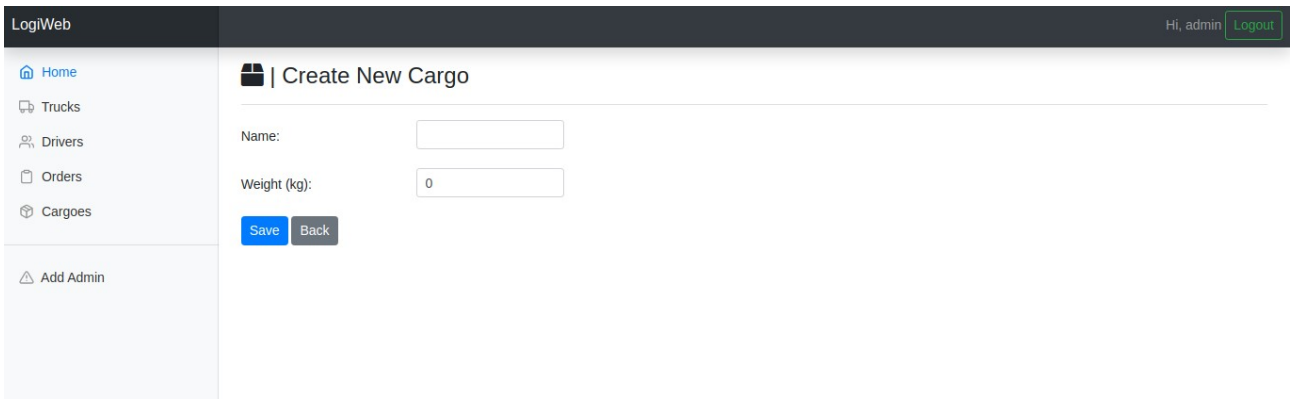

Figure 20 - Create new cargo (only Admin)

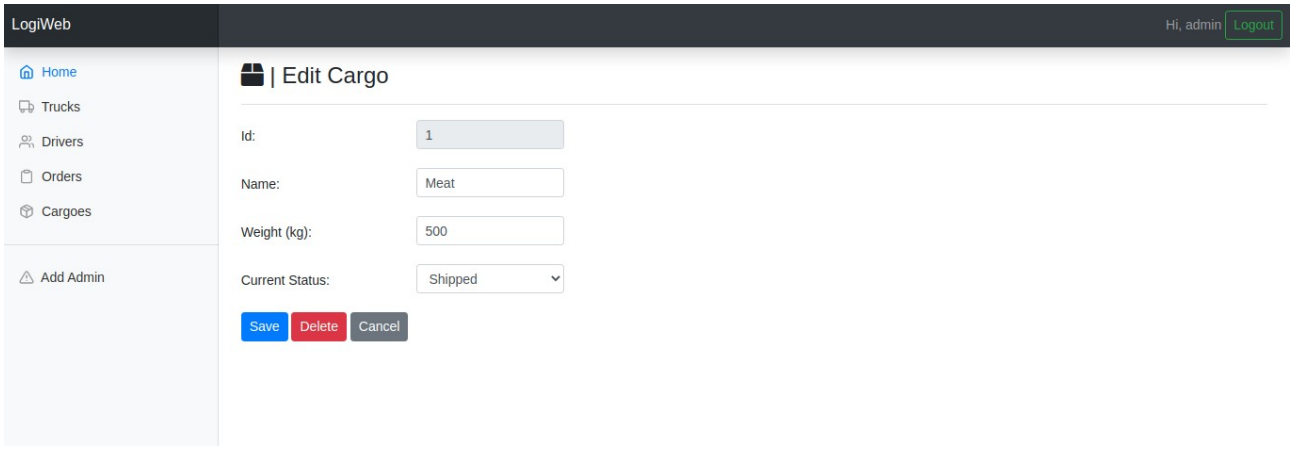

Figure 21 - Edit cargo page (only Admin)

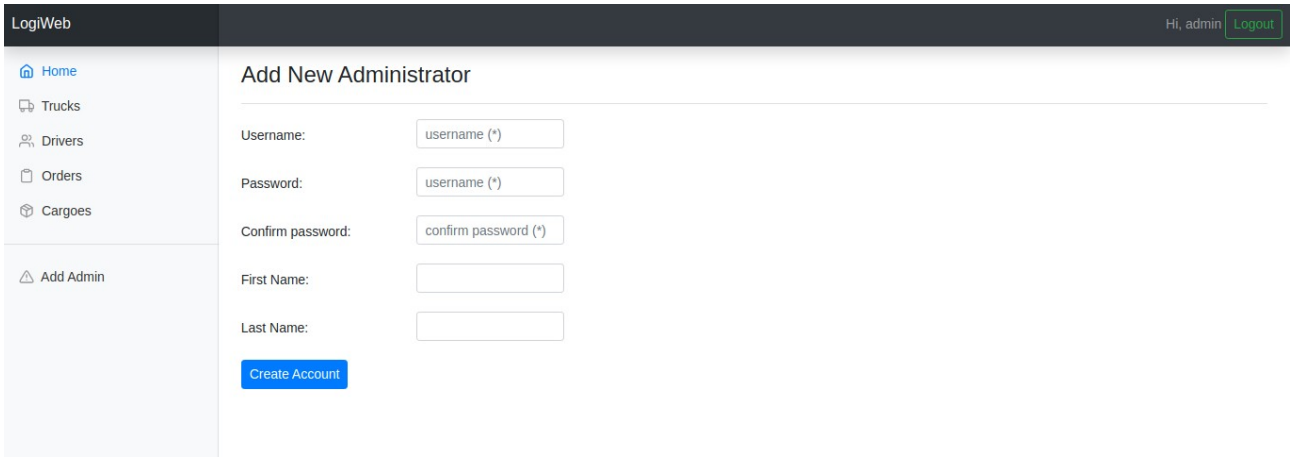

Figure 22 - Create new administrator page (only Admin)

The next pages represent when the user logged with the Driver role:

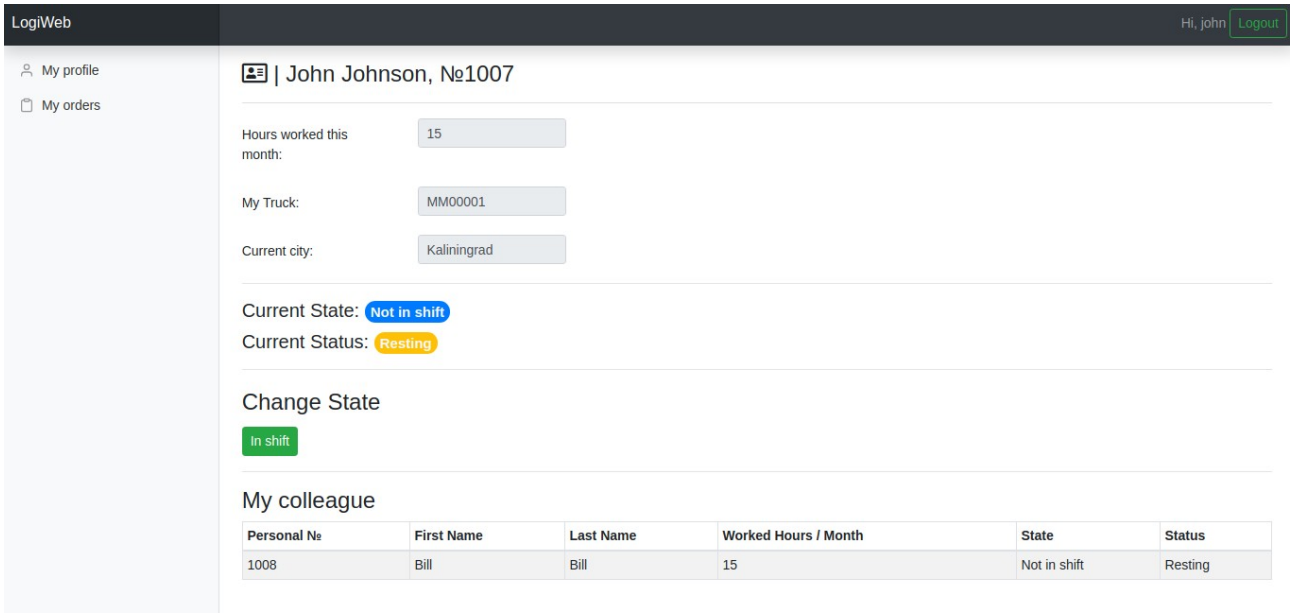

Figure 23 - Driver profile page (only Driver)

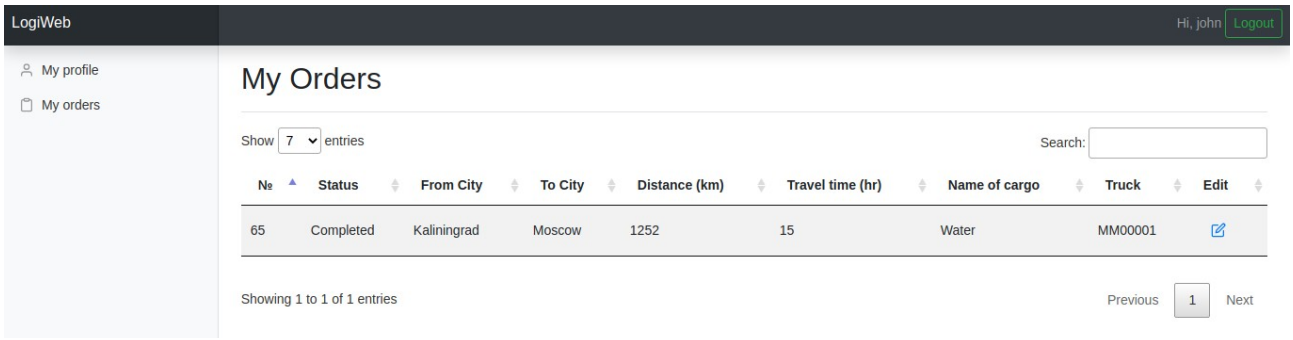

Figure 24 - Orders of driver page (only Driver)

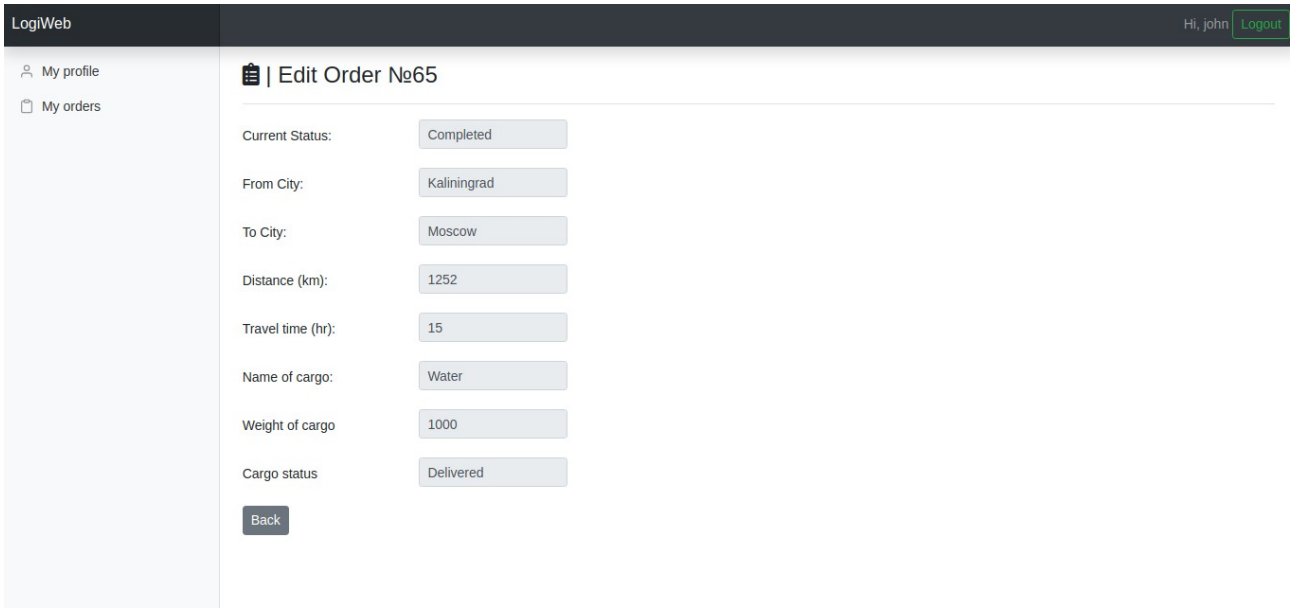

Figure 25 - Edit order (only Driver)

The next page represents the scoreboard with a statistic about trucks and drivers and also the latest orders:

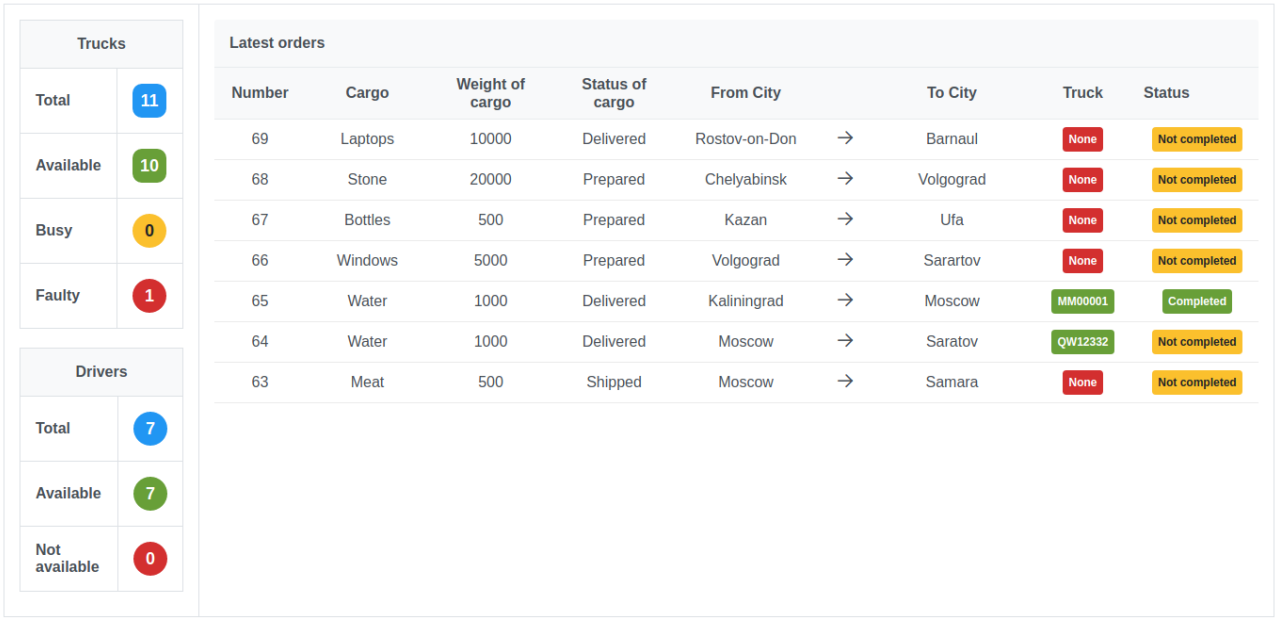

Figure 26 - Scoreboard view

Unit tests

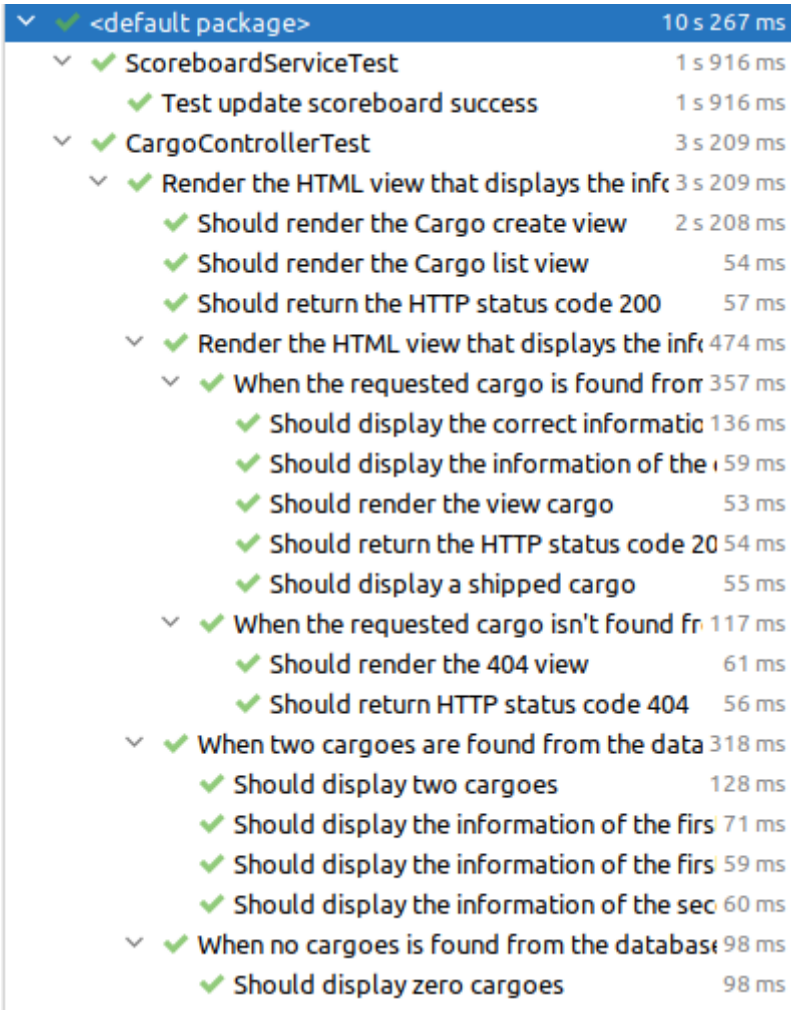

Figure 27 - Small example of successful tests

Service layer coverage is 64% tests

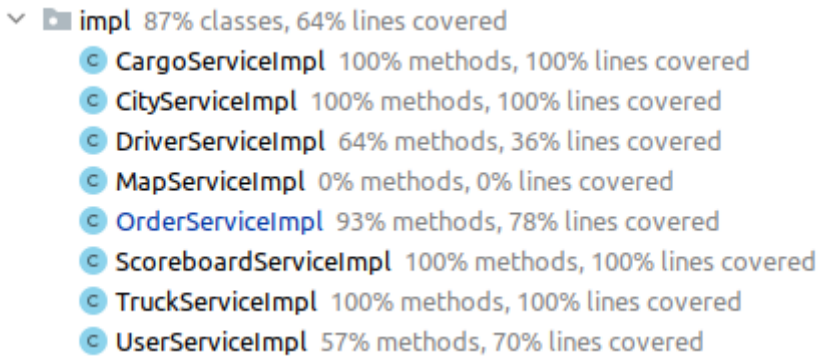

Figure 28 - Tests coverage

Build and deploy applications

1. LogiWeb
   a) Run MySQL Server and create database with tables.
   b) Run Apache Tomcat Server and deploy “LogiWeb.war” to it.

2. Scoreboard for LogiWeb application
   a) Run ActiveMQ message broker
   b) Run WildFly Server and deploy “Scoreboard.war” to it.

Logging

Logging is configured using Log4j. Logs are written to a file.

Small piece of the log file:

```
2021-02-23 12:30:52 INFO  CityServiceImpl:43 - Found city with id = 1
2021-02-23 12:30:53 INFO  CargoServiceImpl:99 - Cargo with id = 1 deleted
2021-02-23 12:30:53 INFO  CargoServiceImpl:88 - Cargo with id = 1 updated
2021-02-23 12:30:53 INFO  CargoServiceImpl:43 - Found cargo with id = 1
2021-02-23 12:30:53 INFO  CargoServiceImpl:72 - Cargo with id = 1 created
2021-02-23 12:30:53 INFO  TruckServiceImpl:92 - Truck with id = 1 created
2021-02-23 12:30:53 INFO  TruckServiceImpl:59 - Found the truck with id = 1
2021-02-23 12:30:53 INFO  TruckServiceImpl:124 - Truck with id = 1 deleted
2021-02-23 12:30:53 INFO  TruckServiceImpl:111 - Truck with id = 1 updated
2021-02-23 12:30:54 INFO  DriverServiceImpl:166 - Available trucks found
2021-02-23 12:30:54 INFO  DriverServiceImpl:134 - Driver with id = 6 deleted
2021-02-23 12:30:54 INFO  DriverServiceImpl:121 - Driver with id = 6 updated
2021-02-23 12:30:54 INFO  DriverServiceImpl:72 - Found driver with id = 6
2021-02-23 12:30:54 INFO  DriverServiceImpl:103 - Driver with id = 6 created
2021-02-23 12:30:55 INFO  OrderServiceImpl:180 - Order with id = 2 updated
2021-02-23 12:30:55 INFO  OrderServiceImpl:162 - Order with id =  created
2021-02-23 12:30:55 INFO  OrderServiceImpl:193 - Order with id = 1 deleted
2021-02-23 12:30:55 INFO  OrderServiceImpl:62 - Found order with id = 1
```

Figure 29 - Example of the log file

The statistic about application code

Code statistic provided by SonarQube:

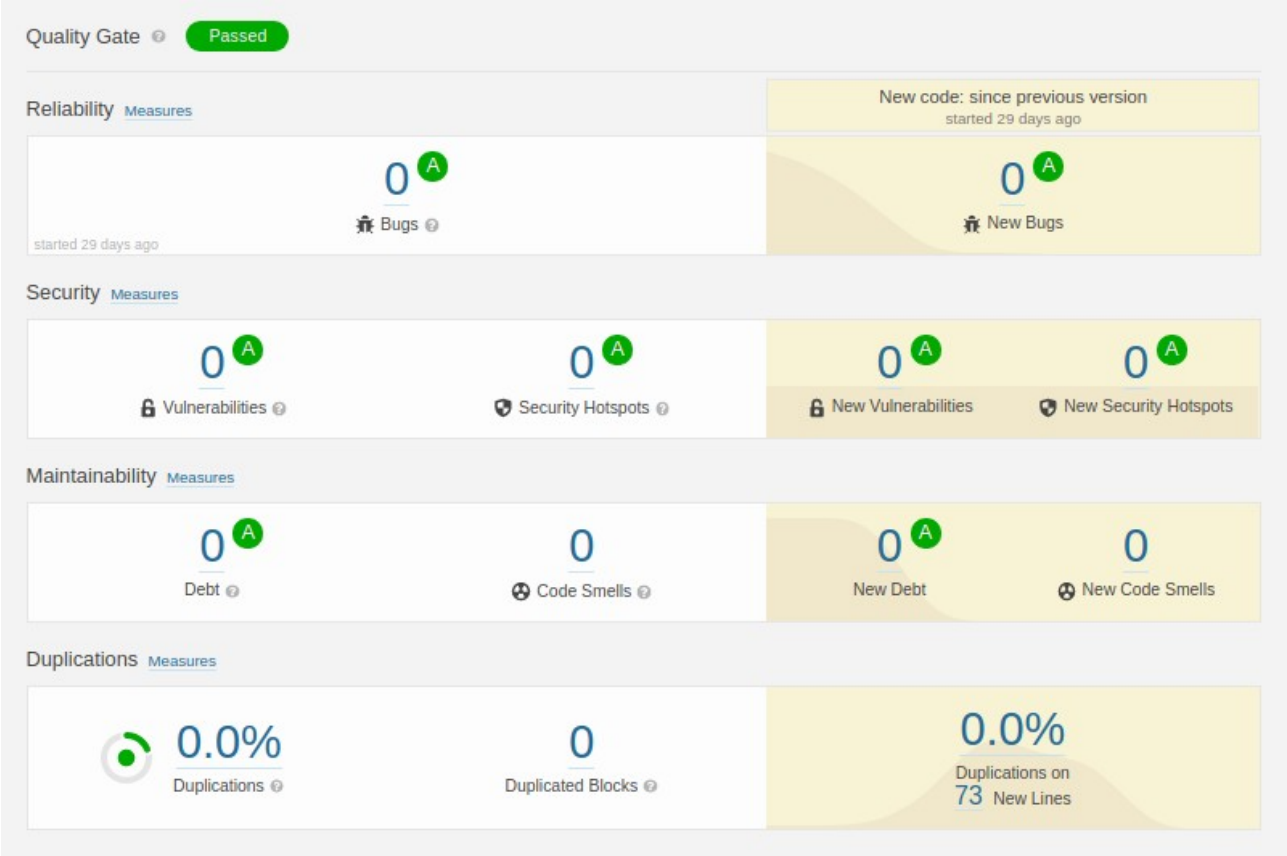

Figure 30 - SonarQube statistic

Planned improvements

1. Full implementation Google Maps API (Place Autocomplete and show map).
2. Improve performance (implement pagination with Hibernate)