Technical solution description

# Logiweb project.

Author:

Alexander Usachev

Saint-Petersburg, 2021

**Content.**

[Logiweb project. 1](#_Toc75105500)

[1. Task. 3](#_Toc75105501)

[2. Project goals. 5](#_Toc75105502)

[3. Application description. 6](#_Toc75105503)

[4. Used Technologies. 7](#_Toc75105504)

[5. Database model. 8](#_Toc75105505)

[6. System infrastructure. 9](#_Toc75105506)

[7. System architecture. 10](#_Toc75105507)

[8. Additional features. 10](#_Toc75105508)

[9. UI. 11](#_Toc75105509)

[10. Code quality. 12](#_Toc75105510)

[Tests. 16](#_Toc75105511)

[Sonar report: 17](#_Toc75105512)

[Logging. 18](#_Toc75105513)

[11. Future improvement. 18](#_Toc75105514)

## 1. Task.

**1. Application (Logiweb)**

To develop an application that simulates the operation of the information system of a certain company that carries out cargo transportation.

User cases:

* For company employees (via the UI interface):
  + View the list, add, edit, and delete trucks and drivers;
  + View the list and add new orders, making sure that:

1. All loaded cargoes must be unloaded somewhere;
2. All unloaded cargoes must be loaded somewhere;
   * View the status of orders and cargoes;
   * Displays a list of trucks that are suitable for the delivery of the order, if:
3. The truck is in good condition;
4. The truck is suitable for capacity (taking into account the loading/unloading of goods in cities along the route);
5. The truck is not currently executing any orders;
   * Selection and assignment of drivers based on the size of the shift of the truck used and the approximate travel time (calculated from the map of cities and waypoints):
6. The time limit for a month (176 hours) for each of the drivers in the shift will not be exceeded during the execution of this order (also take into account the change of months during the order);
7. The driver is not currently fulfilling other orders;
8. When assigned, the driver is located in the same city as the truck.

* For drivers (via the UI interface):
  + Display the following information:

1. Driver's personal number
2. Personal numbers of the co-driver (s)
3. Truck registration number
4. Order number
5. List of waypoints
   * Change the actual working hours and order status:
6. The driver started / finished the shift
7. The driver received / unloaded the cargo (change the order status)

* Uploaded it
* Unloaded it

1. The driver changed the status:

* Driving
* Second driver
* Loading and unloading operations
* Recreation

**2. Application (Display)**

Implement a separate client application such as an electronic scoreboard, which will show full information about the latest orders (at least 10) their number will depend on your UI.

The same screen should display summary information on drivers and trucks for the current month. How many drivers are there in total, how many are available/unavailable. How many trucks are there in total, how many are available/occupied to order/faulty.

The data must be loaded at startup and stored on the client side. Data is reloaded when a notification is received from the server.

## 2. Project goals.

* The robust, useful and reliable system.
* Cohesive data model.
* User-friendly interface.
* Cohesive data model.
* Separate access to different system`s part.

## 3. Application description.

Web-application has two type of user: admins and drivers.

User functionality is described in the task.

Two applications communicate by JMS sending message when something updating.

There is an authentication mechanism in system that control access to portal. Each user in application has access level that display what information he could get and what couldn`t.

Data of users and their options store in reliable database.

## 4. Used Technologies.

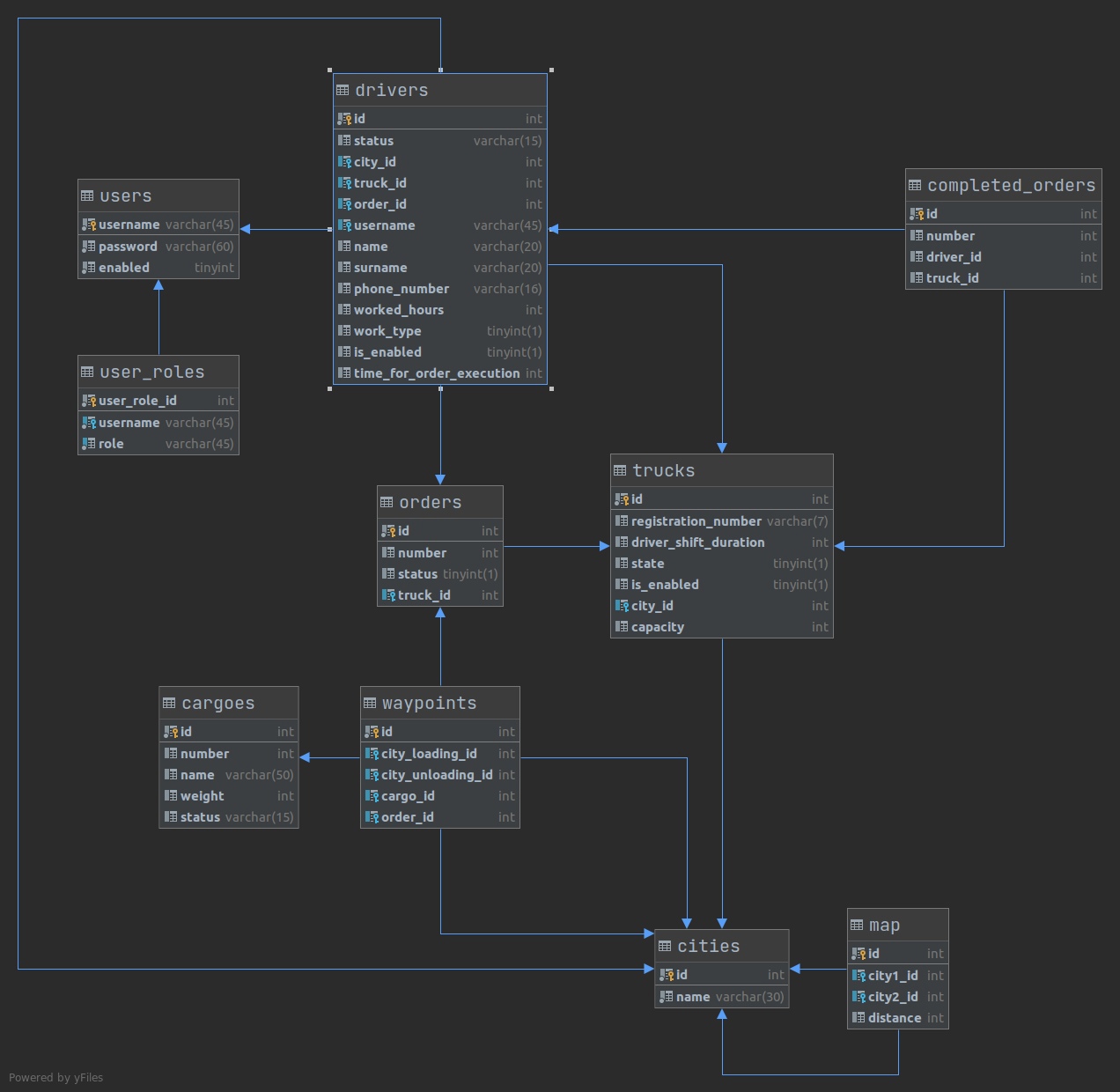
Instruments:

* IDE - IntelliJ IDEA
* Maven

Technologies:

* ActiveMQ 5.7
* Ajax
* Bootstrap 3.3.7
* CI/CD – Travis CI
* DB - MySQL 5.7
* EJB 3
* Java 8
* Javascript
* Jquery
* JPA 2.0
* JSF 2.2.13
* JSP 2.1
* Junit 4.12
* Log4j 1.2.17 1
* Mockito 1.10.19
* Primefaces 5.3
* REST
* Selenium 3.4
* SonarQube 5.1
* Spring 4.3.9
* Spring Security 4.2.3
* WildFly 10

## 5. Database model.



## 6. System infrastructure.

* Front-end (browser presentation level):

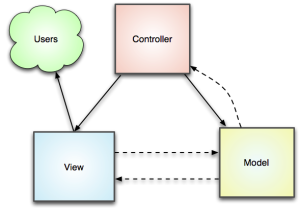
* 1. Web-page structure - HTML
  2. Page-design - CSS
  3. Dynamic content - JavaScript, JQuery, Ajax.

* Back-end (server based level):
  1. Application server - WildFly
  2. Database - MySQL
  3. Server logic - Spring Framework

* Client statistic application:
  1. Web-pages - JSF
  2. JMS - MQ
  3. Application server - WildFly
  4. Server logic - EJB
  5. WS - REST

## 7. System architecture.

Architecture of server-based part presented by MVC - design pattern.



## 8. Additional features.

1. CI/CD

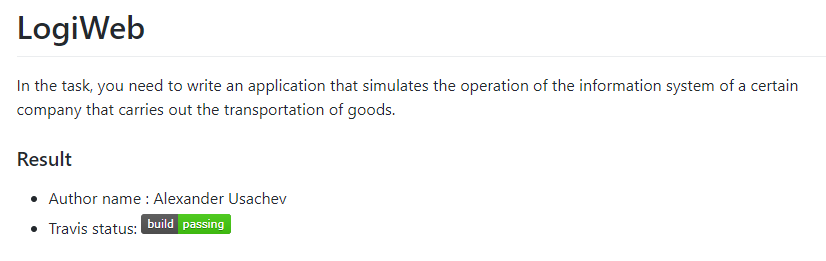
After commit on github, travis automatically building the project and we can see is it correct (Pic. #1).

1. Sonar.

We can see the statistic for the quality of writing code for the future improvements.

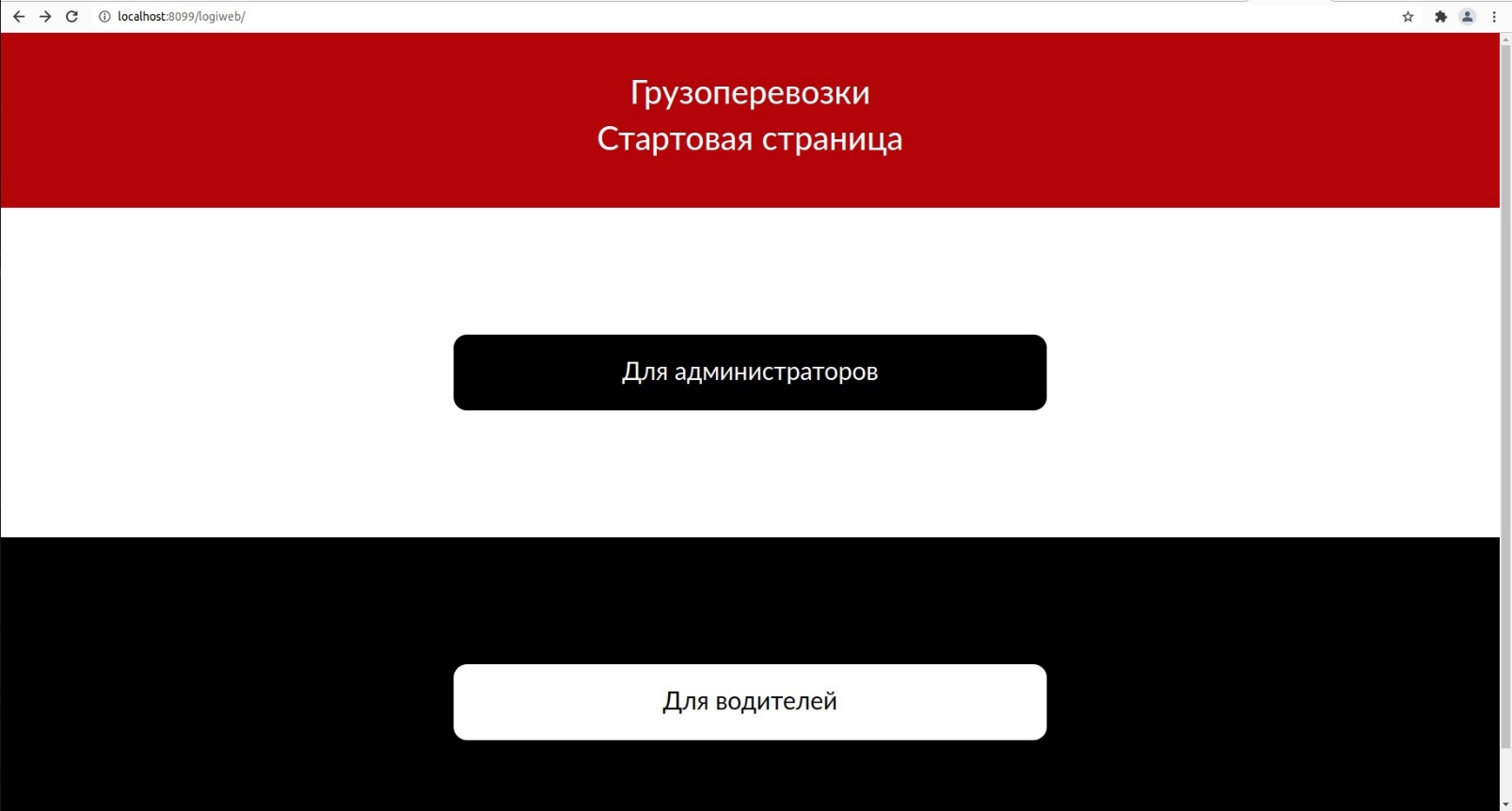
1. Selenium auto-testing.

Tests login page for correct and wrong data.

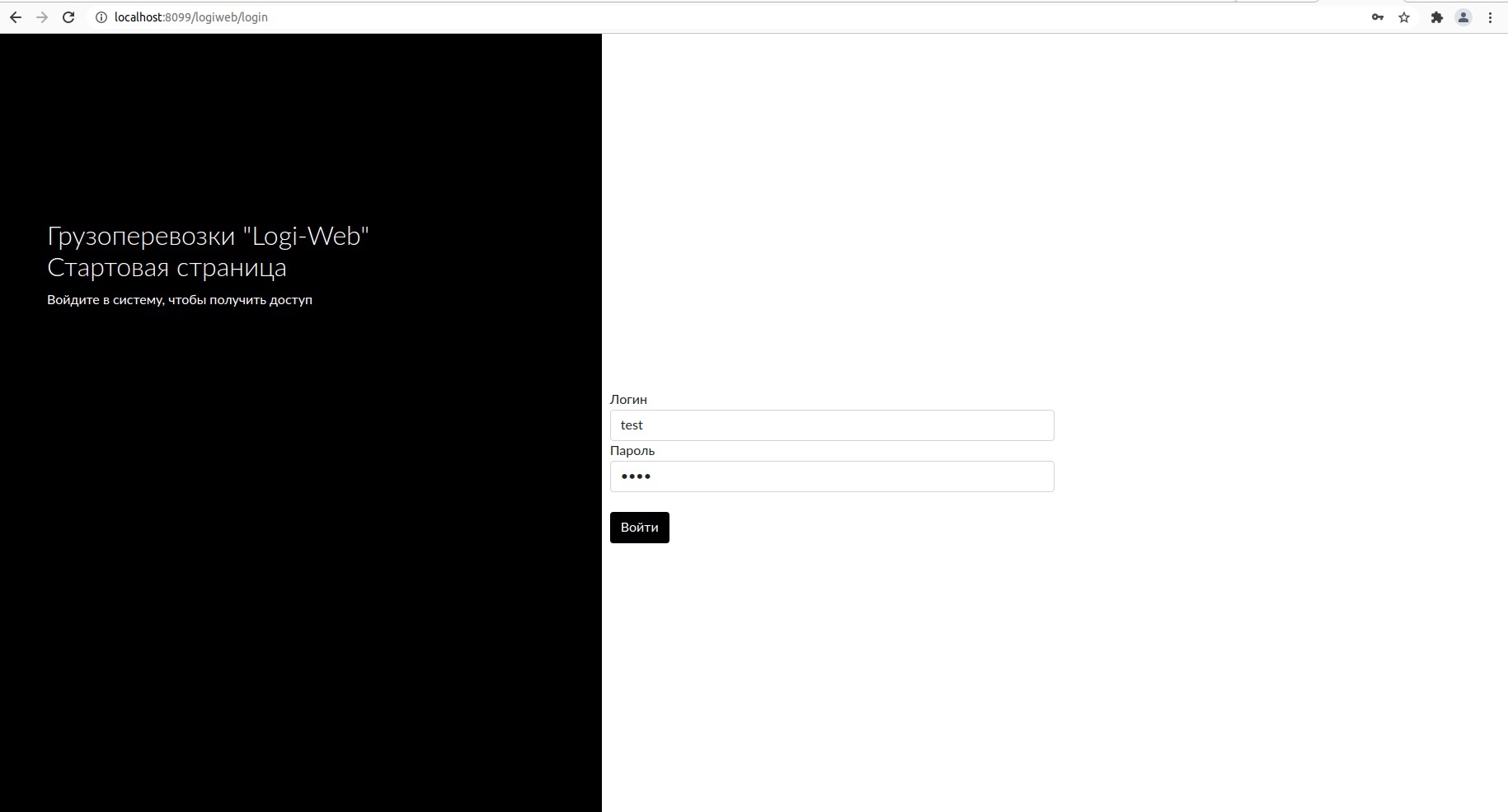


*Picture 1. – Travis build status on GitHub*

## 9. UI.



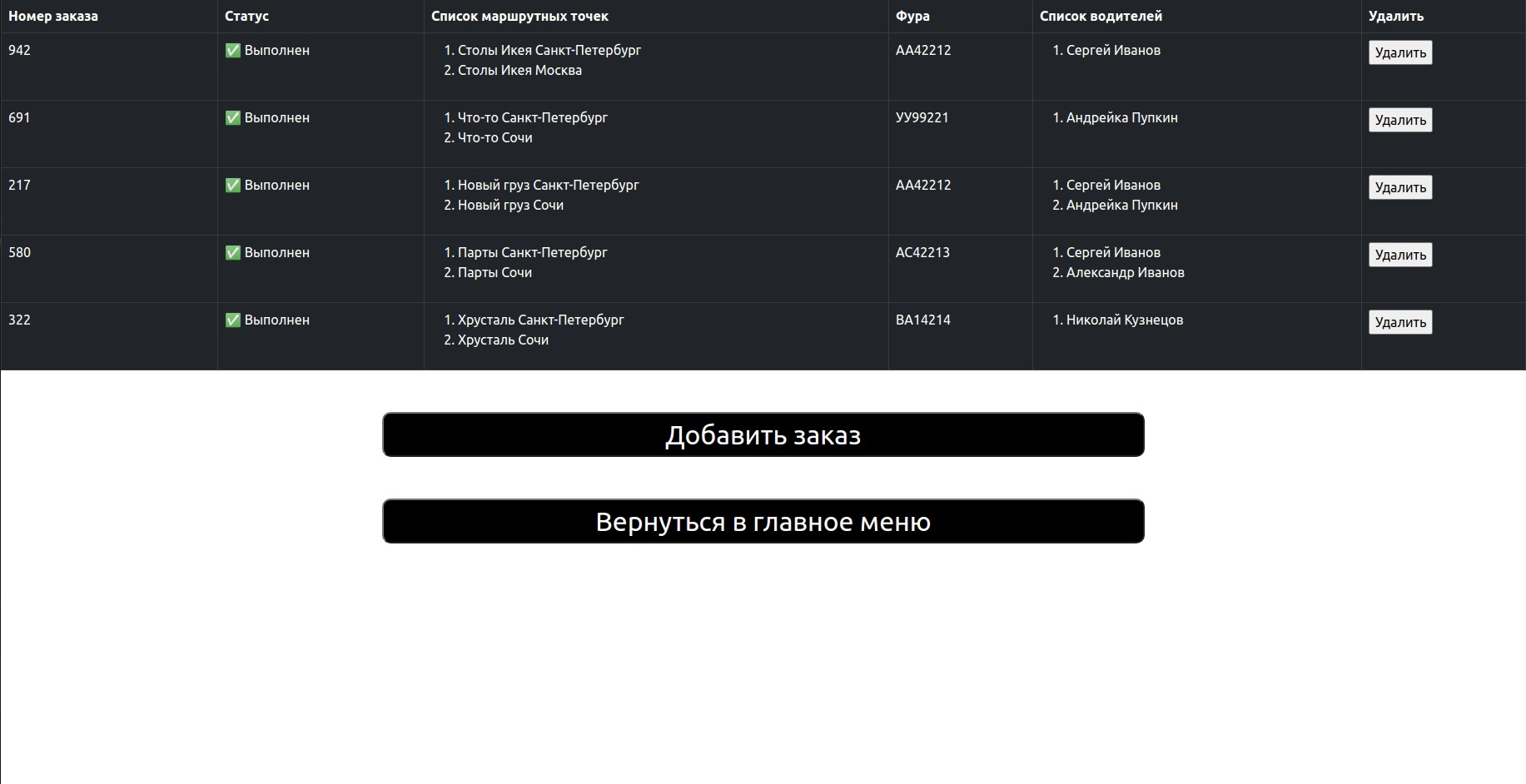
Picture 2. – Start page



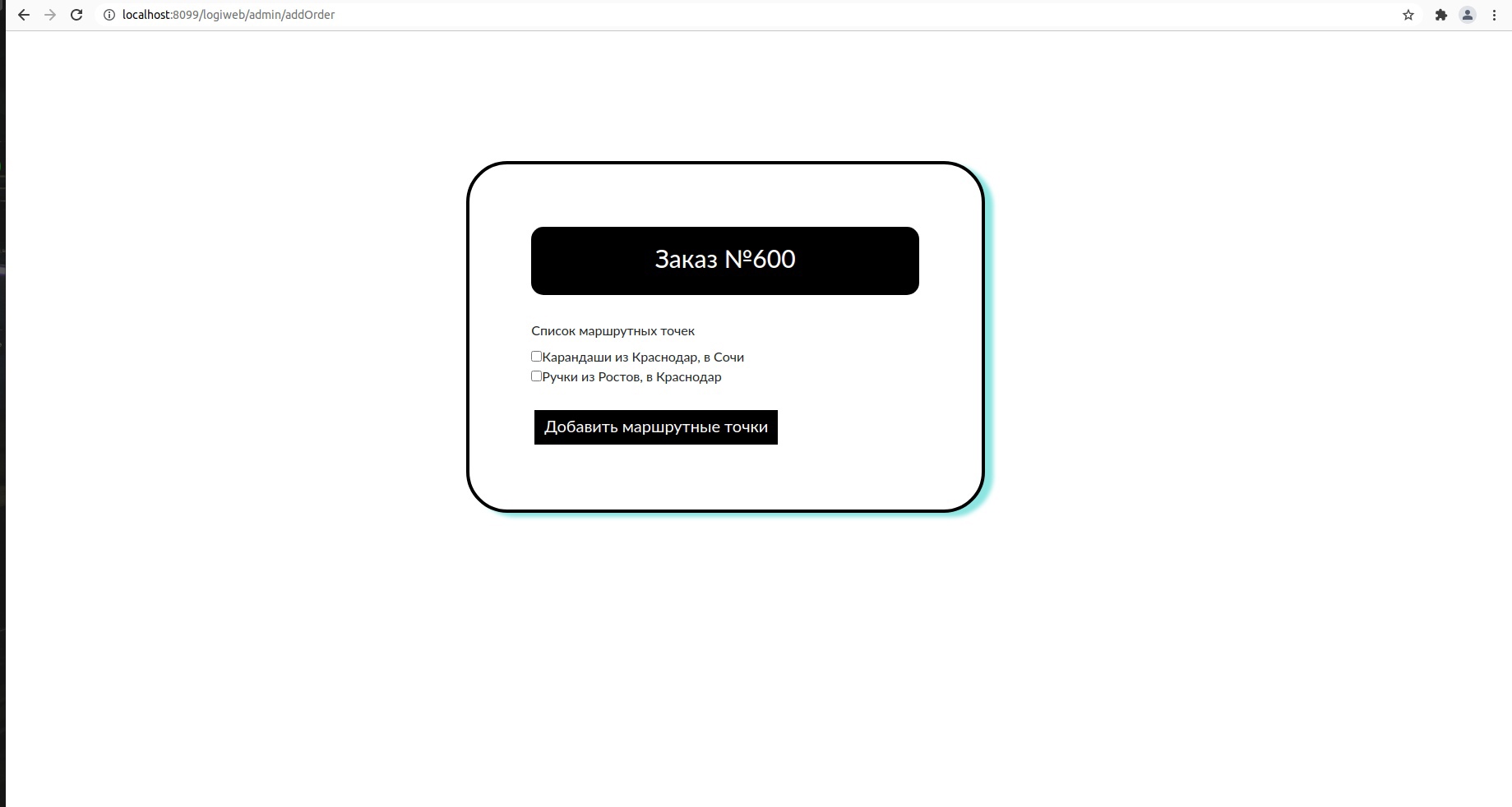
Picture 3. – Login page



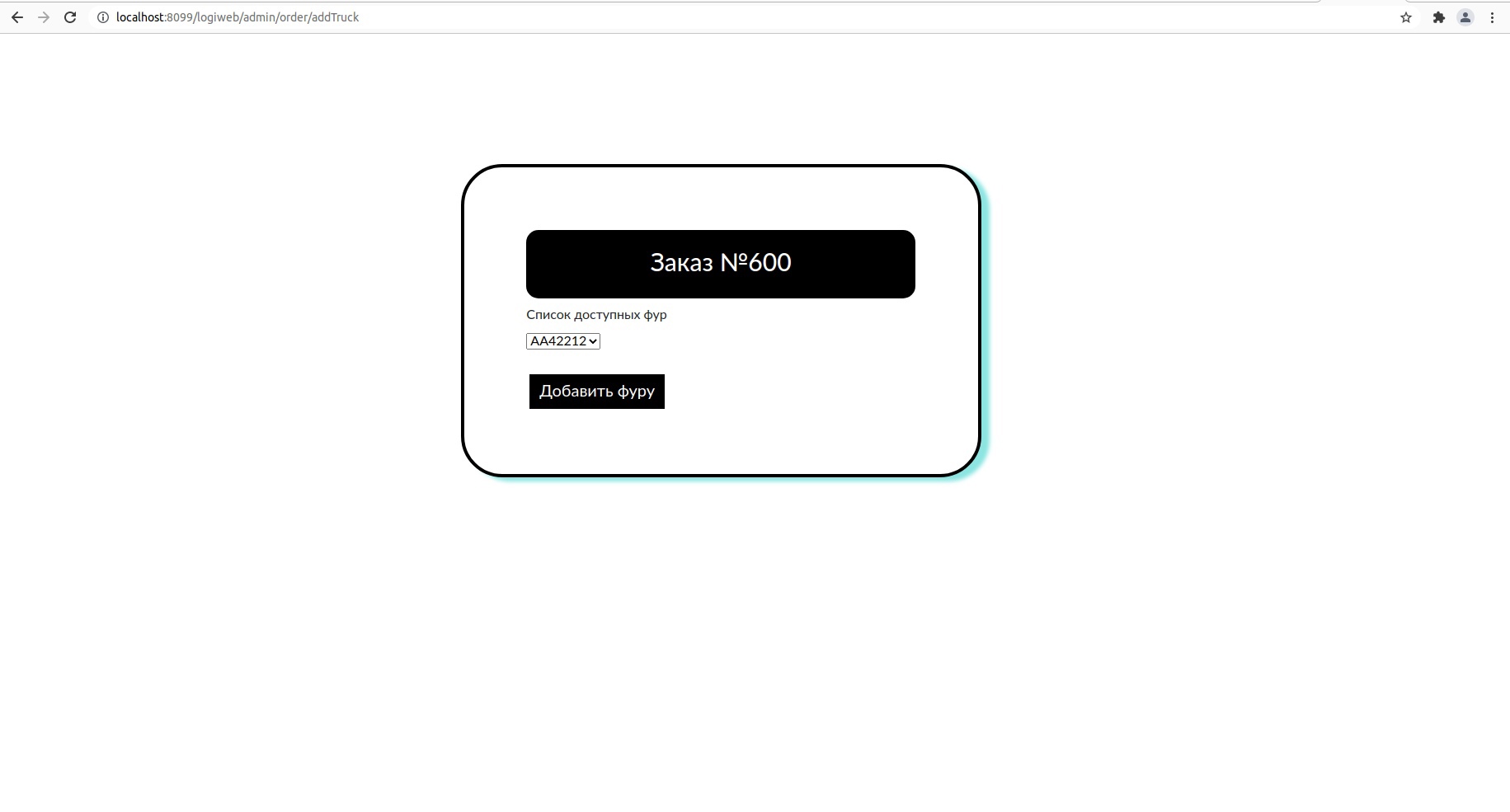
Picture 4. – Admin menu



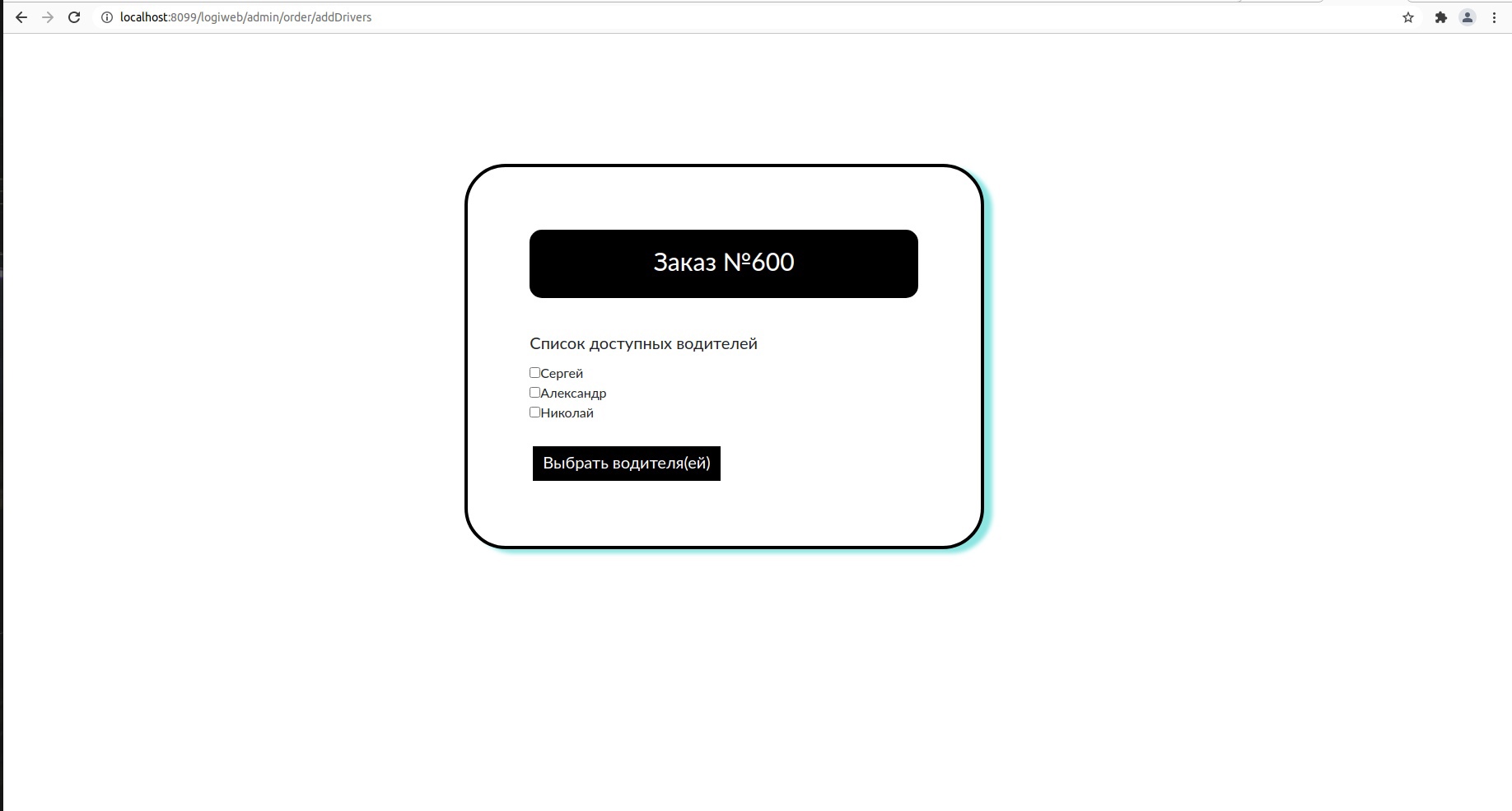
Picture 5. – Viewing of all orders



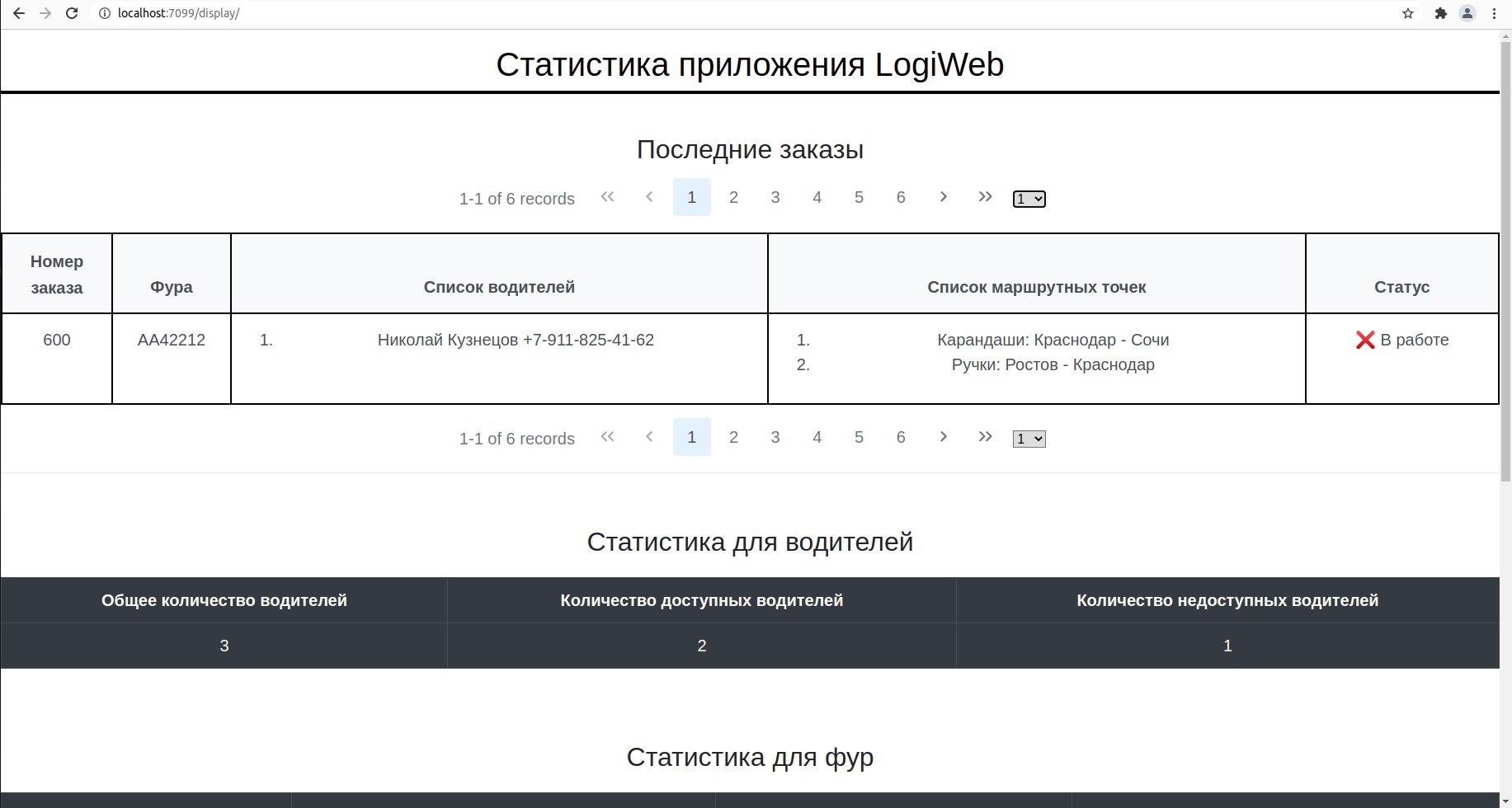
Picture 6. – 1-t step of adding a new order (choosing the waypoint(s))



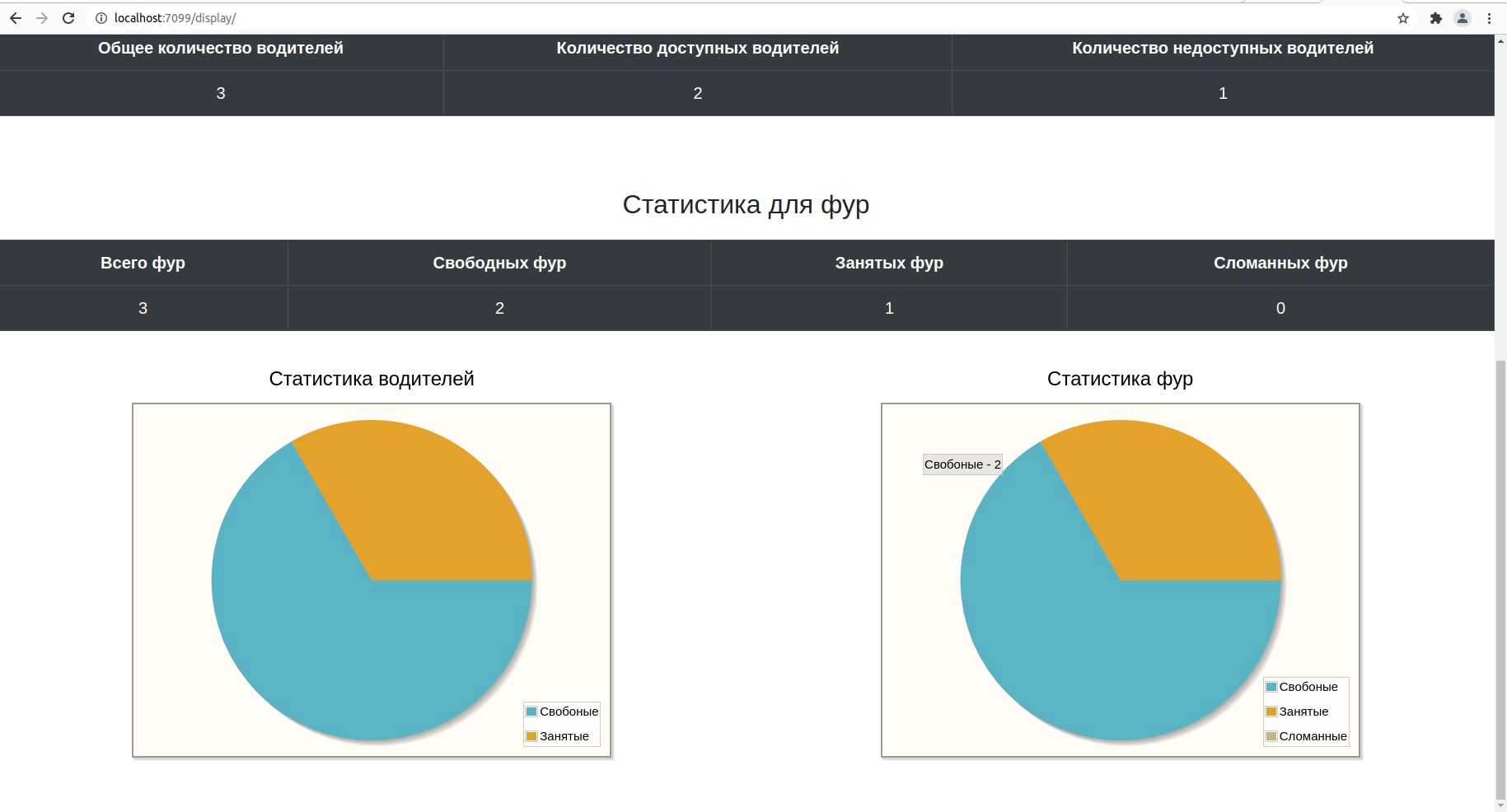
Picture 7. – 2-d step of adding a new order (choosing the truck)



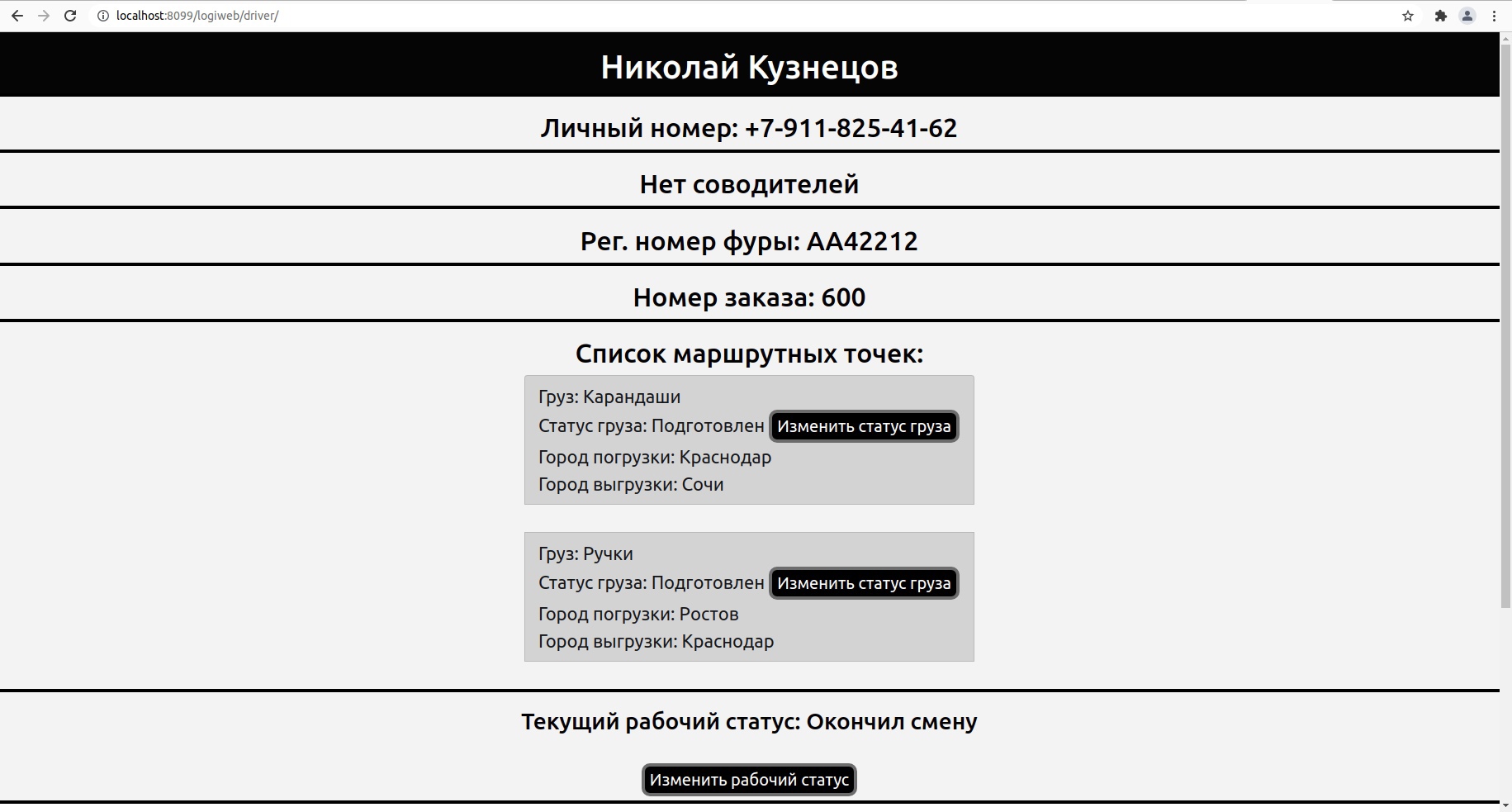
Picture 8. – 3-d step of adding a new order (choosing the driver(s))



Picture 9. – Display application (part 1)



Picture 10. – Display application (part 2)

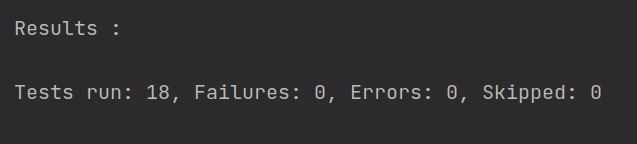


Picture 11. – Driver menu

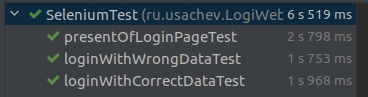
## 10. Code quality.

### Tests.

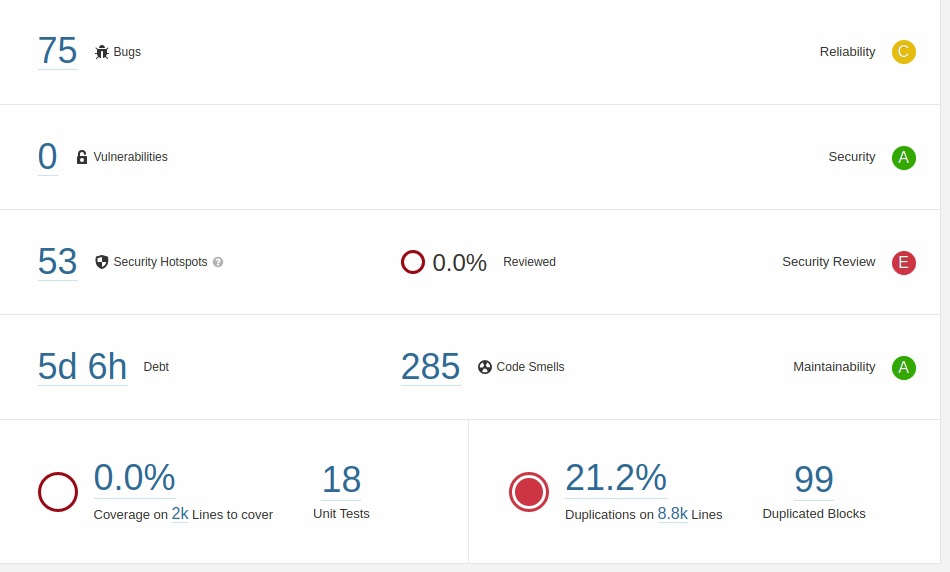
JUnit tests:



Selenium tests.

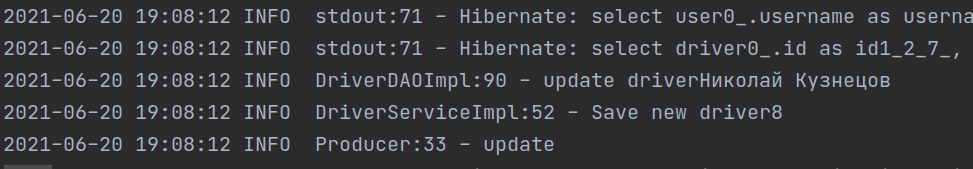


### Sonar report:



### Logging.

*Logiweb/target/log/application.log*



## 11. Future improvement.

1. Adding new functionality (drivers chat, etc.).
2. Refactoring and optimization code.