Contents

System Description

The main goal of the system is to help manage information in logistics company.

The system consists of 3 subsystems:

1. The Managers subsystem;
2. The Drivers subsystem;
3. The WebServices subsystem.

The Managers subsystem solves next tasks:

1. CRUD operations for Trucks;
2. CRUD operations for Drivers;
3. Create, Read and Delete operations for Orders;
4. Read operation for Cargos.

The Drivers subsystem solves the task of giving a Driver the information about his current assignment.

The WebServices subsystem solves next tasks:

1. Open a shift for a Driver;
2. Close a shift for a Driver;
3. Calculate hours of work for a Driver;
4. Change state of a Driver;
5. Change state of a Cargo.

Frameworks and Technologies

1. Java 1.8
2. MySQL
3. JPA 2.1
4. Log4j
5. Spring Framework
6. JAX-RS (Jersey)
7. JUnit
8. JMock
9. JSP/JSTL
10. Bootstrap Framework
11. JSF

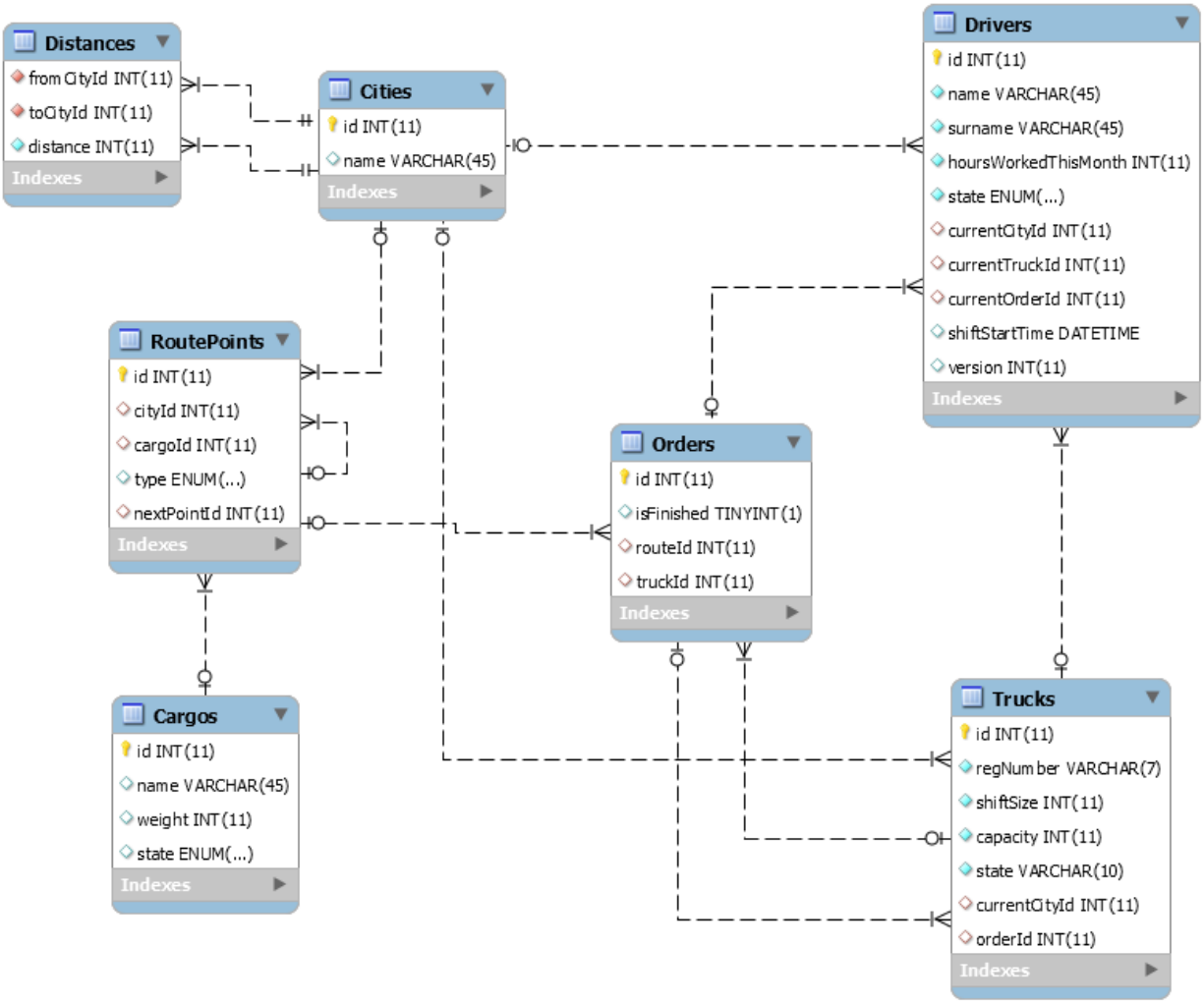
Development Tools

1. IntelliJ Idea
2. Maven
3. MySQL WorkBench
4. Tomcat/WildFly
5. Git

Additional Features

1. If manager tries to create an order with cargos which weigh more than any truck can transport, he will get a hint that it is not possible;
2. If manager tries to create an order and assign a truck for which there is not enough drivers to form a shift, he will get a hint that it is not possible;
3. Validation of user inputs in all subsystems.

Database Scheme



Implementation Description

An order route from the task is implemented using linked list. The Order entity only contains the first RoutePoint, which contains the link to the next RoutePoint etc. The last point in the route contains the link to **null**.

The map from the task is implemented using Distance Class with 3 fields:

1. int fromCityId
2. int toCityId
3. int distance

There is a corresponding table Distances in MySQL database. In this table for each possible pair of cityIds there is a distance. Distance from City to itself equals **zero**.

Modules Interaction

Project consists of 4 modules:

1. persistence;
2. service;
3. view;
4. rs-client.

persistence is not dependant on any modules.

service is dependant on **persistence**.

view is dependant on **service**.

rs-client is dependant on **service**.

UI Description

There are 2 UI clients in this project:

1. LogiWeb – is WEB UI based on jsp pages.
2. LogiRest – is WEB UI based on jsf pages.

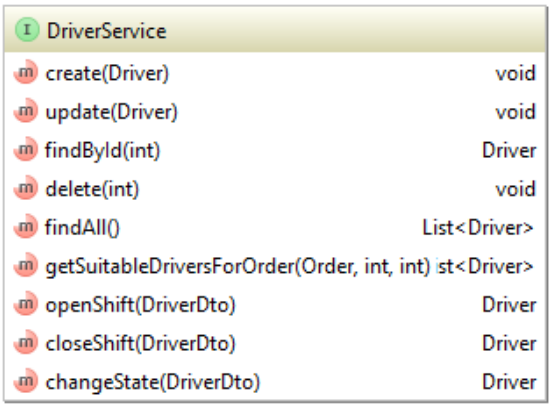
Both UI shares the same structure and css styles, but have different colors of header and footer to easily differentiate between them.

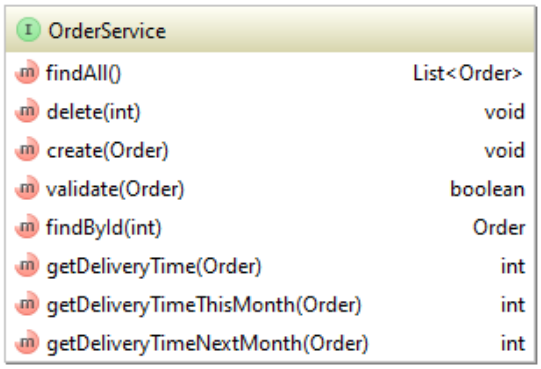
The UI structure consists of 3 parts:

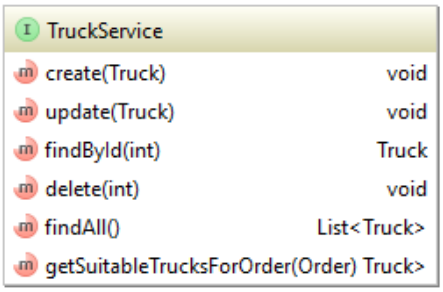
1. Header – here are the links to different web-forms available to user;
2. Content – here are the web-forms themselves;
3. Footer – only company logo, used for better visual experience.

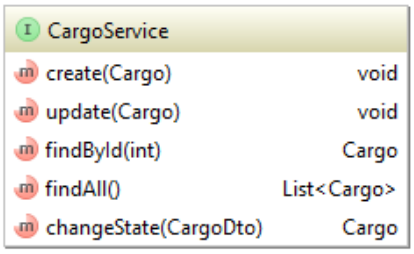
Business Layer Description

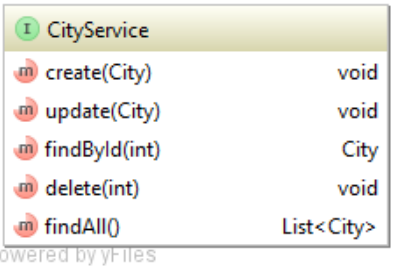
On the picture below you can see interfaces of all services available in the application.

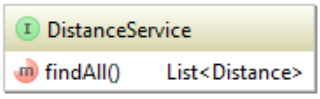


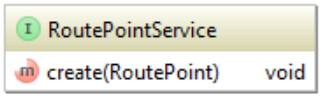






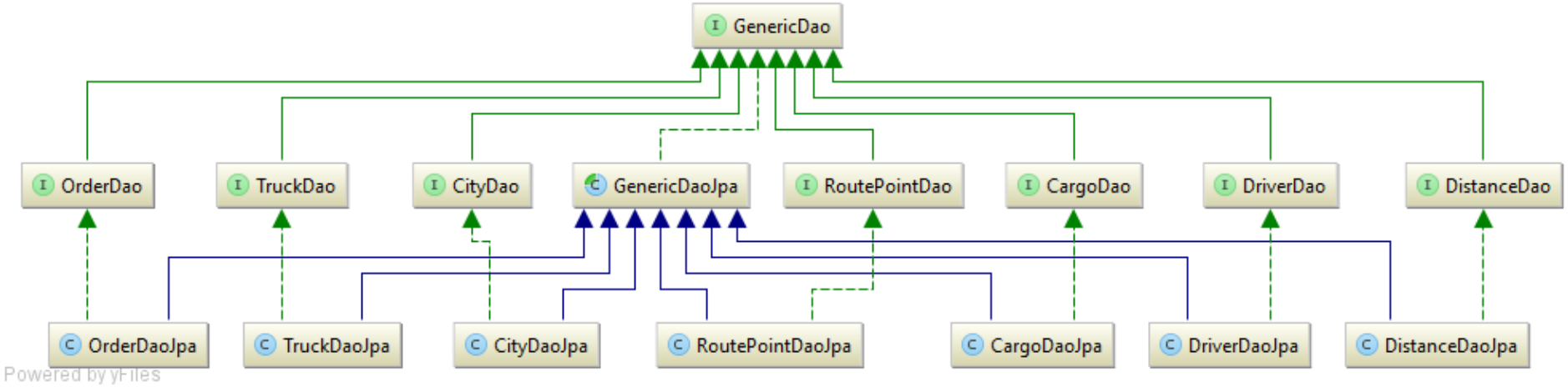






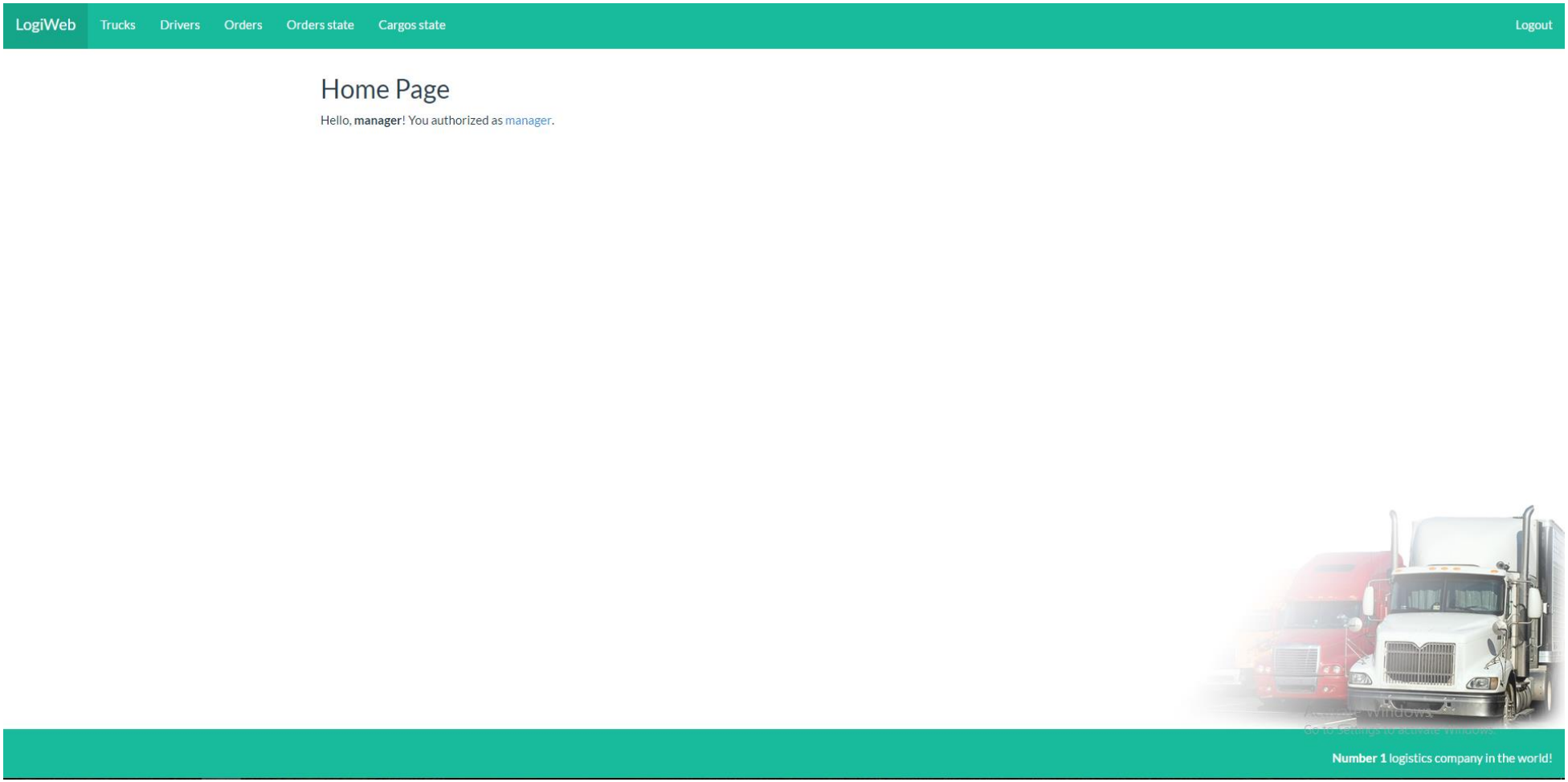
DAO Layer Description

Transactions are handled by Spring @Transactional annotation on services methods.

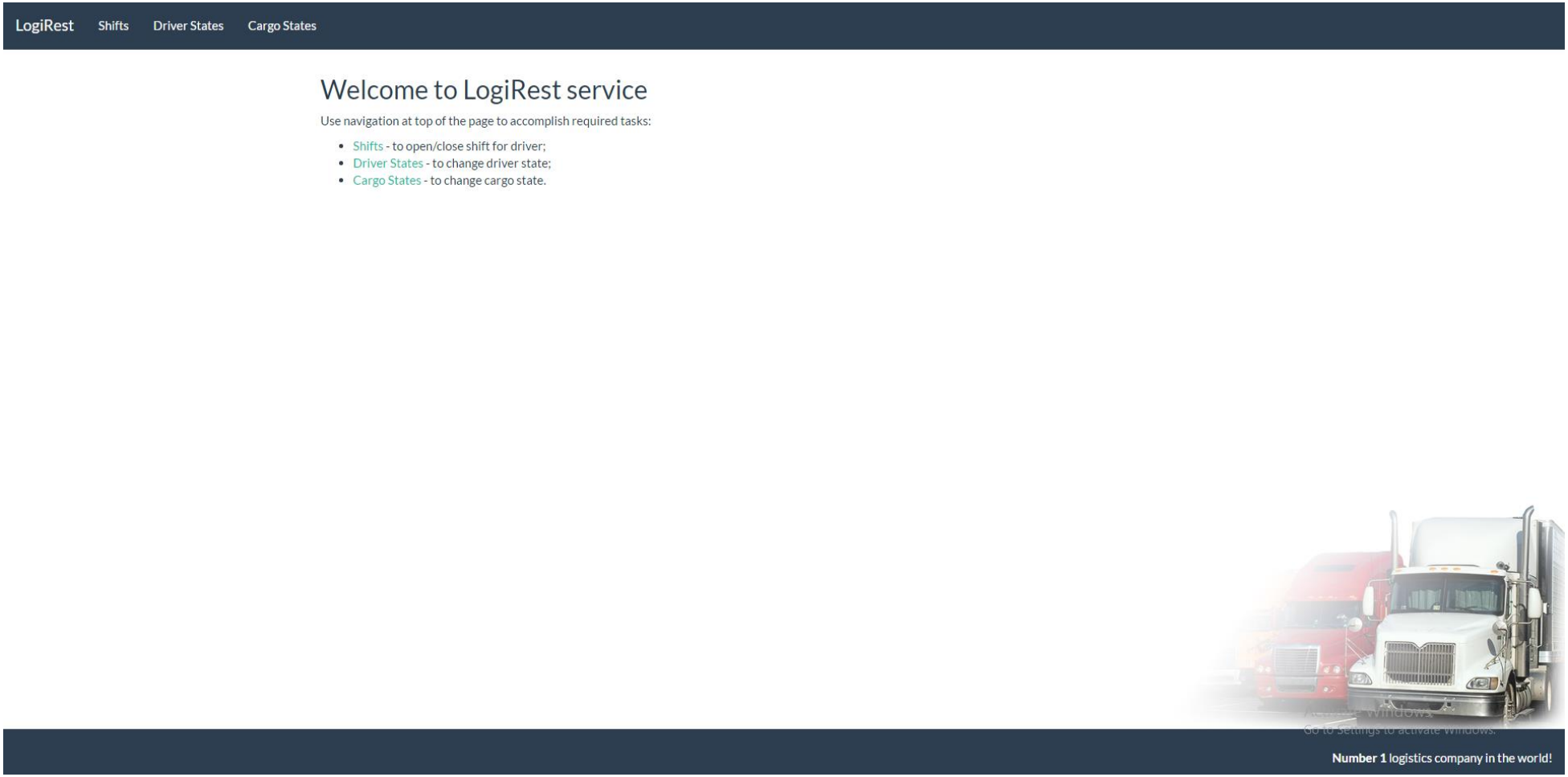


Screenshots

LogiWeb Home Page



LogiRest Home Page



Unit Tests Description

DriverServiceTest Class

1. GetFreeDriversForOrder() – tests that only free drivers are selected for assignment to an orderL;
2. GetDriversInSameCityAsOrder() – tests that only drivers in same city as assigned truck are selected for assignment to an order;
3. GetDriversWhichHaveEnoughTimeForOrder() – tests that only drivers which have enough time are selected for assignment to an order;
4. GetSuitableDrivers() - tests that only free drivers in same city as assigned truck and which have enough time are selected for assignment to an order.

OrderServiceTest Class

1. GetOrdersList() – tests that orderDao findAll() method will only be invoked once;
2. CreateOrderWithDoubleLoadedCargo() - tests that orderService(order) with double loaded cargos will throw DoubleLoadCargoException;
3. CreateOrderWithNotAllCargosUnloaded() - tests that orderService(order) with not all cargos unloaded will throw NotAllCargosUnloadedException;
4. CreateOrderWithNotAllCargosLoaded() - tests that orderService(order) with not all cargos loaded will throw UnloadNotLoadedCargoException.

TruckServiceTest Class

1. FindOkTrucksForOrder() – tests that only trucks with OK state will be returned for order;
2. FindOkTrucksWithNoOrderForOrder() - tests that only trucks not on order will be returned for order;
3. FindOkTrucksWithNoOrderAndEnoughCapacityForOrder() - tests that only ok trucks not on order with enough capacity will be returned for order.

Deployment Guide

Add datasource to WildFly

Open cmd in root of project and use command:

```
mvn clean install wildfly:deploy
```

Further Improvements

1. Multiclient testing
2. Paging
3. AJAX for UI
4. Admin for managing drivers/managers
5. Maps service integration