# WorldWideLogistics

# Technical Solution Description

Gerasimchuk Maksim

2018

## Content

## Introduction

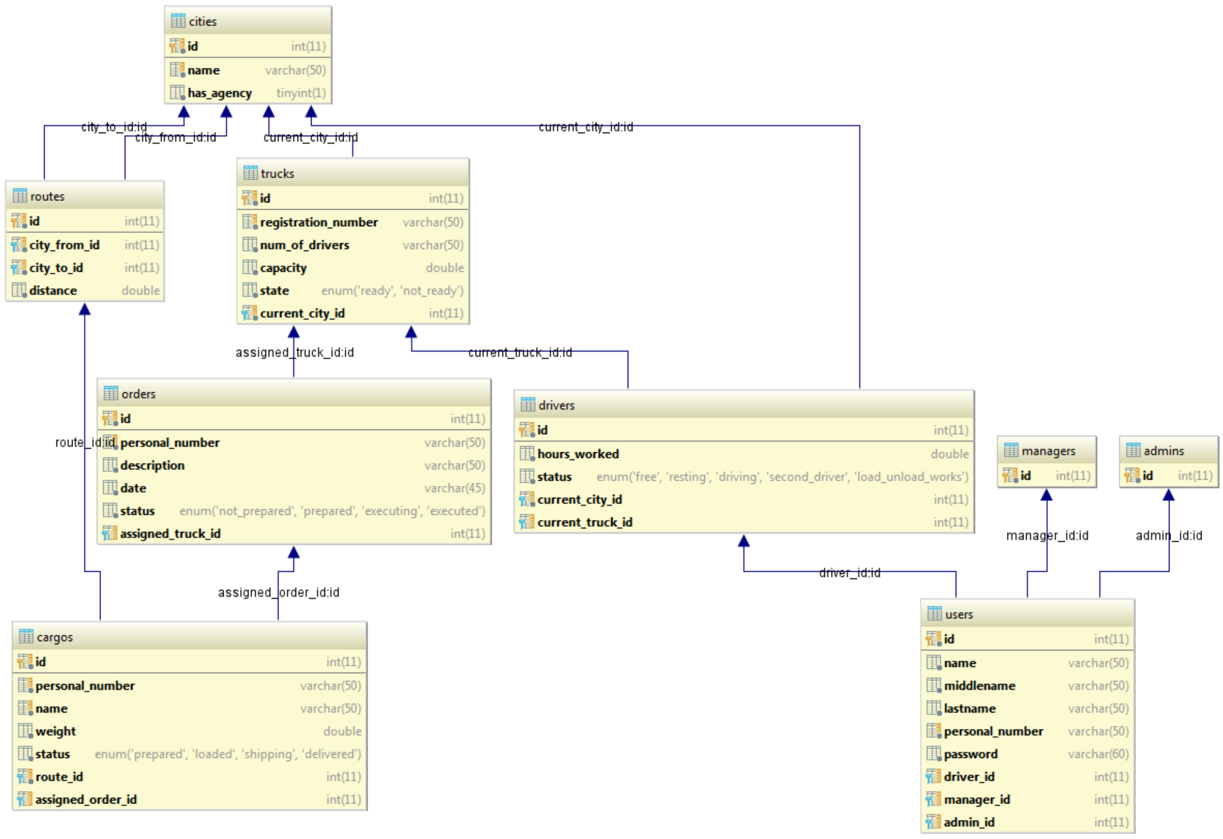
WorldWideLogistics is a web application for logistic business. This application allows to manage cargos, trucks, drivers and orders. Depending on the role (driver, manager, admin) user is allowed to manage current order status (driver), create and edit trucks, drivers, create orders (manager), create, edit and delete orders and other entities.

## Technologies and frameworks

- Spring
  - Spring MVC 5.0.8.RELEASE
  - Spring security 5.0.7.RELEASE
- JPA 2.0 Hibernate 5.3.1.Final
- JSP (JSTL 1.2)
- jQuery, HTML, CSS
- EJB 1.0.1.Final
- JSF 2.3.5.SP1
- Maven 3.5.2
- Mockito 2.23.0Selenium
- Google Maps API
- Jersey WebServices 1.19.2
- Log4j 1.2.17
- Sonarqube 7.3
- Tomcat server 8.5.31
- WildFly server 14.0.1.Final

# Database

## Scheme



**Description**

| Database table | Description |
|---|---|
| Users | Contains basic user information (name, personal number, role etc…) |
| Drivers | Contains specific driver fields (current city, current truck etc…) |
| Managers | Contains manager fields |
| Admins | Contains admin fields |
| Trucks | Contains truck information (registration number, maximal number of drivers, capacity etc…) |
| Cargos | Contains basic cargo information (personal number, name, weight, status etc…) |
| Orders | Contains basic order information (personal number, description, status, etc…) |
| Cities | Contains basic city information (name, has agency ) |
| Routes | Contains basic information about route (origin city, destination city) |

## Model summary

User – basic entity used for authentication, represented by three possible roles: driver, manager, admin.

Manager – entity which has rights to create (add to database) new orders . Also he can create and edit trucks, drivers, cargos.

Admin – entity like manager but with extended rights: in addition to manager abilities he can change or delete every entity (if it doesn’t violate the validity of the database)

Driver – entity which is responsible for executing order. Driver has rights to refresh state of the executing order and cargos included to it.

Truck – entity describing properties of used trucks such as registration number, state (ready to use or not), maximal number of drivers, capacity etc… Truck can be added and modified by manager and admin, deleted by admin only.

Cargo – entity describing properties of cargo such as weight, status (prepared, loaded, shipping or delivered). Cargo can be created and edited both manager and admin, also admin can delete cargos. Driver has an ability to update status of cargos which are included in his current order.

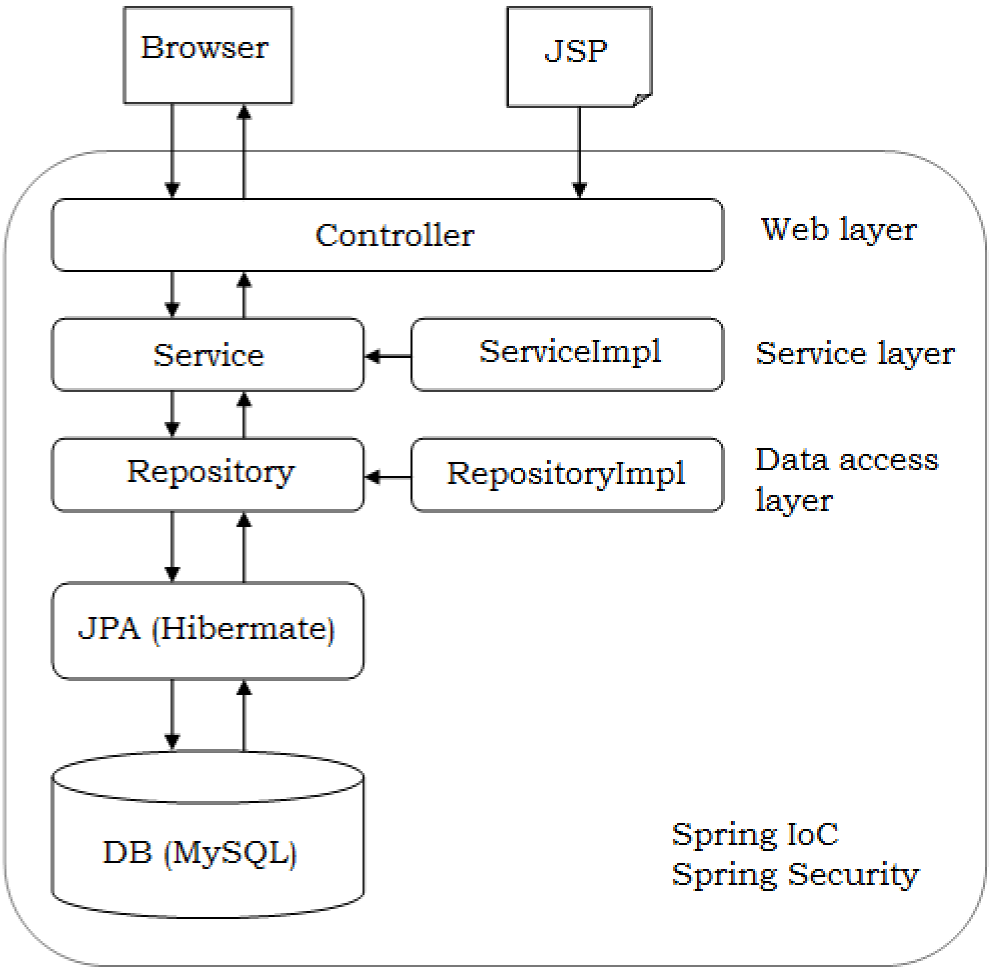
Order – entity describing properties of order such as personal number, description , date, status (not prepared, prepared, executing, executed). Order can be created both by manager and admin, but only admin has rights to update or delete orders (if it doesn’t violate the validity of the database). Order can be assigned to truck (with assigned drivers), so drivers have right to update status of executing order.

City – entity describing properties of city such as name and existence of agency. City can be managed by admin only, manager has rights to use only cities which are already created.

Route – entity describing properties of route such as origin point, destination point and distance between them. Route can be managed by admin only, manager has rights to use only routes which are already created.

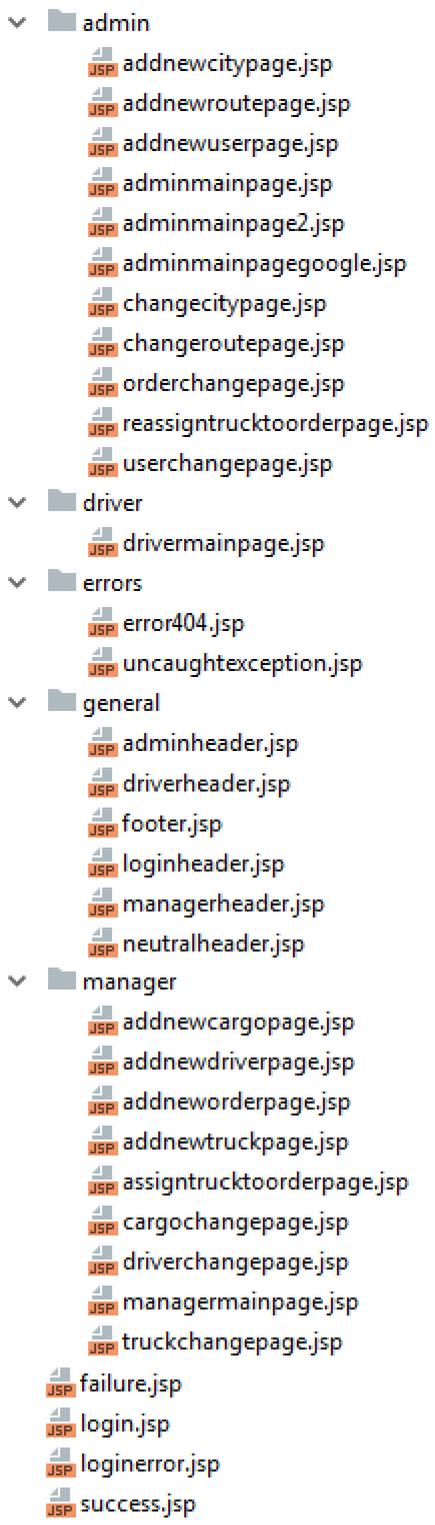
## Modules

Application is designed using the Model-View-Controller design pattern and has following architecture:



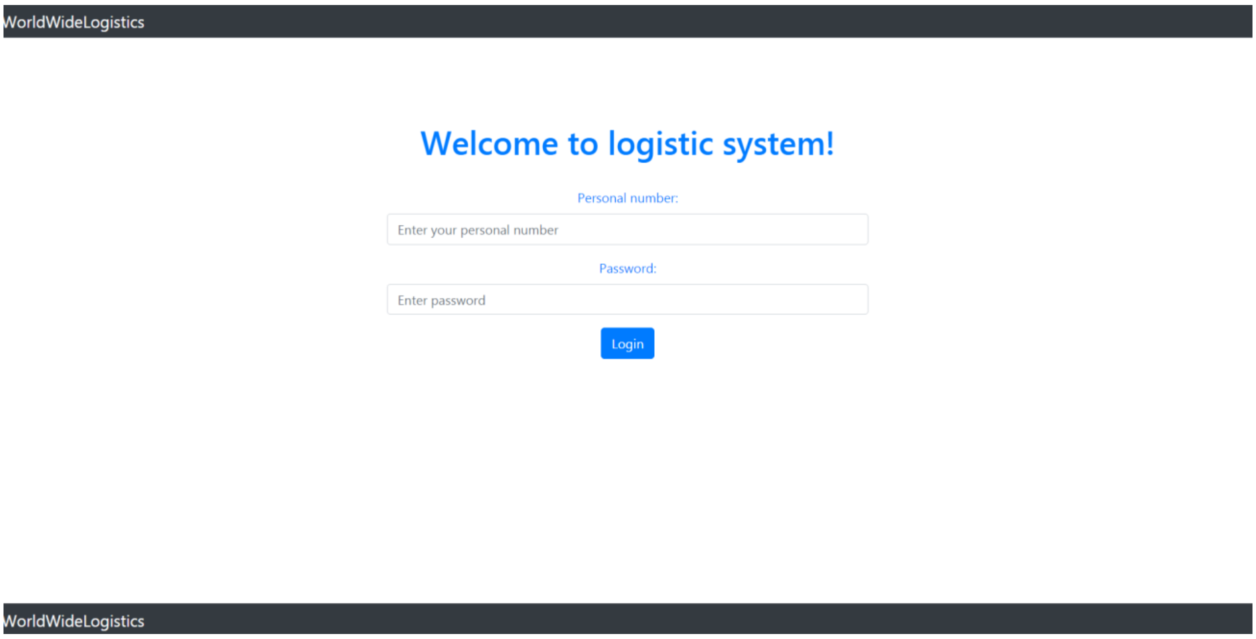
**View**

View layer is represented by JSP pages with CSS and JS. JSP tree is represented on picture below:



```
admin
    addnewcitypage.jsp
    addnewroutepage.jsp
    addnewuserpage.jsp
    adminmainpage.jsp
    adminmainpage2.jsp
    adminmainpagegoogle.jsp
    changecitypage.jsp
    changeroutepage.jsp
    orderchangepage.jsp
    reassigntrucktoorderpage.jsp
    userchangepage.jsp
driver
    drivermainpage.jsp
errors
    error404.jsp
    uncaughtexception.jsp
general
    adminheader.jsp
    driverheader.jsp
    footer.jsp
    loginheader.jsp
    managerheader.jsp
    neutralheader.jsp
manager
    addnewcargopage.jsp
    addnewdriverpage.jsp
    addneworderpage.jsp
    addnewtruckpage.jsp
    assigntrucktoorderpage.jsp
    cargochangepage.jsp
    driverchangepage.jsp
    managermainpage.jsp
    truckchangepage.jsp
failure.jsp
login.jsp
loginerror.jsp
success.jsp
```

Login page:



WorldWideLogistics

Welcome to logistic system!

Personal number:

Enter your personal number

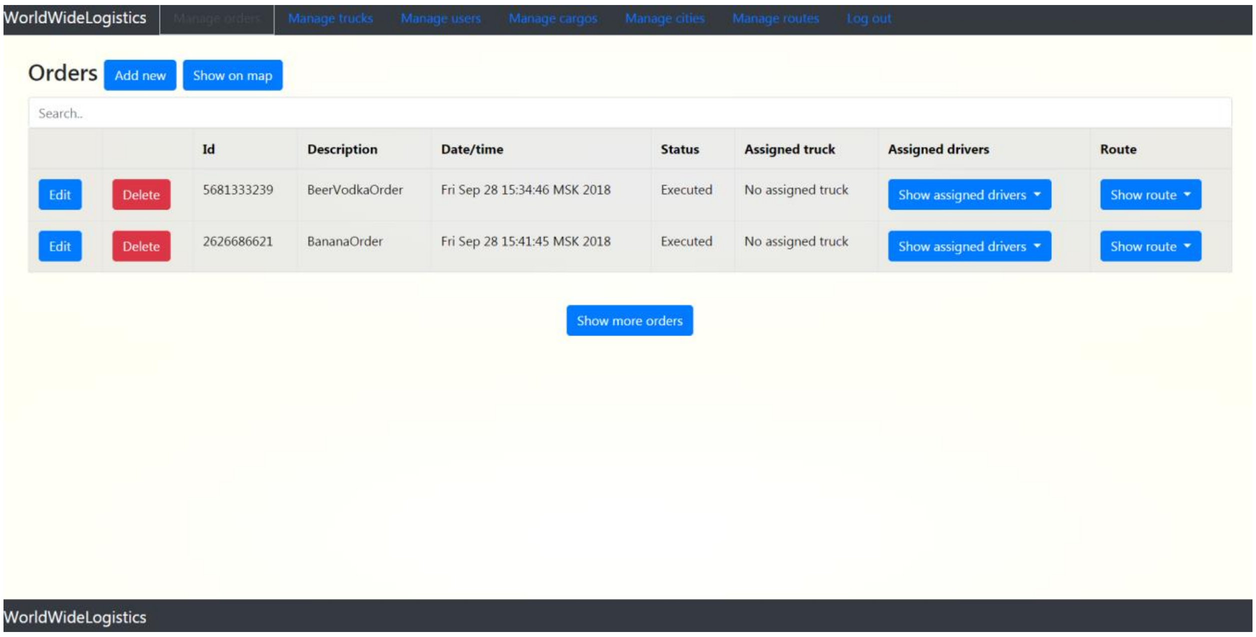
Password:

Enter password

Login

WorldWideLogistics

Admin main page:



WorldWideLogistics | Manage orders | Manage trucks | Manage users | Manage cargos | Manage cities | Manage routes | Log out

Orders Add new Show on map

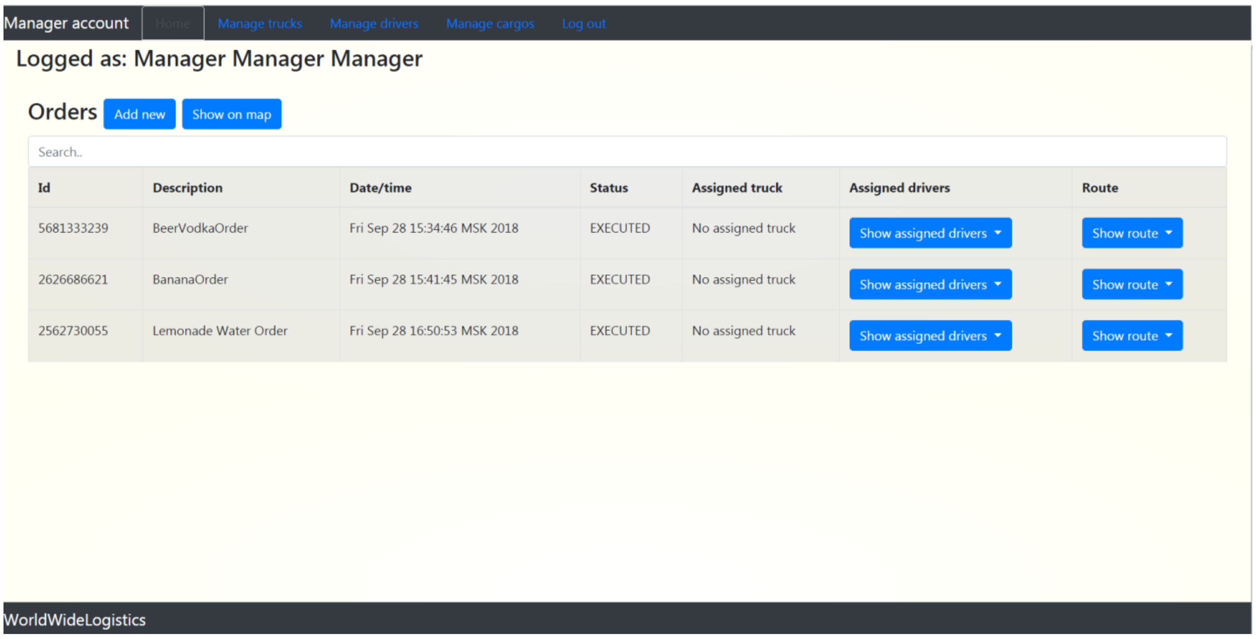
Search..

| | | Id | Description | Date/time | Status | Assigned truck | Assigned drivers | Route |
|---|---|---|---|---|---|---|---|---|
| Edit | Delete | 5681333239 | BeerVodkaOrder | Fri Sep 28 15:34:46 MSK 2018 | Executed | No assigned truck | Show assigned drivers | Show route |
| Edit | Delete | 2626686621 | BananaOrder | Fri Sep 28 15:41:45 MSK 2018 | Executed | No assigned truck | Show assigned drivers | Show route |

Show more orders

WorldWideLogistics

Manager main page:

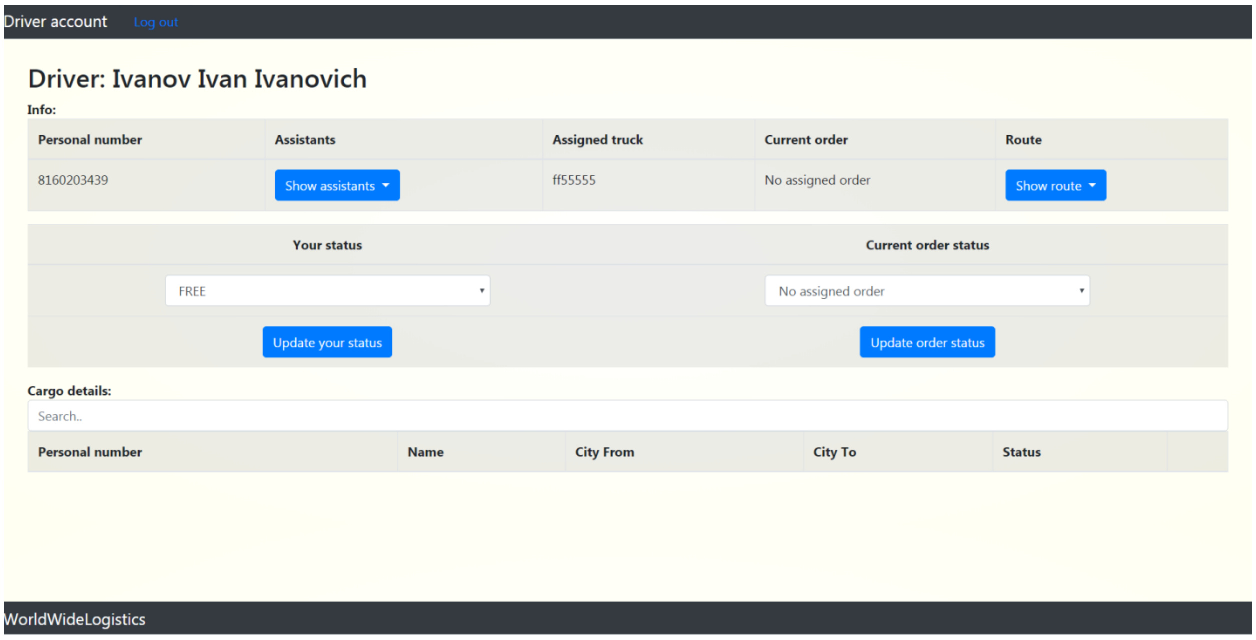


Logged as: Manager Manager Manager

Orders

| Id | Description | Date/time | Status | Assigned truck | Assigned drivers | Route |
|---|---|---|---|---|---|---|
| 5681333239 | BeerVodkaOrder | Fri Sep 28 15:34:46 MSK 2018 | EXECUTED | No assigned truck | Show assigned drivers | Show route |
| 2626686621 | BananaOrder | Fri Sep 28 15:41:45 MSK 2018 | EXECUTED | No assigned truck | Show assigned drivers | Show route |
| 2562730055 | Lemonade Water Order | Fri Sep 28 16:50:53 MSK 2018 | EXECUTED | No assigned truck | Show assigned drivers | Show route |

Driver main page:



Driver: Ivanov Ivan Ivanovich

Info:

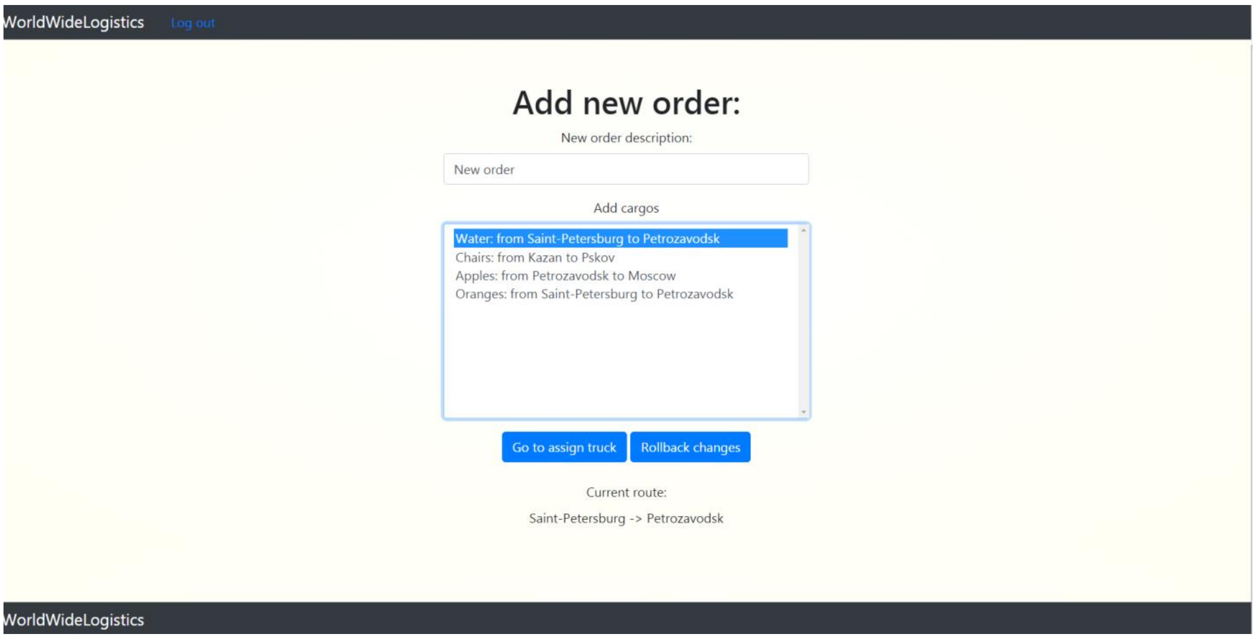
| Personal number | Assistants | Assigned truck | Current order | Route |
|---|---|---|---|---|
| 8160203439 | Show assistants | ff55555 | No assigned order | Show route |

| Your status | Current order status |
|---|---|
| FREE | No assigned order |
| Update your status | Update order status |

Cargo details:

| Personal number | Name | City From | City To | Status |
|---|---|---|---|---|

Add new order page:


WorldWideLogistics Log out

Add new order:

New order description:

New order

Add cargos

Water: from Saint-Petersburg to Petrozavodsk
Chairs: from Kazan to Pskov
Apples: from Petrozavodsk to Moscow
Oranges: from Saint-Petersburg to Petrozavodsk

Go to assign truck | Rollback changes

Current route:

Saint-Petersburg -> Petrozavodsk

WorldWideLogistics

**Entities**

Entities package contains 9 classes:

- User
- Driver
- Manager
- Admin
- Truck
- Cargo
- Order
- City
- Route

*User entity*

| Field | Description |
|---|---|
| Id | User's ID |
| Name | User's first name |
| middleName | User's middle name |
| lastName | User's last name |
| personalNumber | User's unique personal name (used as a login) |
| password | User's password (Encrypted by BCrypt) |
| driver | Reference to driver entity (if user is driver, otherwise null) |
| manager | Reference to manager entity (if user is manager, otherwise null) |
| admin | Reference to admin entity (if user is admin, otherwise null) |

*Driver entity*

| Field | Description |
|---|---|
| id | Driver's id |
| hoursWorked | Driver's hours worked in current month |
| status | Driver's current status (Free, resting, driving, second driver, load/unload works) |
| currentCity | Driver's current city (not null) |
| currentTruck | Driver's current truck |
| user | Reference to user entity |

*Manager entity*

| Field | Description |
|---|---|
| id | Manager's id |
| user | Reference to user entity |

*Admin entity*

| Field | Description |
|---|---|
| id | Manager's id |
| user | Reference to user entity |

*Truck entity*

| Field | Description |
|---|---|
| id | Truck's id |
| registrationNumber | Truck's registration number |
| numOfDrivers | Truck's number of drivers |
| capacity | Truck's maximal capacity |
| state | Truck's current state (ready or not) |
| currentCity | Truck's current city |
| driversInTruck | Set of drivers assigned to truck |
| assignedOrder | Reference to assigned order |

*Cargo entity*

| Field | Description |
|---|---|
| id | Cargo's id |
| personalNumber | Cargo's personal number |
| name | Cargo's name |
| weight | Cargo's weight |
| status | Cargo's status (Prepared, loaded, shipping, delivered) |
| route | Cargo's route |
| order | Reference to assigned order |

*Order entity*

| Field | Description |
|---|---|
| id | Order's id |
| personalNumber | Order's personal number |
| description | Order's description |
| date | Order's date |
| status | Order's status |
| assignedTruck | Order's assigned truck |
| cargosInOrder | Set of cargos assigned to order |

*City entity*

| Field | Description |
|---|---|
| id | City's id |
| name | City's name |
| hasAgency | City's hasAgency Boolean field (true if city has agency) |
| driversInCity | Set of drivers in city |
| citiesFrom | Set of origin cities |
| citiesTo | Set of destination cities |
| trucksInCity | Set of trucks in city |

*Route entity*

| Field | Description |
|---|---|
| id | Route's id |
| cityFrom | Route's origin city |
| cityTo | Route's destination city |
| distance | Route's distance |
| cargosOnRoute | Set of cargos on route |

**Repository**

Repositories package contains 9 classes:

- UserRepository
- DriverRepository
- ManagerRepository
- AdminRepository
- TruckRepository
- Cargo Repository
- OrderRepository
- CityRepository
- Route Repository

All repository classes has next methods:

| Method | Description |
|---|---|
| create | Persists new entity |

| | |
|---|---|
| update | Updates existing entity |
| getById | Returns entity by id |
| getAll | Return collection of all entities |
| remove | Removes entity by id |

**Service layer**

Service layer is represented by 11 services:

- CargoService
- CityService
- DriverService
- OrderService
- RouteService
- StatisticService
- TruckService
- UserService
- SecurityService
- CustomUserDetailService
- AuthenticationSuccessHandler

*CargoService*

| Method | Description |
|---|---|
| **boolean** createCargo(CargoDTO cargoDTO) | Creates new cargo with data from DTO |
| **boolean** updateCargo(CargoDTO cargoDTO) | Updates existing cargo with data from DTO |
| **boolean** deleteCargo(**int** cargoId) | Deletes existing cargo |
| UpdateMessageType deleteCargo(**int** cargoId, **int** val) | Updates existing cargo with data from DTO |
| Collection<Cargo> getAvailableCargos() | Returns collection with available cargos |
| CargoStatus getCargoStatusFromString(String status) | Returns cargo status parsed from string |
| UpdateMessageType updateCargoStatus(**int** cargoId, CargoStatus newStatus) | Updates cargo status |

*CityService*

| Method | Description |
|---|---|
| **boolean** createCity(CityDTO cityDTO) | Creates new city with data from DTO |
| **boolean** updateCity(CityDTO cityDTO) | Updates existing city with data from DTO |
| **boolean** deleteCity(**int** cityId) **throws** Exception | Deletes existing city |
| UpdateMessageType deleteCity(**int** cityId, **int** val) | Deletes existing city |

*DriverService*

| Method | Description |
|---|---|
| DriverStatus getDriverStatusValFromString(String status) | Returns driver status parsed from string |
| **void** updateDriverHoursWorked() | Updates drivers hours worked values every month |

| | |
|---|---|
| UpdateMessageType updateDriverStatus(**int** driverId, DriverStatus newStatus) | Updates driver status |

*OrderService*

| Method | Description |
|---|---|
| Collection<Cargo> getChosenCargos(OrderDTO orderDTO) | Returns collection of chosen cargos according to data from DTO |
| Collection<Truck> getAvailableTrucks(OrderDTO orderDTO) **throws** RouteException | Returns collection of available trucks to execute order with options according to data from DTO |
| List<City> getOrderRoute(OrderDTO orderDTO, Truck truck) **throws** RouteException | Creates order route according to data from DTO |
| ReturnValuesContainer<List<Driver>> checkIfDriversHoursWorkedOverLimit(**double** orderExecutingTime, Date date, Collection<Driver> driversInTruck) | Checks if drivers has too much hours worked to execute order with options according to data from DTO |
| UpdateMessageType createOrder(OrderDTO orderDTO) **throws** RouteException | Creates new order |
| ReturnValuesContainer<Order> createOrder(OrderDTO orderDTO, **int** val) **throws** RouteException | Creates new order |
| UpdateMessageType updateOrder(OrderDTO orderDTO) **throws** RouteException, TooManyHoursWorkedForOrderException | Updates existing order |
| ReturnValuesContainer<Order> updateOrder(OrderDTO orderDTO, **int** val) **throws** RouteException, TooManyHoursWorkedForOrderException | Updates existing order |
| OrderStatus getOrderStatusFromString(String status) | Returns order status parsed from string |
| **boolean** areAllCargosDelivered(Order order) | Checks if all cargos in order have status “Delivered” |
| UpdateMessageType deleteOrder(OrderDTO orderDTO) | Deletes existing order if it’s possible |
| UpdateMessageType deleteOrder(**int** orderId) | Deletes existing order if it’s possible |
| UpdateMessageType updateOrderStatus(**int** orderId, OrderStatus newStatus) | Updates order status |
| **double** getExecutingTime(OrderDTO orderDTO) **throws** RouteException | Counts time needed to execute order |

*RouteService*

| Method | Description |
|---|---|
| **boolean** createRoute(RouteDTO routeDTO) | Creates new route between cities according to data from DTO |
| ReturnValuesContainer<Route> createRoute(RouteDTO routeDTO, **int** val) | Creates new route between cities according to data from DTO |
| **boolean** updateRoute(RouteDTO routeDTO) | Updates existing route between cities according to data from DTO |
| ReturnValuesContainer<Route> updateRoute(RouteDTO routeDTO, **int** val) | Updates existing route between cities according to data from DTO |
| **boolean** deleteRoute(**int** routeId) **throws** Exception | Deletes existing route |

| | |
|---|---|
| UpdateMessageType deleteRoute(**int** routeId, **int** val) | Deletes existing route |

*StatisticService*

| Method | Description |
|---|---|
| **int** getNumOfTrucksTotal() | Returns number of trucks total |
| **int** getNumOfTrucksFree() | Returns number of free total |
| **int** getNumOfTrucksNotReady() | Returns number of trucks which has status “Not ready” |
| **int** getNumOfTrucksExecutingOrders() | Returns number of trucks executing orders |
| **int** getNumOfDriversTotal() | Returns number of drivers total |
| **int** getNumOfDriversFree() | Returns number of free drivers |
| **int** getNumOfDriversExecutingOrders() | Returns number of drivers executing orders |

*TruckService*

| Method | Description |
|---|---|
| UpdateMessageType createTruck(TruckDTO truckDTO) | Creates new truck according to data from DTO |
| UpdateMessageType updateTruck(TruckDTO truckDTO) | Updates existing truck according to data from DTO |
| UpdateMessageType deleteTruck(**int** id) | Deletes existing truck |
| Collection<Truck> getFreeTrucks() | Returns collection of free trucks |

*UserService*

| Method | Description |
|---|---|
| UpdateMessageType createDriver(DriverDTO driverDTO) | Creates new driver according to data from DTO |
| ReturnValuesContainer<User> createDriver(DriverDTO driverDTO, **int** val) | Creates new driver according to data from DTO |
| UpdateMessageType createManager(ManagerDTO managerDTO) | Creates new manager according to data from DTO |
| ReturnValuesContainer<User> createManager(ManagerDTO managerDTO, **int** val) | Creates new manager according to data from DTO |
| UpdateMessageType createAdmin(AdminDTO adminDTO) | Creates new admin according to data from DTO |
| ReturnValuesContainer<User> createAdmin(AdminDTO adminDTO, **int** val) | Creates new admin according to data from DTO |
| UpdateMessageType updateDriver(DriverDTO driverDTO) | Updates existing driver according to data from DTO |
| UpdateMessageType updateManager(ManagerDTO managerDTO) | Updates existing manager according to data from DTO |
| UpdateMessageType updateAdmin(AdminDTO adminDTO) | Updates existing admin according to data from DTO |
| UpdateMessageType deleteDriver(**int** userId) | Deletes existing driver |
| UpdateMessageType deleteManager(**int** userId) | Deletes existing manager |
| UpdateMessageType deleteAdmin(**int** userId) | Deletes existing admin |
| Collection<User> getAllDrivers() | Returns collection of all drivers |

| | |
|---|---|
| Collection<User> getFreeDrivers() | Returns collection of free drivers |
| Collection<UserRole> getRoles() | Returns collection of roles |
| UpdateMessageType createUser(UserDTO userDTO) | Creates new user according to data from DTO |
| ReturnValuesContainer<User> createUser(UserDTO userDTO, **int** val) | Creates new user according to data from DTO |
| UpdateMessageType updateUser(UserDTO userDTO) | Updates existing user according to data from DTO |
| UpdateMessageType deleteUser(**int** id) | Deletes existing user |

*SecurityService*

| Method | Description |
|---|---|
| String findLoggedInUsername() | Returns personal number of logged user |

*CustomUserDetailService*

| Method | Description |
|---|---|
| **public** UserDetails loadUserByUsername(String personalNumber) **throws** UsernameNotFoundException | Returns UserDetails object builded with data from user loaded by personal number |

*AuthenticationSuccessHandler*

| Method | Description |
|---|---|
| **protected** String determineTargetUrl(HttpServletRequest request, HttpServletResponse response) | Returns target URL depends on logged user |

**Controller layer:**

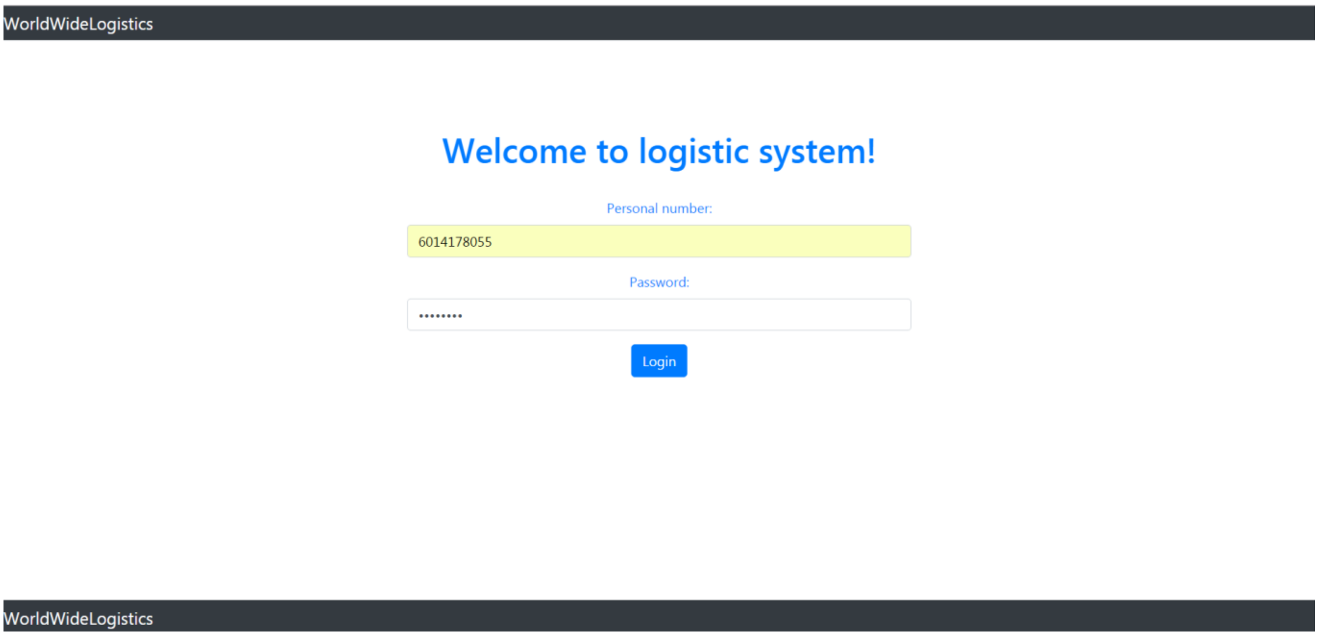
Controller layer is represented by 5 classes annotated with SpringMVC @Controller:

- AdminController
- DriverController
- ManagerController
- UserController
- RestController

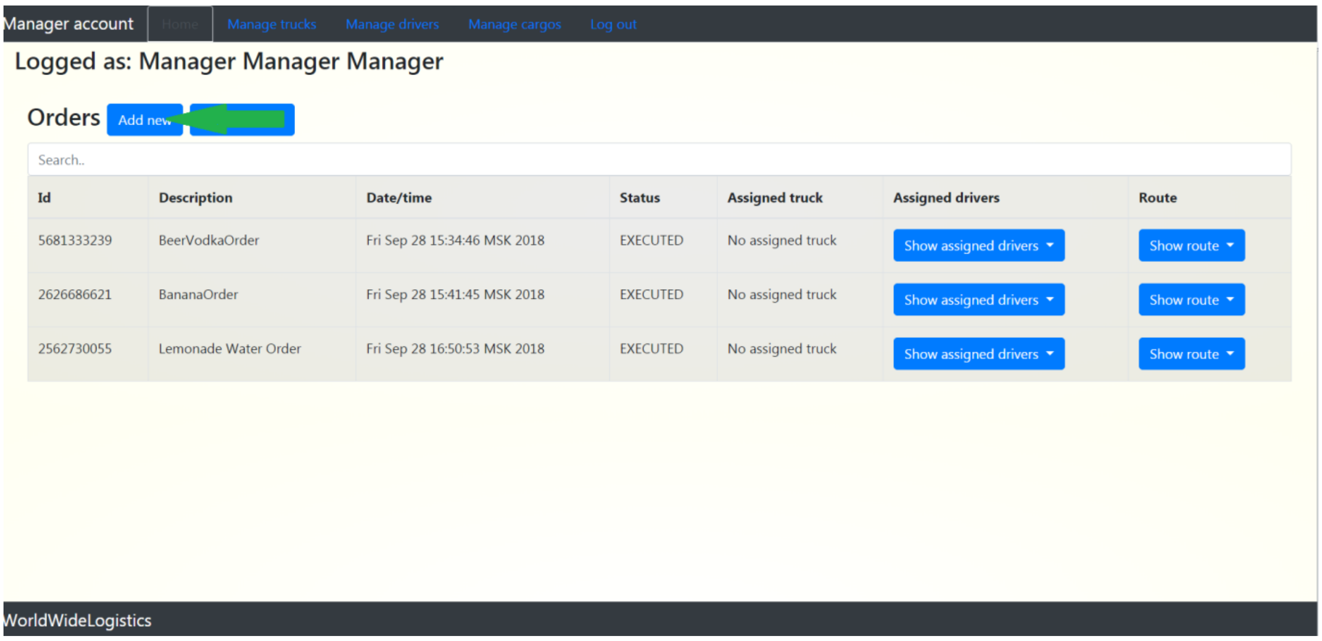
## Main workflow

Main workflow for creating order is represented below:

1. Login as manager:



2. Press "Add new" button on home page:

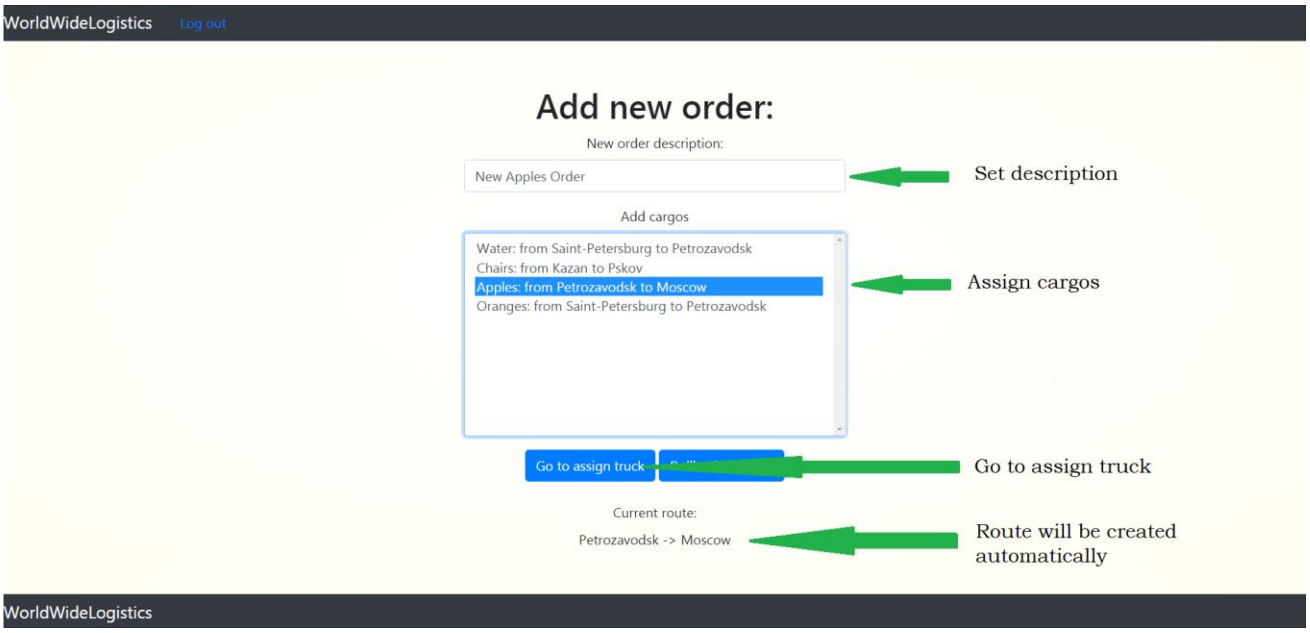


Logged as: Manager Manager Manager

Orders

| Id | Description | Date/time | Status | Assigned truck | Assigned drivers | Route |
|---|---|---|---|---|---|---|
| 5681333239 | BeerVodkaOrder | Fri Sep 28 15:34:46 MSK 2018 | EXECUTED | No assigned truck | Show assigned drivers | Show route |
| 2626686621 | BananaOrder | Fri Sep 28 15:41:45 MSK 2018 | EXECUTED | No assigned truck | Show assigned drivers | Show route |
| 2562730055 | Lemonade Water Order | Fri Sep 28 16:50:53 MSK 2018 | EXECUTED | No assigned truck | Show assigned drivers | Show route |

3. Set description and assign cargos (route will be created automatically):


WorldWideLogistics Log out

Add new order:

New order description:

New Apples Order

Add cargos

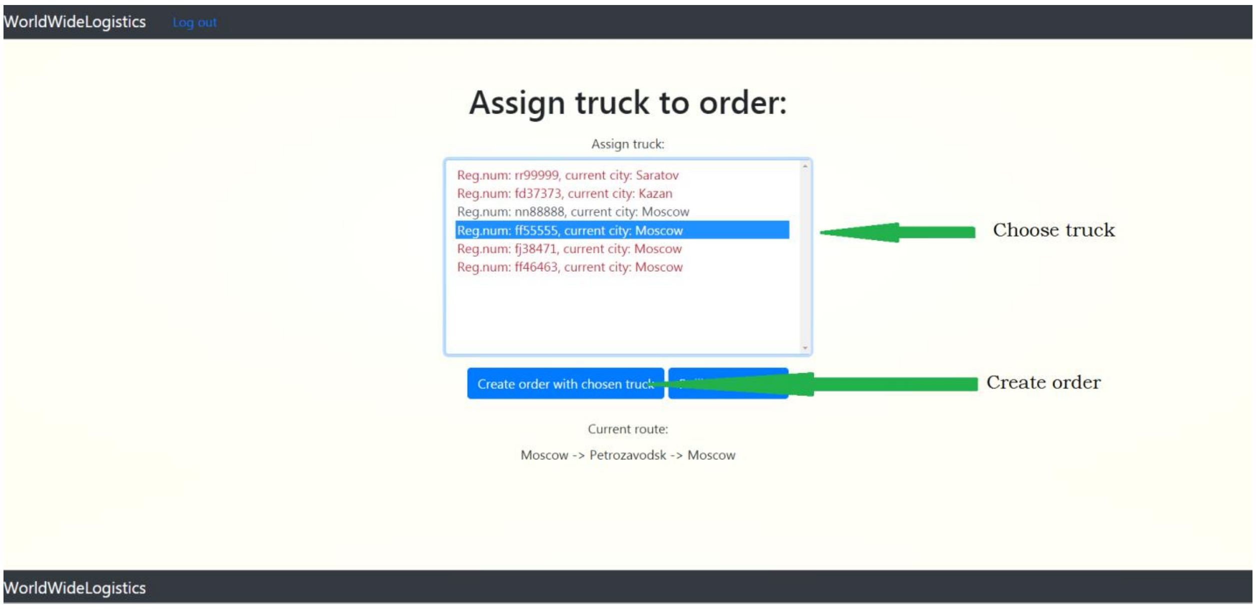
Water: from Saint-Petersburg to Petrozavodsk
Chairs: from Kazan to Pskov
Apples: from Petrozavodsk to Moscow
Oranges: from Saint-Petersburg to Petrozavodsk

Current route:

Petrozavodsk -> Moscow

WorldWideLogistics

4. Assign truck and push “Create order with chosen truck” button:


WorldWideLogistics Log out

Assign truck to order:

Assign truck:

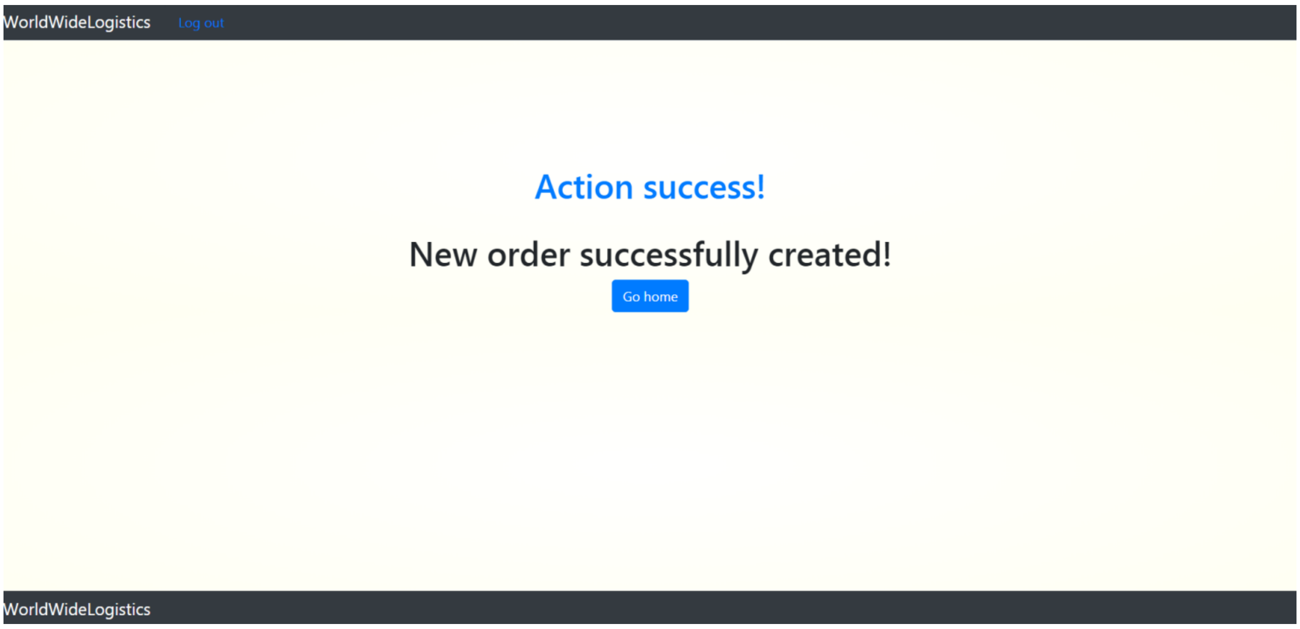
Reg.num: rr99999, current city: Saratov
Reg.num: fd37373, current city: Kazan
Reg.num: nn88888, current city: Moscow
Reg.num: ff55555, current city: Moscow
Reg.num: fj38471, current city: Moscow
Reg.num: ff46463, current city: Moscow

Current route:

Moscow -> Petrozavodsk -> Moscow

WorldWideLogistics

5. If action was successful you’ll see success message:


WorldWideLogistics Log out

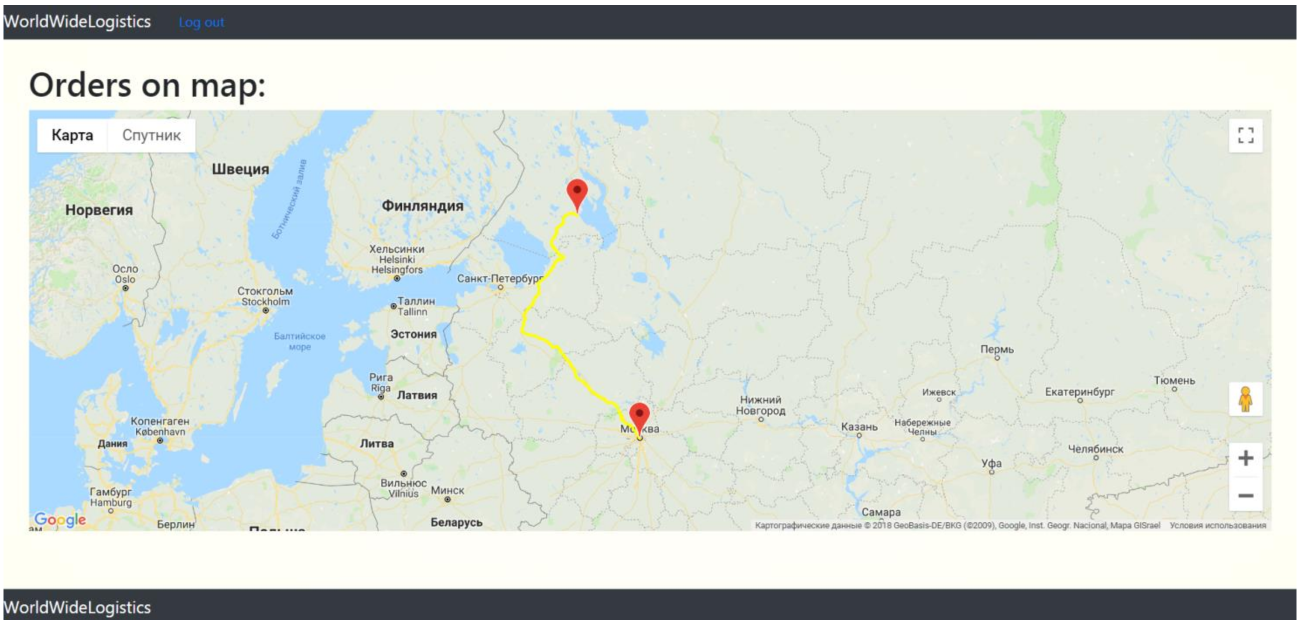
Action success!

New order successfully created!

Go home

WorldWideLogistics

6. To see order on Google Maps press “Show on map” button on home page:


WorldWideLogistics Log out

Orders on map:

WorldWideLogistics

## Additional features

**Build**

Application build and deployed to Tomcat Application Server via maven by command *mvn tomcat7:deploy.*

**Logging**

Application logging configured with log4j library.

**Tests**

Application uses JUnit4 , Mockito and Selenium libraries for testing.

**Google Maps API**

Application uses Google Maps API for showing orders on Google Maps.