Technical solution description

**LogApp**

Author: Sumenko Artem

1. **Content**

During this work an application that emulates the information system of some transportation company was developed. Managers are able to manage company staff, map, trucks and orders (create, read, update, delete) Drivers are able to get information about their appointment (order’s number, points, co-drivers information, truck’s information) and manage order’s state (load and deliver cargos, start and finish shifts). Generally, all functionality from technical task was implemented.

Besides, additional features was implemented:

* Map creation: Drivers are able to see the destination points of an order via the Google Maps
* Localization: Application can be used in English and German languages

## 2. Used technologies

Used instruments:

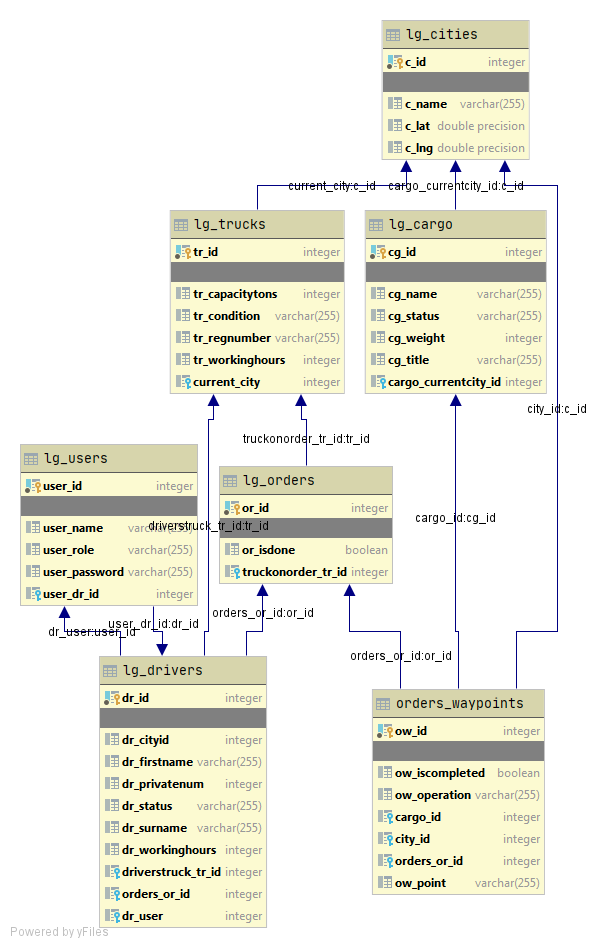
* Intellij IDEA Ultimate 2020
* Apache Maven 3.6.3
* JBoss Wildfly 19.0.0 Final (for rest app)
* Apache Tomcat 8.5.51 (for main app)
* PostgreSQL 9
* pgAdmin 3

Technologies:

* Spring 5.2.2
* Hibernate 6.1.2
* EJB 3
* JSF 2.1
* Jax-rs
* Log4j 1.2.17
* JUnit 4
* Bootstrap 4
* JQuery
* Kafka (as MQ)

# Implementation

* 1. **Database scheme:**

****

## Architecture

The application has 4 layers defining its architecture. Each layer uses only the functionality of the previous layer.

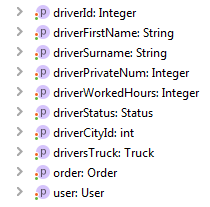
1. Model
   * Entities.
   * DAO.
2. Services, which include business logic.
3. Controllers process requests and parameters from UI and dispatch queries.
4. View layer consists of JSP-pages for the main application and JSF-pages for rest-client.

## Model layer

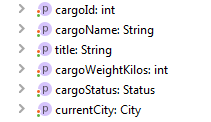
## The entities

My application has 8 entities, defined in the corresponding classes:

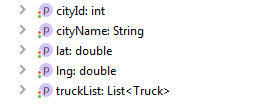
* DriverEntity



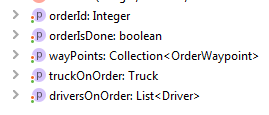
* CargoEntity



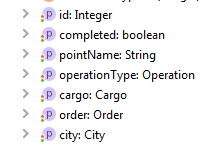
* CityEntity



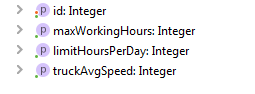
* OrderEntity



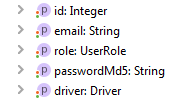
* OrderWaypointEntity



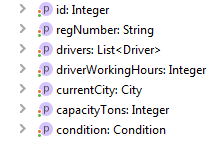
* ShippingCatalog



* UserEntity



* TruckEntity



* + 1. **DAO**

DAO layer consists interface and implementation packages.

There is an implementation with the realization of the methods for each interface. The realization of the CRUD methods is done in the implementation classes.

## Service layer

Service layer consists of API package and its implementation. Each of the service implementation classes is annotated with a @Service annotation. Transactions are declared at this level, each method of the service is performed in a transaction. The methods are annotated with @Transactional annotation, so Spring handles transactions.

Services:

* *CityService*. It does the functionality for cities management.
* *DriverService*. It provides functionality for working with driver entities. CRUD operations, specified selection, etc.
* *UserService*. It saves new users, finds new users, change user’s role.
* *OrderService*. All functionality related to order management.
* *TruckService*. All functionality related to truck management.
* *CargoService* All functionality to work with cargoes.
* *CustomUserDetailsService* This service is used by Spring Security to validate users of the application. Implements “org.springframework.security.core.userdetails.UserDetailsService”.
* *OrderWaypointService* All functionality to work with points on order.
  1. **Controller layer**

This layer holds the classes which are responsible for the interaction between UI and service classes. Controllers are based on the Spring MVC framework, have @Controller annotation.

Controllers:

* DriverController
* AccessDeniedController
* FrontPageController
* MyOrderController
* OrderController
* BasicRestController
* TruckController
* GlobalExceptionHandler

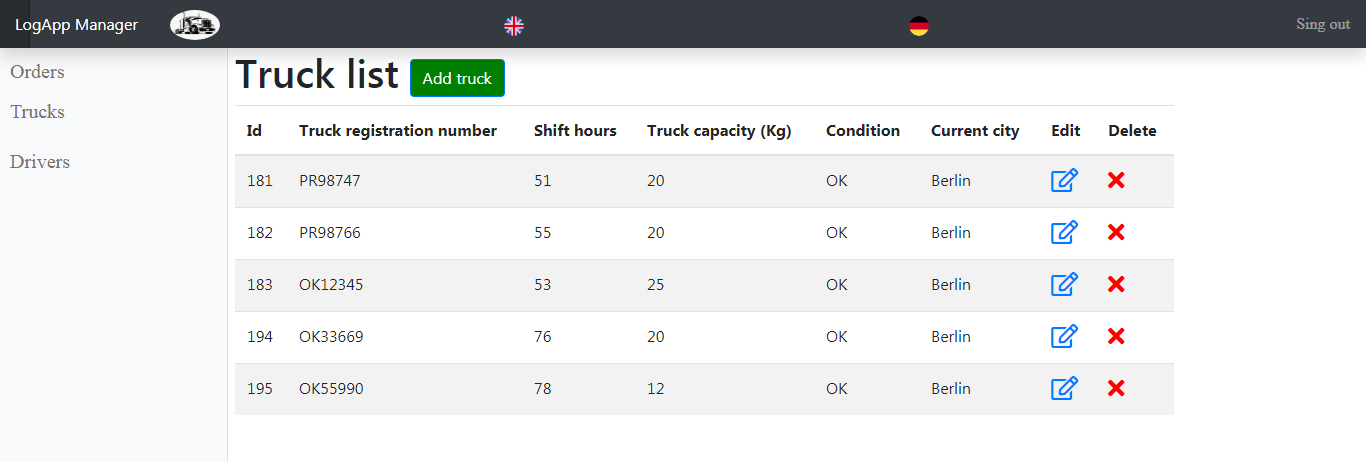
GlobalExceptionController handles all exceptions from controller layer. It is annotated @ControllerAdvice.

* 1. **View layer**

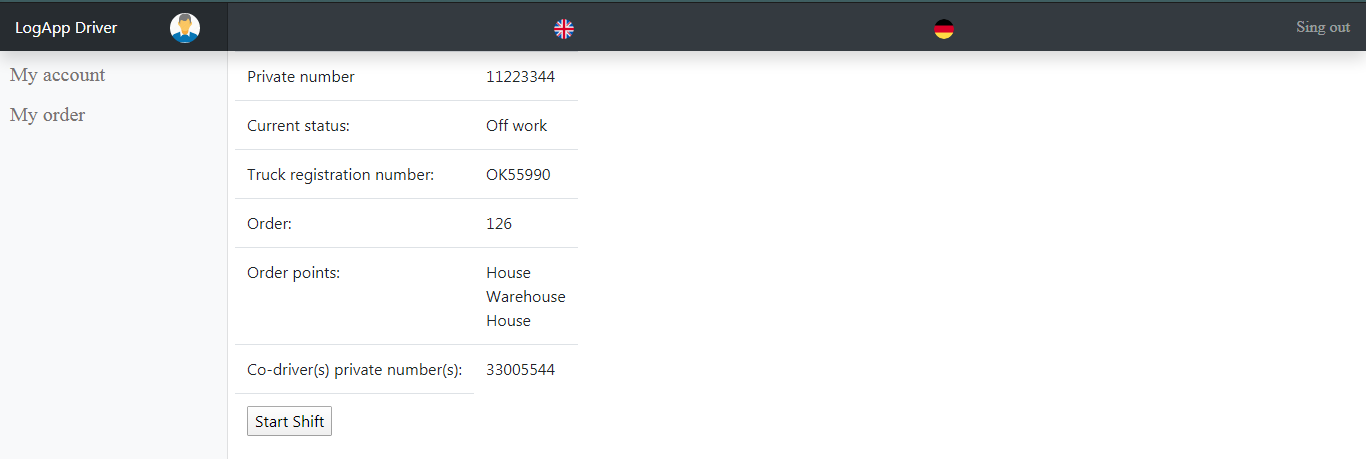
This layer is responsible for UI. Most part of design in main application is made using Bootstrap framework. All jsp, js, css are in webapp package. Rest-client is designed using JSF and PrimeFaces.

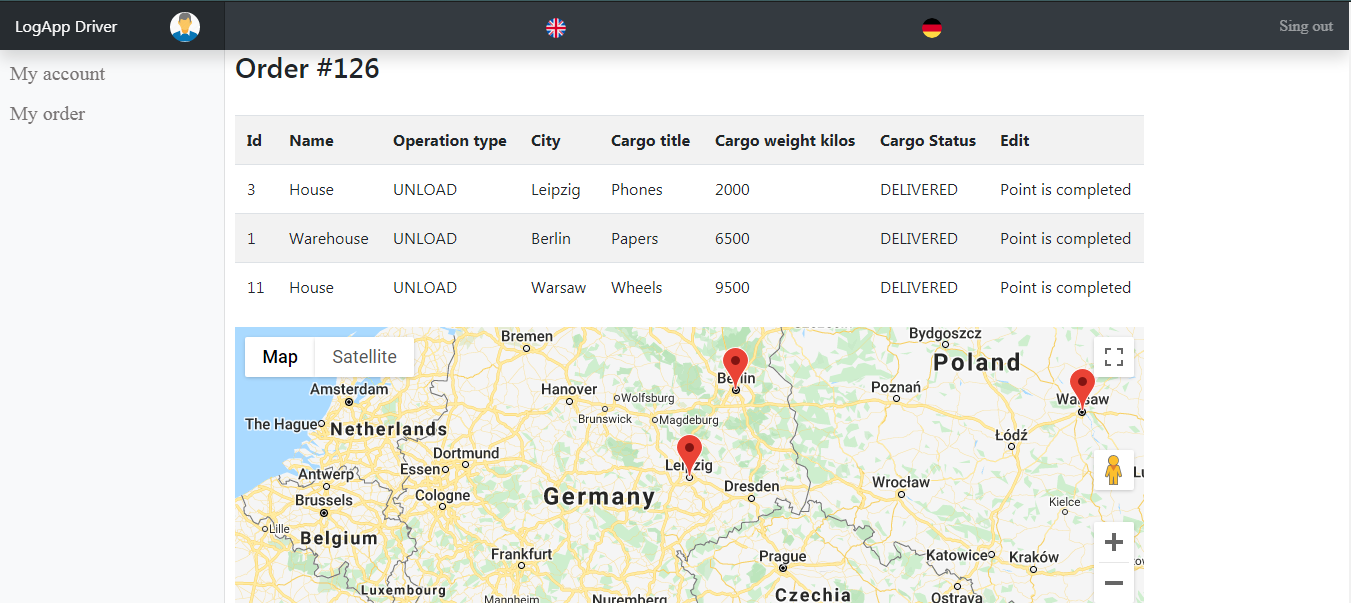
Some pages:

Managers page

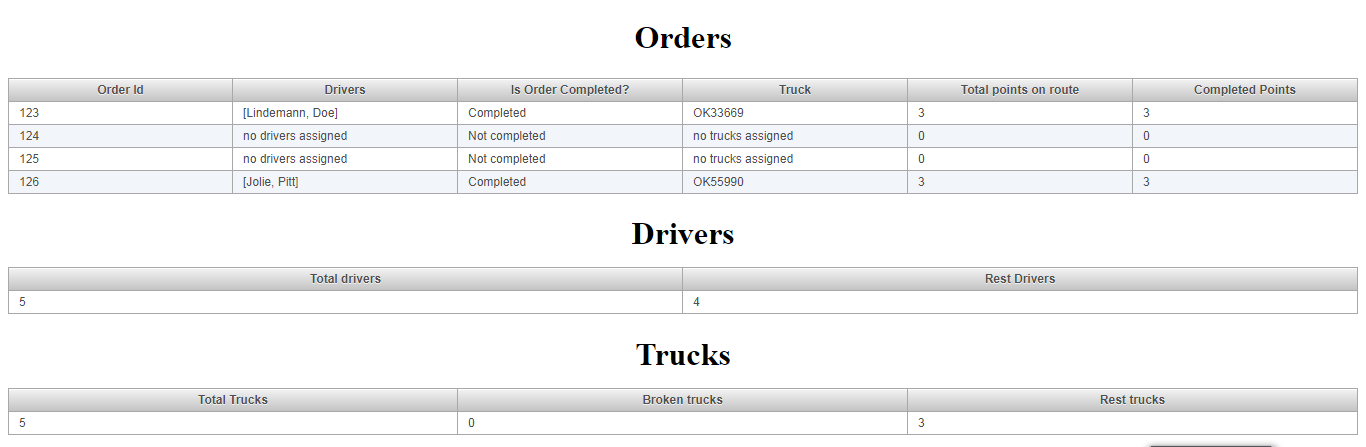


Driver Pages:





Rest application view:



1. **Security**

Application security provided by Spring Security. Spring security is configured in SecurityConfig.class. Login and password are stored in database. Password is encoded. So drivers and usual users can’t access to manager’s pages without correct login and password.

1. **Logging**

Log4j is used for logging. It is configured to make FATAL, ERROR, WARN  logs to file. Each module has own log file. Log files are stored on wildfly and tomcat servers.