Technical Solution Description
“Logiweb”

Author: Ihsanov D.F.

Saint-Petersburg
2021

Contents

1 Introduction 3
2 Business Scenarios 4
2.1 Accessing profile 4
2.2 Interaction with trucks 4
2.2 Interaction with drivers 4
2.3 Interaction with cargos 4
2.4 Interaction with orders 4
2.4 Interaction with driver profile page 4
3 Architecture 6
3.1 Description 6
3.2 Endpoints 6
4 Tech Stack 8

1 Introduction

This document is the Logiweb project's technical solution description. Logiweb is a web application that simulates system of a certain company that carries out the transportation of different goods. It will be useful for both company employees and drivers.

2 Business Scenarios

2.1 Accessing profile

Both employees and drivers can access their own profiles using username and password on the login page. After logging in employee will see a page with his/her credentials and a navbar. Drivers will see a page with a current order.

2.2 Interaction with trucks

By clicking on the “Фуры” button in the navbar, an employee can see cards with current trucks and information about them. An employee can easily update or delete trucks by clicking on the buttons at the right side of the card. A new truck can be added by clicking button “Добавить фуру”. Note that employee cannot delete busy trucks.

2.2 Interaction with drivers

By clicking on the “Водители” button in the navbar, an employee can see cards with current drivers and information about them. An employee can update or delete drivers by clicking on the buttons at the right side of the card. A new driver can be added by clicking button “Добавить водителя”. Employee cannot delete busy drivers.

2.3 Interaction with cargos

By clicking on the “Грузы” button in the navbar, an employee will see cards with current cargos and information about them. An employee can interact with cargos in the same way as said above about drivers and trucks.

2.4 Interaction with orders

By clicking on the “Заказы” button in the navbar, an employee will see cards with current or completed orders and information about them. Clicking the “Создать пустой заказ” button will cause creation of an empty order with only a creation date. First, an employee needs to add cargos to the order, then assign a truck, driver from a list of suitable trucks and drivers. At the end, an employee should set start and end dates for the order, because only after this driver will see the order in his own profile.

2.4 Interaction with driver profile page

Driver can always change their status by using the “Изменить статус водителя” button on their profile page. On the same page, the driver can see all the information

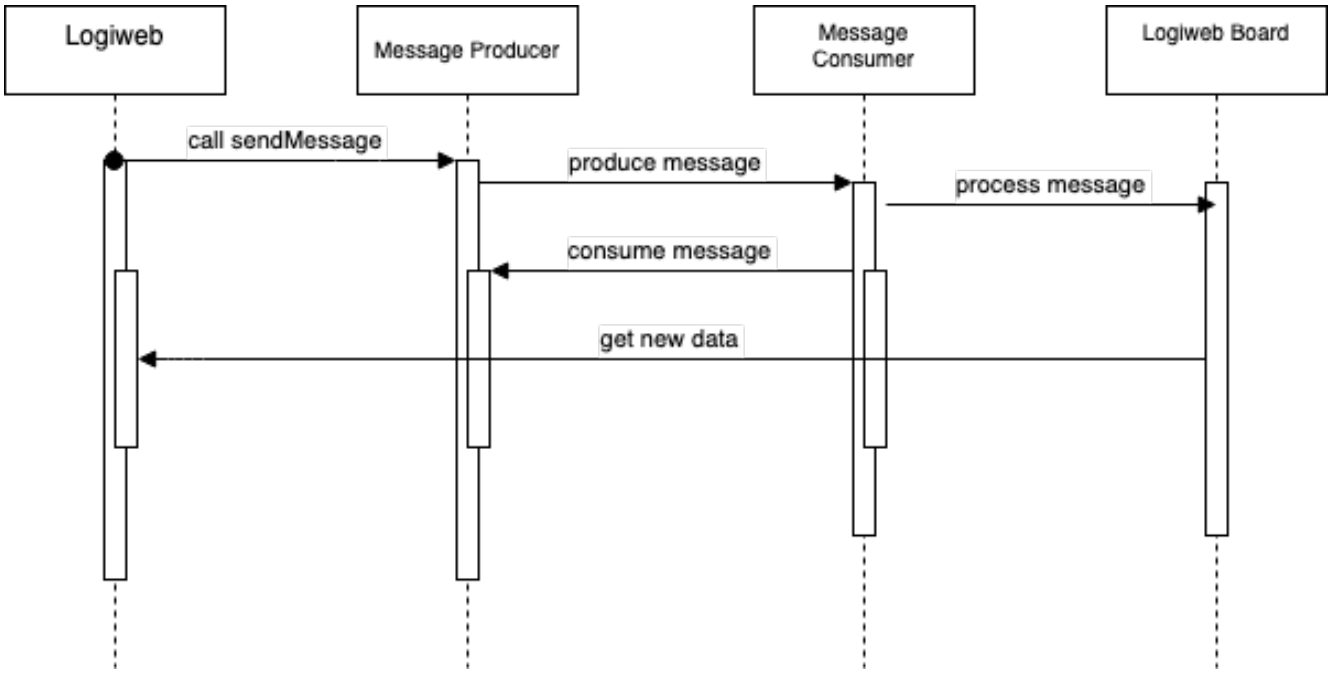
about the current order. Waypoint's status can be easily changed by clicking the “Изменить статус заказа” button. When all waypoints are done, driver can click the “Завершить заказ” button to complete the order. As soon as the button is pressed, on the orders page an employee will see the order with the status “completed”.

3 Architecture

3.1 Description

The Logiweb app is built using the MVC architectural pattern. It has DAO layer classes to interact with database, service layer classes for business logic and controller layer classes to processing all the incoming requests, manipulate data using the model component and interact with the views to render the final output pages. There are DTO classes which are used to transfer data between html page forms and controllers.

App also has a Kafka message producer. This producer sends messages to Logiweb Board app's message consumer every time when some order, truck or driver was updated, deleted or created. Consumer uses those messages for updating board's database data. Below you can see message queueing diagram.



3.2 Endpoints

1. [/login] Access login page;
2. [/logout] Logout from account;
3. [/driverPage] Get driver profile page;
4. [/employeePage] Get employee's profile page;
5. [/employeeTruck] Get employee's trucks page;
6. [/employeeTruck/trucks/{id}] Get truck by given id;
7. [/employeeTruck/trucks] Get all trucks;

8. [/employeeDriver] Get employee’s drivers page;
9. [/employeeDriver/drivers/{id}] Get driver by given id;
10. [/employeeTruck/drivers] Get all drivers;
11. [/employeeCargo] Get employee’s cargos page;
12. [/employeeCargo/{id}] Get cargo by given id;
13. [/employeeOrder] Get employee’s orders page;
14. [/employeeOrder/edit/{id}] Get employee’s order edit page by given id;

4 Tech Stack

Application was built using Java and Spring Framework. Hibernate was used for mapping object-oriented domain model to a relational PostgreSQL database. Connection pooling mechanism was implemented through HikariCP. Tests were written using Junit 5, Mockito, AssertJ and PostgreSQL embedded database. Log4j was used for logging. Template engine – Thymeleaf.

Below you can see database's ER Diagram.