Nikita Voronin
SBB Project
T-Systems Java School
St. Petersburg, Russia
November 11, 2020

Technical Solution Description

Introduction

The goal of the project is to develop a web application simulating an information system of a railway company.

The application consists of two standalone modules:

a) a client-server application connected over a network with data stored in a database on the server side;

b) a client application representing a live timetable, which automatically refreshes if any changes have been made in the first module.

The modules communicate with each other via a messaging system. Once the client module receives an update message from the other module, it sends a REST GET request to fetch updated data and refreshes the web page.

User Stories

The application provides a different scope of available actions based on two distinct roles – User and Admin. User refers to a client, while Admin is the company's employee. Below is a list of available actions for each role.

User

- searching for a train to get from Station A to Station B on a given date
- viewing a timetable for a specific station
- purchasing a ticket providing that the following conditions are met:
    - there are empty seats available on the train
    - no passenger with the same first name, last name, and birthdate on the train
    - at least 10 minutes left until departure
- viewing and cancelling purchased tickets

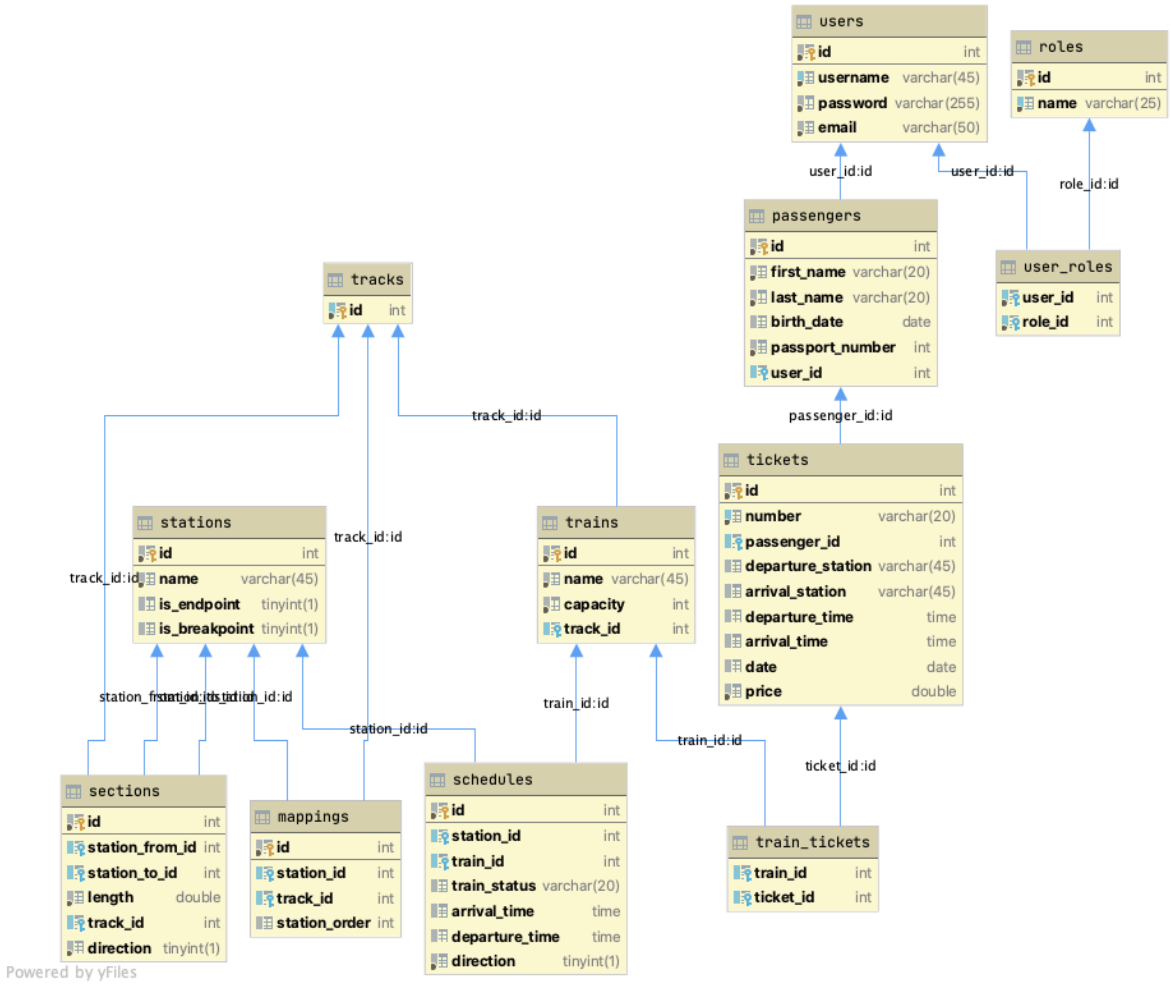
Admin

- adding new stations and trains
- viewing a list of all passengers on a specific train
- monitoring current trains with the ability to delay or cancel them

Technology Stack

Java 8		Jakarta EE	
IntelliJ IDEA Ultimate		Apache Maven	
Tomcat 9		Wildfly 21	
Spring 5 Framework (MVC)		Spring Security	
MySQL		JPA (Hibernate)	
JSP		JSF	
PrimeFaces		OmniFaces	
JMS (Apache ActiveMQ)		REST API	
jUnit 5		Mockito	
Lombok		MapStruct	
Apache Log4j2		Apache Commons	
Google Gson		Eclipse Jersey	
Bootstrap 4		JavaScript	

Data Model

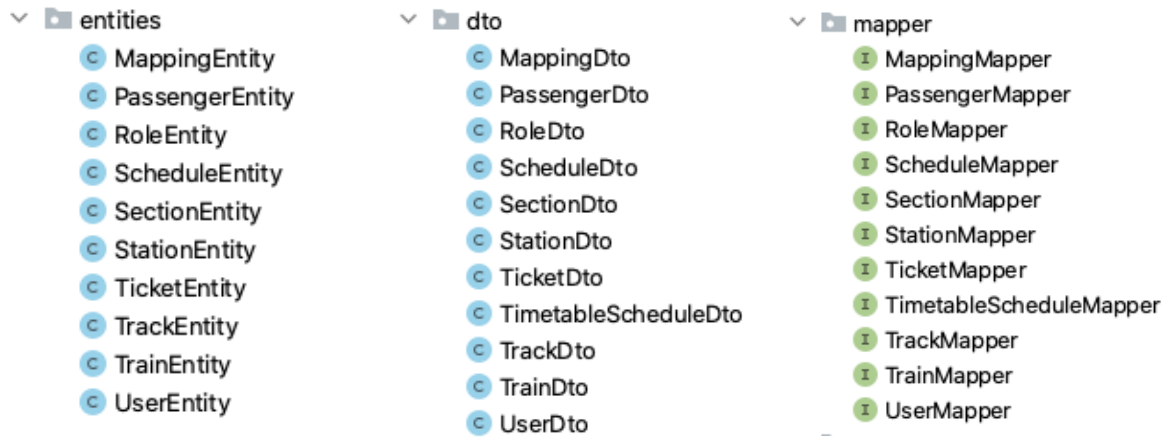


Database schema representing the relationships between entities

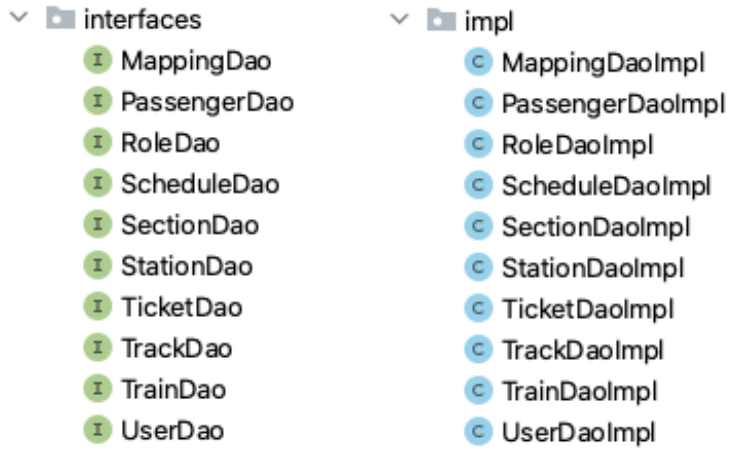
Application Architecture

The client-server application follows a standard MVC pattern:

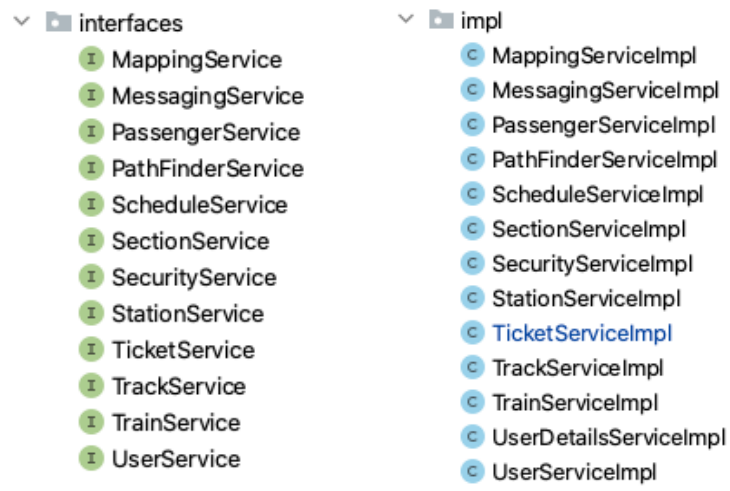
Model



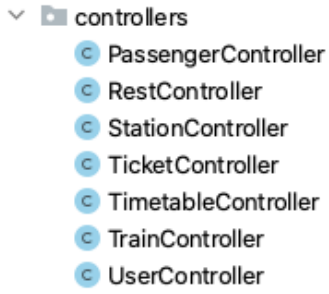
DAO



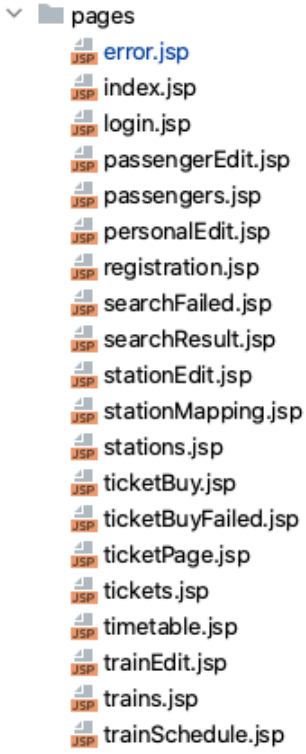
Service



Controller

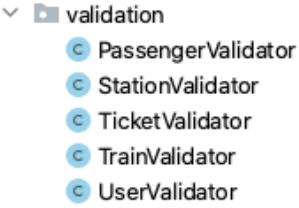


View

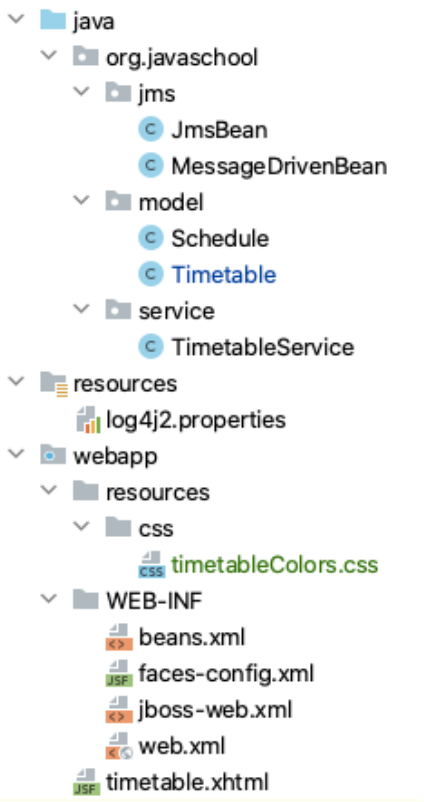


Config

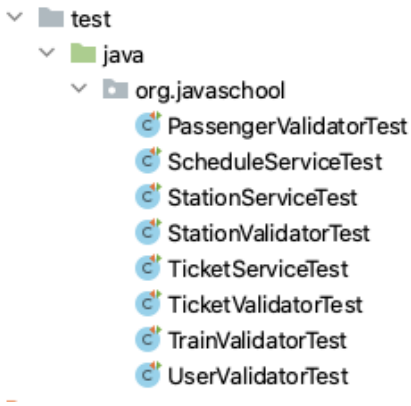
Validation



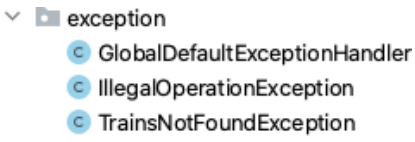
Client application



Tests (66 unit tests)

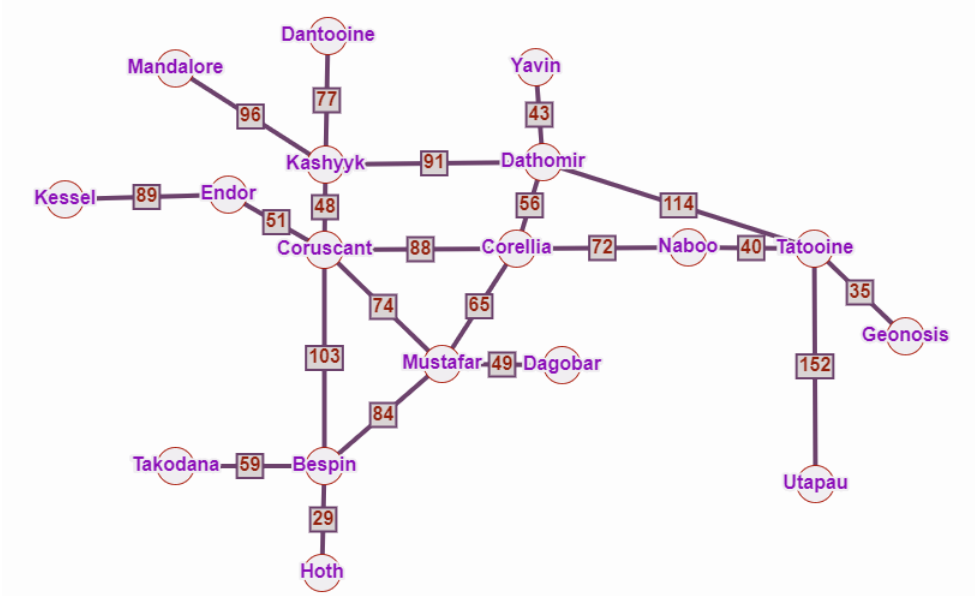


Exception

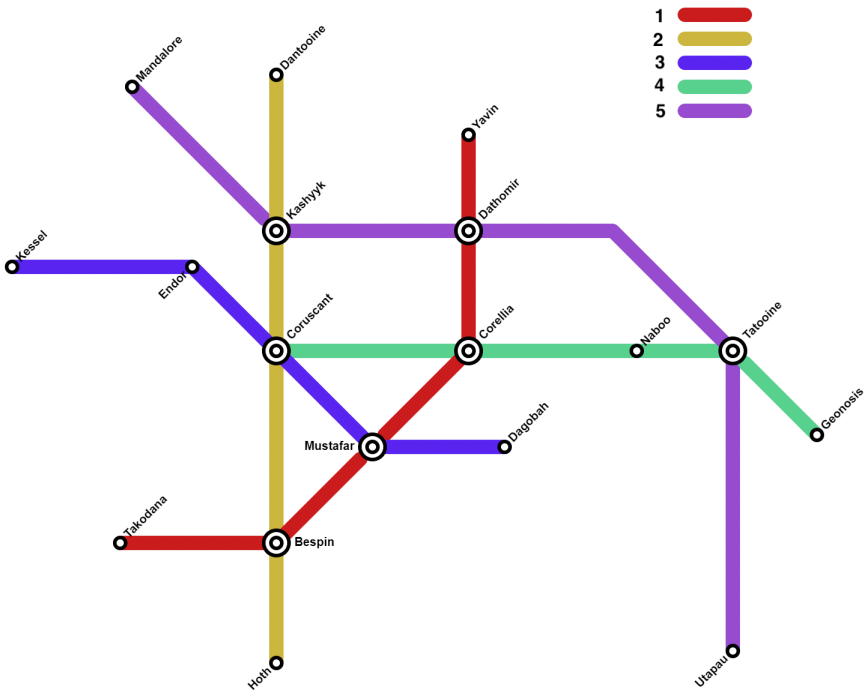


Railway System

The modeled railway system is represented by a weighted graph where stations serve as nodes. Pathfinding is implemented using Dijkstra's shortest path algorithm.



The railway system consists of five tracks. Each train is bound to a specific track, so changing tracks implies changing trains (similar to how a subway operates).



Further Improvements

The application's functionality can be further enhanced by adding the following features:

- optimization of heavy pages to make them fetch data more efficiently
- multiple route options offered in the search results
- ability to book a return ticket
- pagination for long lists
- PDF ticket generation
- internationalization
- code refactoring