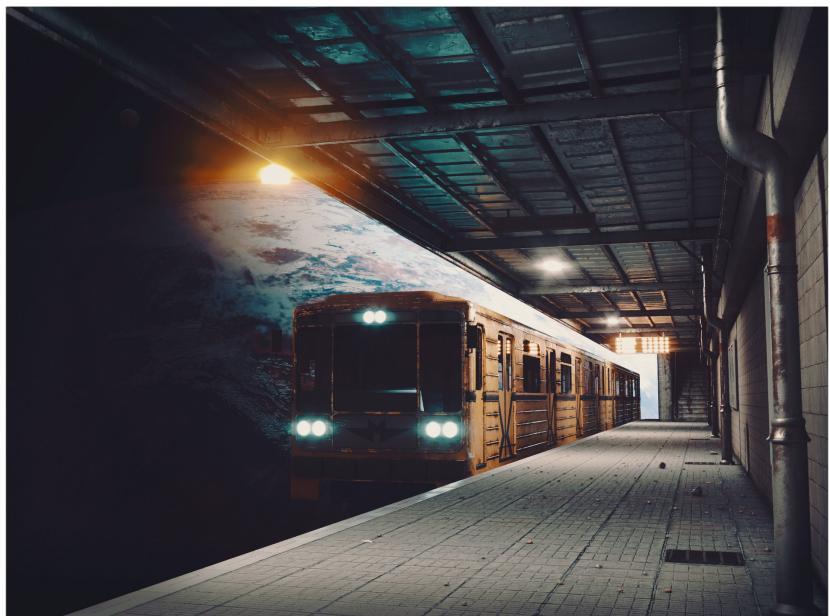


Nikita Voronin  
SBB Project  
**T-Systems Java School**  
St. Petersburg, Russia  
November 11, 2020

# Technical Solution Description



## **Introduction**

**3**

## **User Stories**

**4**

## **Technology Stack**

**5**

## **Data Model**

**6**

## **Application Architecture**

**7**

## **Railway System**

**10**

## **Further Improvements**

**11**

# 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

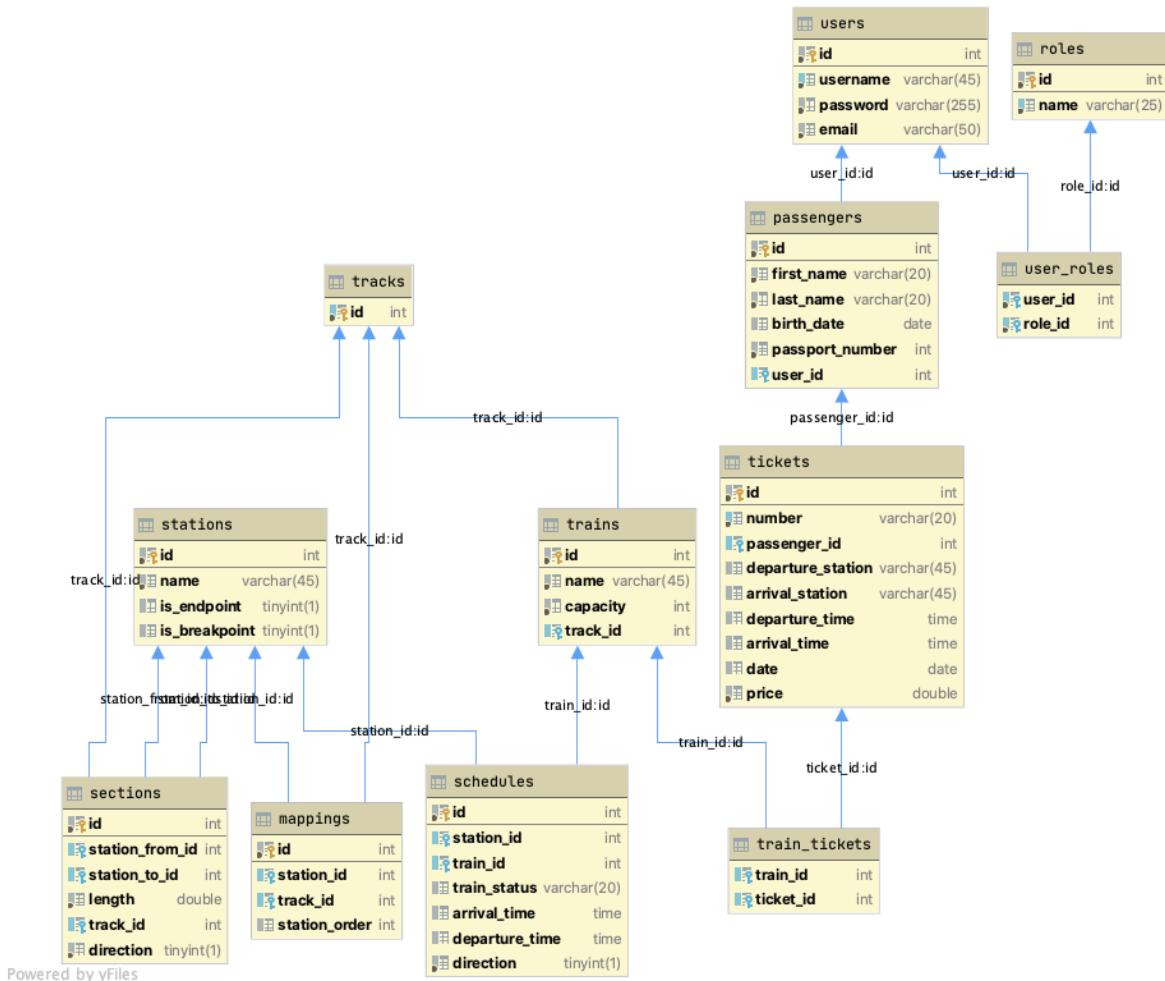
### Admin

- 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



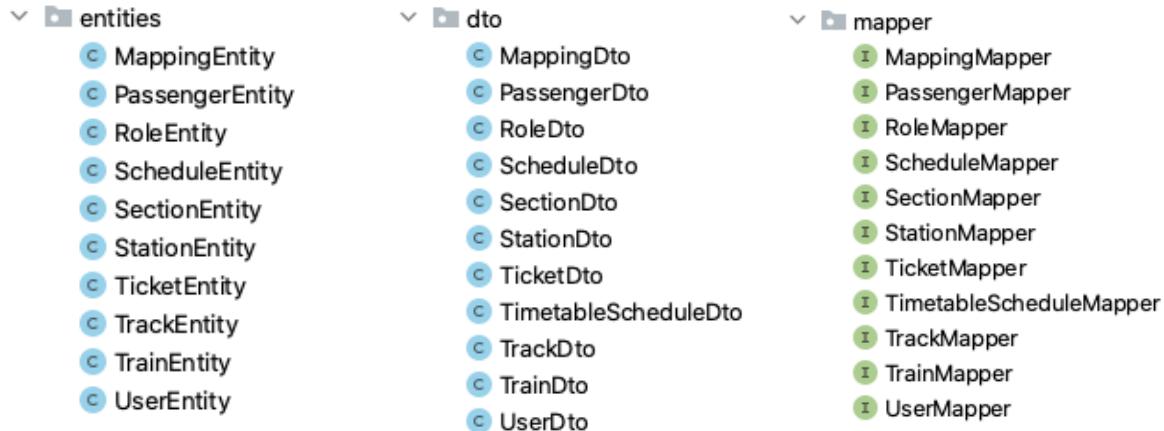
Powered by yFiles

*Database schema representing the relationships between entities*

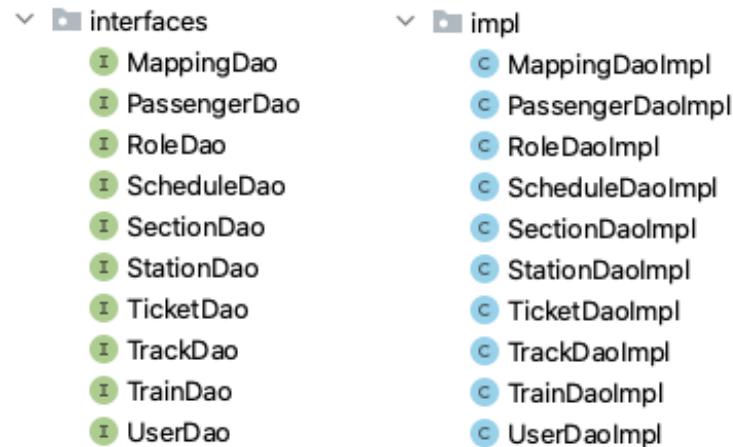
# Application Architecture

The client-server application follows a standard MVC pattern:

## Model



## DAO



# Service

interfaces	impl
I MappingService	C MappingServiceImpl
I MessagingService	C MessagingServiceImpl
I PassengerService	C PassengerServiceImpl
I PathFinderService	C PathFinderServiceImpl
I ScheduleService	C ScheduleServiceImpl
I SectionService	C SectionServiceImpl
I SecurityService	C SecurityServiceImpl
I StationService	C StationServiceImpl
I TicketService	C TicketServiceImpl
I TrackService	C TrackServiceImpl
I TrainService	C TrainServiceImpl
I UserService	C UserServiceImpl

# Controller

controllers
C PassengerController
C RestController
C StationController
C TicketController
C TimetableController
C TrainController
C UserController

# Config

config
C MessagingConfig
C PersistenceJPACConfig
C WebAppInit
C WebConfig
C WebSecurityConfig

# View

pages
JSP error.jsp
JSP index.jsp
JSP login.jsp
JSP passengerEdit.jsp
JSP passengers.jsp
JSP personalEdit.jsp
JSP registration.jsp
JSP searchFailed.jsp
JSP searchResult.jsp
JSP stationEdit.jsp
JSP stationMapping.jsp
JSP stations.jsp
JSP ticketBuy.jsp
JSP ticketBuyFailed.jsp
JSP ticketPage.jsp
JSP tickets.jsp
JSP timetable.jsp
JSP trainEdit.jsp
JSP trains.jsp
JSP trainSchedule.jsp

# Validation

validation
C PassengerValidator
C StationValidator
C TicketValidator
C TrainValidator
C UserValidator

## Client application

```
java
org.javaschool
  jms
    JmsBean
    MessageDrivenBean
  model
    Schedule
    Timetable
  service
    TimetableService
resources
  log4j2.properties
webapp
  resources
    css
      timetableColors.css
WEB-INF
  beans.xml
  faces-config.xml
  jboss-web.xml
  web.xml
timetable.xhtml
```

## Tests (66 unit tests)

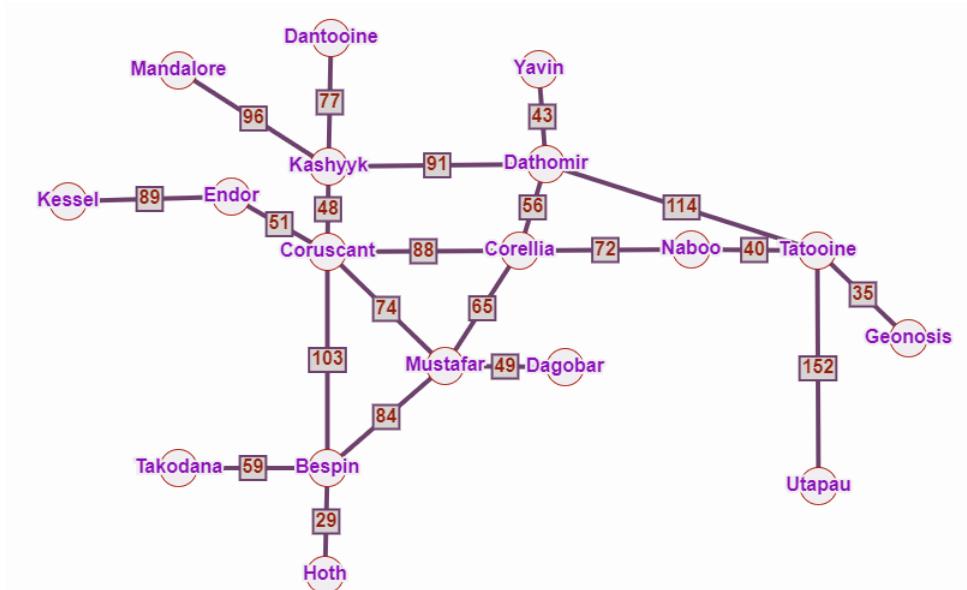
```
test
java
org.javaschool
  PassengerValidatorTest
  ScheduleServiceTest
  StationServiceTest
  StationValidatorTest
  TicketServiceTest
  TicketValidatorTest
  TrainValidatorTest
  UserValidatorTest
```

## Exception

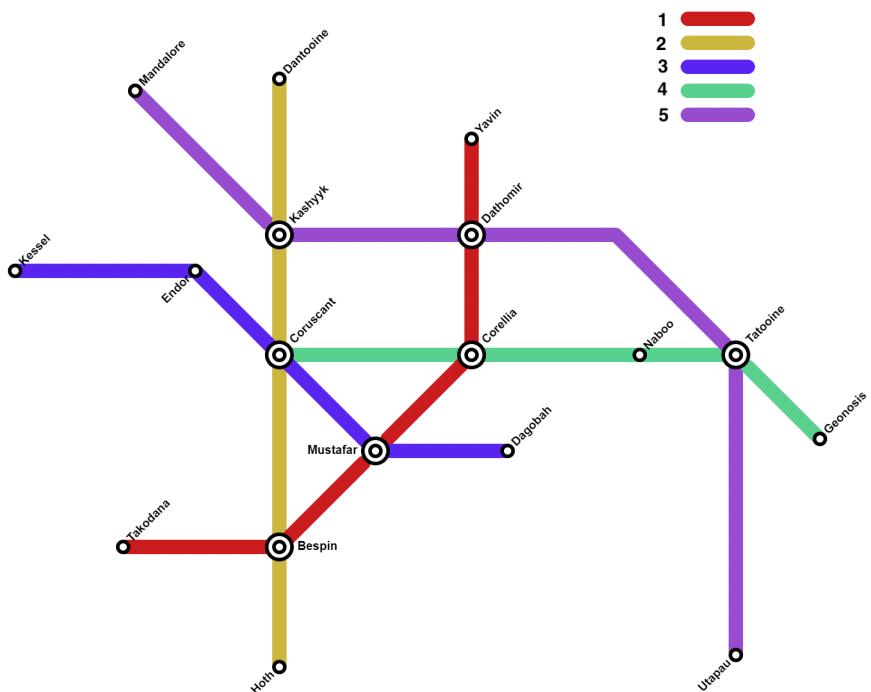
```
exception
  GlobalDefaultExceptionHandler
  IllegalOperationException
  TrainsNotFoundException
```

# 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