Technical solution description

Railway Travels

Chernov Ilya
2018

Table of contents

Introduction

This system was developed for automation of the process of interaction of members of railway company and their clients. The system must allow to publish and edit information about departures for administrators and to watch this information and to purchase tickets for clients.

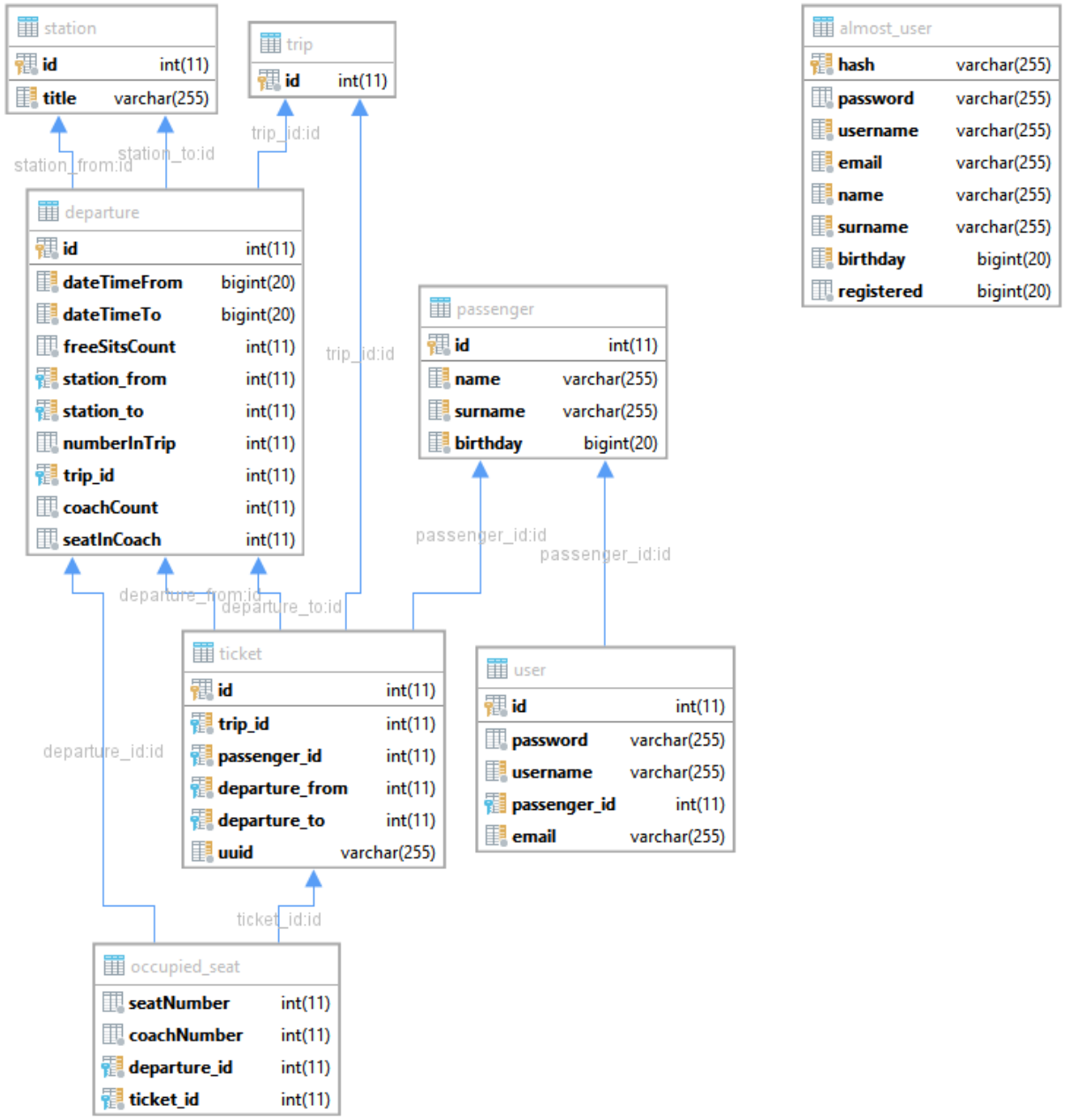
Used technologies and frameworks

- MySQL 5.6
- Wildfly 10.*
- IntelliJ IDEA 2018
- WebStorm 2018
- Maven 3.*
- Tomcat 9
- RabbitMQ 3.7.7
- Spring Framework 5.0.6
- AngularJS 1.7.2
- Angular 6.0.3
- Node 9.2.0
- AngularCLI 6.0.8
- Angular Material
- Bootstrap
- JSP
- JSF 2.3
- PrimeFaces 6.2
- Rest WS
- JPA 2.0
- EJB 3.*
- Flyway 5.1.4

Additional features

- Registration through link in mail message
- Registration through google reCaptcha
- Opportunity to search the way from one station to another with transfers
- Opportunity to return a ticket
- Send mail message with ticket in PDF format with barcode
- Opportunity to edit or delete defined departures

Database scheme



Description of table and relations

Table name	Table description
user	Contains information about user. Not every user is passenger.
passenger	Contains information about passenger.
almost_user	Contains information about user who did not confirm account through link in email message.
trip	This entity unites departures in one trip.
departure	Contains information about departure.
station	Contains titles of stations
ticket	Contains information about passenger's ticker: passenger, trip, first departure, last departure and occupied seats.

occupied_seat	Contains information about occupied seat in one departure. It needs for constraint (2 passengers can't occupy the same seat on the same departure)

Description of implementation of the model

There is a core table in a model: departure. It contains next information:

1. Departure station and time
2. Arrival station and time
3. Coaches count in train
4. Serial number in whole trip

Trip contains one or more departures.

Next tables are responsible for next constraints for buying ticket process: ticket, occupied_seat. Ticket contains foreign keys trip_id and passenger_id and this combination is unique which is not allow to register one passenger to the trip more than one time.
Ticket is the owner of relation with occupied_seat. Occupied_seat contains information about seat number, coach number and departure and this combination is unique which is not allow to register more than one passenger on concrete seat in departure.

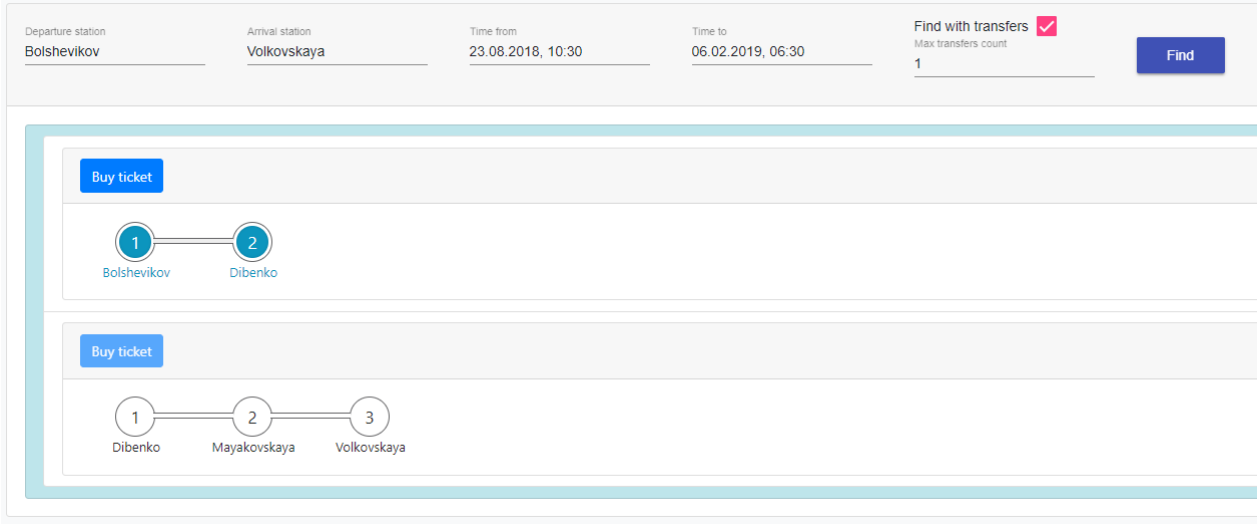
Description of modules

Module name	Module description
config	Contains configuration classes.
controller	Contains classes with next roles: controller, controller advice, rest controller.
service	Contains service classes with business logic, business exceptions and dto converters.
dto	Contains dto objects and few validators connected with it.
dao	Contains DAO objects for working with database.
entity	Contains entities mapped to tables in database and few converters to convert some data to objects.

Description of user interface

1. Filters

There is an opportunity to find a way with or without transfers. Next filter allows client to enter necessary data and try to find what he need. For station inputs there is autocompletes components provided by Angular Material. For date inputs there is datetimepickers components provided by Bootstrap DateTimePicker. After response is shown client can choose departures to travel and buy ticket.



2. Templates

Views built by JSP or angular templates. JSP is used for login and registration pages, angular templates are used for client and admin pages.

3. Frameworks and libraries

- Angular is used on client page
- AngularJS is used on server page
- Jquery as dependency for bootstrap
- Jquery-ui as dependency for bootstrap-datetimepicker
- Bootstrap for styling pages
- Bootstrap-datetimepicker
- Angular Material

Description of services

Service name	Service description
AlmostUserService	Contains some CRUD operations for AlmostUser entity and job which executes at midnight and removes old entries.
DepartureService	Contains CRUD operations for Departure entity.
MailSender	Contains asynchronous methods for sending emails with html templates and attachments.
PassengerService	Contains CRUD operations for Passenger entity and logic for buying and returning tickets.
RabbitService	Contains asynchronous methods for sending an object to RabbitMQ exchange.
ReCaptchaApiClient	Contains method for verifying reCaptcha code by google.api
SearchTripWithTransfers	Contains logic to find necessary ways with transfers.
StationService	Contains CRUD operations for Station entity.
TripService	Contains CRUD operations for Trip entity.
UserService	Contains CRUD operations for User entity.

Description of entities and DAO

Ticket and OccupiedSeat entities describe ticket.

Trip and Departure entities describe when and where train will go.

Station entity describes stations.

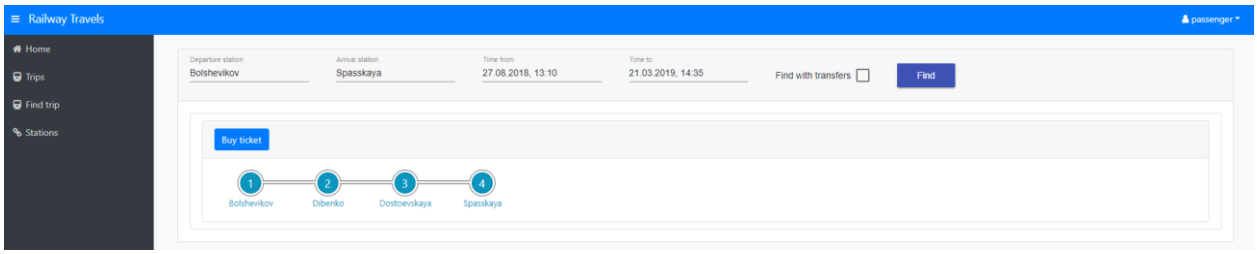
User and Passenger entity describe users and passengers. If user has passenger then it's client, otherwise – admin.

AlmostUser entity describes users which went through a registration form, but didn't follow the link in email message.

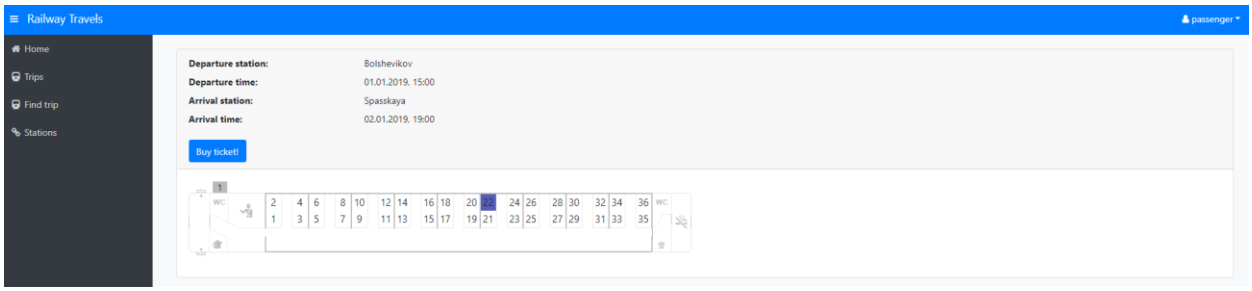
DAO is generic. There is an abstract class which contains most common DAO operations and specific DAO must extend this class to have these common operations automatically.

Screenshots of main pages

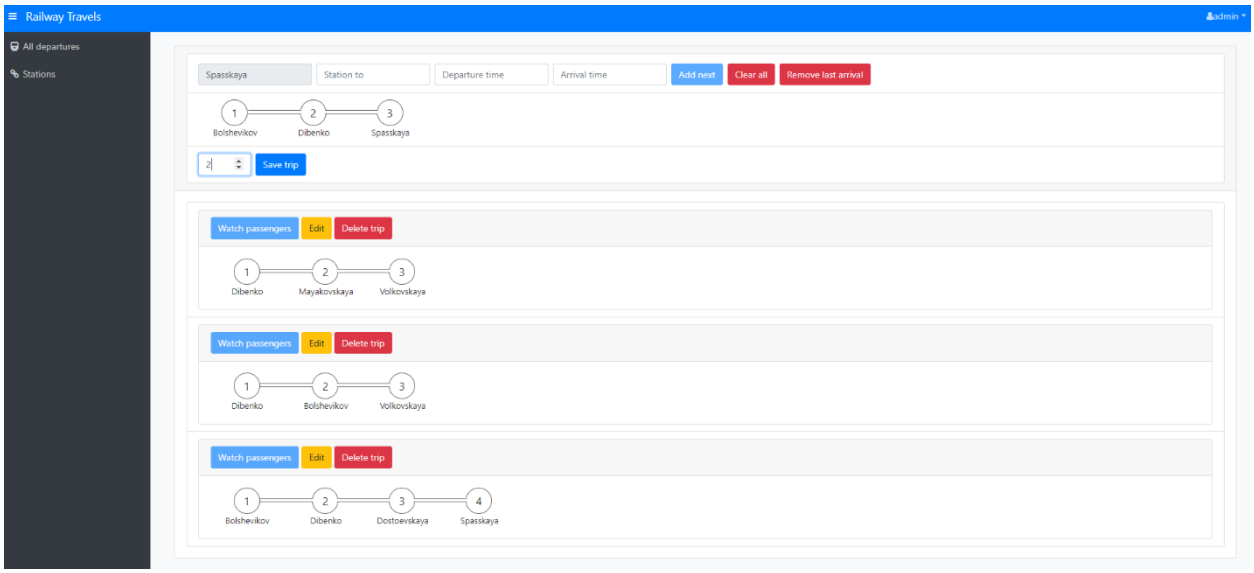
1) Find trip view



2) Choose ticket view



3) Admin view



Build and deploy

To build and deploy application to wildfly AS run **mvn clean install**.

Also you must configure **mail.properties** in **resource** directory of main module if you want to correct sending messages. You can configure connection to database and RabbitMQ in **application.properties** file.