Technical description

**JUSTRAILWAYS PROJECT**

Author:

Alexander Prendota

Saint-Petersburg,2017

Content

[Task 3](#_Toc483151990)

[Project goals 4](#_Toc483151991)

[Application description 5](#_Toc483151992)

[Used technologies 6](#_Toc483151993)

[Database Model 7](#_Toc483151994)

[UI Model 8](#_Toc483151995)

[System infrastructure 9](#_Toc483151996)

[System architecture 10](#_Toc483151997)

[*The Justrailways application* 10](#_Toc483151998)

[*The timetable schedule application:* 13](#_Toc483151999)

[*The schedule by station appication:* 13](#_Toc483152000)

[Code quality 14](#_Toc483152001)

[Additional features 16](#_Toc483152002)

[GUI 17](#_Toc483152003)

[*JustRainWays* 17](#_Toc483152004)

[*TimeSchedule* 17](#_Toc483152005)

[*TimeStatations* 17](#_Toc483152006)

[Future improvement 20](#_Toc483152007)

# **Task**

To develop web-application for JustRailways company. The application have to perform the required user’s cases.

**Users cases:**

*For clients***:**

* Getting schedule between two stations.
* Getting schedule by name of station.
* Purchase tickets with different conditions
* To change clients information.

*For managers:*

* To create and manage new trains.
* To create and manage new stations.
* To find all clients.
* To manage client.
* To view passengers on specific train.
* To send notification to all clients.
* To make schedules.

Additionally to develop co-application with electronic tableboard.

# **Project goals**

* The robust, useful and reliable system.
* Cohesive data model.
* Intuitive, user-friendly interface.
* Cohesive data model.
* Separate access to different system’s part.

# **Application description**

Web-application has two type of user: clients and managers.

Clients can find different train-schedules by one station departure or two station and specific date. Client also can buy tickets and change his information.

Managers can add, update, delete schedules, train, station and users. Managers also can looking for list of user on specific train and send notification to all users.

Before using application user must sign up in system. After login user can find and buy rides.

Each user after buying ticket gets the mail-notification with QR-code. Also each user can see all of rides detail in his account page.

Data of users and their options store in reliable data base.

# **Used technologies**

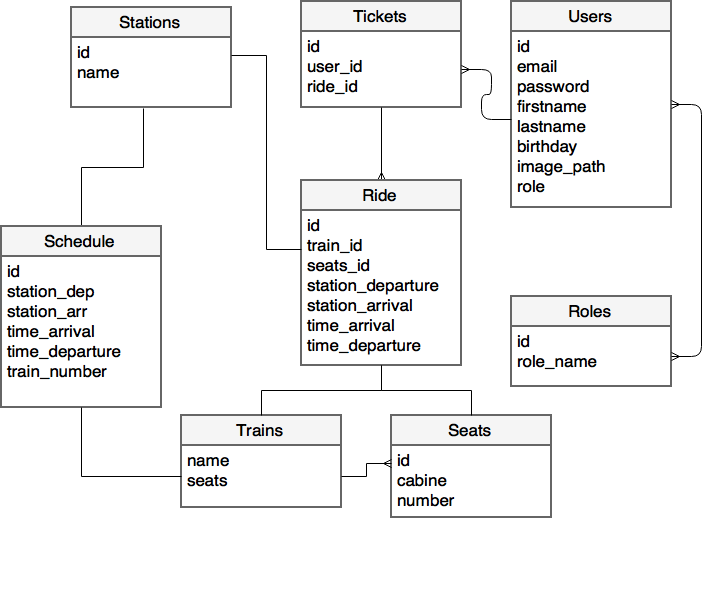
*Instruments:*

* IDE – Intellij IDEA
* Gradle 3.1

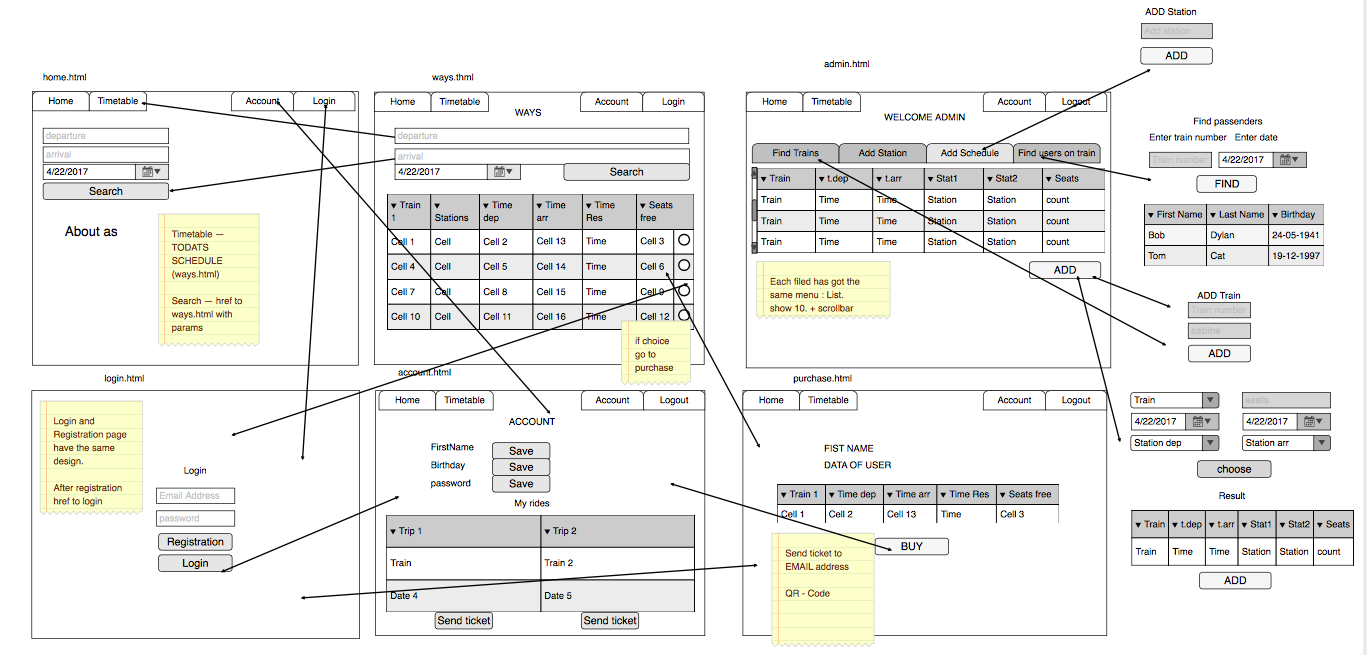
*Technologies:*

* PostgreSQL
* Java 8
* JUnit
* Selenium
* Lombok
* Google zxing
* JavaScript
* JQuery
* Bootstrap 3.0
* Hibernate 5.2.2
* Spring mail
* Spring Boot 1.5.2
* Spring security
* Tomcat 8.5.11
* JSF 2.1.7
* RabbitMQ 3.6.9
* Wildfly 10.1.0
* Angular JS
* armCharts JS

# **Database Model**



# **UI Model**



# **System infrastructure**

* Front-end (browser presentation level):

1) Web-page structure - HTML

2) Page-design - CSS

3) Dynamic content – JavaScript, JQuery.

* Back-end (server based level):

1) Application server - Tomcat

2) Database – PostgreSQL

3) Server logic - Spring Framework

* Client schedule application:

1) Web-pages - JSF

2) JMS - RabbitMQ

3) Application server – WildFly

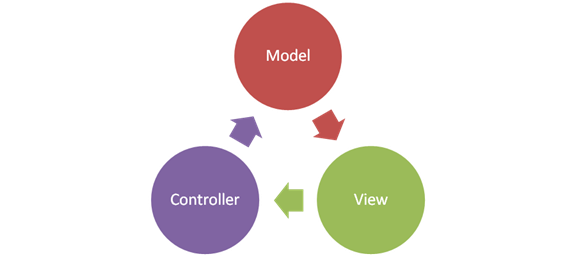
4) WS - REST

# **System architecture**

## *The Justrailways application*

Architecture of server-based part presented by MVC - design pattern.

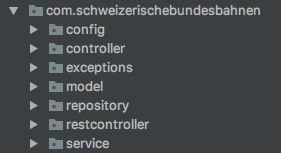
(Pic. #1.1)



Picture 1.1 – System architecture

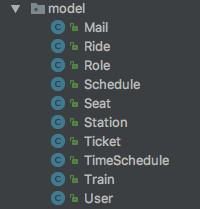
*Class structure*

According MVC-pattern application has next structure (Pic. #1.2):



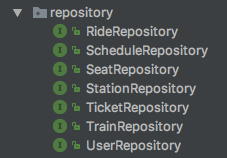
Picture 1.2

Model level (Pic. #1.3):



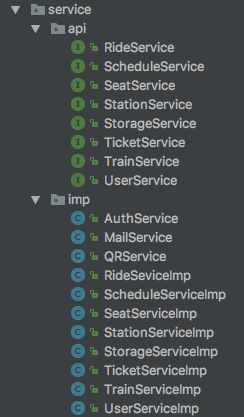
Picture 1.3

Model-service level (Pic. #1.4):



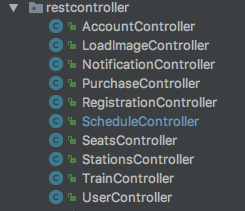
Picture 1.4

Service level (Pic. #1.5):



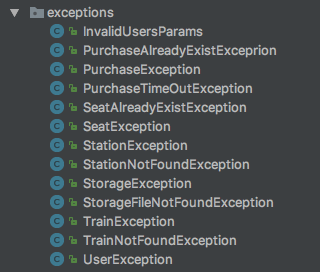
Picture 1.5

View-service level(Pic. #1.6):



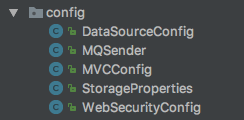
Picture 1.6

Exception handling presented (Pic. #1.7):

****

Picture 1.7

Configuration (Pic. #1.8):



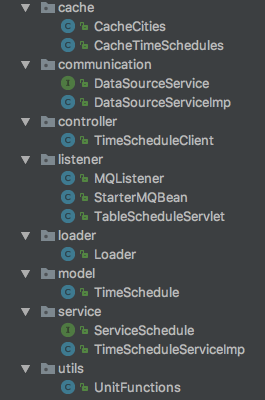
Picture 1.8

## *The timetable schedule application:*

Client can see schedules for today in real time.

When justrailways’s managers update/add/delete schedules The table will update.

Architecture (Pic. #1.9):



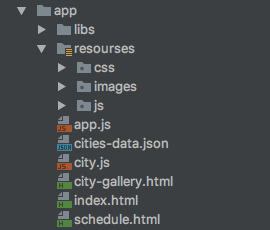
Picture 1.9

## *The schedule by station appication:*

Client can see future schedules in real time by cities:

* Yaroslavl
* Moscow
* Saint-Petersburg
* Basel

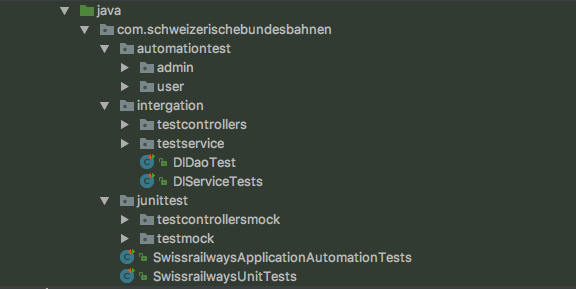
Architecture (Pic. #2.0):



Picture 2.0

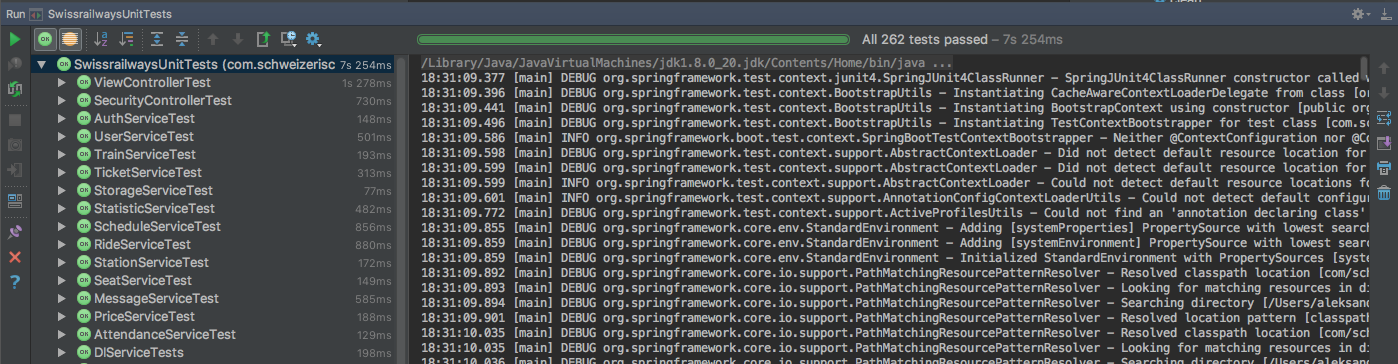
# **Code quality**

Test structure (Pic. #2.1):



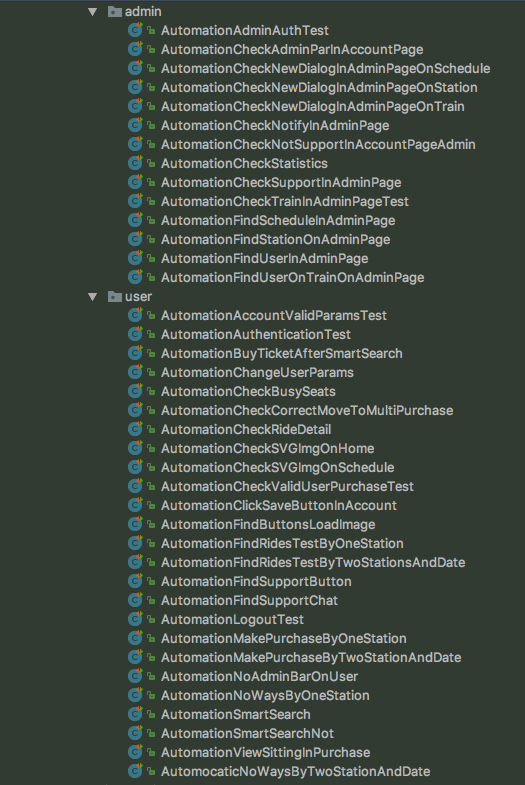
Picture 2.1

Junit-tests resulst(Pic. #2.2):



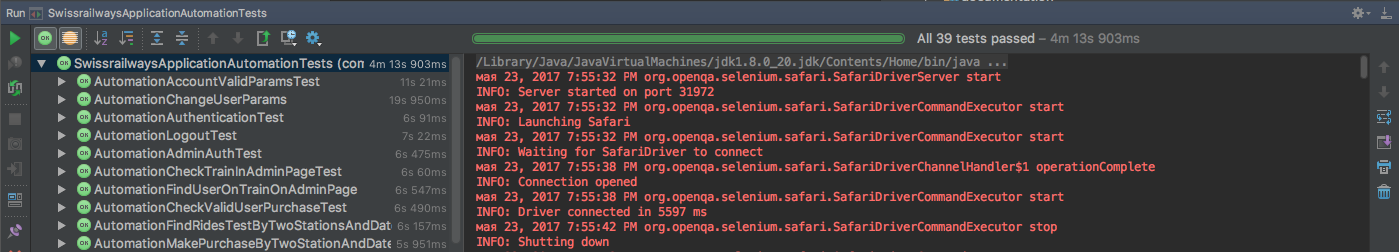
Picture 2.2

Selenium auto-testing structure (Pic. #2.3):



Picture 2.3

Selenium test-result (Pic. #2.4):



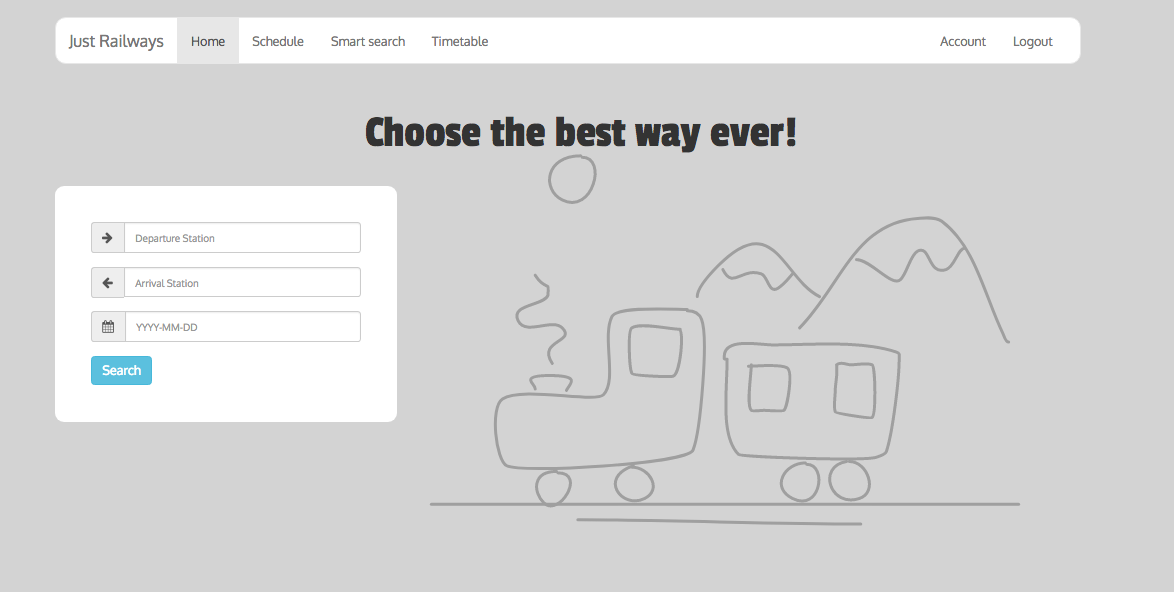
Picture 2.4

# **Additional features**

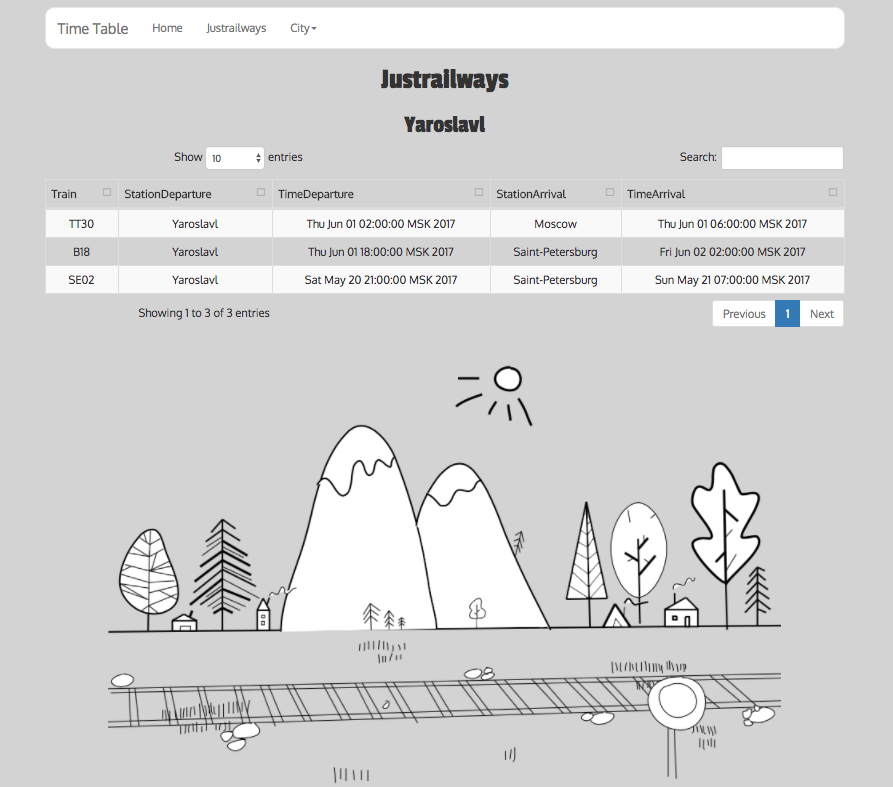
1. Selenium auto-test
2. Pretty design: draw effect
3. Admin can add/update/delete schedule
4. Generate QR-Code after purchase
5. User can choose the seat at the train
6. Send notification by email
7. User can upload his image
8. Password encoding
9. Stress testing
10. User can see his ride detail
11. Support: chat user with admin
12. Statistics: attendance by day, money statistics, rides statistics
13. Angular JS Application

# **GUI**

## *JustRainWays*

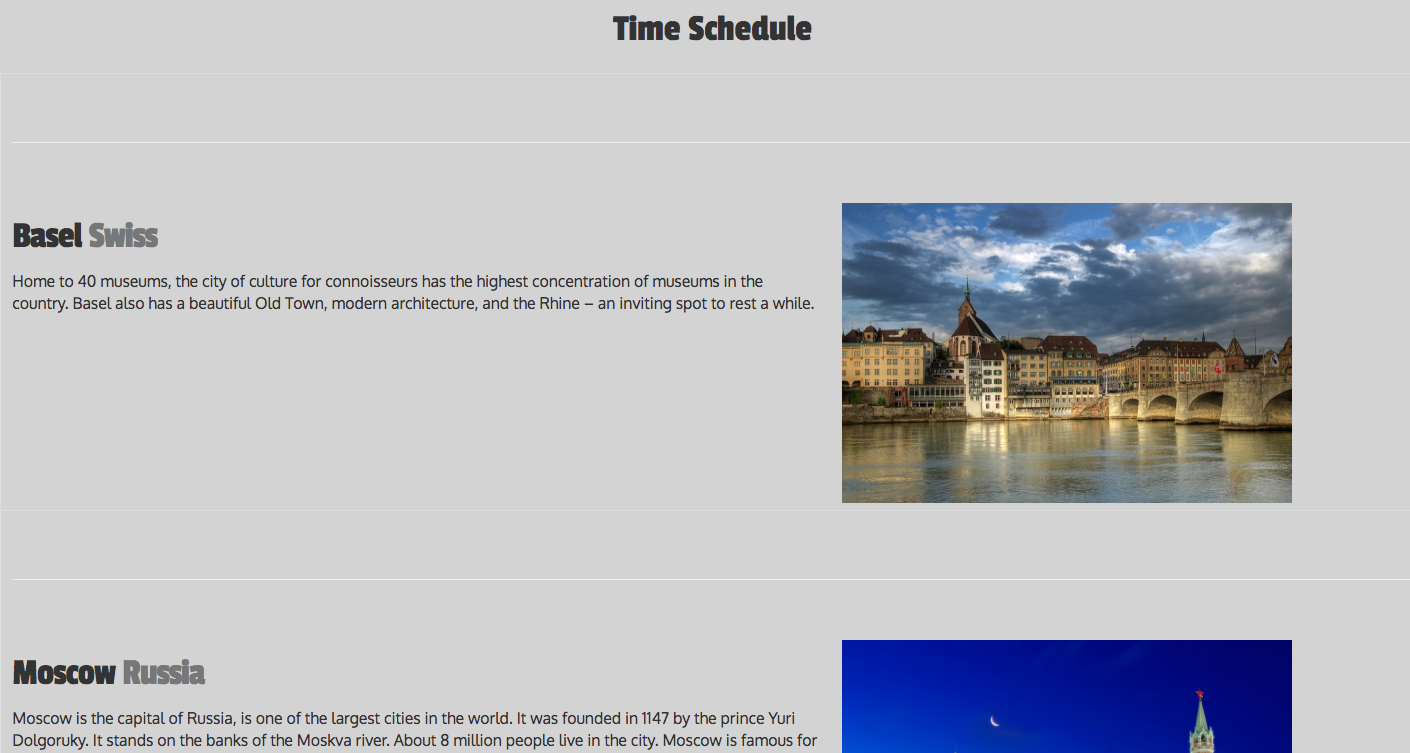


## *TimeSchedule*



## 

## *TimeStatations*

**

# **Build and deploy**

**JustRialways application:**

**$** gradle clean

**$** gradle bootRepackage

**$** java -jar swissrailways-0.0.1-SNAPSHOT.jar

**Starting Rabbint MQ:**

*Downloading and Installing:*

<https://www.rabbitmq.com/download.html>

**$** ./rabbit-server

**Time Schedule application:**

*Start Wildfly application server*

**$** ./standalone.sh

*Main application installation (in Time Schedule directory)*

**$** mvn clean install

**$** mvn wildfly:deploy

# **Future improvement**

1. Add the payment system
2. Add more statistics
3. New UI
4. Add more data