Technical description

**JUSTRAILWAYS PROJECT**

Author:

Alexander Prendota

Saint-Petersburg,2017

Content

[Task 3](#_Toc480801027)

[Project goals 4](#_Toc480801028)

[Application description 5](#_Toc480801029)

[Used technologies 6](#_Toc480801030)

[Database Model 7](#_Toc480801031)

[UI Model 8](#_Toc480801032)

[System infrastructure 9](#_Toc480801033)

[System architecture 10](#_Toc480801034)

[*The Justrailways application* 10](#_Toc480801035)

[*The timetable schedule application:* 13](#_Toc480801036)

[Code quality 14](#_Toc480801037)

[Additional features 15](#_Toc480801038)

[GUI 16](#_Toc480801039)

[Future improvement 19](#_Toc480801040)

# **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 scoreboard.

# **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**

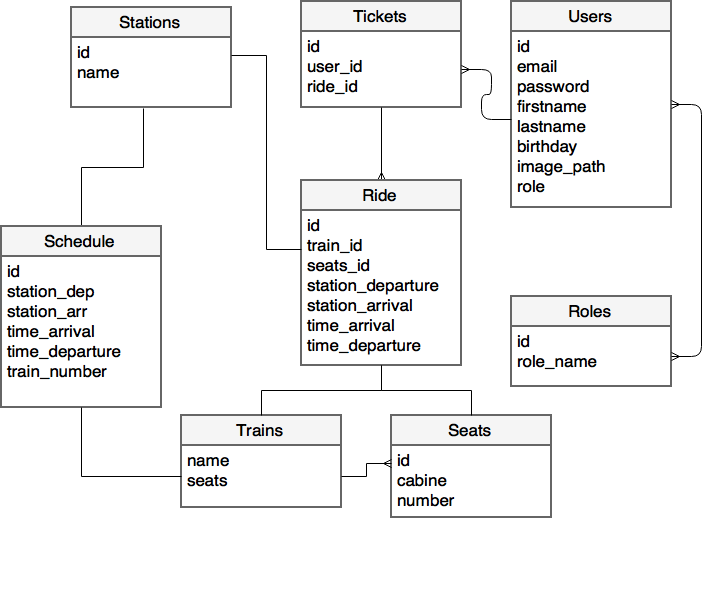
*Instruments:*

* IDE – Intellij IDEA
* Gradle 3.1

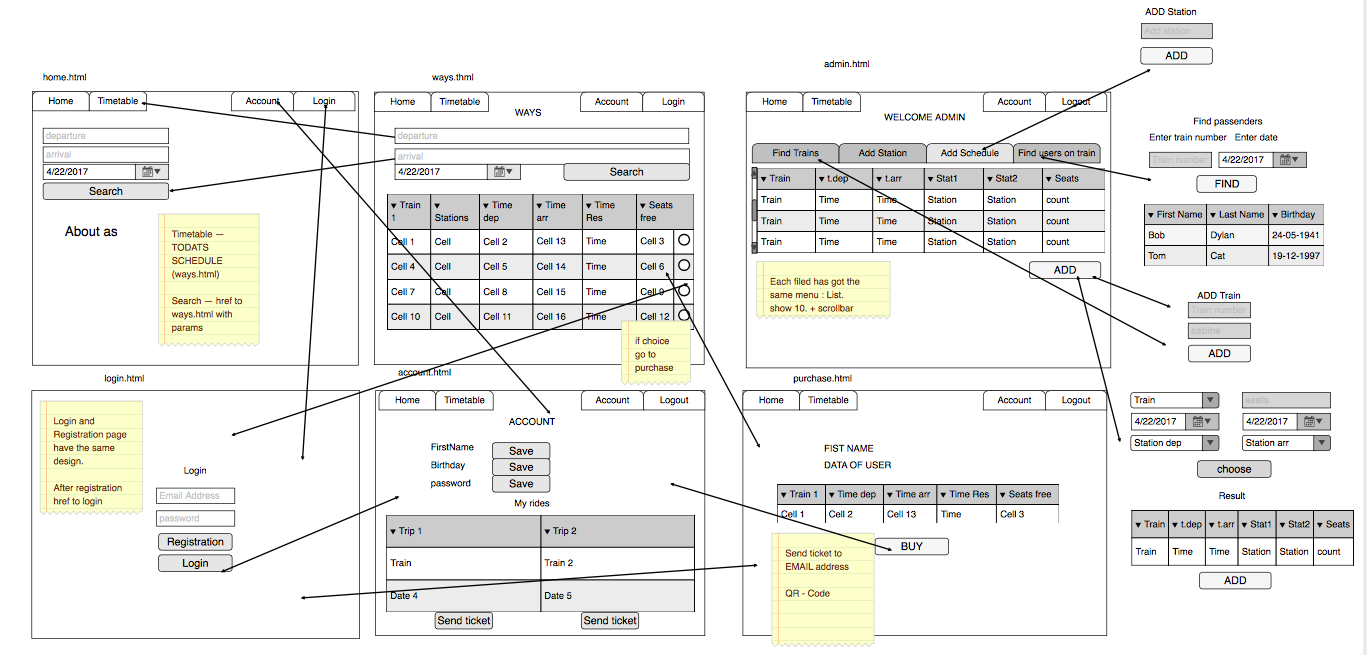
*Technologies:*

* 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

# **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) ORM - Hibernate

4) Server logic - Spring Framework

* Client schedule application:

1) Web-pages - JSF

2) JMS - RabbitMQ

3) Application server – WildFly

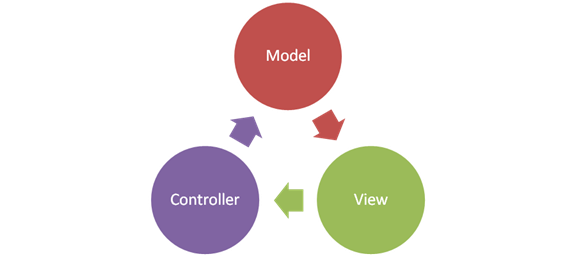
4) WS - REST

# **System architecture**

## *The Justrailways application*

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

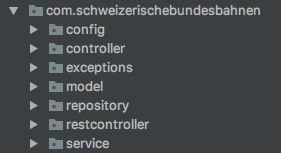
(Pic. #1.1)



Picture 1.1 – System architecture

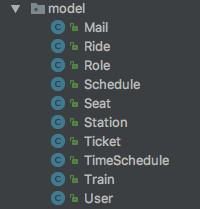
*Class structure*

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



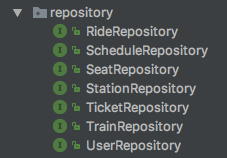
Picture 1.2

Model level (Pic. #1.3):



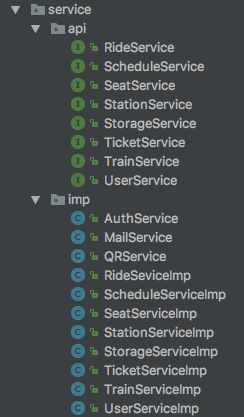
Picture 1.3

Model-service level (Pic. #1.4):



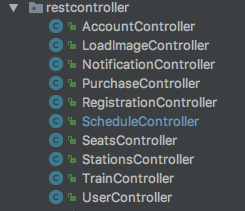
Picture 1.4

Service level (Pic. #1.5):



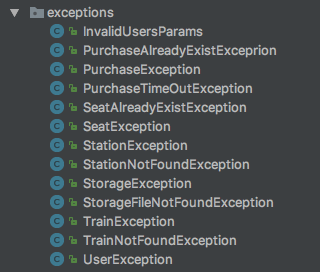
Picture 1.5

View-service level(Pic. #1.6):



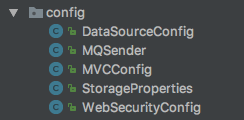
Picture 1.6

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.

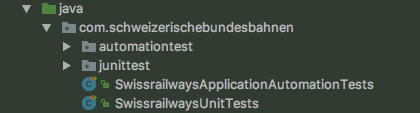
When justrailways’s manages updates/add/delete schedule The table will update.

Architecture (Pic. #1.9):

Class structure

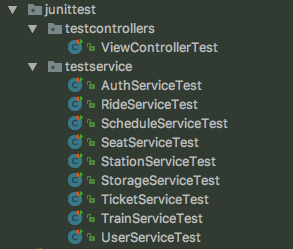
# **Code quality**

Test structure (Pic. #2.1):



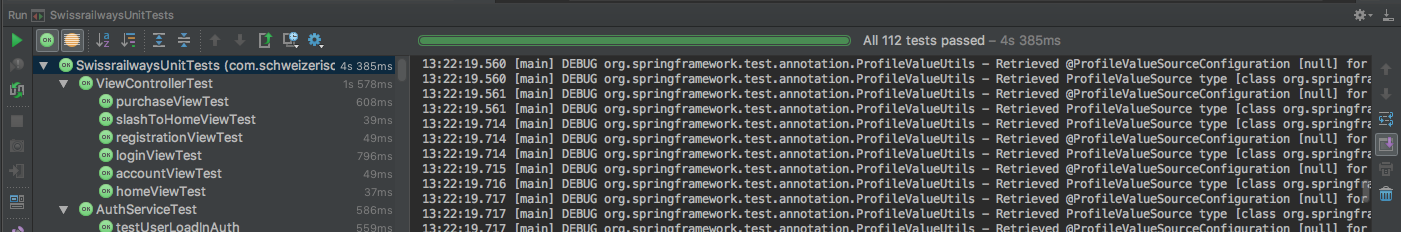
Picture 2.1

Junit‑tests structure (Pic. #2.2):



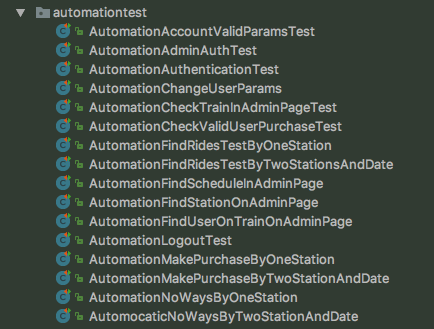
Picture 2.2

Junit-tests resulst(Pic. #2.3):



Picture 2.3

Selenium auto-testing structure (Pic. #2.4):



Picture 2.4

# **Additional features**

1. *Селениум тесты*
2. *Красивый дизайн: эффект рисования.*
3. *Добавление/Удаление/Изменение расписание*
4. *Генерация QR-кодов поездок*
5. *Отправка нотификаций по электронной почте*
6. *Визуальный выбор мест в поезде*
7. *Загрузка пользовательской картинки*
8. *Шифрование пароля пользователя*
9. *Нагрузочное тестирование*
10. *Просмотр подробной информации о поездке*

# **GUI**

**Build and deploy**

**Glossary**

# **Future improvement**