Technical solution description

Mobile operator

Content

1 Task

Main goal was to develop an application that models information system of mobile operator.

Application must provide following functionality:

- For clients:
  - View contracts;
  - View all tariffs, change tariff;
  - View all tariff options, add new options.
  - Block / unblock contract.

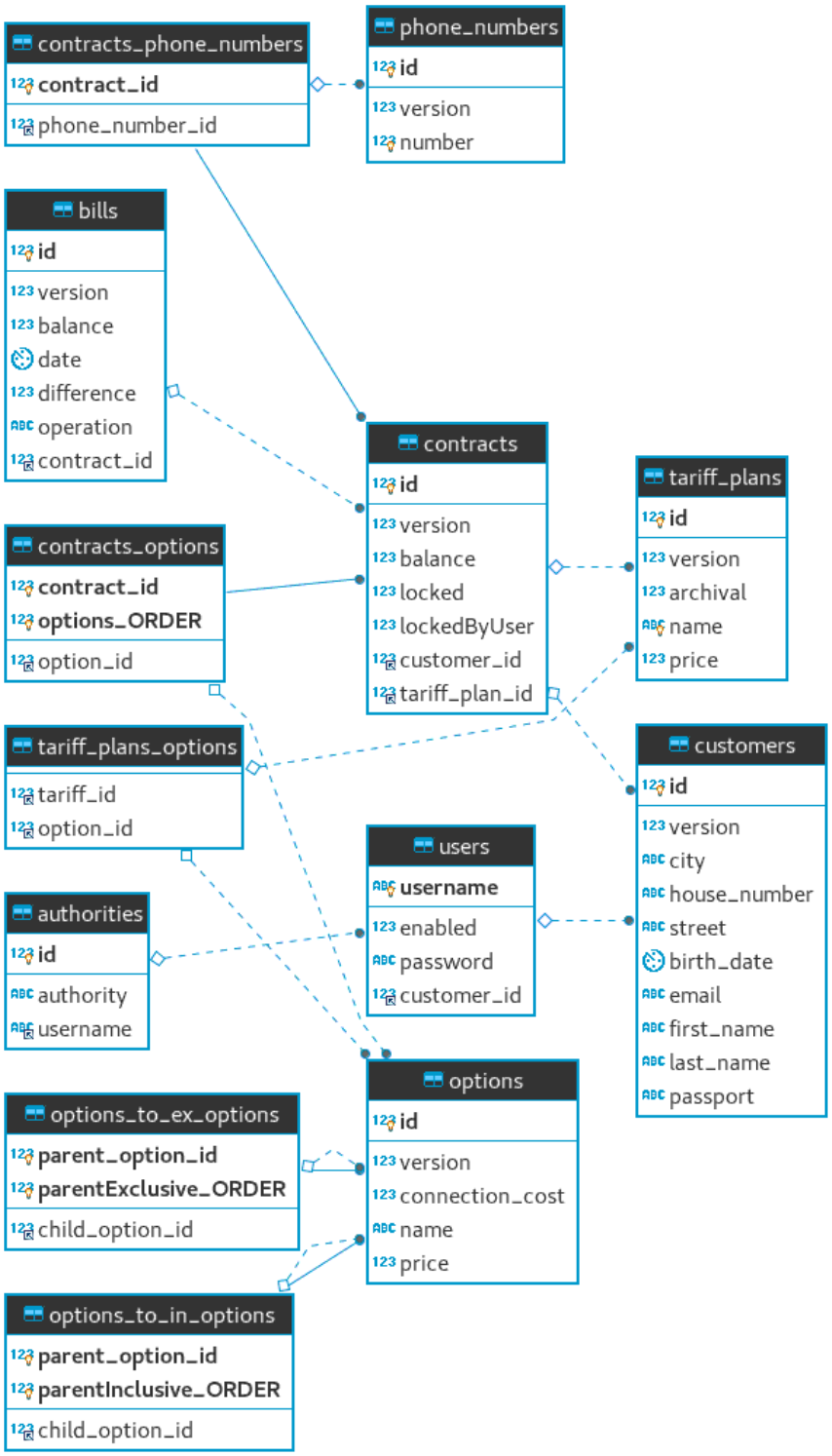
- For managers:
  - Create new contracts with unique numbers/change old contracts;
  - View all clients and their contracts;
  - Block / unblock contracts;
  - Find client by number;
  - Manage tariffs and options.

Second goal was to develop an application for advertising stand, showing tariffs from first application. Applications should communicate by MQ.

2 Used technologies

Technology	Version
AspectJ	1.8.11
FasterXML Jackson	2.9.1
H2 Database	1.4.196
Hibernate	5.2.12
IntelliJ IDEA	2017.2.5
Java	8
JSP	2.2
JUnit	4.12
Log4J	1.2.17
Maven	3.5.0
MySQL	5.7
Primefaces	6.1
SonarQube	5.6.0
Spring and Spring security	5.1.0
Wildfly	14.0.1
Jenkins	2.150.1

3 Database schema



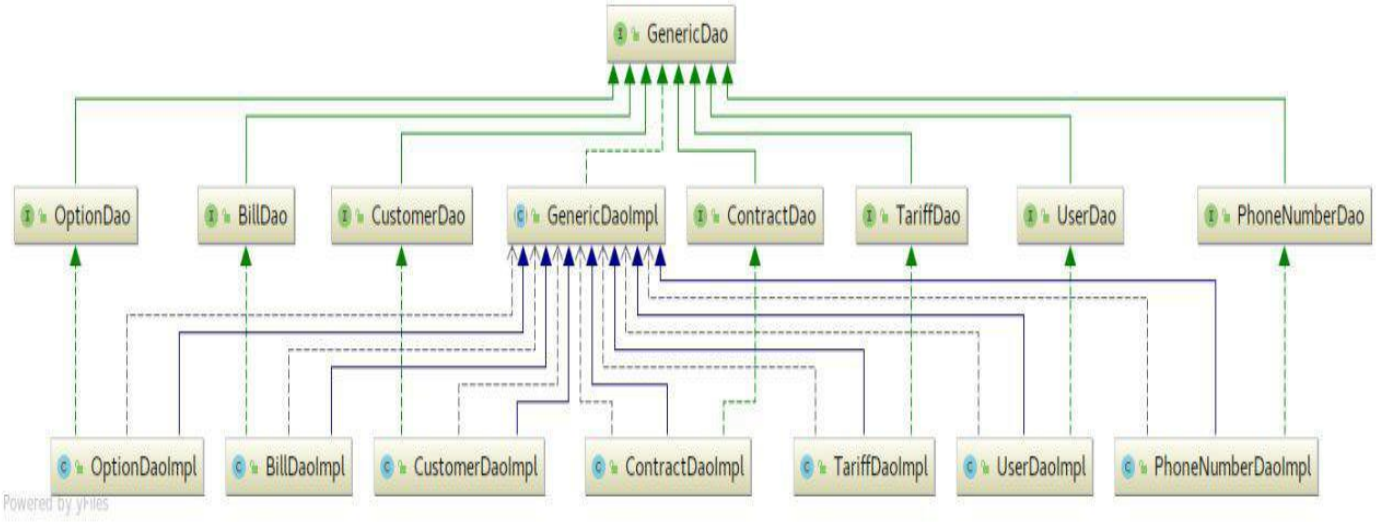
Picture 1 - database schema

4 Architecture

4.1 Main application

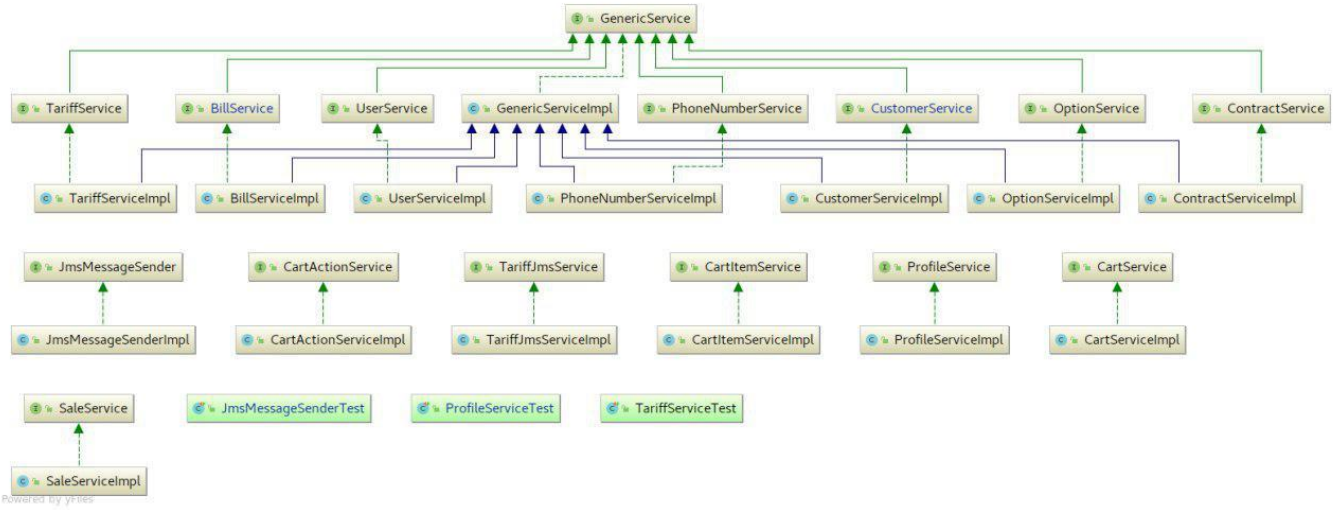
According to MVC, main application has next structure:

- Model level:



Picture 2 - model level diagram

- Controllers level:



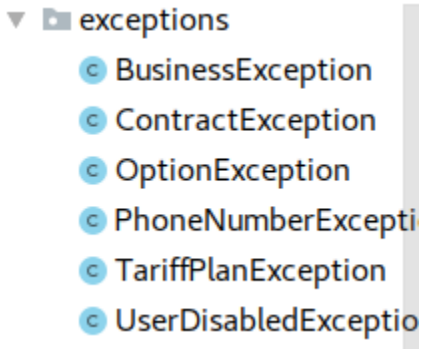
Picture 3 - controllers level diagram

- View level:

Picture 4 - server side view level

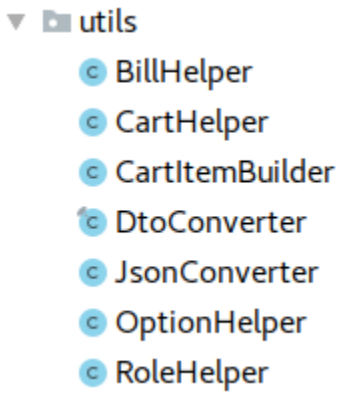
Picture 5 - client side view level

- Custom exceptions:



Picture 6 - custom exceptions list

- Utility classes:



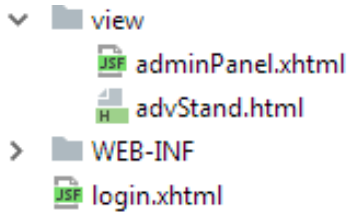
Picture 7 - utility classes

- For logging was used AOP.

4.2 Advertising stand

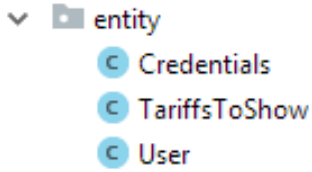
Advertising stand has next structure:

- Views:



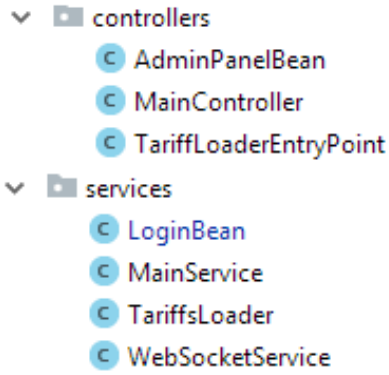
Picture 8 - advertising stand views

- Dtos:



Picture 9 - advertising stand entities

- Business logic:



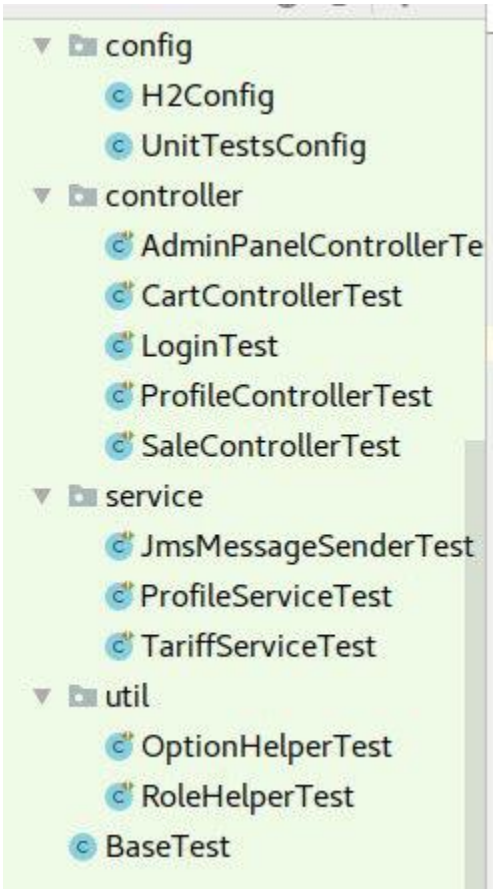
Picture 10 - advertising stand views

5 Tests and quality of code

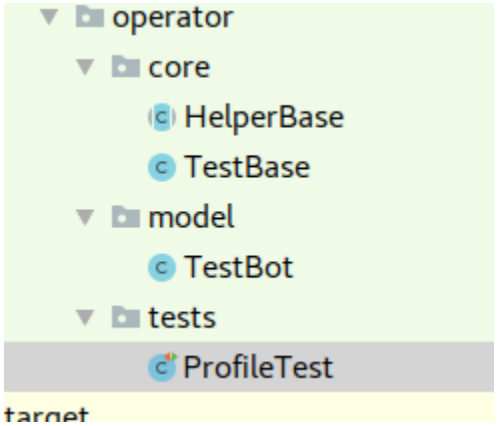
JUnit tests:

```
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.102 sec

Results :

Tests run: 33, Failures: 0, Errors: 0, Skipped: 2
```

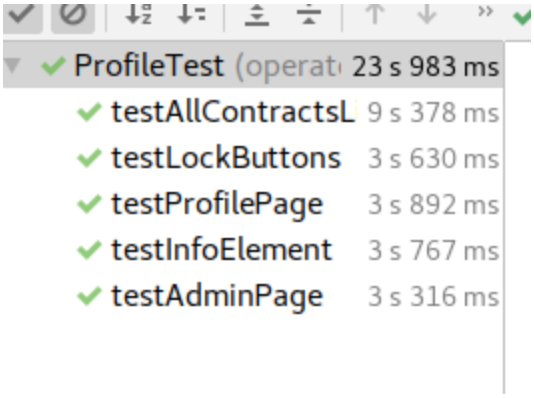
Picture 11 - JUnit tests result



Picture 12 - Unit tests structure



Picture 13 - UI tests structure



14 – UI tests results

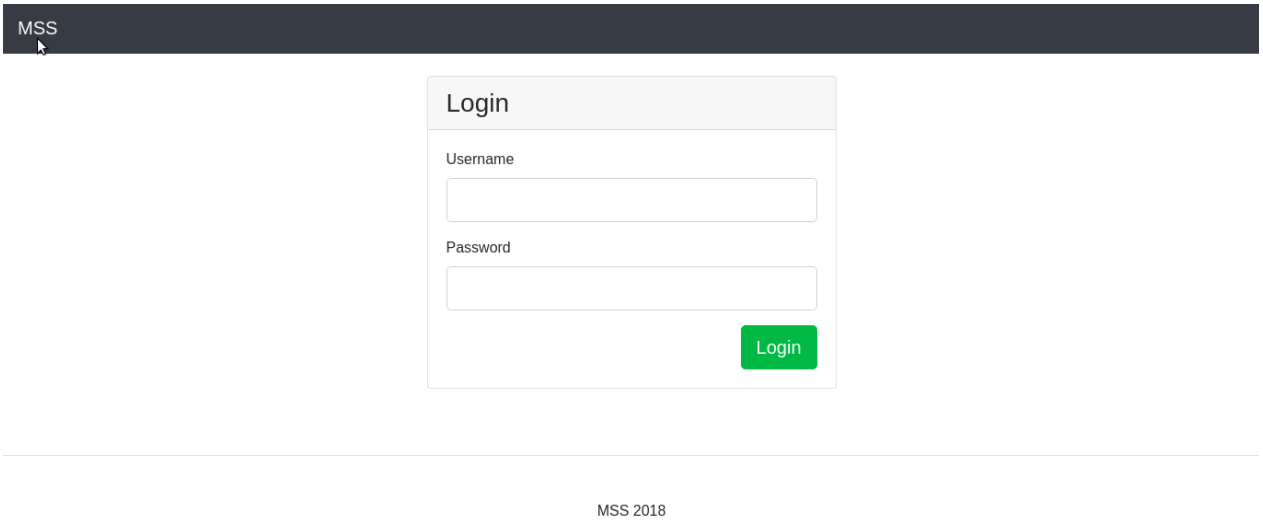
6 Deployment

Available deploy to wildfly via Jenkins.

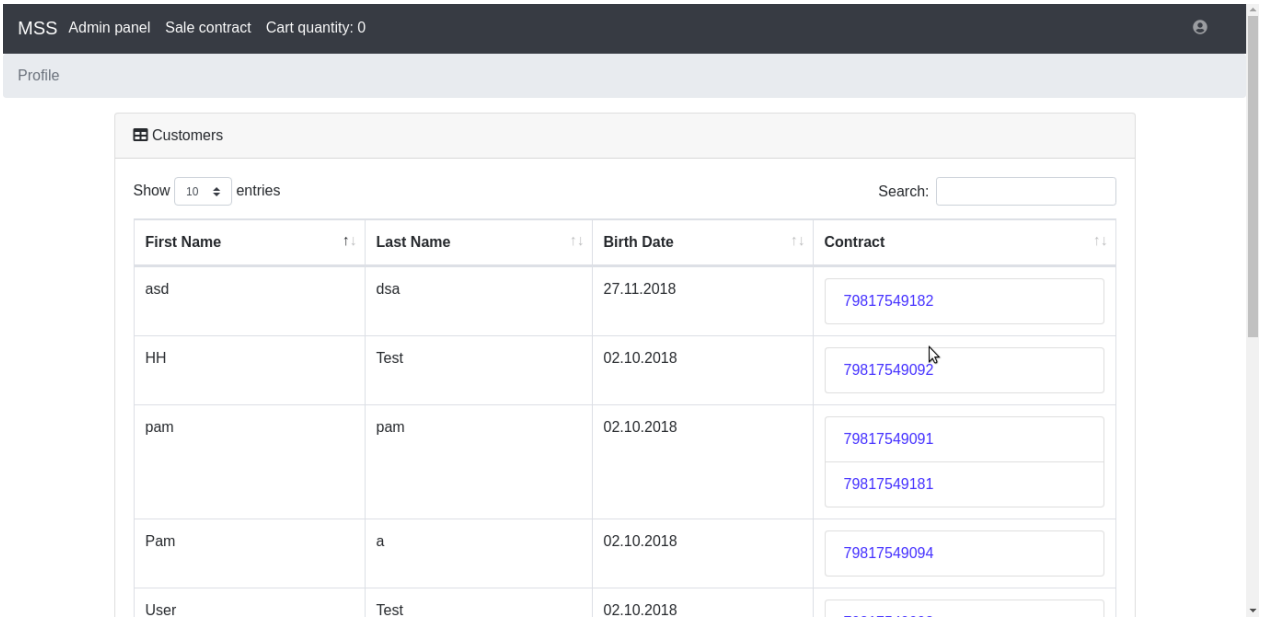
Manual build and deploy with maven is also available:

1. In working directory enter in console: *mvn clean install*;
2. Then enter: *mvn wildfly:deploy*. (if wildfly is running)

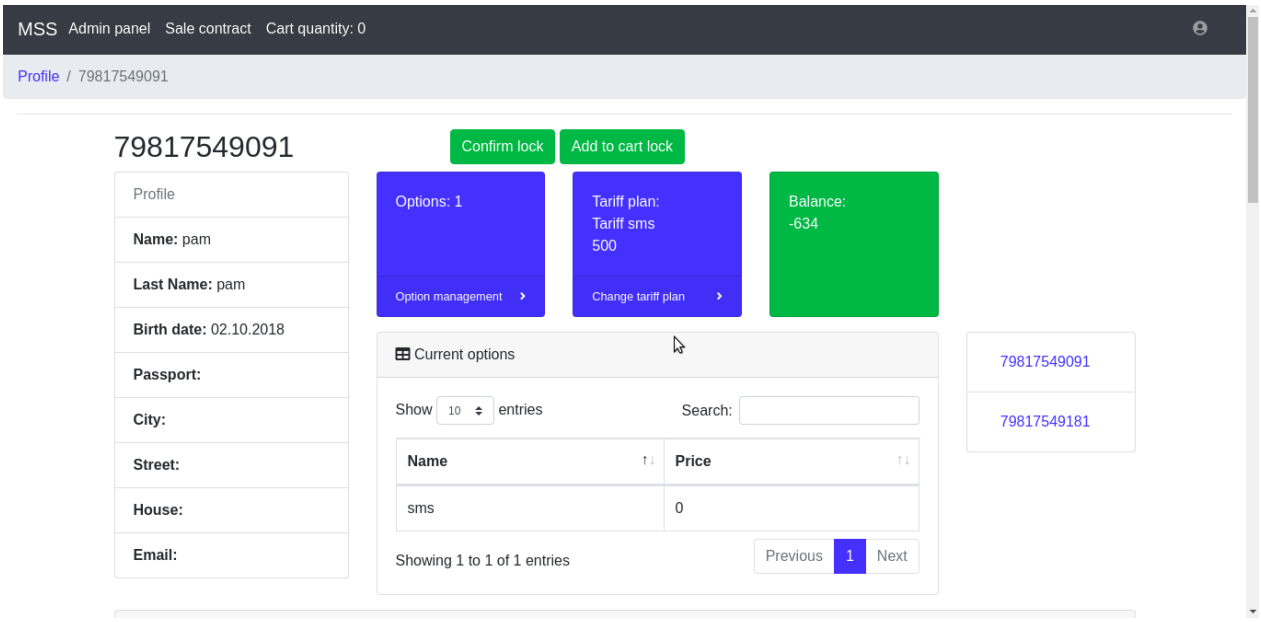
7 GUI



Picture 11 - login page



Picture 12 - managers index page



Picture 17 - clients index page