TECHNICAL DESCRIPTION

Web-portal for mobile operator «Spting Line»

Ekaterina Kochurova

Saint-Petersburg

16.05.2019

Contents

Revision history

Date	Version	Description	Author
01.05.2019	1.1	Creation of document	Ekaterina Kochurova
15.05.2019	1.2	Description added	Ekaterina Kochurova
16.05.2019	1.3	Sonar metrics	Ekaterina Kochurova

1 System requirements

The task is to make a web-application for “Spring Line” mobile operator. Application should be able to perform the following actions:

- For administrators
  - To create new options and tariffs
  - To edit existing options and tariffs
  - To delete options and tariffs
  - To manage options
    - Set compatible tariffs
    - Set incompatible options
    - Set supplementary options
  - To create new contracts
  - To update existing contracts
  - To view all users and contracts
  - To block/unblock sim card
  - To search user by phone number

- For clients
  - To view the contract in your personal cabinet
  - To view all possible tariffs for migration, change of tariff
  - To view possible tariff options, connecting new options, disabling existing ones;
  - To block/unblock number

Additionally an application to advertise tariffs promotions will be created.

2 Project Goals

- Implementation of all system requirements
- A stable and reliable system for mobile operator
- Cohesive data model
- User-friendly interface
- Scalable architecture

3 Short Application Description

Web-application is made according to provided requirements. It has two type of users: admins and clients. Admins have an access to the whole functionality of application, including tariffs and options administration.

During creation of new tariff administrator can set name, description, monthly price, compatible options and validity. Invalid tariffs cannot be chosen during contract management. User can update only description, compatible options and validity. Price and name cannot be changed after creation. Tariffs cannot be deleted. It can be only set as invalid so it is not seen during contract management.

Option management has all the same features as tariff management with some additional points. Instead of compatible options you can set compatible tariffs here. Options have one time price, which is charged during contract creation. This price also cannot be changed after creation.

Some options can be only chosen only if other option has been chosen before. A condition option is called “parent”. An option can have only one parent, but parent option can have several dependent options – “children”. A child cannot have other children. So we have only one layer of dependency. Parent and children options can be set during option creation and edited later.

Some options are incompatible with each other. To make options incompatible you should add them to an option group. An option can belong only to one option group and is incompatible with all other options within this group. Children options cannot have an option group. Option groups can be set during option creation and can be changed later.

Administrator can also create and edit option groups. During creation administrator can set name, description, compatible options and validity. Name cannot be changed during editing stage. When option group is set as invalid all compatible options are removed from it automatically.

Administrator also can create new contracts. First, he need to choose a tariff, than options that are compatible with this tariffs. He can chose none options though. Finally, he can add a contract to existing client or create a new one. In both cases he also choses a phone number for new contract. If we create a new client a new user will be created automatically.

Administrator can view all existing clients and contracts. He can search clients by phone numbers. He can edit existing options, removing existing and adding new options. He can also change tariff. All current options will be removed then. Administrator can lock the contract. Than no one could change tariff or options of this contract. Only administrator can unlocked contract, that is locked by administrator.

Additionally administrator during creation or editing stage of tariff can set is as promoted. Than a signal will be send to supplementary application and the tariff will be shown on advertisement stand.

Clients can have view their current contracts, details of each contract. They can change tariff and options. They can also block and unblock contract. They cay unblock contract if it was not blocked by administrator.

4 Used Technologies

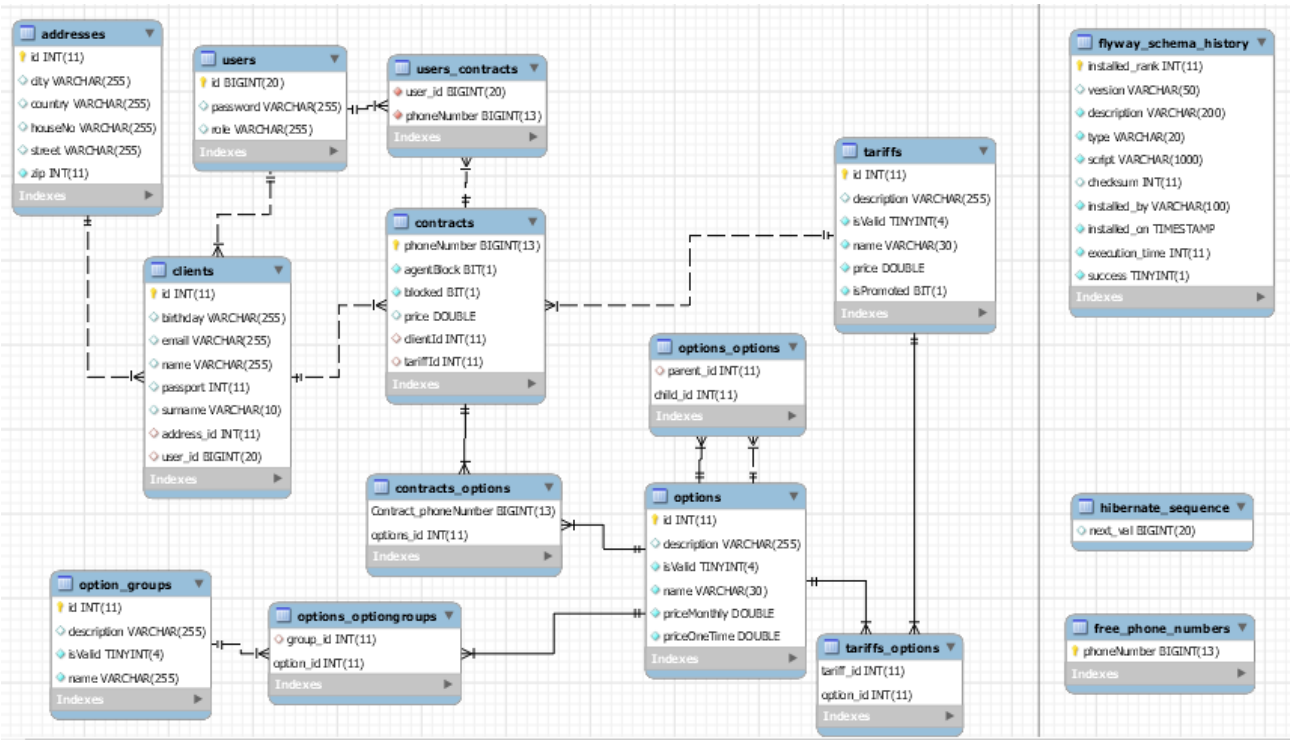
Instruments:

- IDE – Intellij IDEA
- Maven
- SQL WorkBench
- Docker

Technologies:

- Bootstrap 4.0
- DB – MySQL
- EJB
- ItxtPdf 5.5.13
- FlyWay 5.2.4
- Hibernate 5.4.1
- Java 8
- JavaScript
- JSF 2.3
- JSP
- Junit 5.5
- Log4j2
- Mockito 2.25
- Wildfly 16
- JMS
- REST
- SonarQube 7.7
- Spring 5.1.4
- Spring security 5.1.4
- Tess4j 4.3.1
- Velocity Engine 2.1
- WebcamJS v1.0.25

5 Data Base Diagram



6 System Architecture

6.1 System infrastructure

Front-end (browser presentation level):

1) Web-page structure - *HTML*
2) Page-design - *CSS*
3) Dynamic content – *JavaScript* (e.g. *AJAX*)

Back-end (server based level):

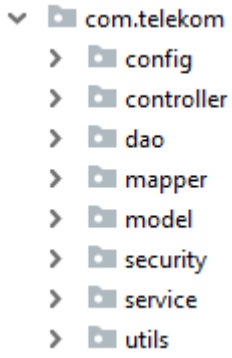
1) Application server - *WildFly*
2) Database – *MySQL*
3) ORM - *Hibernate*
4) Server logic - *Spring Framework*

Rest client (transaction server level):

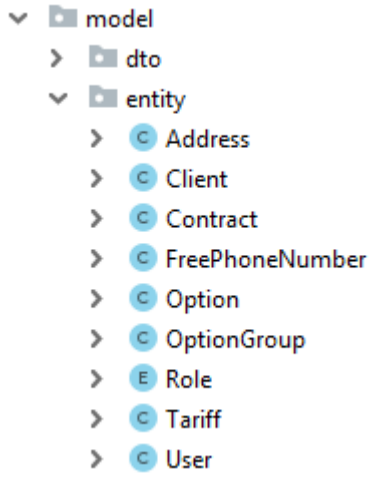
1) Web-pages - *JSF*
2) Server logic – *EJB*

6.2 Application structure

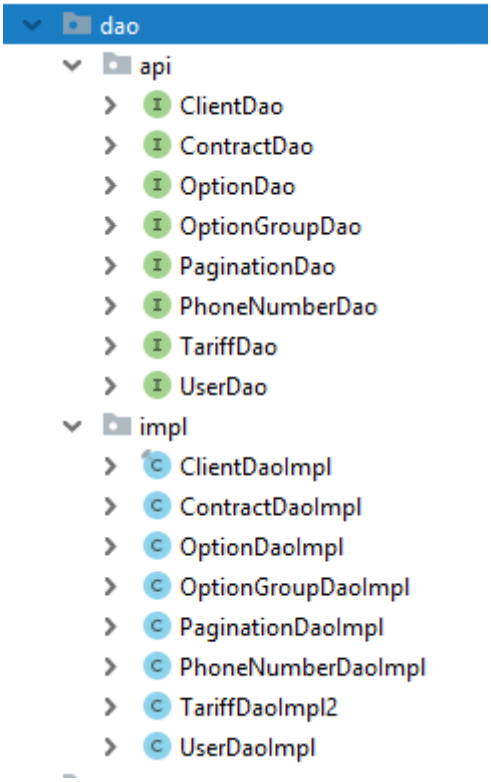
The system is built basing on, MVC pattern. According to the pattern application has the following structure:

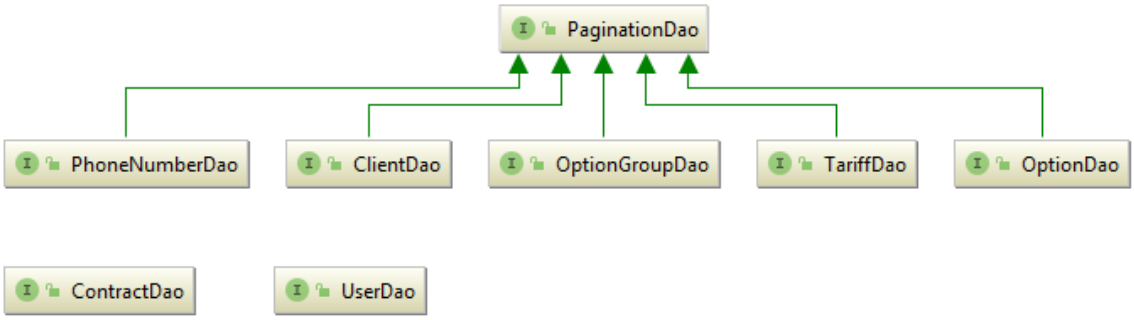


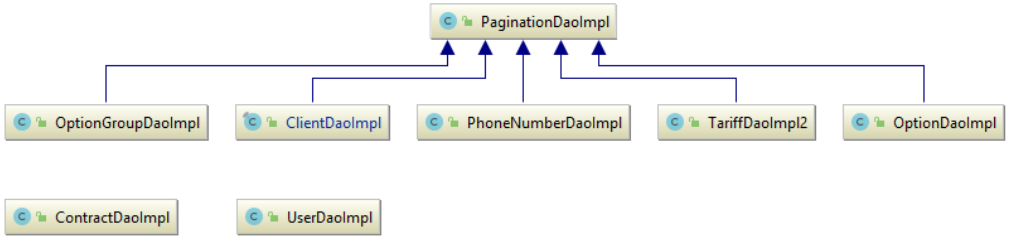
Model level:



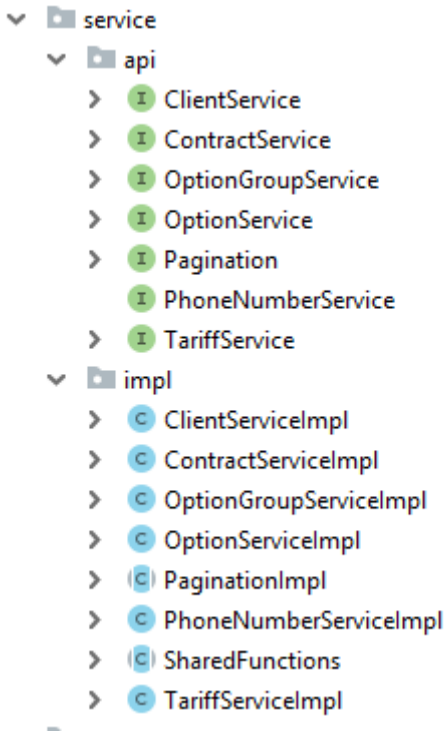
Model-service level:



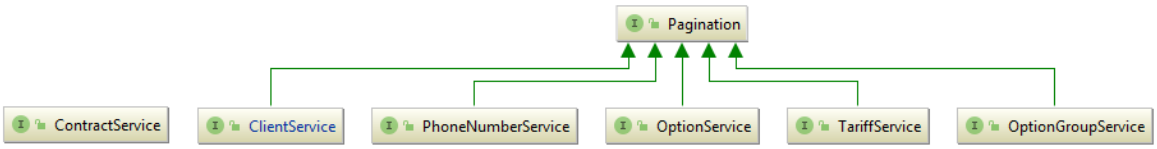


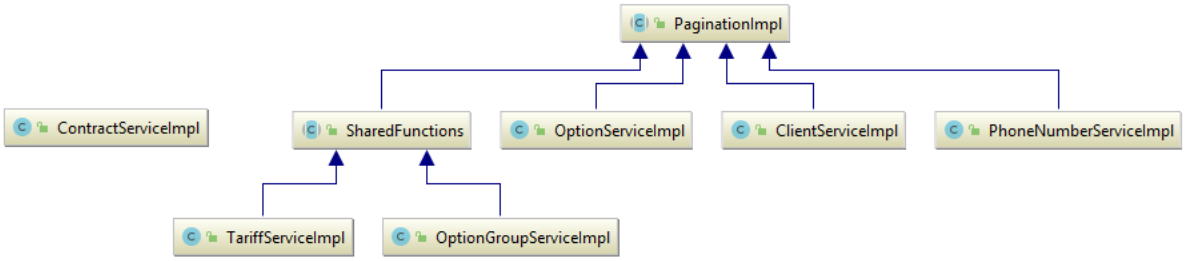


Service level:

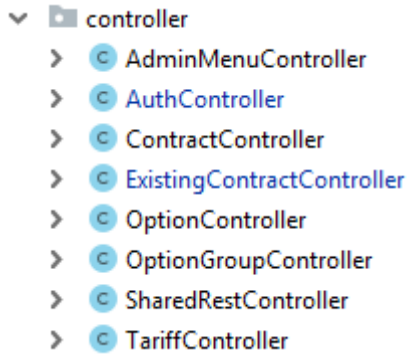


Services that support pagination are inherited for Pagination class or interface:

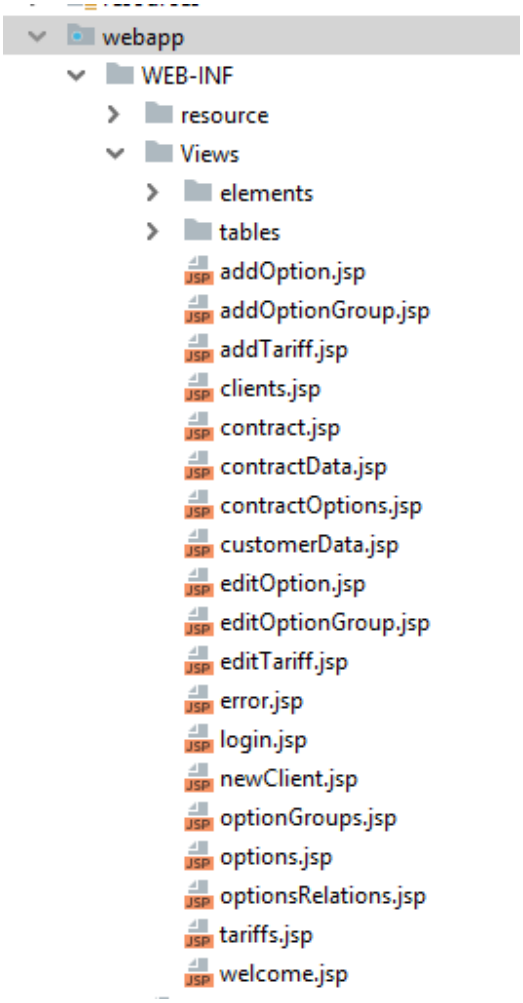




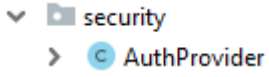
View-service level:



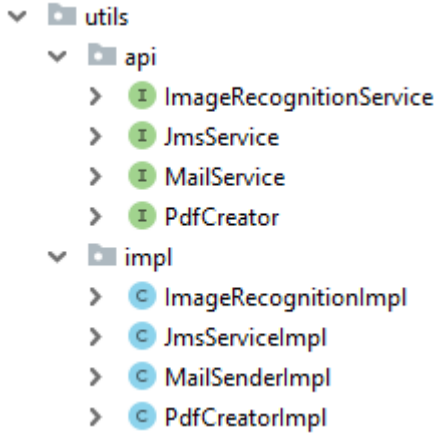
View level:



Security mechanisms:



Support utilities:

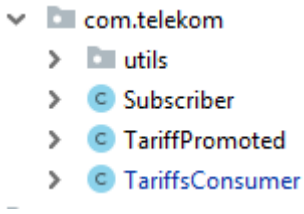


Second application is built on EJB and JSF technologies.

Architecture:

Message Driven Bean (Subscriber) triggers Get promoted tariffs request upon new message from JMS Topic. Received JSON is mapped to Promoted Tariff Bean Objects. Changes are pushed via web socket on JSF page.

Business logic:



View:

7 User Workflow

Log in

Each registered user has login and password (stored in data base). Login for users are equal to their contract phone number. It is possible to choose any of contract phone numbers that user have to login. The password however is only one. After entering correct personal data (login and password) users get access to their scope. Important actions change information stored in data base. Actions are also stored in log file – *myapp.log*

User actions

Within scope of application clients can change their tariffs or options. They can also block or unblock contracts. They can unblock a contract only if it was blocked by them.

Admins have an access to the whole scope of an application, including tariffs and options management, new contract creation and existing clients management.

Advertisement stand

Promoted tariffs should be marked by administrators in main application. After change of the tariff status a notification is sent to supplementary application. Then second application requests all promoted tariffs via web services and updates the page with the help of web socket.

8 Additional features

1. Pagination
   Entities that have a big amount of objects are not uploaded at once but by pages. On FE results are updated through ajax and rest controllers

2. Email service
   Emails with information and login link are send to users after update of contract.

3. Pdf generation
   Invoice with contract data is generated after contract change. Invoices are sent to customers via email.

4. OCR implementation
   Id of the customer can be shoot by web camera. Than data is recognized by external library. Data is parsed by application so that some of the customer features can be filled up automatically

5. Flyway
   Database is set up using FlyWay migration

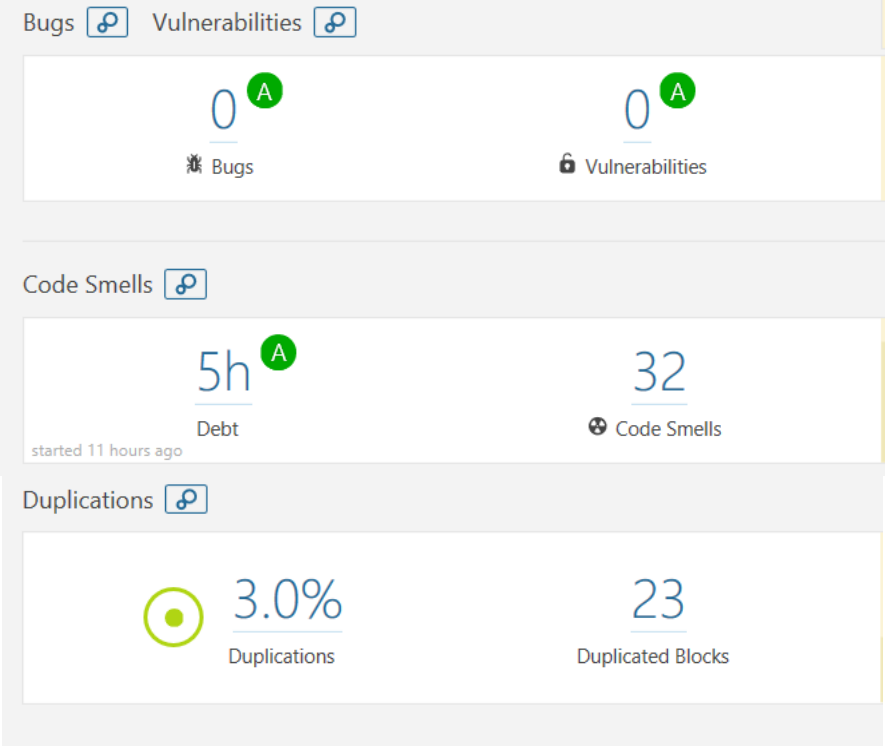
6. Docker
   Both applications as well as Sonar and Data Base can be deployed using docker

9 Code quality

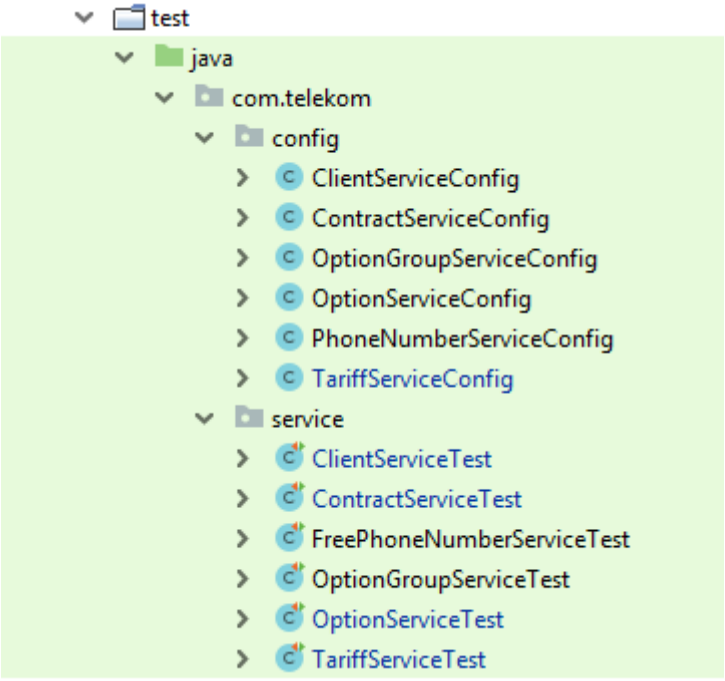
SonarQube measures presented below.

Code structure:

Code quality:



Test Structure:



Unit Test Coverage:

Element	Class, %	Method, %	Line, %
telekom	35% (23/64)	42% (208/493)	36% (731/2002)

Unit Test Results:

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.telekom.service.ClientServiceTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.289 s - in com.telekom.service.ClientServiceTest
[INFO] Running com.telekom.service.ContractServiceTest
[INFO] Tests run: 31, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.33 s - in com.telekom.service.ContractServiceTest
[INFO] Running com.telekom.service.FreePhoneNumberServiceTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.06 s - in com.telekom.service.FreePhoneNumberServiceTest
[INFO] Running com.telekom.service.OptionGroupServiceTest
[INFO] Tests run: 12, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.089 s - in com.telekom.service.OptionGroupServiceTest
[INFO] Running com.telekom.service.OptionServiceTest
[INFO] Tests run: 42, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.359 s - in com.telekom.service.OptionServiceTest
[INFO] Running com.telekom.service.TariffServiceTest
[INFO] Tests run: 18, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.075 s - in com.telekom.service.TariffServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 112, Failures: 0, Errors: 0, Skipped: 0
```

10 Build and deploy

Build and deploy project using Docker

1. Build:

```
./mvnw clean package
```

2. Deploy:

```
docker-compose up
```

Analyze project using SonarQube

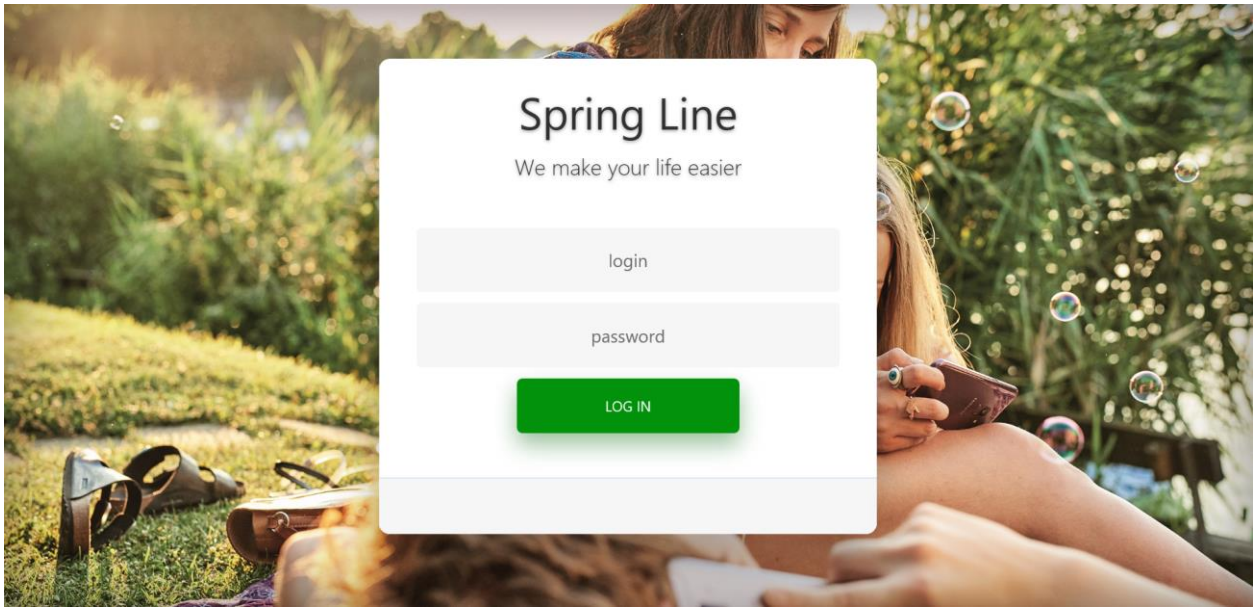
1. Start docker container with SonarQube

```
docker-compose -f sonar/docker-compose.yml up
```

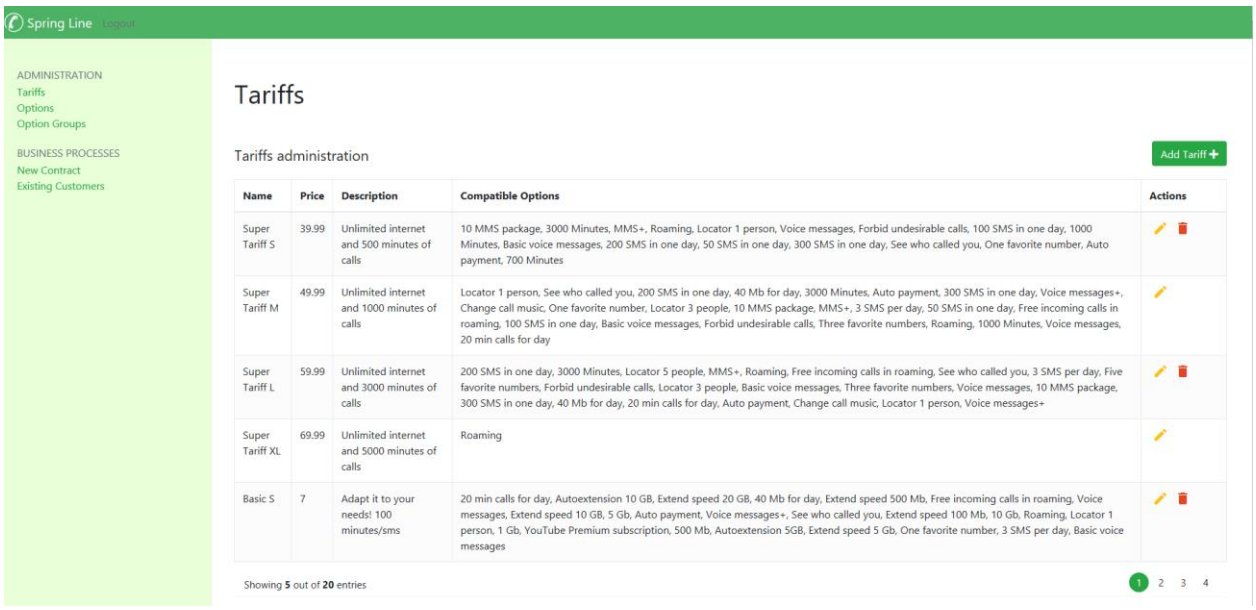
2. Create token using SonarQube UI (default login/password is admin/admin)
3. Perform analyzing

```
./mvnw clean verify sonar:sonar \
     -Dsonar.host.url=http://localhost:9000 \
     -Dsonar.login={token}
```

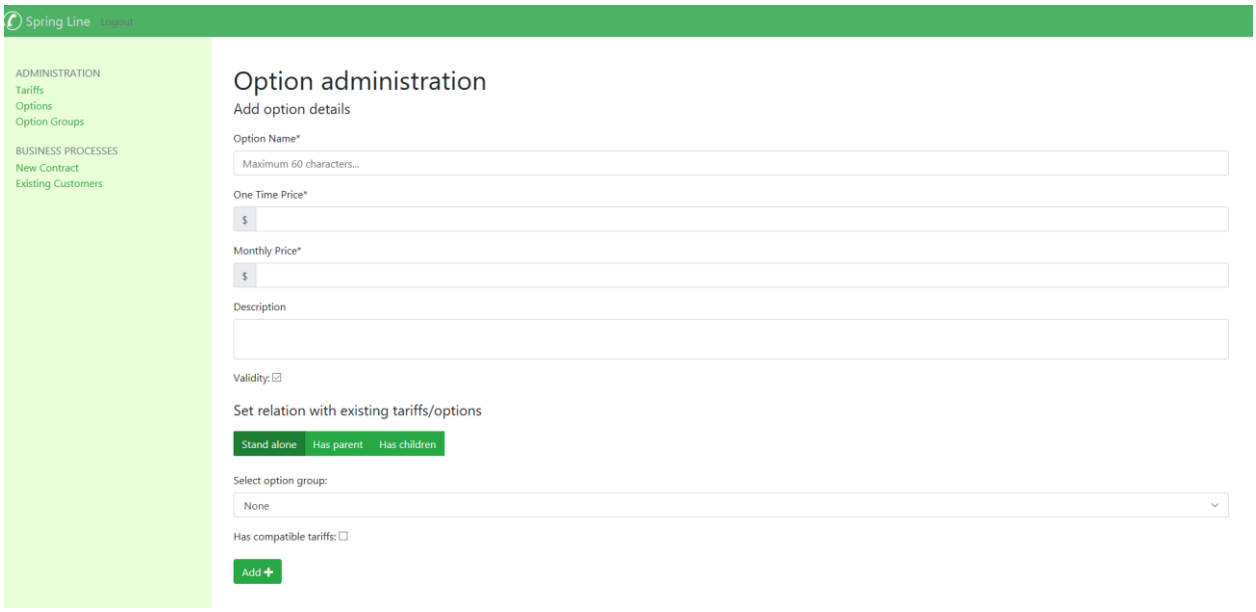
11 GUI



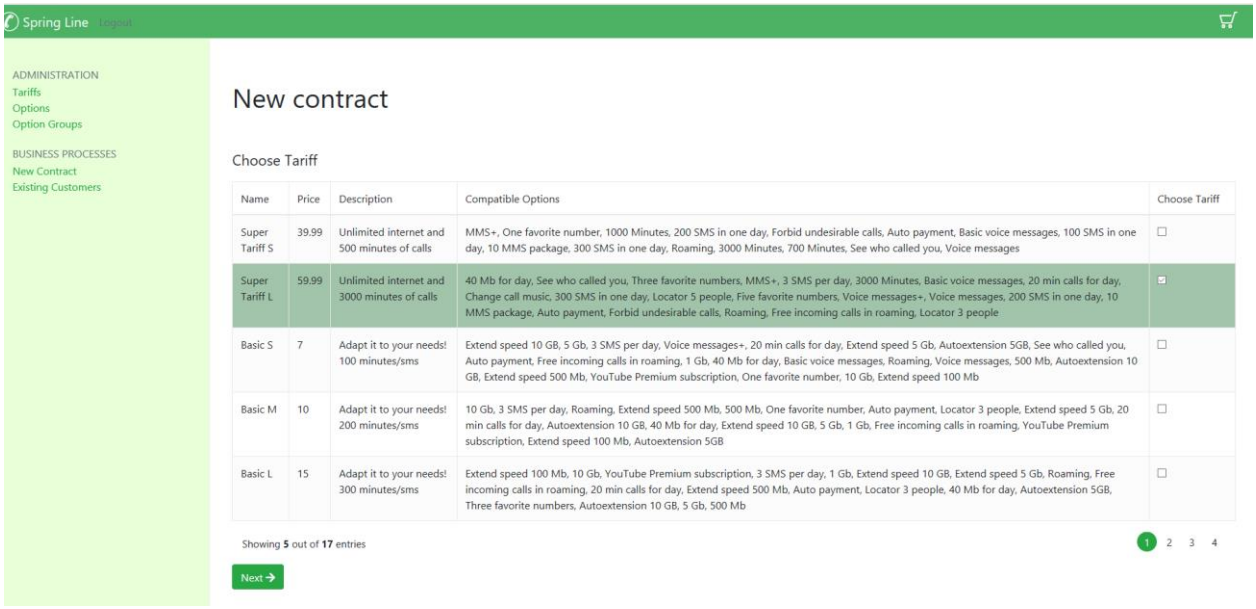
Picture 1.1 – Login Page



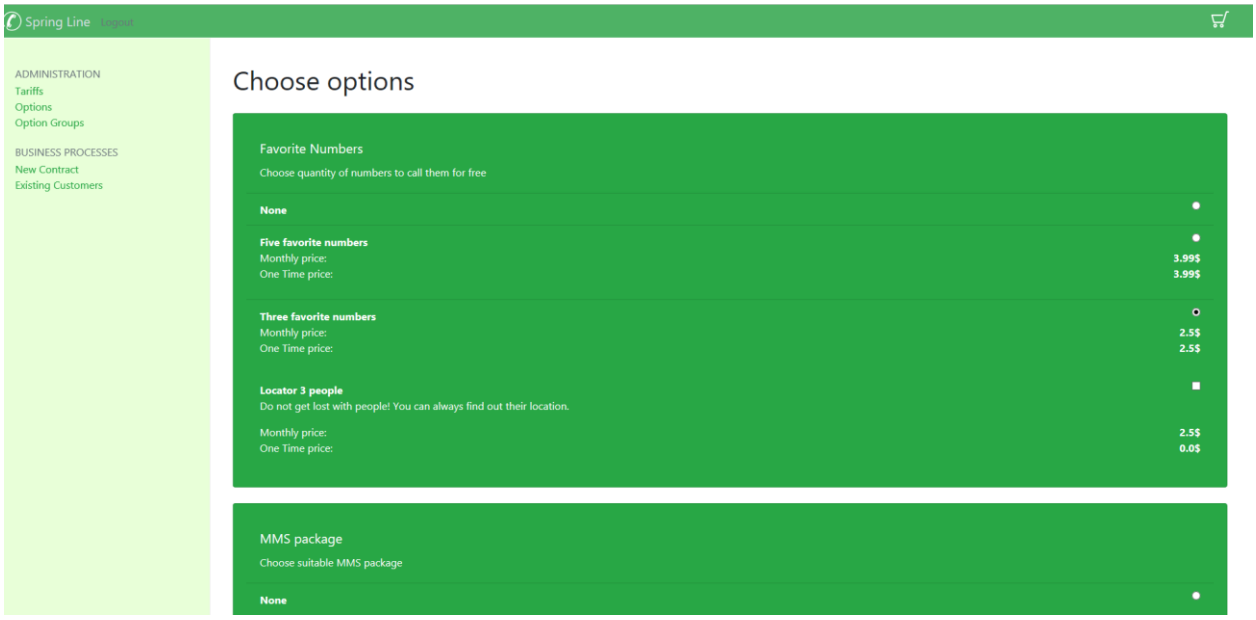
Picture 1.2 – Tariffs Administration Page



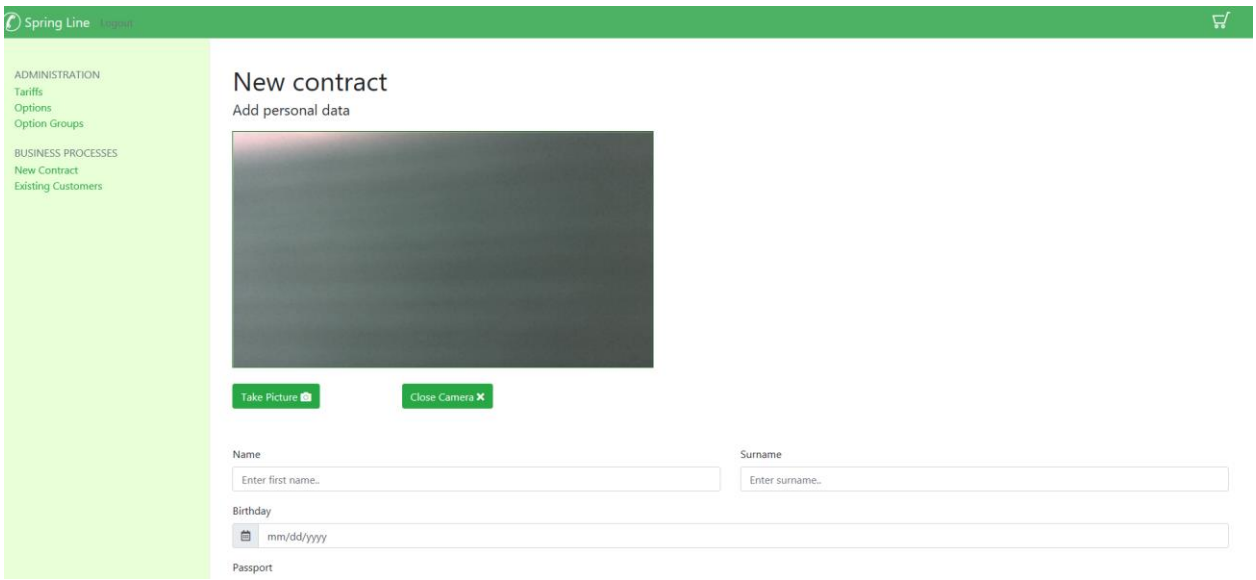
Picture 1.3 – Adding Option Page



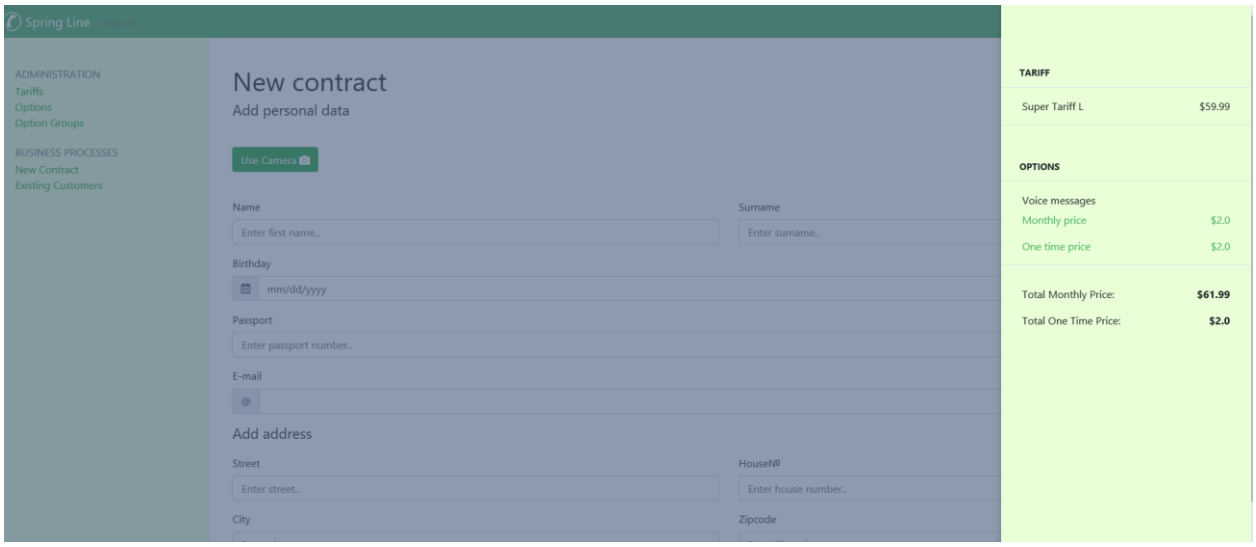
Picture 1.4 – Choosing Tariff For New Contract



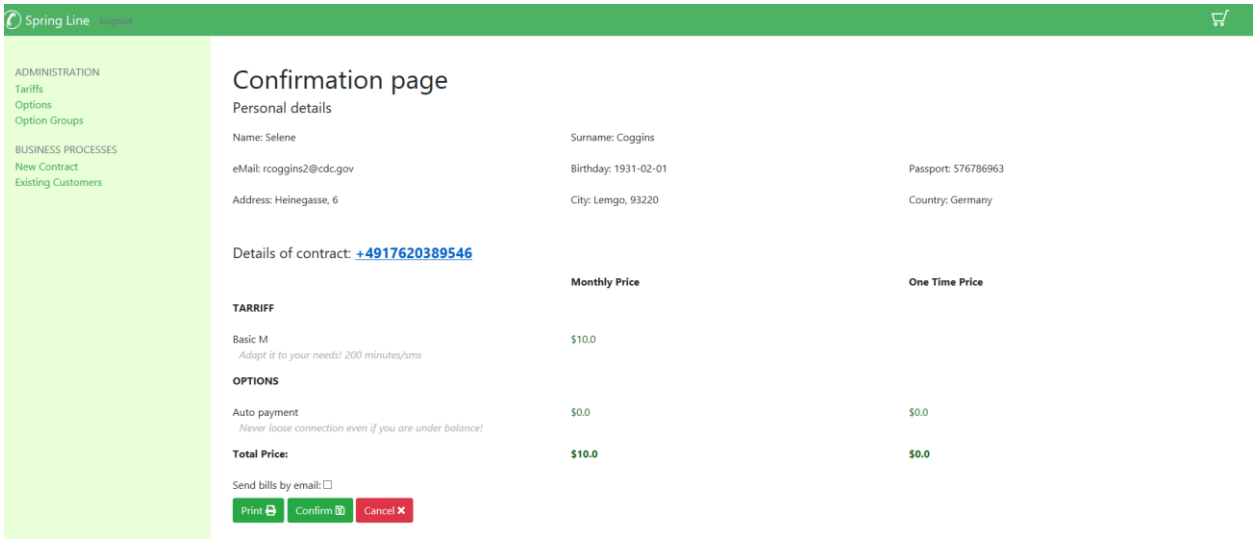
Picture 1.5 – Choosing Options For New Contract



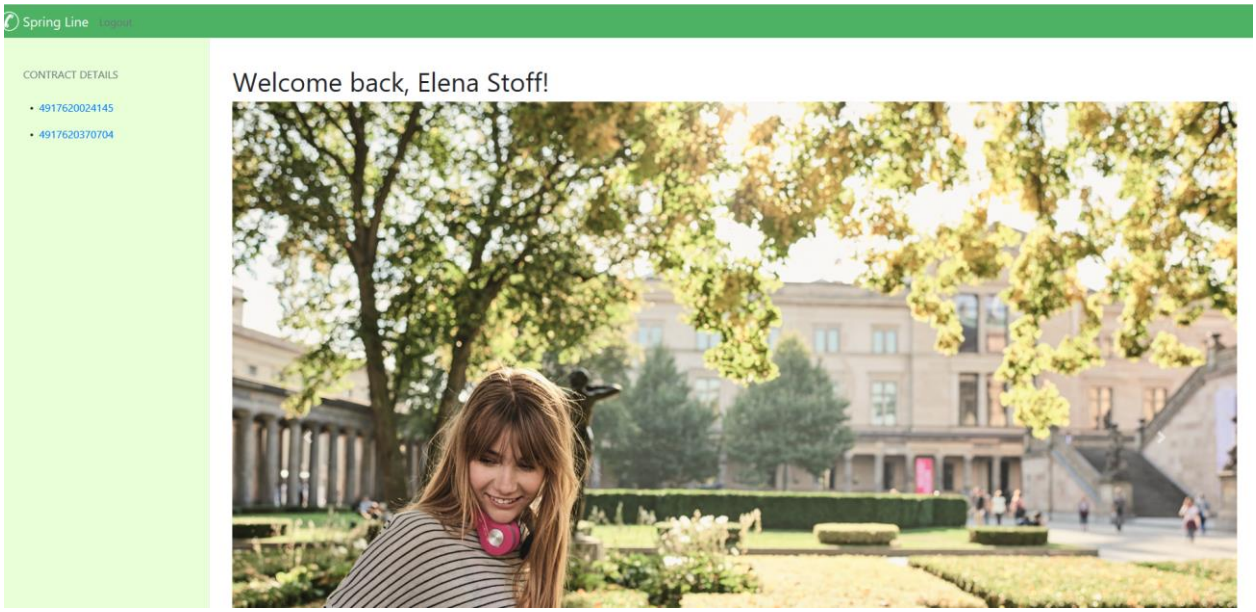
Picture 1.6 – New Client Form



Picture 1.7 – Shopping Cart



Picture 1.8 – Confirmation Page



Picture 1.9– Client Welcome Page

12 Future improvement

1) Micro Services –split up application to several micro services (eg client administration, option administration and tariff administration)

2) Make application work even if cookies are disabled. Now only warning appears

3) Improve security (password rules, two-factor authentication, csrf)

4) Add functionality (deleting contract, choosing accessories etc)