Technical solution description
Ecare - mobile operator imitation

Author: Julia Zhukova

2021

Task

To develop an application that simulates the operation of the information system of a mobile operator.

First application should ensure following functions:

· For the clients:

- View the contract in personal client profile;
- View all possible tariffs and ability to transition, change the tariff;
- View possible options for tariffs, connect new options, disable existing ones;
- Blocking / unblocking the number (if the number is blocked, then you cannot change the tariff and options; if the number was blocked not by the user, then he cannot unblock it himself);

· For employers:

- Conclusion of a contract with a new client: auto-generation of a unique new phone number with tariff activation.
- View all users and contracts;
- Blocking a user / unblocking a user contract;
- Search for a user by number;
- Change of tariff, connection and disconnection of options to the user;
- Adding new tariffs, deleting old ones;
- Add / remove options for the tariff;
- Option management: some options may be incompatible with each other or may be added only with certain options, employee adding and removing these rules.

When performing actions with contracts on each page, before saving changes, a basket should be displayed, in which the selected positions of the user are displayed.

A basket is also implemented for the manager to work with the user, where all the user's choice will be saved until payment. 2 the application is a separate client application for the advertising stand. The application displays a list of new tariffs and available options with their prices. Data reloading is carried out in case of receiving a notification from the server about tariff changes.

Technologies:

Back-end

IDE:

- § IntelliJ IDEA 2020.3.2 (Ultimate Edition)

Database:

- MySQL 8.0.14

Project build management tool:

- Apache Maven 3.6.3

Application Server:

- Wildfly-21.0.1.Final

Servlet Container:

- Tomcat 9.0.331

Git:

- GitHub

Framework:

- Apache ActiveMQ 5.16.0 • Jackson 2.12.1 • JSF 2.3 • JSP 2.3 • EJB • Hibernate • Lombock 1.18.12 4 • Log4J 1.2.3 • ModelMapper • Slf4j 1.7.30 • Spring Boot 2.1.9
- Gson 2.7 (json format serialization)

Tests**:**

• JUnit 5 • Mockito 3.5.7

Front-end

- Bootstrap 4

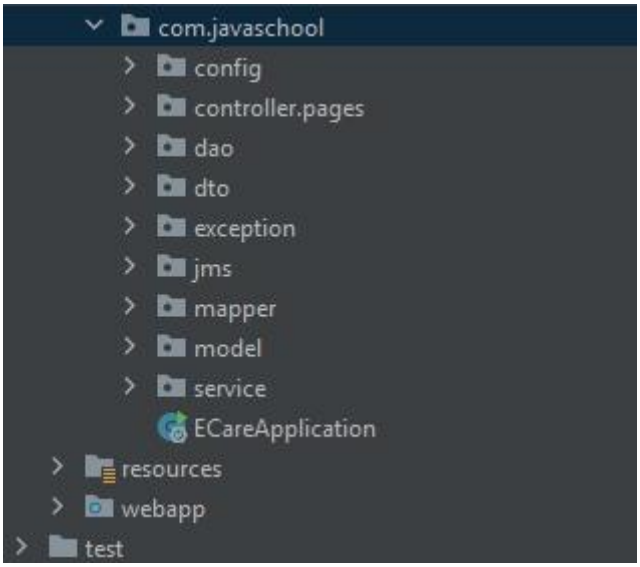
- HTML/CSS
- Primefaces 8.0

Architecture

The application structure implements client-server model in which many clients (web-browsers) request and receive service from a centralized server (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them.

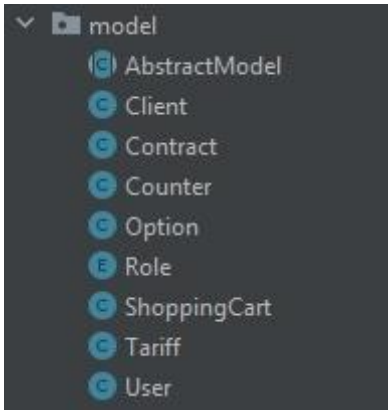
The server module implements model-view-controller design pattern: - the model is responsible for managing the data. - the view represent data as web-pages; - the controller responds to the user input and performs interactions on the data model objects. The controller receives the input, passes the input to the model and then updates views.

Picture 1 represents general application modules:

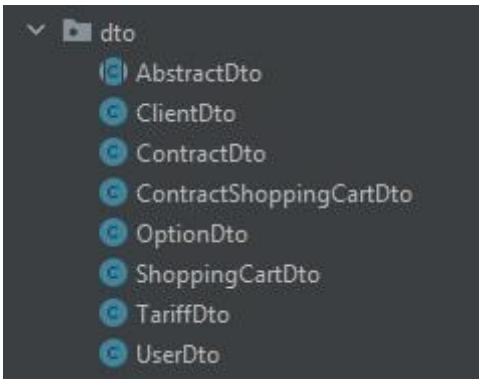


Picture 1

Picture 2 represents entities stored in database. Picture 3 represents data transfer object (DTO): helper objects that keep user inputs when pass data from controllers to model services or keeps entities properties when pass data from model to views. DTO objects prevent persisted objects from unchecked mutates and isolate them from controller and view layer. Java Validation API is used here to implement DTO property validation.

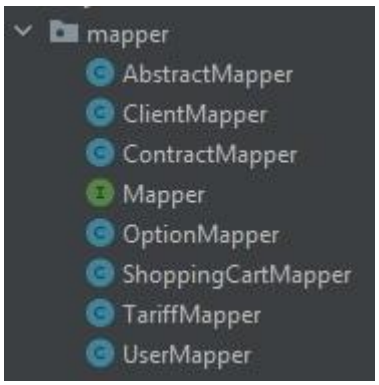


Picture 2



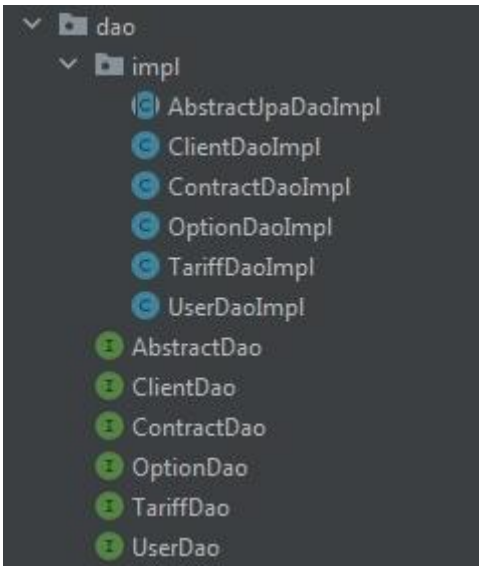
Picture 3

ModelMapper serves to transform entity into dto and back.

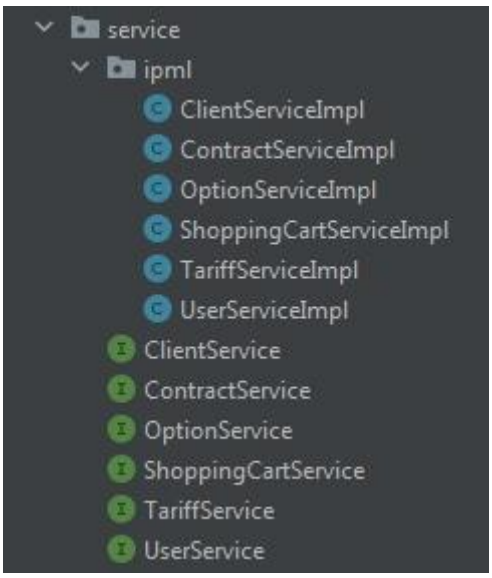


Picture 4

Picture 5 represents object used to access database (data accessible objects, or DAO). Each DAO implements corresponding interface. Picture 6 represents model services. These services perform all business logic. Each service implements corresponding interface.

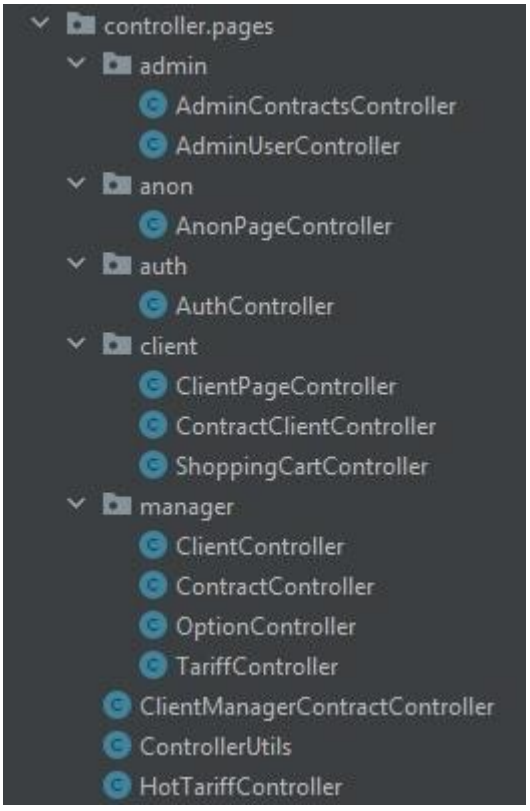


Picture 5



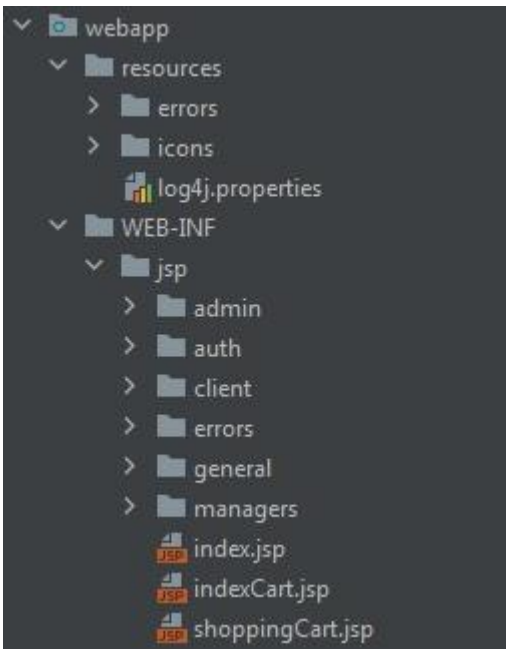
Picture 6

Controller layer Picture 7 represents controller layer



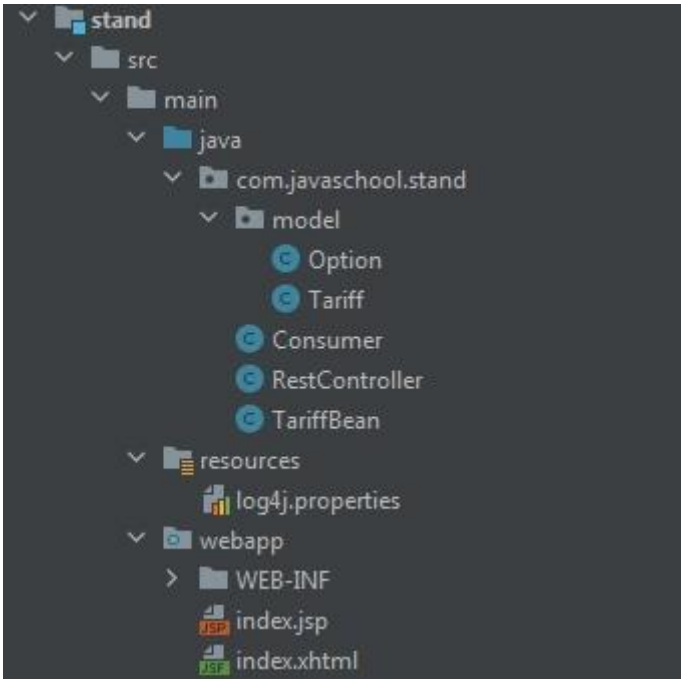
Picture 7

View layer Picture 8 represents view layer.



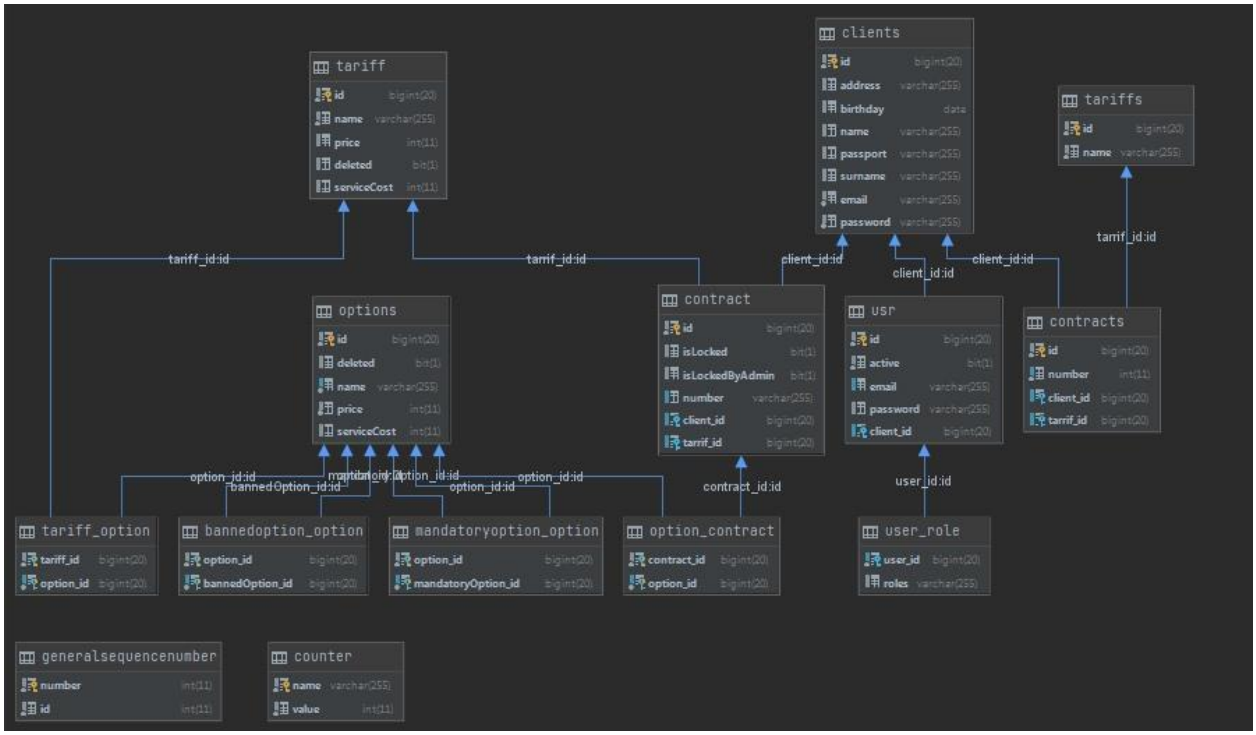
Picture 8

App2 architecture Extra application is a one-page application based on EJB and JSF frameworks:



Picture 8

Database schema



Picture 9

Testing environment Unit-tests are written for business logic using Junit5 and Mockito. The coverage is around 50% for the service layer.

Showcase is a simple one-page application deployed on WildFly application server. It communicates with another application via rest requests. When the main application is uploaded, it sends to this application a message about that in topic on the server (using JMS). After that second app send REST request to take a data - 6 last tariffs. When data receives, a message is pushed to the JSF side via websocket, after that data on this page is updating with AJAX. When second app could not connect to remote source, warning message appears.