Mobile operator information system (eCare)

Technical Solution Description

Author: Gitenko Oleg

Table of contents

Project

This project constitutes an manage system for a mobile operator called eCare. It consists of two parts: the main application that provides functionality for both managing the mobile operator network and using its services, and a separate ad board that displays all tariffs and options.

Basic functionality

The main application provides following possibility:

For employees

- create, edit and delete options. Create rules for options binding.
- create, delete tariffs. Add and remove options from tariff.
- create a new client and edit in further (change personal data).
- sign a new contract with client and edit in further (change tariff, add or remove options from contract, block or unblock phone number).
- searching contracts by phone. Getting all clients and contracts.

For clients

- getting information about personal tariff.
- Block/unblock phone number.
- Change tariff and options using a shopping cart.
- Connect/disconnect options in your own tariff.

Second application is an ad board. All tariffs and options are presented here. It also shows the tariff with the best price and the tariff with the longest list of options.

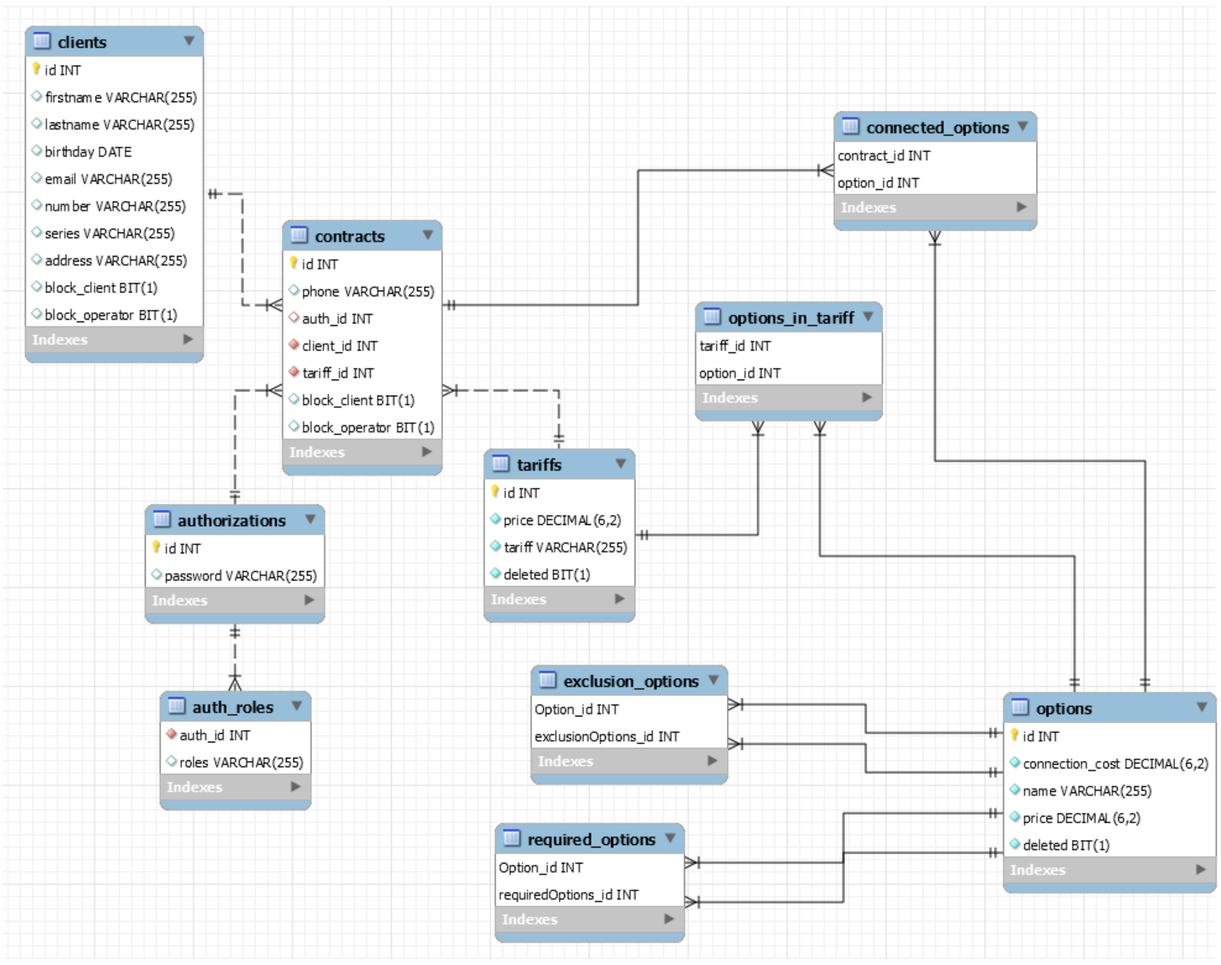
Technology stack

• **Both applications:**
• Maven 3
• Junit4
• Mockito
• SLF4J
• Log4j12
• Lombok
• Bootstrap 4
• ActiveMQ 5.16
• JMS
• **Main application:**
• Spring MVC
• Spring Security
• Spring JMS
• JSP/JSTL
• MySQL 5.6
• Hamcrest 2.1
• JPA
• javascript
• Tomcat 8.5.51
• Docker 19.03.12

• **Ad board:**
• Wildfly 20.0.1
• Java EE 8
• EJB
• JSF 2.3
• Gson 2.8
• Jersey 1.19
• Docker 19.03.12

Data model

Database scheme



Model implementation

Tables AUTHORIZATION and AUTH_ROLES store information for authorization and authentication on the website. Each contract in CONTRACTS table have a link on AUTHORIZATION table.

The table CONTRACTS store contracts information to identify each client by phone. Phone is a unique field. It also has links on a CLIENTS table and connected TARIFF.
TARIFFS table contains information about tariff and binds with the table OPTIONS_IN_TARIFF which shows available options in each tariff.

The table CLIENTS describes each client. There are all personal data in this table.
Fields series and number form a composite unique index.

The table CONNECTED_OPTIONS has two links, on CONTRACTS table and OPTIONS table. Ensures that information is saved for each client when options or tariffs are changed e.g. if option is marked as deleted, clients will use it until it won't be disconnected by client or operator.

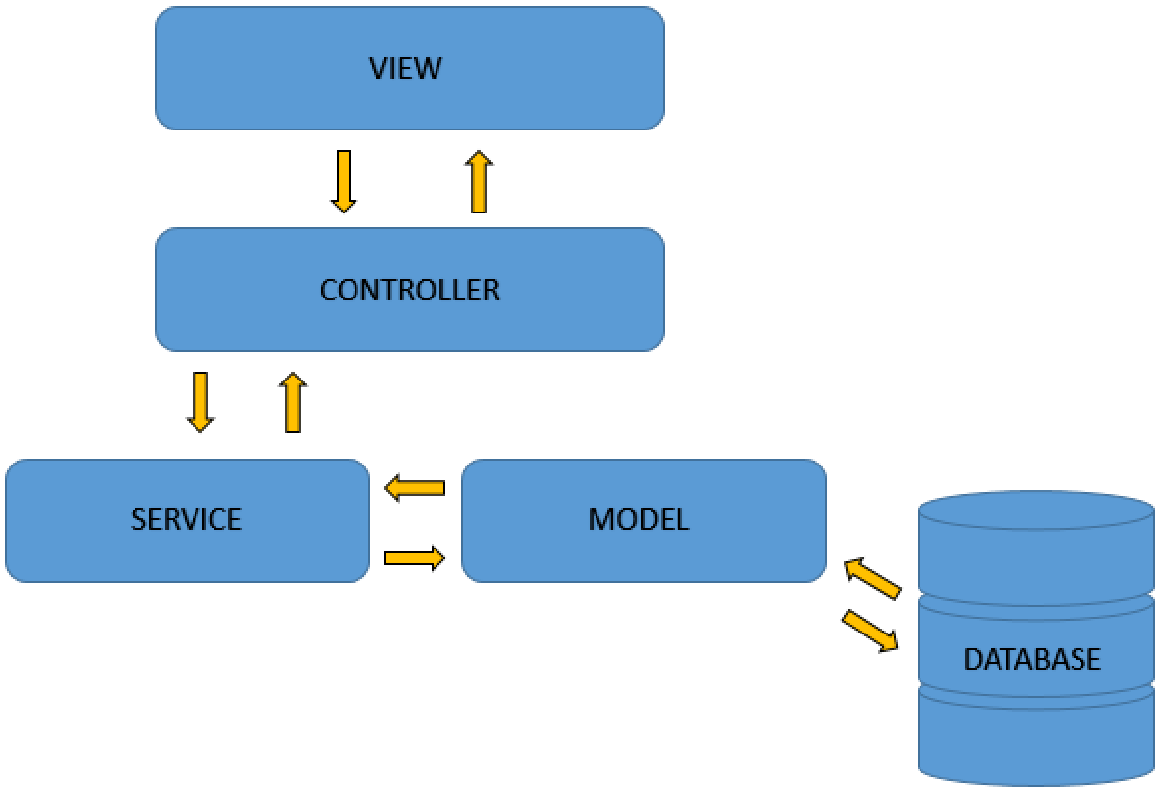
The table OPTIONS describes each option.

The table REQUIRED_OPTIONS has two links. Both of it bind with the table OPTIONS. First column contains an options ID, second column contains another options ID that is required to connect for the first option (means that you have to connect second option before you can use first option).

The table EXCLUSION_OPTIONS has two links and both of it bind with the table OPTIONS again. But in this case, second column contains an options ID which can’t be connected at the same time in one contract with an option in first column.

Application

The application builds on MVC architecture.

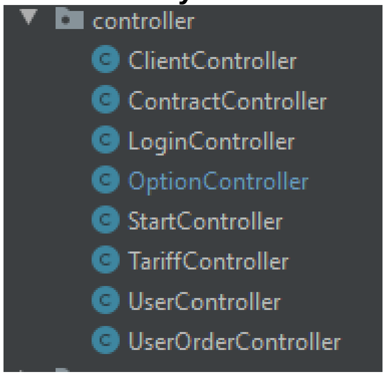


View is responsible for display all information from server and transfer all request from user to server. Controller receives requests and processes them. Controller interacts by DTO objects with service layer which processes all business logic. Service layer converts all DTO objects to DAO objects, validates the data and then makes request to model layer.
Model layer interacts with Database (all CRUD operation) and returns results to service layer. Service layer gets data from model layer, realizes all logic, converts DAO objects to DTO objects and returns them to the Controller.

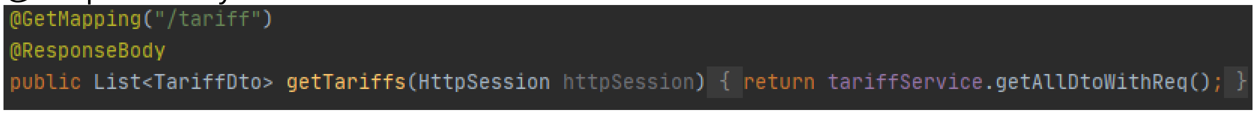
Controller level.

This level consists of plain («view») controllers that are responsible for processing requests and providing data for UI rendering, rest controller is responsible for processing REST requests and providing response containing necessary data, and error handlers.
View controllers communicate with service layer in order to obtain necessary model attribute maps.

Controller layer

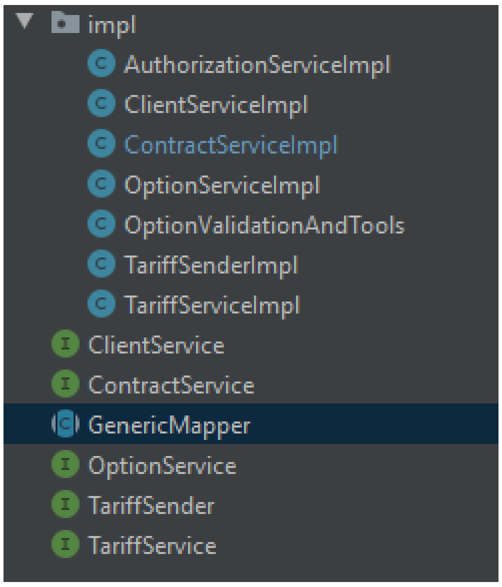

controller
ClientController
ContractController
LoginController
OptionController
StartController
TariffController
UserController
UserOrderController

Each controller responsible for one entity or business logic e.g. tariffcontroller is responsible for all tariff requests. In most of case controllers are simple “view” controller which return name of JSP page. If a controller method returns data in a JSON format it will be marked as @ResponseBody


```
@GetMapping("/tariff")
@ResponseBody
public List<TariffDto> getTariffs(HttpSession httpSession) { return tariffService.getAllDtoWithReq(); }
```

Service level.

Services are responsible for communication with DAO and conversion of entities into DTO. For this each service class implements Generic Mapper abstract class. Also there are all business logic here. They also provide transaction handling via Spring @Transactional annotation. Specific methods have an additional separate @Transactional annotation in order to achieve necessary transaction propagation and isolation.


impl
AuthorizationServiceImpl
ClientServiceImpl
ContractServiceImpl
OptionServiceImpl
OptionValidationAndTools
TariffSenderImpl
TariffServiceImpl
ClientService
ContractService
GenericMapper
OptionService
TariffSender
TariffService

Service layer has an auxiliary class - OptionValidationAndTools. This class validates all requirements for options e.g. it is possible or not to connect some options with others.

All services are responsible for one entity or business logic e.g. contract service is responsible for all CRUD operation with contracts.

TariffSenderImpl is an implementation of a TariffSender interface. He is responsible for interacting with REST requests.

Model level

This level includes entities that represent database tables and DAO that are used to communicate with the database.

- **Entities**

• *Authorization* – store password and roles;
• *Client* – contains all client's personal data;
• *Contract* - represents contract which contains phone number, client, authorization, tariff ;
• *Option* – describes option and also stores information about connection requirements;
• *Role* - represents a user account role associated with a certain set of permissions;
• *Tariff* – stores information about tariff and connected options;

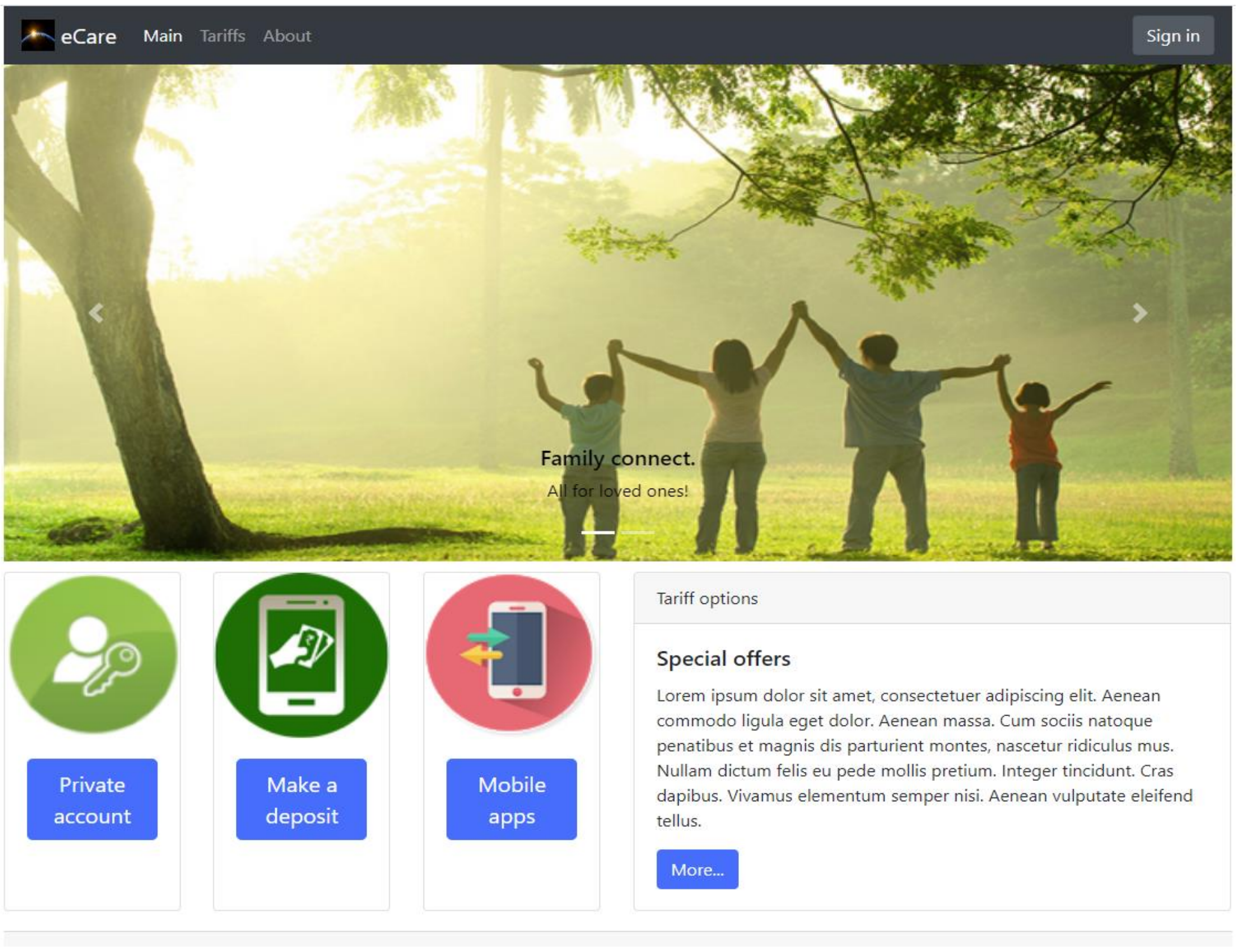
- **DAO**

• *ClientDao* - contains specific methods for obtaining client's information. Implemented by *ClientDaoImpl*;
• *ContractDao* – contains specific methods for obtaining contract's information. Implemented by *ContractDaoImpl*;
• *OptionDao* - contains specific methods for obtaining option's information. Implemented by *OptionDaoImpl*;
• *TariffDao* - contains specific methods for obtaining tariff's information. Implemented by *TariffDaoImpl*;
• *GenericDao* - contains all CRUD methods for managing DAO's information. Implemented by *GenericDaoImpl*;

View level.

This level based on a JSP technology. It also uses bootstrap CSS styles and javascript for frontend validation.

Start page example.

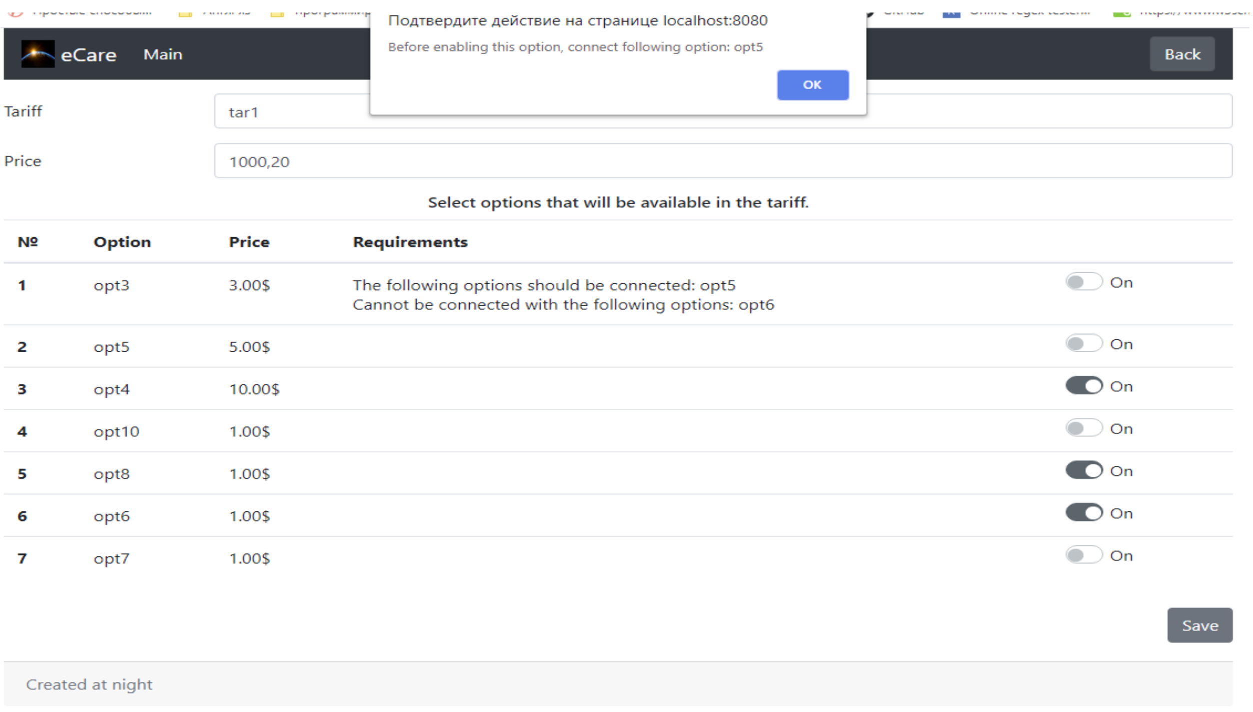


Tariff options

Special offers

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus.

JS validation example.



Подтвердите действие на странице localhost:8080

Before enabling this option, connect following option: opt5

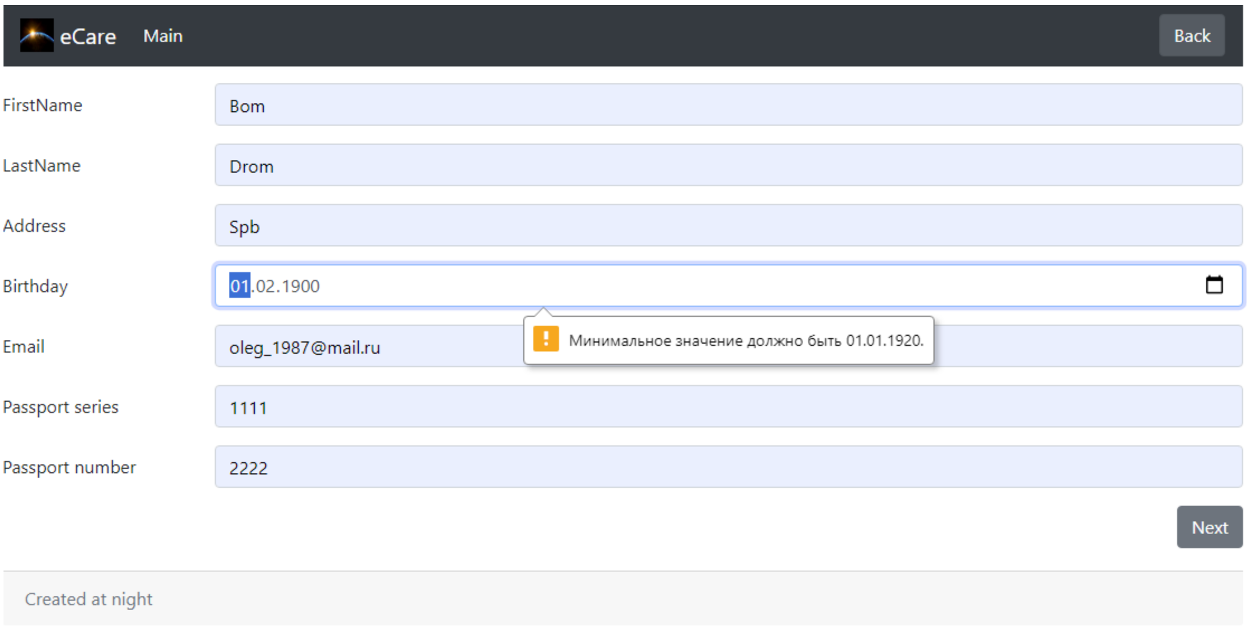
Tariff	tar1
Price	1000,20

Select options that will be available in the tariff.

№	Option	Price	Requirements	
1	opt3	3.00$	The following options should be connected: opt5 Cannot be connected with the following options: opt6	On
2	opt5	5.00$		On
3	opt4	10.00$		[✓ On]
4	opt10	1.00$		On
5	opt8	1.00$		[✓ On]
6	opt6	1.00$		[✓ On]
7	opt7	1.00$		On

Created at night

HTML validation.



FirstName	Bom
LastName	Drom
Address	Spb
Birthday	01.02.1900
Email	oleg_1987@mail.ru
Passport series	1111
Passport number	2222

Минимальное значение должно быть 01.01.1920.

Created at night

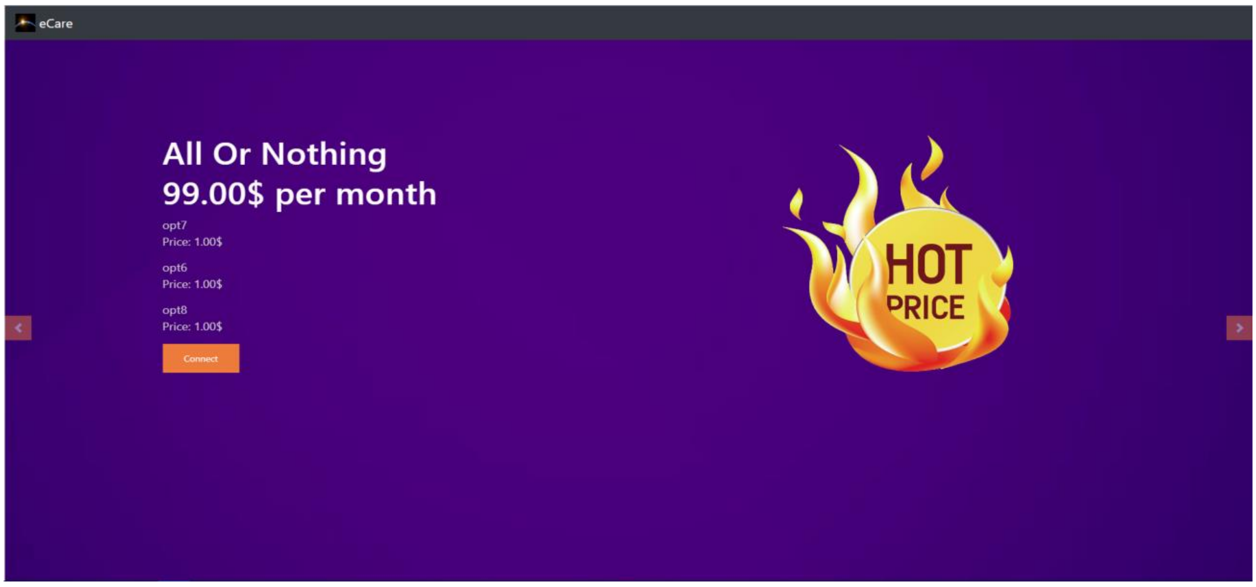
Ad board (eCareAd).

This module is a Java EE application. It consists of *RestClient* that obtains
necessary data from the «eCare» module through REST requests;
@ApplicationScoped @Named *UiBean* is managed bean which binds with a JSF page.
@Singletone JmsConsumer is responsible for connecting to ActiveMq broker.
@Singleton @LocalBean JmsMessageListener is a broker's listener. Also it is responsible for receiving information from broker and updating tariff's list in UiBean.
Also UiBean is responsible for communicating to user interface about any update via PushContext.

User interface

User interface in this module is rendered by JSF 2.3. Communication with @Named bean is done with the help of <f:ajax> (when another station is chosen from the list) and <f:websocket> (when there are updates in the schedule). CSS is provided via Bootstrap 4.

Screenshots



All Or Nothing
99.00$ per month

opt7
Price: 1.00$

opt6
Price: 1.00$

opt8
Price: 1.00$

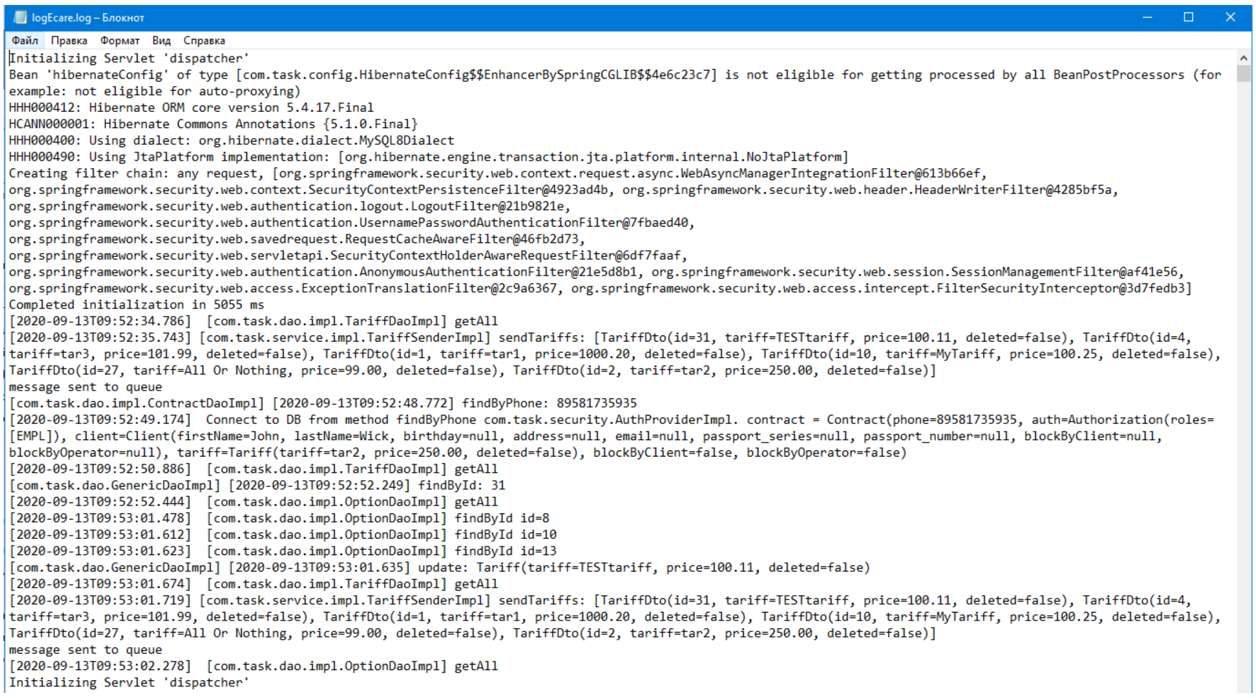
Additional information

Authorization and authentication

Authorization and authentication is handled by Spring Security framework with URL-based access control. I also added a custom login page and a custom AccessDeniedHandler that is responsible for logging an attempt to access a restricted URL and redirecting the user to the custom error page. Also I have a custom AuthSuccessHandler that is responsible redirecting the user to page according to roles.
BCrypt encryption is used for authorization.

Logging

Logging for both projects is handled by SLF4J over Log4J2. Logger
properties are configured in log4j.properties files and provide for both console and rolling file logs.

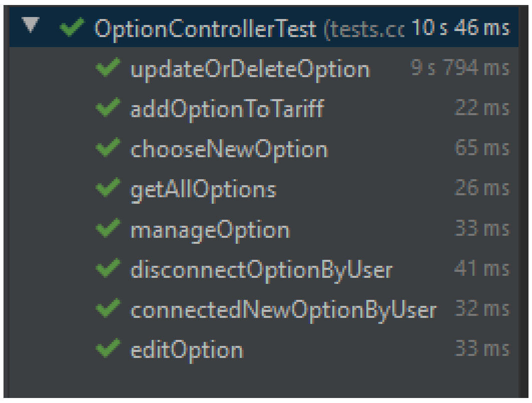


```
Initializing Servlet 'dispatcher'
Bean 'hibernateConfig' of type [com.task.config.HibernateConfig$$EnhancerBySpringCGLIB$$4e6c23c7] is not eligible for getting processed by all BeanPostProcessors (for
example: not eligible for auto-proxying)
HHH000412: Hibernate ORM core version 5.4.17.Final
HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Creating filter chain: any request, [org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@613b66ef,
org.springframework.security.web.context.SecurityContextPersistenceFilter@4923ad4b, org.springframework.security.web.header.HeaderWriterFilter@4285bf5a,
org.springframework.security.web.authentication.logout.LogoutFilter@21b9821e,
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter@7fbaed40,
org.springframework.security.web.savedrequest.RequestCacheAwareFilter@46fb2d73,
org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@6df7faaf,
org.springframework.security.web.authentication.AnonymousAuthenticationFilter@21e5d8b1, org.springframework.security.web.session.SessionManagementFilter@af41e56,
org.springframework.security.web.access.ExceptionTranslationFilter@2c9a6367, org.springframework.security.web.access.intercept.FilterSecurityInterceptor@3d7fedb3]
Completed initialization in 5055 ms
[2020-09-13T09:52:34.786]  [com.task.dao.impl.TariffDaoImpl] getAll
[2020-09-13T09:52:35.743] [com.task.service.impl.TariffSenderImpl] sendTariffs: [TariffDto(id=31, tariff=TESTtariff, price=100.11, deleted=false), TariffDto(id=4,
tariff=tar3, price=101.99, deleted=false), TariffDto(id=1, tariff=tar1, price=1000.20, deleted=false), TariffDto(id=10, tariff=MyTariff, price=100.25, deleted=false),
TariffDto(id=27, tariff=All Or Nothing, price=99.00, deleted=false), TariffDto(id=2, tariff=tar2, price=250.00, deleted=false)]
message sent to queue
[com.task.dao.impl.ContractDaoImpl] [2020-09-13T09:52:48.772] findByPhone: 89581735935
[2020-09-13T09:52:49.174]  Connect to DB from method findByPhone com.task.security.AuthProviderImpl. contract = Contract(phone=89581735935, auth=Authorization(roles=
[EMPL]), client=Client(firstName=John, lastName=Wick, birthday=null, address=null, email=null, passport_series=null, passport_number=null, blockByClient=null,
blockByOperator=null), tariff=Tariff(tariff=tar2, price=250.00, deleted=false), blockByClient=false, blockByOperator=false)
[2020-09-13T09:52:50.886]  [com.task.dao.impl.TariffDaoImpl] getAll
[com.task.dao.GenericDaoImpl] [2020-09-13T09:52:52.249] findById: 31
[2020-09-13T09:52:52.444]  [com.task.dao.impl.OptionDaoImpl] getAll
[2020-09-13T09:53:01.478]  [com.task.dao.impl.OptionDaoImpl] findById id=8
[2020-09-13T09:53:01.612]  [com.task.dao.impl.OptionDaoImpl] findById id=10
[2020-09-13T09:53:01.623]  [com.task.dao.impl.OptionDaoImpl] findById id=13
[com.task.dao.GenericDaoImpl] [2020-09-13T09:53:01.635] update: Tariff(tariff=TESTtariff, price=100.11, deleted=false)
[2020-09-13T09:53:01.674]  [com.task.dao.impl.TariffDaoImpl] getAll
[2020-09-13T09:53:01.719] [com.task.service.impl.TariffSenderImpl] sendTariffs: [TariffDto(id=31, tariff=TESTtariff, price=100.11, deleted=false), TariffDto(id=4,
tariff=tar3, price=101.99, deleted=false), TariffDto(id=1, tariff=tar1, price=1000.20, deleted=false), TariffDto(id=10, tariff=MyTariff, price=100.25, deleted=false),
TariffDto(id=27, tariff=All Or Nothing, price=99.00, deleted=false), TariffDto(id=2, tariff=tar2, price=250.00, deleted=false)]
message sent to queue
[2020-09-13T09:53:02.278]  [com.task.dao.impl.OptionDaoImpl] getAll
Initializing Servlet 'dispatcher'
```

Unit tests

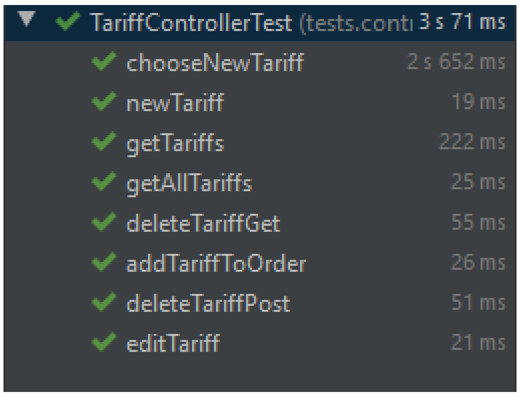
Unit tests are written with JUnit 4 and Mockito Extension. First and foremost, I wrote the tests to cover controllers and services.

- *OptionControllerTest* - tests handling of BindingResult with or without errors;



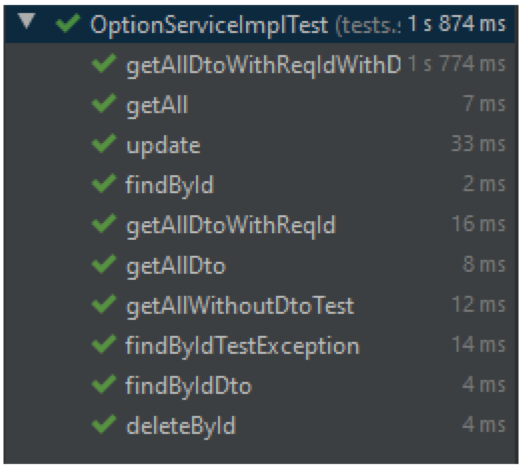
```
✓ OptionControllerTest (tests.cc 10 s 46 ms
    ✓ updateOrDeleteOption      9 s 794 ms
    ✓ addOptionToTariff             22 ms
    ✓ chooseNewOption               65 ms
    ✓ getAllOptions                 26 ms
    ✓ manageOption                  33 ms
    ✓ disconnectOptionByUser        41 ms
    ✓ connectedNewOptionByUser      32 ms
    ✓ editOption                    33 ms
```

• *TariffControllerTest* - tests handling of BindingResult with or without errors;



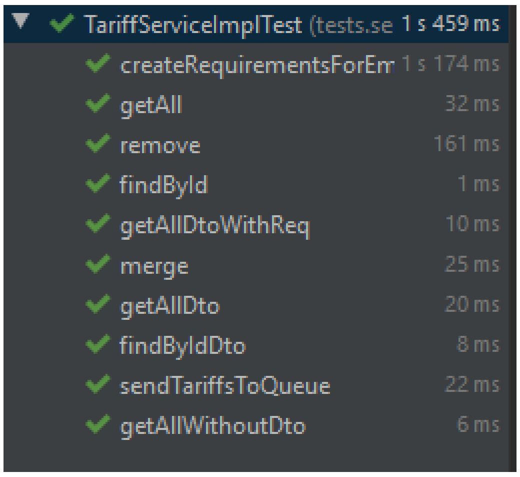
```
▼ ✓ TariffControllerTest (tests.conti 3 s 71 ms
    ✓ chooseNewTariff          2 s 652 ms
    ✓ newTariff                     19 ms
    ✓ getTariffs                   222 ms
    ✓ getAllTariffs                 25 ms
    ✓ deleteTariffGet               55 ms
    ✓ addTariffToOrder              26 ms
    ✓ deleteTariffPost              51 ms
    ✓ editTariff                    21 ms
```

•*OptionServiceImplTest* - checks all CRUD operation with MOCK. It also checks business logics and conversation Dto into Dao and vice versa.



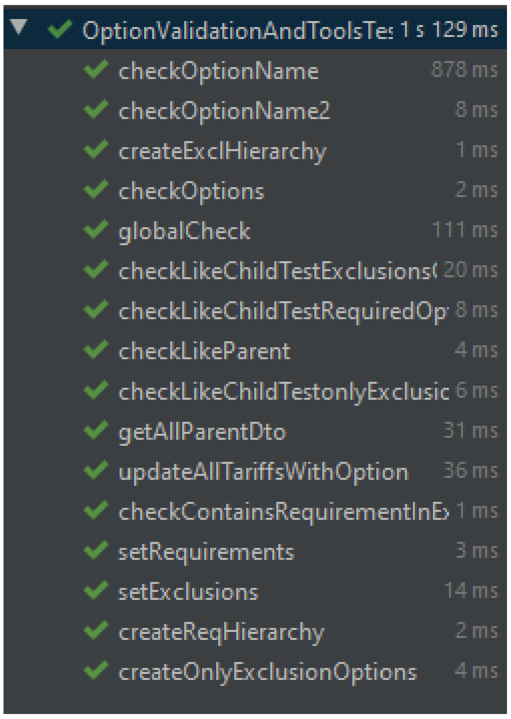
```
▼ ✓ OptionServiceImplTest (tests.: 1 s 874 ms
    ✓ getAllDtoWithReqIdWithD 1 s 774 ms
    ✓ getAll                       7 ms
    ✓ update                      33 ms
    ✓ findById                     2 ms
    ✓ getAllDtoWithReqId          16 ms
    ✓ getAllDto                    8 ms
    ✓ getAllWithoutDtoTest        12 ms
    ✓ findByIdTestException       14 ms
    ✓ findByIdDto                  4 ms
    ✓ deleteById                   4 ms
```

•*TariffServiceImplTest* - checks all CRUD operation with MOCK. It also checks business logics and conversation Dto into Dao and vice versa.



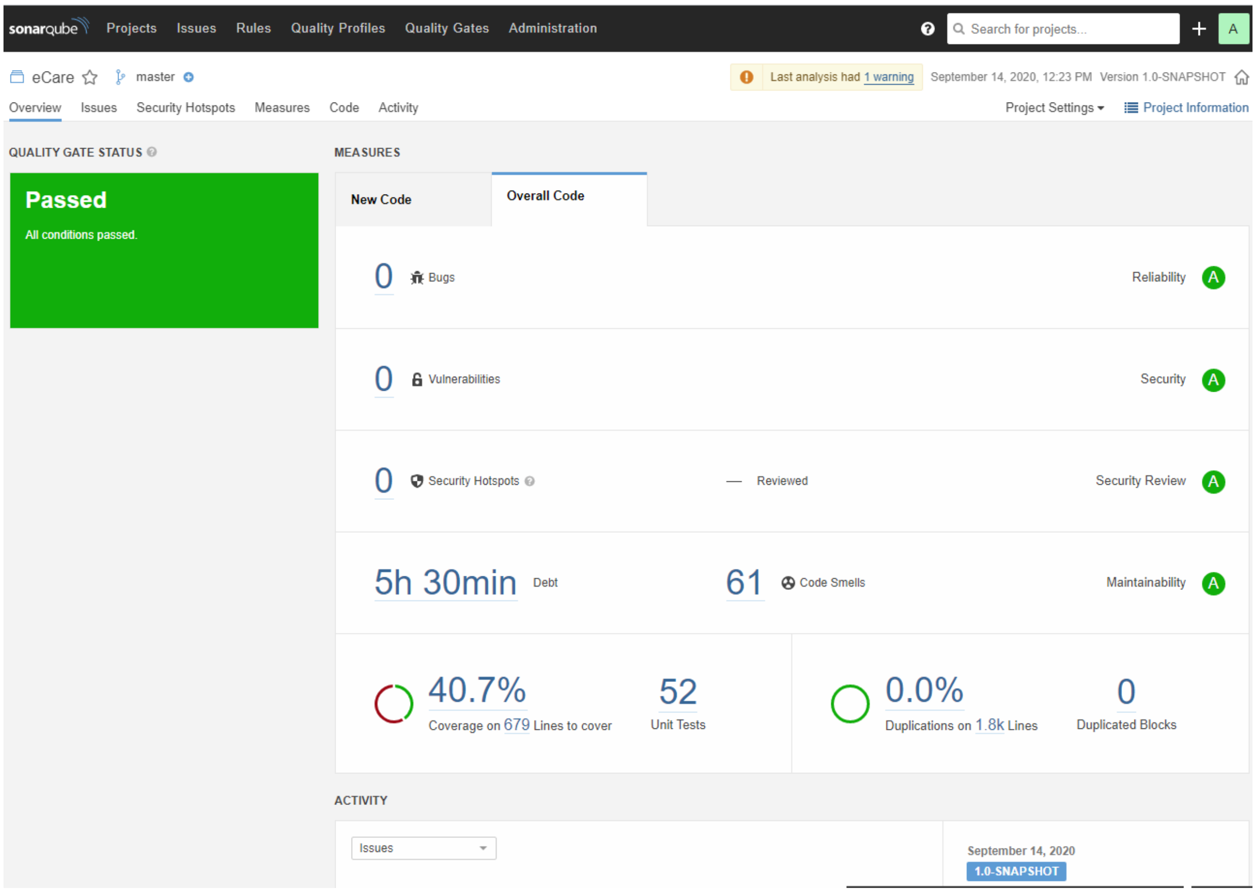
```
▼ ✓ TariffServiceImplTest (tests.se 1 s 459 ms
    ✓ createRequirementsForEm 1 s 174 ms
    ✓ getAll                      32 ms
    ✓ remove                     161 ms
    ✓ findById                     1 ms
    ✓ getAllDtoWithReq            10 ms
    ✓ merge                       25 ms
    ✓ getAllDto                   20 ms
    ✓ findByIdDto                  8 ms
    ✓ sendTariffsToQueue          22 ms
    ✓ getAllWithoutDto             6 ms
```

•*OptionValidationAndToolsTest* - checks all validation operation with options in a tariff.



```
▼ ✓ OptionValidationAndToolsTe: 1 s 129 ms
    ✓ checkOptionName                878 ms
    ✓ checkOptionName2                 8 ms
    ✓ createExclHierarchy              1 ms
    ✓ checkOptions                     2 ms
    ✓ globalCheck                    111 ms
    ✓ checkLikeChildTestExclusions( 20 ms
    ✓ checkLikeChildTestRequiredOp   8 ms
    ✓ checkLikeParent                  4 ms
    ✓ checkLikeChildTestonlyExclusic 6 ms
    ✓ getAllParentDto                 31 ms
    ✓ updateAllTariffsWithOption      36 ms
    ✓ checkContainsRequirementInE:   1 ms
    ✓ setRequirements                  3 ms
    ✓ setExclusions                   14 ms
    ✓ createReqHierarchy               2 ms
    ✓ createOnlyExclusionOptions       4 ms
```

SonarCloud report.



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

eCare master

Last analysis had 1 warning September 14, 2020, 12:23 PM Version 1.0-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

5h 30min Debt — 61 Code Smells — Maintainability A

40.7% Coverage on 679 Lines to cover — 52 Unit Tests

0.0% Duplications on 1.8k Lines — 0 Duplicated Blocks

ACTIVITY

Issues

September 14, 2020
1.0-SNAPSHOT

Built and deployed.

Both projects are created by one command - build in maven and deployed in a Docker container.
Main project eCare have two config files: Dockerfile and docker-compose.yml.
Dockerfile based on tomcat image ver. 8.5.51.
docker-compose.yml. binds several containers in one network.
Also docker-compose.yml creates two additional containers. First container for database which based on mysql image ver. 5.7.31 and second container for adminer (UI for database) which based on adminer image.

In docker-compose.yml we have following settings:

Database:

- test DB which will be deployed on first start.
- all necessity settings for creating root password and user.
- time zone settings.
- transferring port 3306 to 3307 on a host machine.

Application

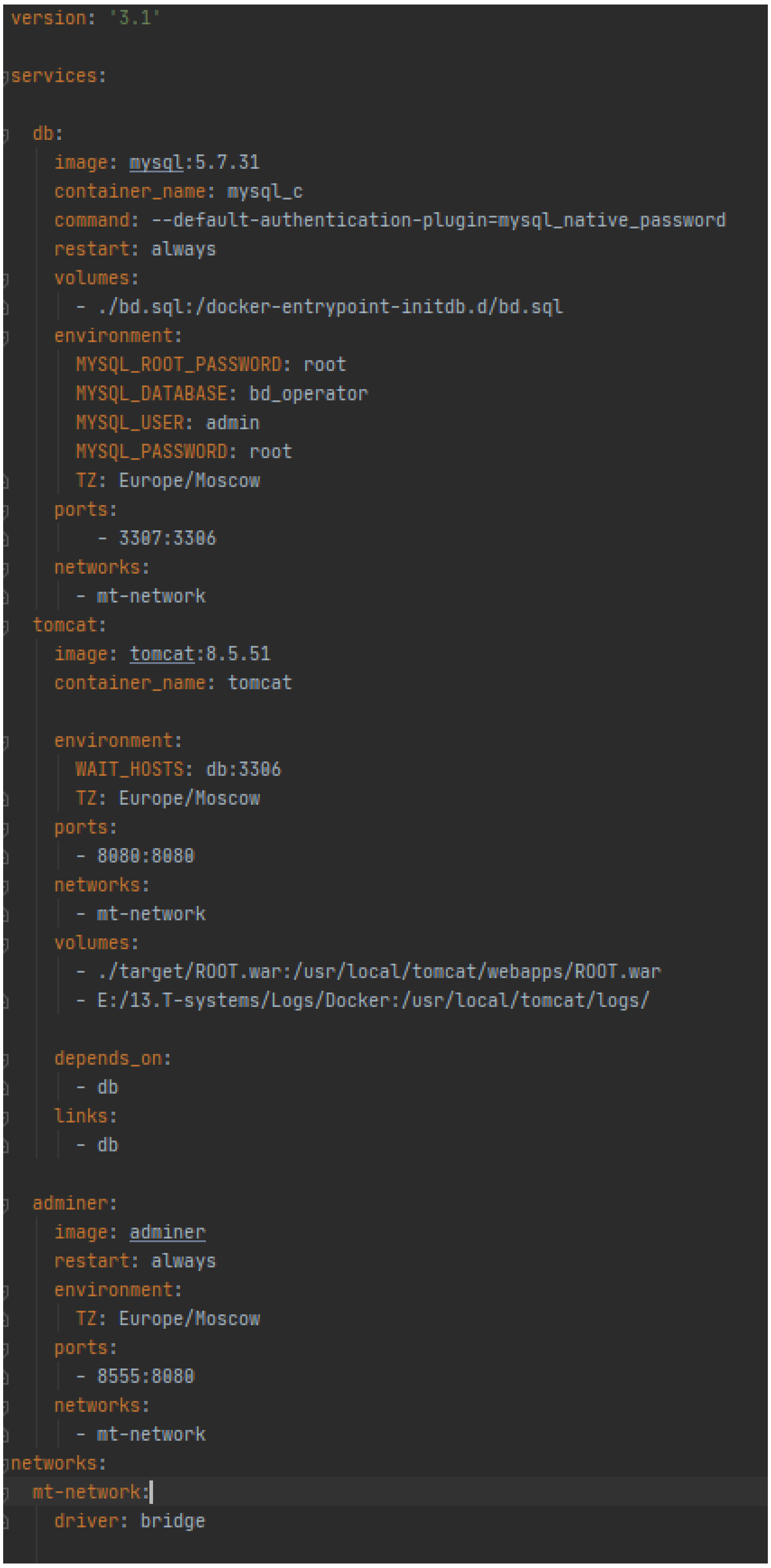
- binding a .war file on a host machine to a .war file in a container.
- sharing a folder on a host machine with log folder in a container
- time zone settings.
- transferring port 8080 to 8080 on a host machine.

Adminer

- time zone settings.
- transferring port 8080 to 8555 on a host machine.

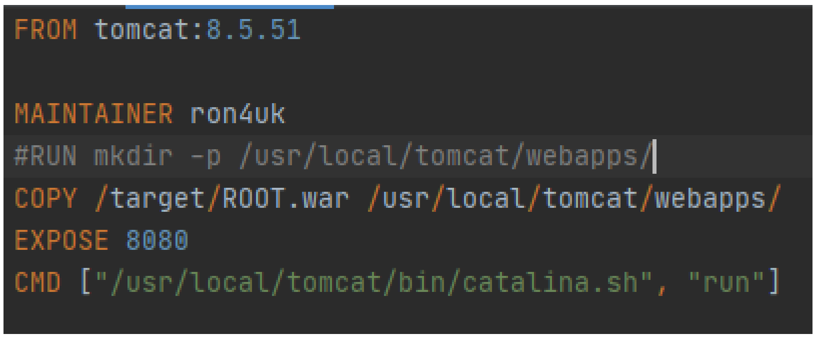
Creating common network for all containers.

docker-compose.yml



```yaml
version: '3.1'

services:

  db:
    image: mysql:5.7.31
    container_name: mysql_c
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    volumes:
      - ./bd.sql:/docker-entrypoint-initdb.d/bd.sql
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: bd_operator
      MYSQL_USER: admin
      MYSQL_PASSWORD: root
      TZ: Europe/Moscow
    ports:
        - 3307:3306
    networks:
      - mt-network
  tomcat:
    image: tomcat:8.5.51
    container_name: tomcat

    environment:
      WAIT_HOSTS: db:3306
      TZ: Europe/Moscow
    ports:
      - 8080:8080
    networks:
      - mt-network
    volumes:
      - ./target/ROOT.war:/usr/local/tomcat/webapps/ROOT.war
      - E:/13.T-systems/Logs/Docker:/usr/local/tomcat/logs/

    depends_on:
      - db
    links:
      - db

  adminer:
    image: adminer
    restart: always
    environment:
      TZ: Europe/Moscow
    ports:
      - 8555:8080
    networks:
      - mt-network
networks:
  mt-network:
    driver: bridge
```

Dockerfile



```
FROM tomcat:8.5.51

MAINTAINER ron4uk
#RUN mkdir -p /usr/local/tomcat/webapps/
COPY /target/ROOT.war /usr/local/tomcat/webapps/
EXPOSE 8080
CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]
```

Running all applications in Docker is carried out only by one command:

$ docker-compose up

eCareAd

Second application also is deployed in a Docker container.
We have only one Dockerfile with following settings.

- time zone settings.
- expose ports 9990, 8080, 9000.

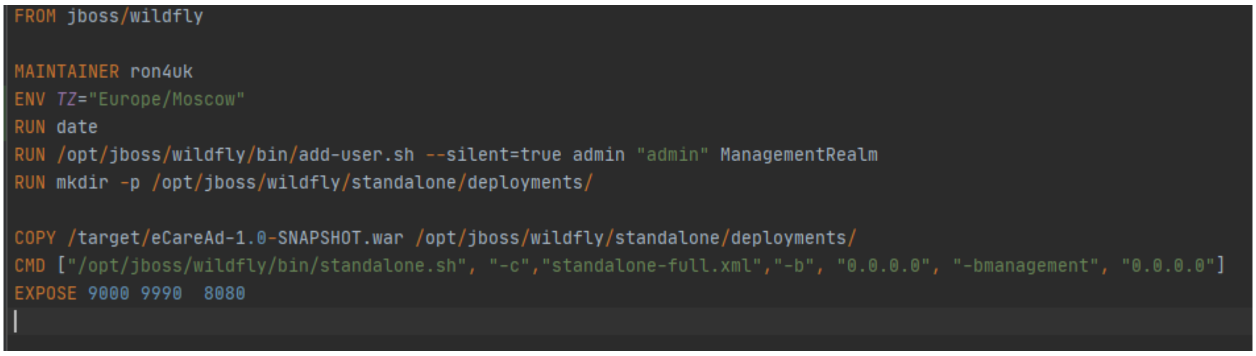
Running application in Docker is carried out by following commands:

$ dicker build –t wild .
 Creates image jboss/wildfly with name “wild”.

$ docker run --name ecareAd -p 9080:8080 -v E:/13.T-systems/LogsAd/Docker:/opt/jboss/wildfly/standalone/log/ wild

-t, --name – can be any.
-v E:/13.T-systems/LogsAd/Docker:/opt/jboss/wildfly/standalone/log/ - sharing a folder on a host machine with log folder in a container

Dockerfile



```
FROM jboss/wildfly

MAINTAINER ron4uk
ENV TZ="Europe/Moscow"
RUN date
RUN /opt/jboss/wildfly/bin/add-user.sh --silent=true admin "admin" ManagementRealm
RUN mkdir -p /opt/jboss/wildfly/standalone/deployments/

COPY /target/eCareAd-1.0-SNAPSHOT.war /opt/jboss/wildfly/standalone/deployments/
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-c","standalone-full.xml","-b", "0.0.0.0", "-bmanagement", "0.0.0.0"]
EXPOSE 9000 9990  8080
```