Ecare Project Specification

(T-Systems Java School 12 project)

Author: Starostin Konstantin, 2014.

1. Task

In the task required to write an application that simulates the operation of mobile network operator information system. Below is a more detailed description of subject area and technical requirements.

1.1 Subject area

There are following kinds of entities:

- Tariff
  - Title
  - Price
  - List of available options
- Option
  - Title
  - Price
  - Cost of connection
- Client
  - Name
  - Last name
  - Birth date
  - Passport data
  - Address
  - List of contracts (telephones numbers of client)
  - E-mail
  - Password
- Contract
  - Telephone number
  - Tariff
  - Connected options for tariff

The application must provide the following functionality:

- For clients
  - Browse of the contract in a personal cabinet;
  - Browse of all available tariffs and changing of tariff;
  - Browse of all available options for tariff, adding new options, disable the existing ones;
  - Lock / Unlock of number (if number is locked, you cannot change tariff and options; if number is not locked by client, he can't unlock it);
- For employees
  - Conclusion of a contract with a new client: the choice of a new telephone number with the tariff and options. The phone number should be unique.
  - Browse of all clients and contracts;
  - Lock / Unlock of clients contract;
  - Search of client by phone number;
  - Changing tariff, adding and removing of options to contract;
  - Adding new tariffs, removing of old;
  - Adding / removing option for tariff;
  - Option management: some options may not be compatible with each other or could be added to certain options, employee adds and removes these rules.

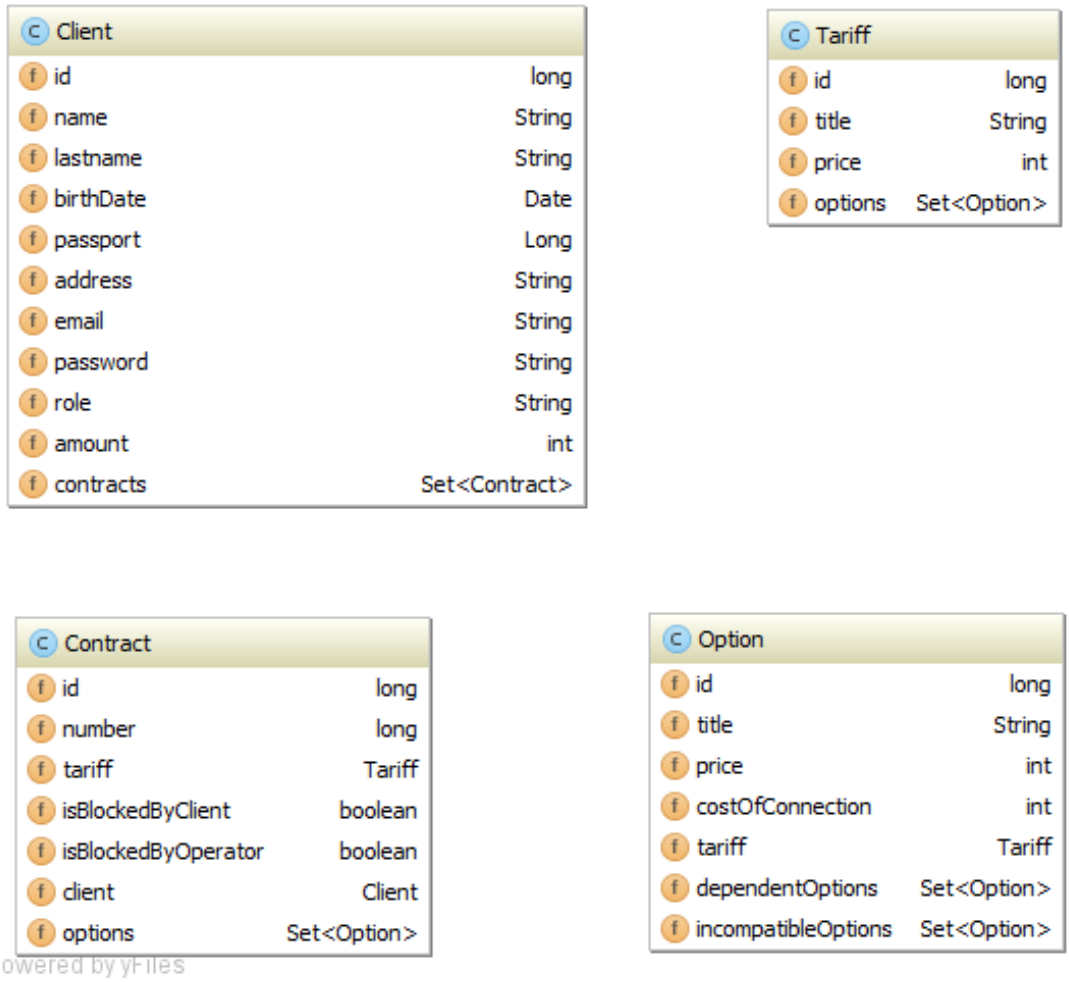
When performing operations with contracts before saving the changes on each page will be displayed basket, which displays the selected client's positions.

2. Model

Result web application model contains DAO (Data Access Object) and service layers and entities set for clients, contracts, tariffs and options.

2.1 Entities

For model created four entities. They represented as the same name java classes and shown in the following UML-diagram.



Each entity marked by @Entity annotation for working with JPA and Spring Framework, mapped on the database by the Java Persistence API (JPA), have standard auto generated ID field and relationship with other entities (One-To-Many, Many-To-Many, etc…).

Client entity have client info fields and set of contracts, which linked with Contract entity by @OneToMany relationship.

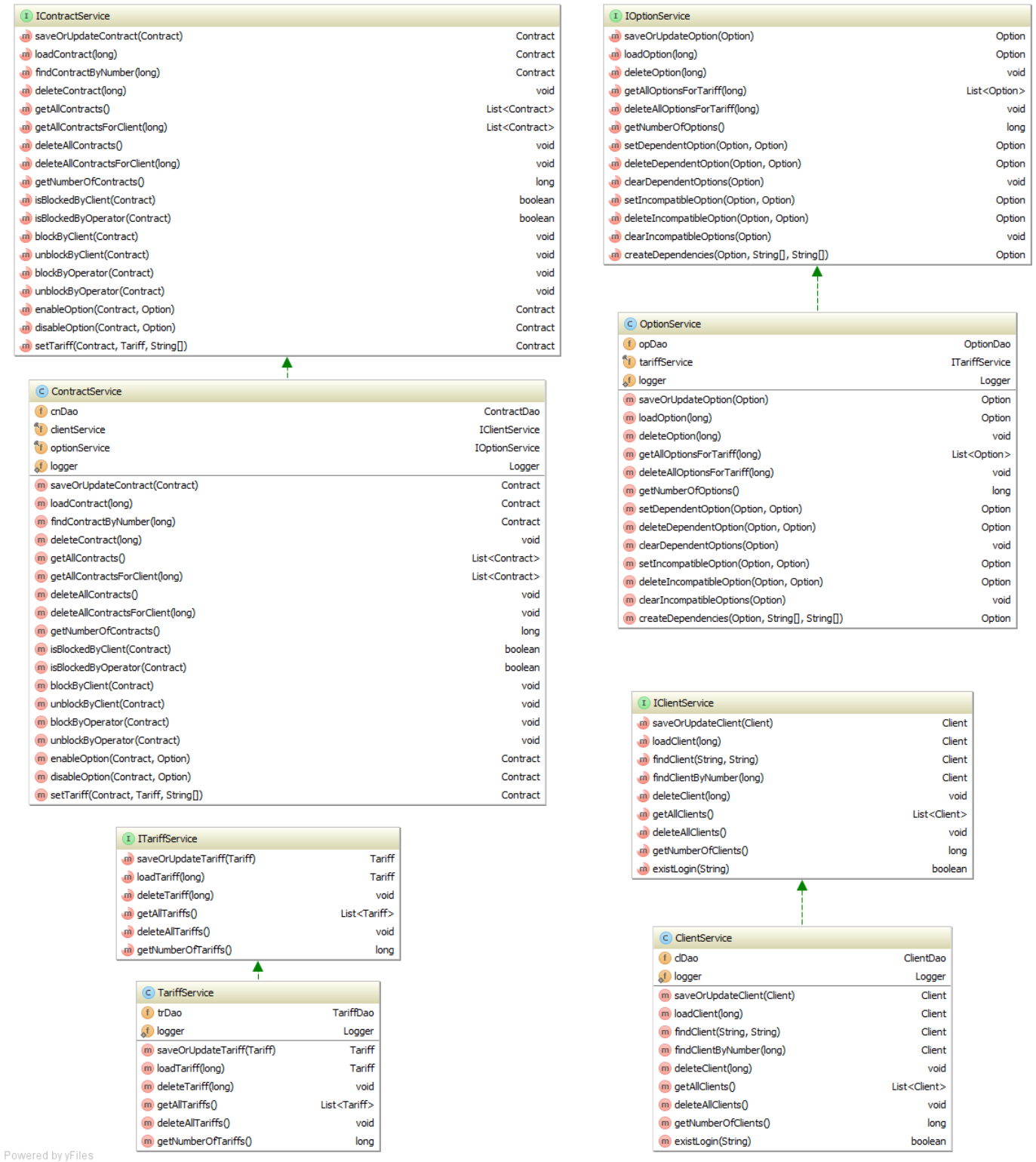
Contract entity have contract info and property fields, client field linked with Client entity by @ManyToOne annotation, tariff field linked with Tariff entity @ManyToOne annotation and set of connected options, which linked with Option entity by @ManyToMany relationship.

Tariff entity have tariff info fields and set of available options, which linked with Option entity by @OneToMany relationship.

Option entity have option info fields, tariff field linked with Tariff entity by @ManyToOne annotation and sets of dependent and incompatible options, which linked with another Option entities by @ManyToMany relationships.

2.2 Services

For each entity created separate service interface and class-implementation for working with the database. All service implementation classes annotated as @Service for definition it as a service by Spring Framework. Created services and their methods shown in the following UML-diagram.



Constructors of these classes have @Autowired annotations for automatically connection these services by Spring Framework in any place of the application. Methods of service classes have @Transactional annotations. In each method carried out logging of performed actions and results of operations. For directly working with the database, each service implements the necessary DAO class and uses its methods. Some services implementing other services for working of their specific methods.

2.3 Data Access Object (DAO)

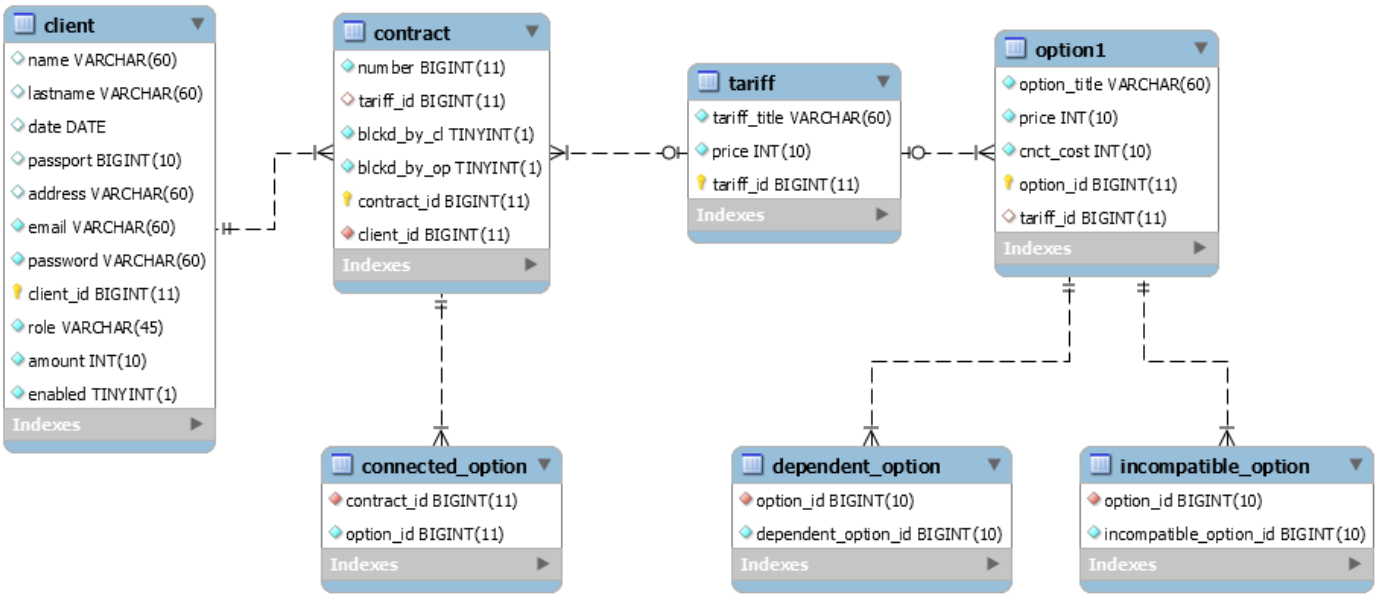
DAO provides an abstract interface to the database (MySQL in that application). Each DAO class extends a generic AbstractDAO class and provides methods of work with the database for each separate entity. It is ClientDAO for Client entity, ContractDAO for Contract entity, TariffDAO for Tariff entity and OptionDA for Option entity. All of them extends AbstractDAO class.

Most DAO methods use overridden AbstractDAO methods, but some of the methods used are specific for each entity requests like “findClientByLoginAndPassword”, described in the @NamedQueries fields of these entities.

Each DAO class marked by @Repository annotation and implements EntityManager object by @PersistenceContext annotation for working with Spring Framework.

3. Database

By the task in project should be used MySQL database. For each entity and their inner sets of relationships created seven tables in “ecare” schema: client, contract, connected_option, tariff, option, dependent_option, and incompatible_option. All relationships between tables shown in the UML-diagram.



Each table of entity linked with that entity by the special annotation @Table in Entity class. Additional tables are associated with sets in the entities using annotations @JoinTable.

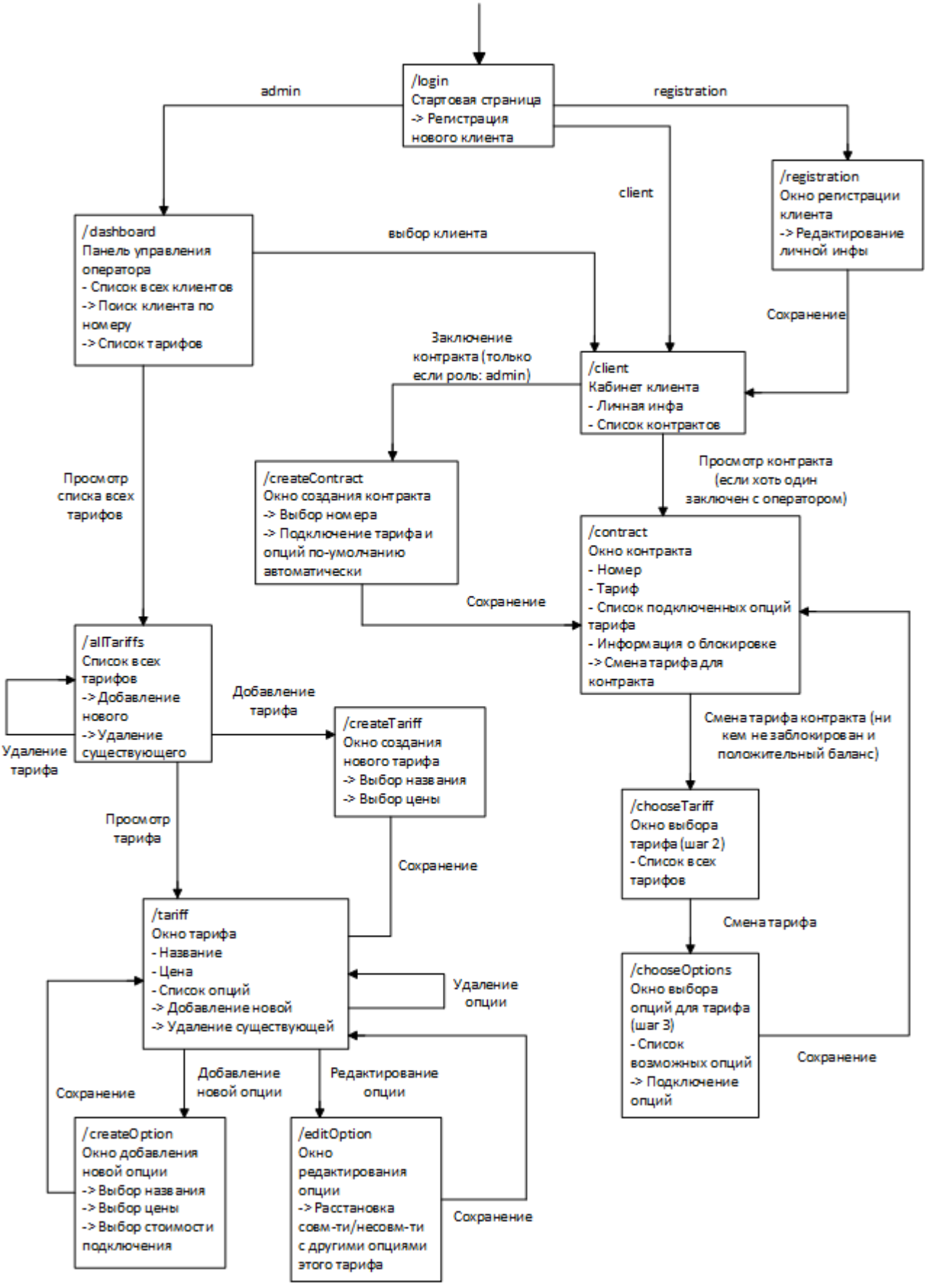
Application communicates with the database using a special “dataSource” bean in “spring-datasource.xml” file.

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/ecare" />
    <property name="username" value="root" />
    <property name="password" value="123456" />
</bean>
```

Filling, updating and deleting of related entities occurs according to the cascade principle.

4. User interface

User interface of application presented as JSP pages, which communicate with each other using Sparing controller classes with @Controller annotations. The next schema can show all relationships between JSP pages.



5. Security

Safe operation carried out in the application using Spring Security. A user can have one of three roles: “ROLE_ANONYMOUS”, “ROLE_USER” or “ROLE_ADMIN”. On the login page and register used “ROLE_ANONYMOUS”, on operator pages - “ROLE_ADMIN”, on client pages - “ROLE_USER”. These roles are stored in a database table “client” and loaded by using the following settings of application manager in spring-security.xml:

```
    <access-denied-handler error-page="/showLogin" />

    <form-login
            login-page="/login"
            default-target-url="/loginUser"
            always-use-default-target="true"
            authentication-failure-url="/showLogin?error"
            username-parameter="username"
            password-parameter="password"/>
    <logout logout-success-url="/logoutUser" />
    <csrf/>
    <anonymous username="guest" granted-authority="ROLE_ANONYMOUS" />
</http>

<authentication-manager alias="authenticationManager" erase-credentials="false">
    <authentication-provider>
        <jdbc-user-service data-source-ref="dataSource"
                           users-by-username-query=
                                   "select email, password, enabled from client where email=?"
                           authorities-by-username-query=
                                   "select email, role from client where email =?" />
        <password-encoder ref="encoder" />
    </authentication-provider>
</authentication-manager>

<beans:bean id="encoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>
```

On the login page checked entered by the user login and password in the form and giving him the appropriate role from the database. If the entered a username or password is incorrect, it will redirect to the login page with an error message. When you log out, also occurs redirect to the login page and assign user role “ROLE_ANONYMOUS”.

All passwords are stored in the database in encrypted form using the SHA algorithm.

email	password	client_id	role
admin	$2a$10$991vQrXGj0Gcx7Wx6aXJzO41AsEGB/m42oRI	1391	ROLE_ADMIN
young@mail.com	$2a$10$zLj8rLOrEsFmN5zJ5ghZhuUbQXiC5TIOqJBm	3312	ROLE_USER
ritchieb@mail.com	$2a$10$QJ9r.HWCVVm2jwUsy4emzeSUNjR.AMWZSuZ5	3313	ROLE_USER
hudson@mail.com	$2a$10$Q132xlw8aAcCJTjTpGHyXepDkot6IZPQ3hrW	3314	ROLE_USER
bucket@mail.com	$2a$10$QJ9r.HWCVVm2jwUsy4emzeSUNjR.AMWZSuZ5	3315	ROLE_USER

Spring Security using the “encoder” bean decrypts password when the user logs into the system.