|  |  |
| --- | --- |
| **Гордеев Алексей Aleksei.a.Gordeev@gmail.com** |  |

**Overview**

The application has a common three-layered desing: presentation, services and persistence.

The DAO-layer is used for basic (create, read, update and delete) operations on the database. The DAO layer is based on Java Persistence API the the Application Server provides the implementation of it by the Hibernate.

As a database management system the application uses MySQL.

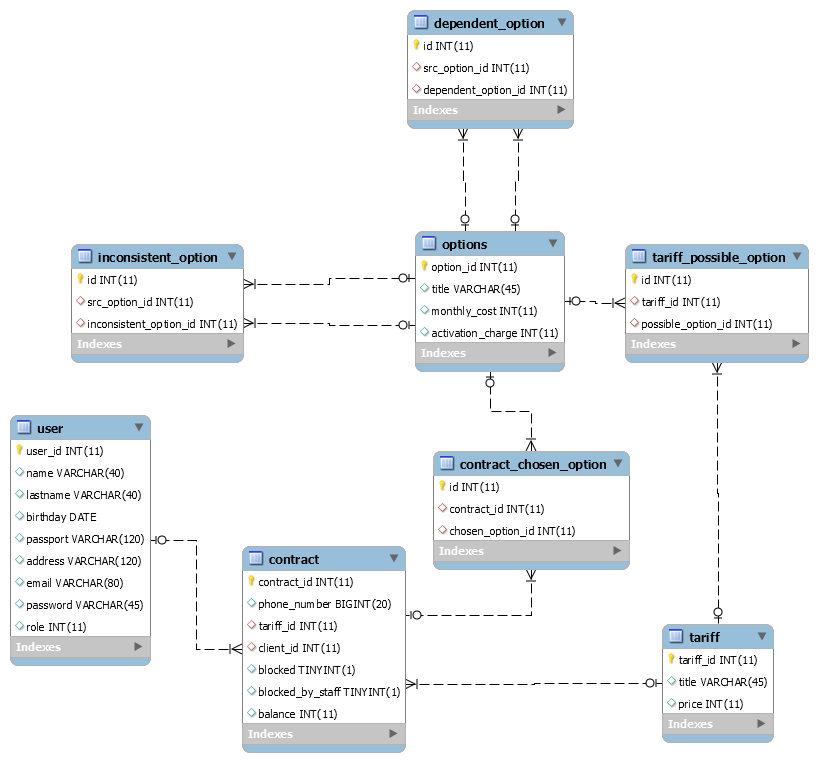
The application runs with a help of the Spring Framework. To be exact there are Spring Core and Spring MVC here. The Spring Core helps to solve the task of injecting of dependencies. While the Spring MVC is responsible for presentation layer.

The application runs under WildFly 9.0 Application Server.

**Datasource**

The Application Server is configured in such a way that it creates and managed the Entity manager. And the Entity managers is obtained from JNDI.

**Database design**

****

First let’s take a look at **tariff** and **options** tables. By condition of the task each tariff may have a specific set of options. At the same time an option may have been linked to the different tariffs. The **tariff\_possible\_option** table serves for the storing this data. The similar things are with **contract** and **options.** Each of the contract may have a definite assortment of option and vice versa. The **contract\_chosen\_option** helps with storing this linking information.

This diagram shows that a user may have several contracts. While a contract may has the only tariff.

It is considered that several options may have been linked to a contract and several contracts may have the same option. So there is an Many-to-Many relationship here.

**Input data validation**

For the purpose of validation the user input data I used Java Validation API in Spring MVC. It [requires](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/validation.html) a class for binding it to form’s inputs. But my existing classes (like UserDTO) were not what I need because Java Validation API does not provide @Pattern annotation for Integer fields. Consequently I created a bunch of classes for form validation. All those classes located in «*controllers\_mvc.validationFormClasses*» package.

**Options**