**Technical solution description**

**Store project**

**System description:**

This project is a prototype of an online phone store.

Used technologies and frameworks:

* Maven
* JPA
* JQuery
* JDBC (MySQL)
* JUnit
* Log4j
* Spring MVC
* Glassfish
* Spring-Security

**Features:**

1. JS sort phone list.
2. JQuery update cart value after add new phone, change quantity phone in cart, delete phone from cart.

**The scheme of database with description of tables and their relations:**

1. Users table - a table with users personal data.

Many-To-One relation with role and orders table.

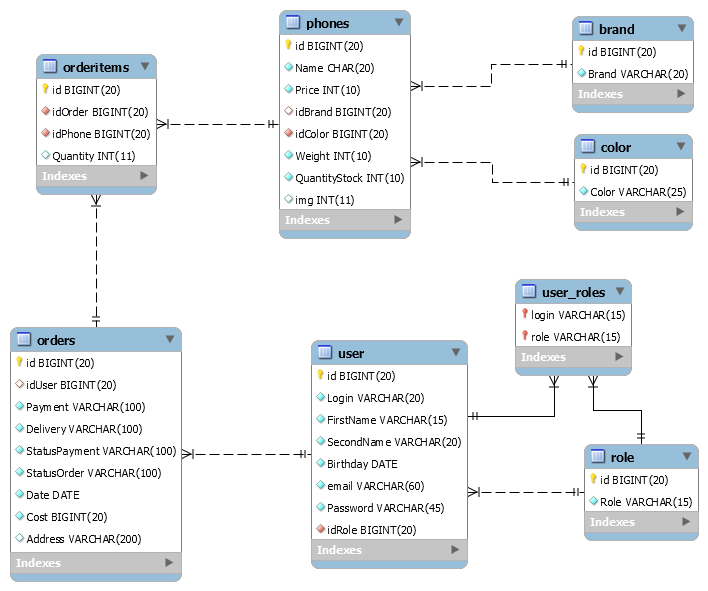
2. Phones table - a table with characteristics phones.

Many-To-One relations with brand and color tables.

One-To-Many relations with orderitems table.

2. Orders - a table with order details.

One-To-Many relations with orderitems table.



**Explication and implementation of the model from the task:**

I decided that the main entities in the application are: goods(phones), orders and users. We also need the tables for the basis fields of these entities: brand, color, order lines, role.

**UI:**

All view pages are in the folder ‘WEB-INF/views’. Special pages for the admin or user in the ‘WEB-INF/views/admin’ and ‘WEB-INF/views/user’ respectively. All pages contain ‘header.jspf’ and all except ‘index.jsp’ contains its specific left-menu. For the interface I use one of the template bootstrap. All resources allocate in the resource folder.]

**Business logic:**

The service layer contains all the business logic of the application. Methods of this layer is called only from the controllers (or from service layer). The service layer methods call dao layer methods for their business logic operations.

1. BrandService - provides methods to interaction with Brand entity. There are simple CRUD operations at all.

2. ColorService - provides methods to interaction with Color entity. There are simple CRUD operations at all.

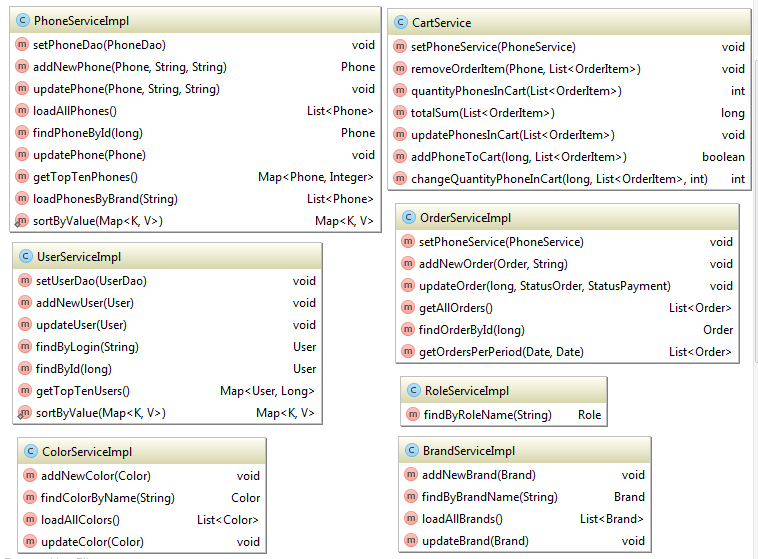
3. OrderService -provides methods to interaction with Order entity. There are simple CRUD operations and method for getting all orders per period.

4. PhoneService -provides methods to interaction with Phone entity. There are simple CRUD operations and method for getting most popular phones.

5. RoleService -provides methods to interaction with Role entity. There is only a method read.

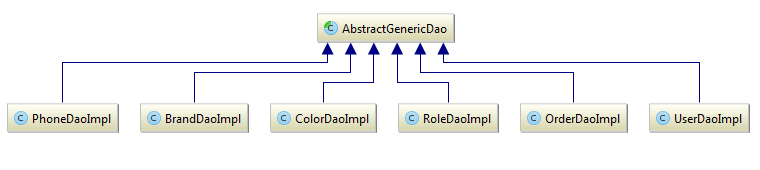
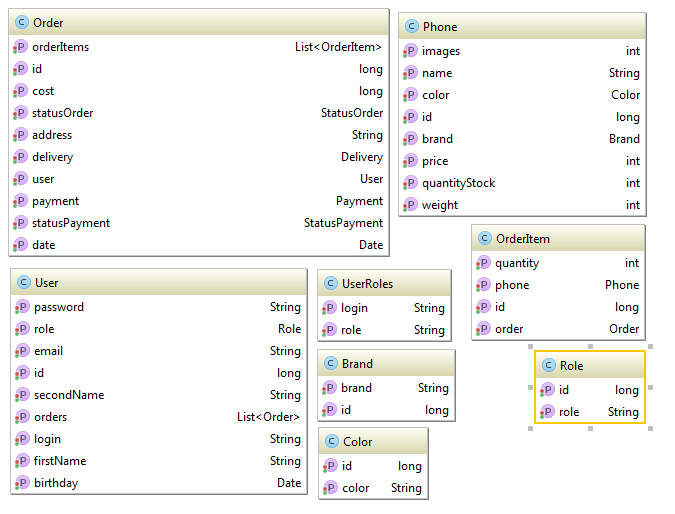
6. UserService- provides methods to interaction with User entity. There are simple CRUD operations and method for getting users most shop.

7. CartService - provides methods to interaction with Cart.

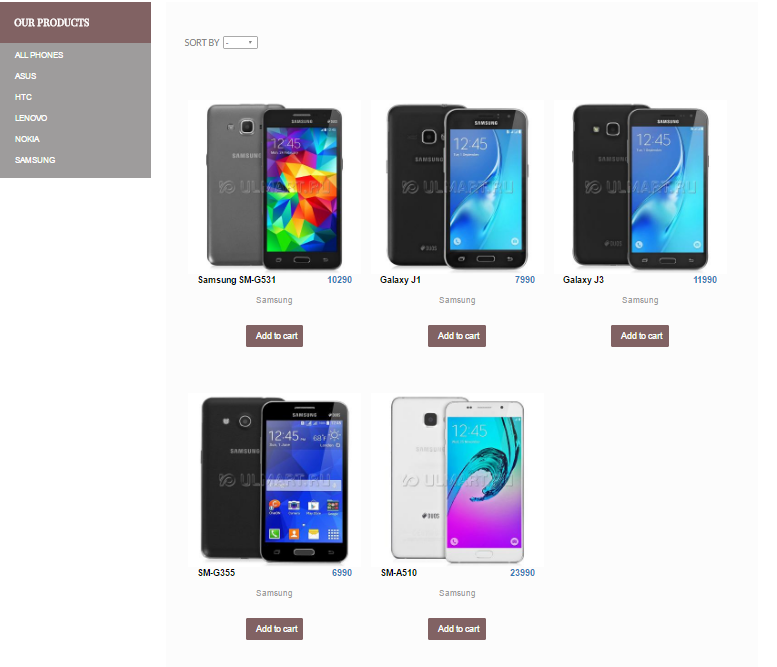
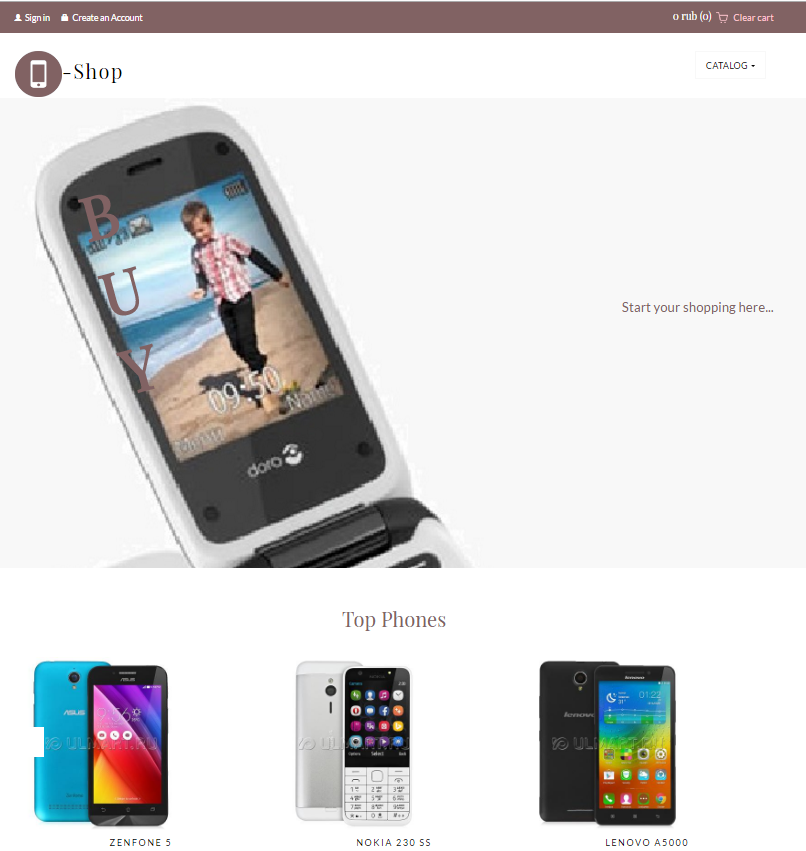


**Entities and DAO:**

There are located the entities which mapped in the database. User entity also contains the list of all your orders. Order entity also contains the list of all your order items. Transaction handling occurs in the service layer.



**Screenshots of the application:**

****

**Junit tests:**

* testTopTenPhones –testing sorted map<Phone, Integer>
* testTopTenUsers – testing sorted map<User,Long>
* testCreateOrder – testing expected NotEnoughExeception
* testRemoveOrder – testing remove order item from cart
* testQuantityPhonesInCart – testing quantity phones in cart
* testTotalSummCart – testing total sum cart