This project is an internet store.Functionality consists of 2 parts: User functionality, and admin functionality.

1. **User functionality**

Functionality of the user includes the following features.

**1.Registration:**

Registration takes place on the “registration.jsp”. Validation is realized by springframework.

**2.Log In:**

User can login on all pages. Controller for login invokes from the ajax function "login" in "loginActivity.js" which sends data to controller "login" by json. This function builds buttons for entering the user profile, logging out from account, and if user is an administrator, then function builds button for entering the admin panel. Also the function hides registration and basket buttons in case the user is an administrator. If login or password is incorrect, the function sends an error message.

3**.Finding products**

The user can find products by a part of name on all pages. After that the user can filter result by price and by cities.

**4.Filter products by price and cities**

User can filter products by price and by cities. On the home page the user can also choose a category of products. The price filter is realized by jquery plugin "slider-range".

**5.Viewing products**

After clicking on a product, the user will be directed to the product page, where he can read more information about the product and add the product to the basket.

**6.Adding products to the basket**

User can add products to the basket on the homepage, search the page and the product page. Adding product is realized by ajax function "addToBasket" in store.js. The basket is saved in session, and if the user logs out, products are deleted from the basket.

**7.Buying products**

The user can buy products from the basket. There he can also delete products from the basket. Deletion and buying are realized by ajax function in basket.js which sends data to controller "BasketController" by json. If the user is not logged in, the function sends him a warning message about the necessity to register or log in.

**8. Changing contact data and a password.**

The user can change a password, @mail, and phone number on the User page . Changing data is realized by ajax functions which sends data to controller "userController" by json.

**2.Admin functionality**

The User has 5 tabs that correspond to each class of the entity. And in every tab admin can operate with objects of these classes. In tabs "Cities", "Categories", "Products" functionality s realized by ajax functions which send data to REST controllers. Entity REST api is not realized in other objects because after logining the user with an empty password is logged into session, for higher securityю And a model "reserves" includes a model "user".

1. **Users tab**

Admin can see all data of users and ban these users. Banning of users is realized by the ajax function "change Activity" in users.js which sends user data to the controller by JSON.

1. **Reserves tab**

Admin can see all reserves and accept them. Accepting of reserves is realized by the ajax function "acceptReserve" in reserves.js which sends user data to the controller by JSON.

1. **Cities**

Admin can see all cities, change their names, add a new city .This is realized by the ajax functions in cities.js which send data to the REST controllers by JSON.

1. **Categories**

Admin can see all categories, change their names, add a new category .This is realized by the ajax functions in categories.js which send data to the REST controllers by JSON.

1. **Products**

Admin can see all products, change their name, add a new product .This is realized by the ajax functions in products.js which send data to the REST controllers by JSON.

**Libraries and frameworks**

1.Hibernate – for connection to a database, operating with model objects and their validation;

2.spring mvc- for web project configuration, mapping controllers, realizing validation;

3.jquery and Ajax- for sending data to controllers by JSON without reloading a page, for dynamic work with data and views;

4.Log4j-for logging project errors and warnings;

5.javax.persistence- for mapping entity;

6. commons-fileupload – for uploading image to a project folder;

7. org.codehaus.jackson – for reading Jackson

**Patterns**

1. singleton - in com.markoproject.util.HibernateUtil.class and in

com.markoproject.dao.DaoFactory.class for returning only one object of session initialized in static field.;

1. Factory method- in com.markoproject.dao.DaoFactory.class for returning model's dao classes;
2. Lazy initialization- in com.markoproject.dao.DaoFactory.class for initialization model's dao classes;
3. MVC- work of a web application based on this pattern;
4. Data Access Object –for work with entity;

**Instruction**

1. import database from Mystore/database/mystore.sql

2. edit hibernate.cnf.xml ,set your name and pussword for connection to mySQL DB

3. change path to hibernate.cnf.xml in file src.com.markoproject.util. Set your path to hibernate.cnf.xml in File f = new File("(your path)\\MyStore\\src\\main\\java\\hibernate.cfg.xml");( like C:\\Users\\Marko\\Documents\\NetBeansProjects\\MyStore\\src\\main\\java\\hibernate.cfg.xml)

4. set your path to project in src/main/resources/log4j.properties

5. set your path to project in сдфіі otherAdminFuncController in method “UploadImage”

6. change tomCat's port on 8083;