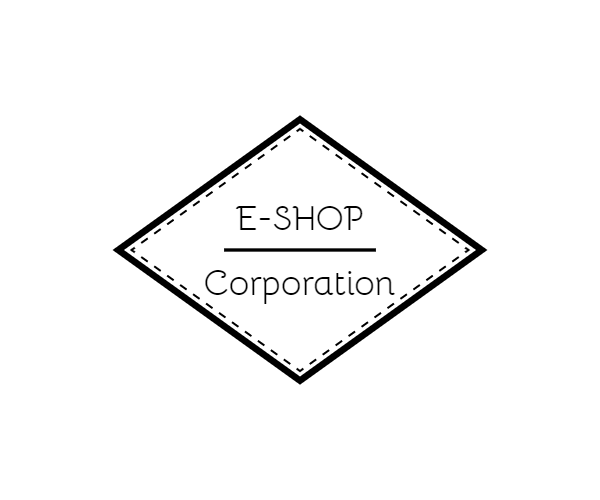
**E-shop project.**

Technical Solution Description

****

**Content**

Assigned Task………………………………………………………………………………………………………3

List of used technologies and frameworks……………………………………………………………..4

Implemented features…………………………………………………………………………………………..5

Database………………………………………………………………………………………………………………6

Explanation of the model………………………………………………………………………………………9

System architecture………………………………………………………………………………………………10

List of pages………………………………………………………………………………………………………….14

Main Page……………………………………………………………………………………………………………..14

Test list…………………………………………………………………………………………………………………15

**Assigned Task**

The task required writing an application that simulates the work of e-shop information system. As a result, the application is required to obtain multi-user, client-server network connection. All data stored on the server side. Each client may download some data after each data change operation must be synchronized with the server. The client must have a graphical interface. The application must handle the hardware and software errors.

**List of used technologies and frameworks**

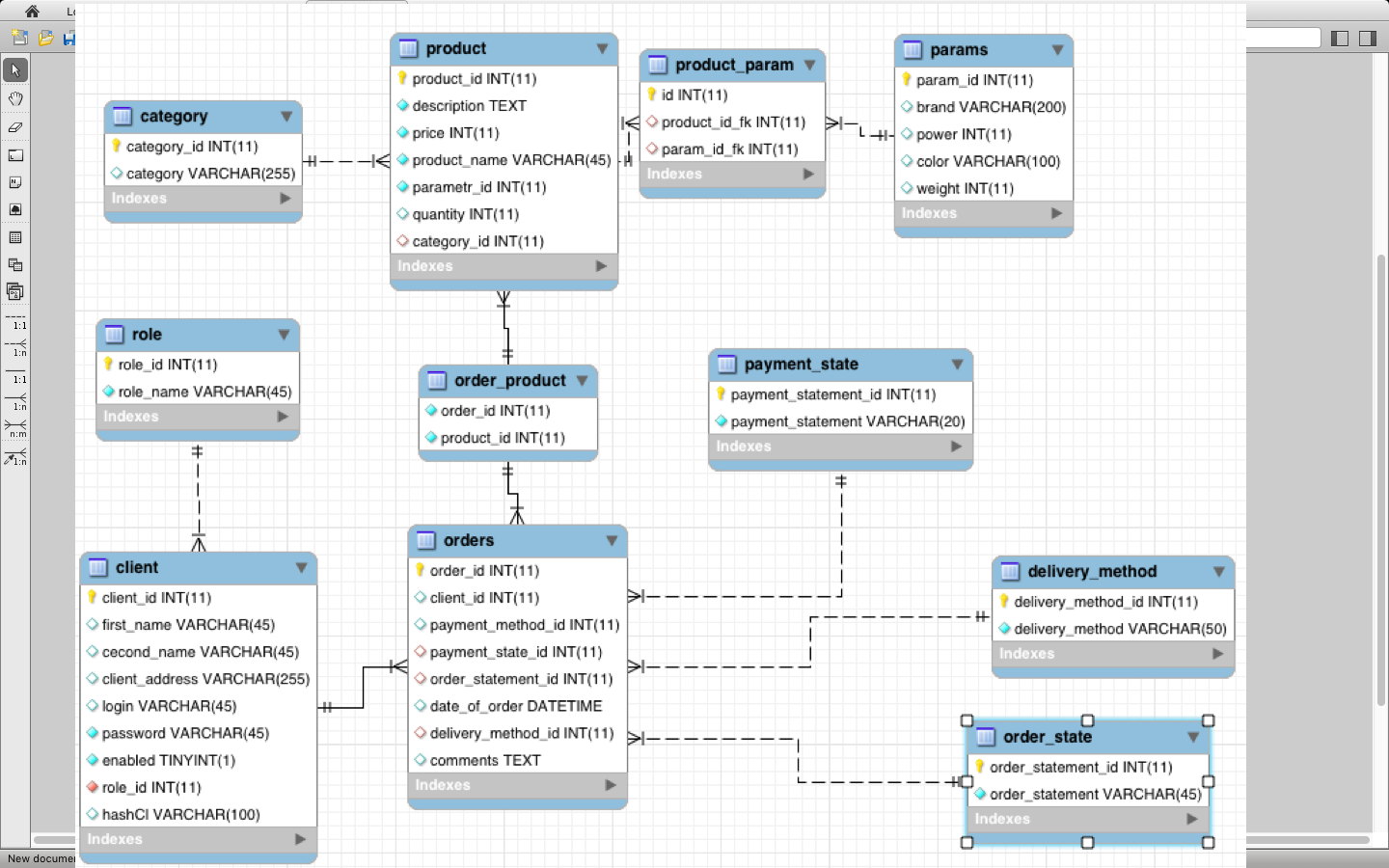
In this application was written on Java language and used such technologies and frameworks as:

* CSS 3.0,
* JSTL 1.2,
* JSF,
* AJAX 3.1.0,
* JavaScript
* Spring framework 4.3.2:
* Spring Security
* Spring MVC
* Spring Data JPA
* MySQL 5.5+
* JPA 2.0
* Hibernate 5.1.0
* GSON 2.7
* Log4j 1.2.17
* Junit 4.12
* Java Mail 1.4
* Mockito 2.2.5
* Jbcrypt
* Common-validator 1.4.1
* Selenide 4.0
* Jax-ws
* Web-Service

**Implemented features**

Client registration goes through mail registration. After filling the form on main Site, automatically generated message to client. Message contains link, as hash code to complete registration.

**Database**



1. Client table:

Main database table of client. This table contains fields:

* client’s first name,
* client\_id,
* client’s last name,
* client’s address
* client’s login,
* client’s password,
* enabled Boolean field, the confirmation of registration,
* client’s role

Relation of this table:

* role table, contains name of client’s role with ManyToOne relation
* orders table, contains full information about order with OneToMany relation

1. Role table:

Include information about client’s role, with one field:

* Role

Relation of this table:

* OneToMany relation with client table

1. Orders table:

Include information about client’s order. This table contains fields:

* order\_id,
* client\_id,
* Payment method id,
* Payment state id,
* Order statement id,
* Date of order,
* Delivery method id,
* Comments.

Relation of this table:

* Client id ManyToOne relation client table ,
* Payment method id ManyToOne relation payment\_method table
* Payment state id ManyToOne relation payment\_state table,
* Order statement id ManyToOne relation order\_state table
* Delivery method id ManyToOne relation delivery\_method table

1. Order\_state table:

Include information about order’s statement, with one field:

* order\_statement contain information about order state

Relation of this table:

* OneToMany relation with orders table

1. Delivery\_method table:

Include information about delivery method, with one field:

* delivery\_method contain information about delivery method

Relation of this table:

* OneToMany relation with orders table

1. Payment\_state table:

Include information about payment state, with one field:

* Payment\_state contain information about payment state

Relation of this table:

* OneToMany relation with orders table

1. Order\_product table:

This table is connection table between Orders table and Product table, with relation ManyToMany.

1. Product table:

This table includes information about product:

* Product id,
* Product description,
* Price of product,
* Product name,
* Parameter id,
* Quantity of product in stoke,
* Category id.

Relation of this table:

* category\_id ManyToOne relation with category table,
* param\_id ManyToOne relation with param table,

1. Product\_param table:

This table is connection table between product table and parameter table, with relation ManyToMany.

1. Param table:

This table includes information about parameters of product:

* Brand of product,
* Color of product,
* Wight of product,
* Power of product.

1. Category table:

This table contain information about name of product categories, with one field name of category.

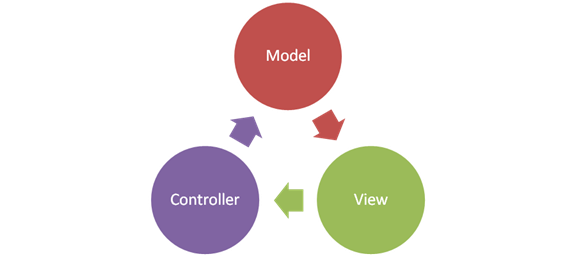
**Explanation of the model**

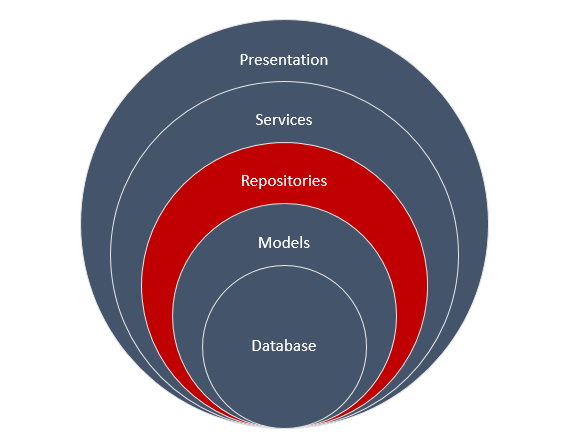
**Product cart** - For the client, on every page has to display product cart. Product cart hasn’t to disappear after client authentication, reload current page, movement to another page, reopen page in browser. Product cart contains list of products witch client is going to buy. On the page of product cart client may to choose quantity of product.

**Client –** client of E-Shop application, has access to such functions like: registration, logging, choose product, choose quantity of product, make an order.

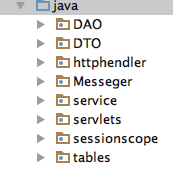
**Admin -** person who has administration privileges. Has permission to create new Category, create new product, check order information and change order status. Has permission for getting statistic information and access for the subsidiary application.

**System architecture**

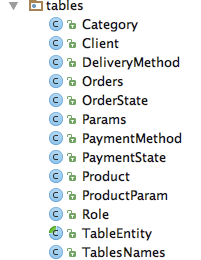




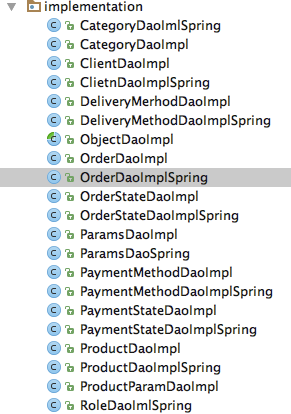
According MVC patter application has architecture:



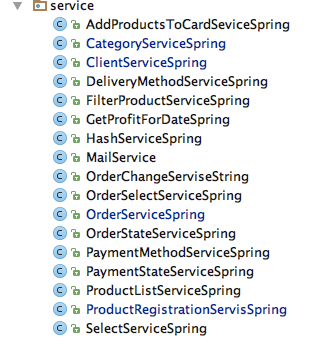
Model level:



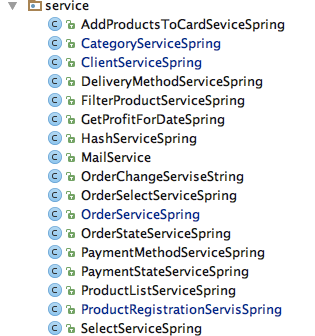
Model-Service level:



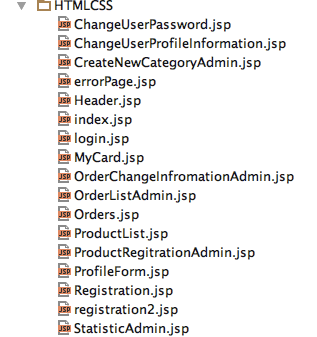
Service level:



Controller level:



View level:



**List of pages**

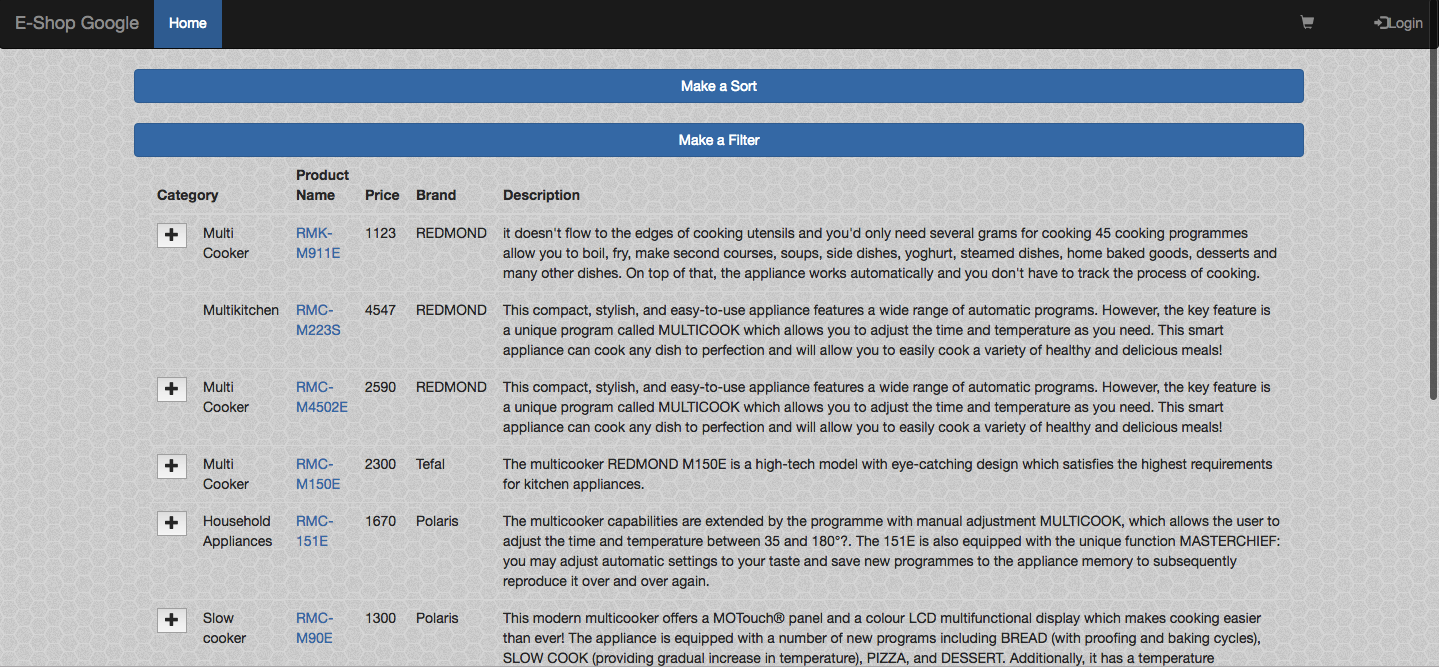
User Pages

* ChangeUserPassword – page for changing password.
* ChangeUserProfileInformation –page for changing user’s fist name, last name, address
* errorPage – page for illegal entrance.
* Index – login user page
* MyCart – page for the product on user’s cart.
* Orders page – page for buy products and make an order
* ProductList – main page with list of products
* ProfileForm page – user profile form page
* Registration page – page for new client registration

Admin Pages

* CreateNewCategoryAdmin page for new category creation
* OrderChangeInformationAdmin page for changing order information
* OrderListAdmin page for getting order list
* ProductRegistrationAdmin page for create new product
* StatisticAdmin page for getting statistic information

**Main Page**

****

**Test list**

Tests created for business logic and client’s interface.

1. Test AdminApptest

Test created for testing interface of administrator. Used technologies : Selenide

|  |  |
| --- | --- |
| Method | Description |
| adminFullTest() | Automatic test. Step by step testing admin pages |

1. OrderSelectServiceStringTest

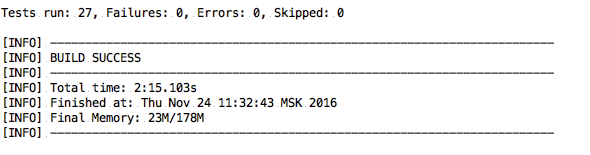
Test for checking sorted method. Used technologies: mockito

|  |  |
| --- | --- |
| Method | Description |
| testMockCreation() | Availability of mocks |
| getSortedOrdersListByDateOfOrder() | Test for sorting Orders list |
| getSortedOrdersListByOrderState () | Test for sorting Orders list |
| getSortedOrdersListByClient() | Test for sorting Orders list |
| getSortedOrdersListByDeliveryMethod() | Test for sorting Orders list |

1. AddProductTOCartTest

Test for checking of process adding product on cart and profit display

|  |  |
| --- | --- |
| Method Method | Description |
| GetListifProductsOnCartManyProducts() | Check quantity products on client’s product cart |
| GetListifProductsOnCartWithOneElement() | Check quantity products on client’s product cart |
| GetListifProductsOnCartWith6Element() | Check quantity products on client’s product cart |
| GetListifProductsOnCartEmptyList() | Check quantity products on client’s product cart |
| GetListifProductsOnCarteithEnatherMethod() | Check quantity products on client’s product cart |
| getQuantityOfProducOnCardTest2\_GetListOnproductOnCardMethod | Check quantity products on client’s product cart |
| getTotalPriceOfProductsOnCartwithGetListOnProductOnCartMethod | Get total profit |
| getTotalPriceOfProductsOnCart | Get total profit |

**Test results**