Technical Solution Description MMS (E-commerce site)

Last revised 23-11-2016

Status done

Author Mikhail Pankin

Table of Contents

[**Introduction**](#_gjdgxs)[**3**](#_gjdgxs)

[**Technologies and frameworks**](#_30j0zll)[**4**](#_30j0zll)

[**Database description**](#_1fob9te)[**5**](#_1fob9te)

[**Explication and implementation**](#_3znysh7)[**7**](#_3znysh7)

[**Modules**](#_2et92p0)[**8**](#_2et92p0)

[**User interface**](#_fwyi0izh5c4j)[**10**](#_fwyi0izh5c4j)

[**Service layer**](#_3dy6vkm)[**11**](#_3dy6vkm)

[**Entities and DAO layer**](#_1t3h5sf)[**12**](#_1t3h5sf)

[Entities](#_4d34og8) [12](#_4d34og8)

[DAO layer](#_2s8eyo1) [13](#_2s8eyo1)

[**Screenshots of the application**](#_17dp8vu)[**15**](#_17dp8vu)

# Introduction

E-commerce site is to be developed. Essential functionality includes catalog of products, search by catalog, personal accounts where client can manage order, get history of orders and update personal information.

User can manage basket before login or registration. All data must be stored in basket on the client side in this case.

Information management involves update personal information (name, sername, login, birthday) and password;

The buying process includes filling a basket, address for delivery, type of payment;

Users with privileges of admin must be possible to add new product to catalog, create and manage categories of products, manage orders (change the delivery and payment state).

Also this type of users can access for statistics (top ten product, client, monthly and weekly revenue);

Search includes filtering by all parameters of products;

As addition second application is to be developed.

Essential functionality includes retrieving mobile plan usage statistics from the web service provided by the main application. Report in .pdf format must be generated using retrieved statistics.

# Technologies and frameworks

● Java 1.8;

* MySQL

● Hibernate;

● Spring MVC;

● Spring Security;

* Spring validation

● JSP, CSS3, JQuery;

● JUnit;

● Glassfish 4;

● Mockito;

● Log4j;

Also used in additional part:

● EJB;

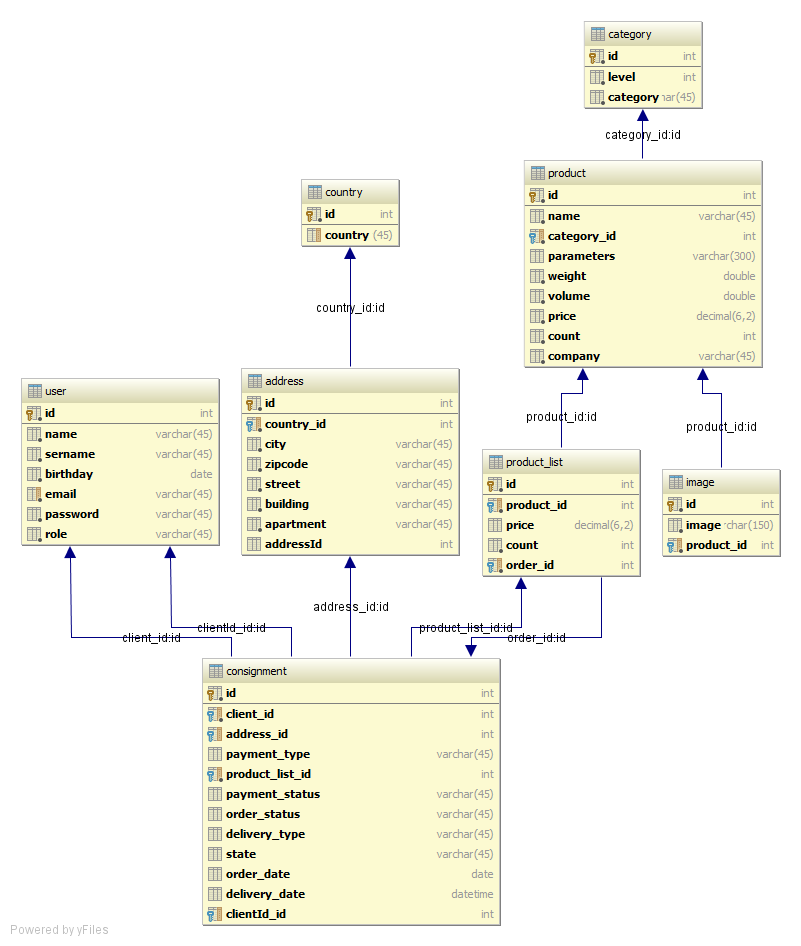
● JSF;

# Database description

All database tables are listed in the table below.

|  |  |
| --- | --- |
| **Table** | **Description** |
| address | Stores all addresses for each order |
| category | Stores all product’s categories |
| consignment | Stores all information about orders and basket (depends on state) |
| image | Stores all links to images of products |
| product | Stores all information of products |
| user | Stores all information about user (Clients and Admins) |
| **Many-to-many support tables** | |
| product\_list | Stores all information about products that added to basket/orders; |

Database diagram with fields and types is shown at Picture 3.1.



Picture 3.1 – Database diagram

# Explication and implementation

**User** is a user of application. He can be a client of shop or a manager who works for it. Client can have any orders and only one basket.

**Profile** is special page where client can manage his personal data, password and get order’s history;

**Basket** is special page that is accessible when client is in the middle of buying process. It’s possible for anonymous user; User can update count and delete product from it;

**Category** is page for managing category of products; Admin can create new, rename and delete exist category. Category that contains products cannot be deleted;

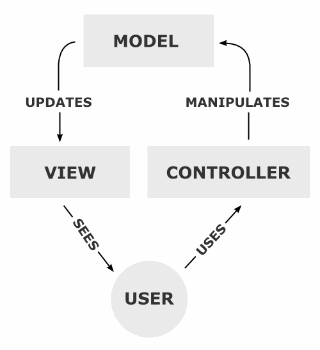
**Orders** is page for managing orders by admin; Admin can search it by namber, client login, type; It’s possible to modify status delivery and payment of each of it;

**Add** is page for add new product to the catalog. It’s possible fill all parameters and add images;

**Statistic** is page for display statistic;

# Modules

Application is designed using MVC template



Picture 5.1 – Interaction between client and server parts of application.

Client side consumes HTTP. Controllers call appropriate service method. Services interact with objects using data access objects. DAO use entities for storing database objects data.



Picture 5.2 – Application levels interaction

# User interface

Frontend is developed using JSP, JavaScript (JQuery library) and CSS (Twitter Bootstrap).

Below all pages are listed:

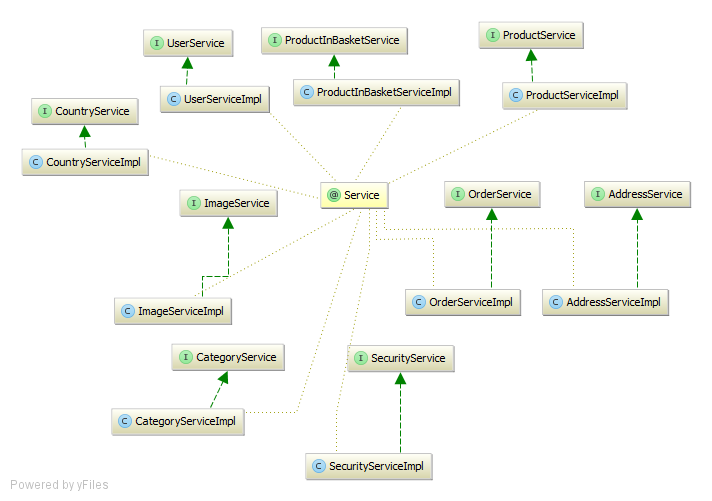
* index – main page with catalog and search;
* product – page that contains information of product;
* login – page for login and registration;
* basket – page for displaying and managing basket of client;
* profile – page for displaying and managing profile and orders history of client;
* category – page for managing category of products;
* orders – page for managing orders by admin;
* add – page for add new product to the catalog;
* statistic – page for display statistic;

# Service layer

Business logic is represented by services:

* AddressService - containing address processing logic;
* CategoryService - containing category processing logic;
* CountryService - containing country processing logic;
* ImageService - containing image processing logic;
* OrderService - containing orders processing logic;
* ProductInBasketService - containing basket processing logic;
* ProductService - containing product processing logic;
* SecurityService - containing authorization processing logic;
* UserService - containing user processing logic;

These services are described by interfaces and have Spring-specific implementation. All methods in service classes are annotated as transactional. All changes are being committed automatically at the end of successful method execution.



Picture 7.1 – Service layer structure

# Entities and DAO layer

## Entities

User – represents user of application. Each client with role CLIENT has many orders;

AddressEntity - represents address of each orders;

CategoryEntity – represents category of product;

CountryEntity – represents dictionary of country for address;

ImageEntity – represents link for images on server;

OrderEntity – represents order/basket of client;

ProductEntity – represents product of shop;

ProductListEntity – represent product in basket/order with saved price and count;

BasketOrderState – enum of order state. It can be as basket as order (after finished of payment process);

DeliveryType - enum of delivery type;

OrderStatus – enum of order status type (Draft, placed, delivery, recived);

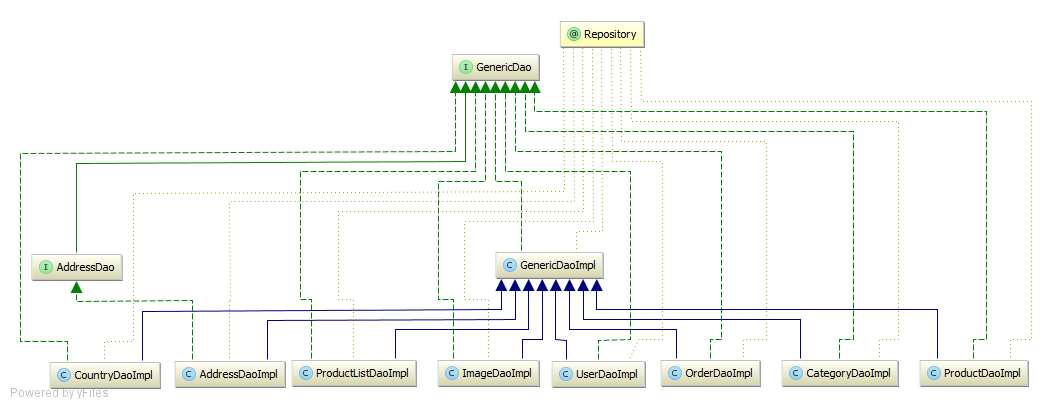
PaymentStatus - enum of payment status (paid, not paid);

PaymentType - enum of payment type;

Role – enum of roles;

## DAO layer

DAO layer is designed to be generified: interface GenericDao contains some methods for operation over entities that are common for all types of entities. This methods are implemented in GenericDao class that is Spring specific implementation.



Picture 8.1 – DAO layer structure

# Screenshots of the application