

Technical Solution Description

https://github.com/Aleksander1995/T-Systems

Aleksander Lobov,

[alex1995nazarovo@gmail.com](mailto:alex1995nazarovo@gmail.com)

St. Petersburg 2016

**Contents**

[1.Introduction](#page3) [3](#page3)

[2.Technologies](#page3) [3](#page3)

[3.Project Structure](#page4) [4](#page4)

[4.Database Scheme](#page4) [5](#page4)

[5.Database Interaction](#page6) [7](#page6)

[6.Services](#page7) [8](#page7)

[7.Views](#page7) [8](#page7)

[8.Unit Tests](#page14) [12](#page14)

[9.Client Application](#page14) 12

**1. Introduction**

InStore is an internet store. It allows you to shop from home. This web site is very simple, which designed for any age group.

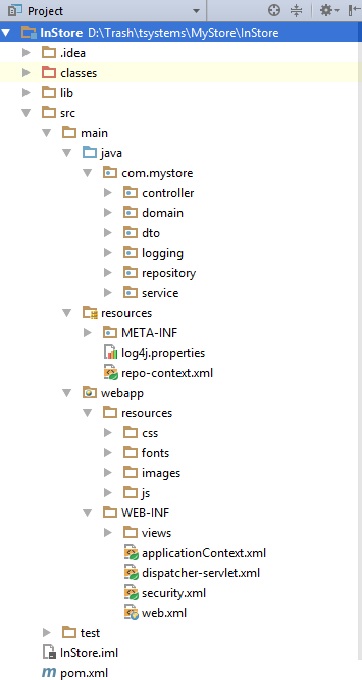
**2. Technologies**

Using the following technologies:

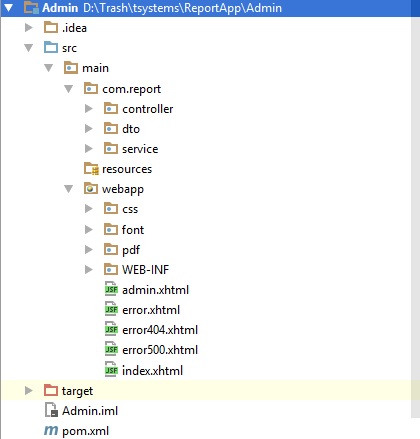
* MySQL 5.5
* Hibernate 5.1.2
* Spring 4.3.1 (core, MVC, Data JPA, Security, REST)
* mcat 8.5.5 Application Server
* GlassFish 4.1.1 Application Server(for client application)
* JSP 2.0, JSTL
* JQuery 2.1.1
* JSF 2.2.14, CDI (for client application)
* Log4j 1.2.12, slf4j 1.7.21
* Junit 4.12
* Mockito 1.9.5

**3. Project Structure**

Main(InStore) application:



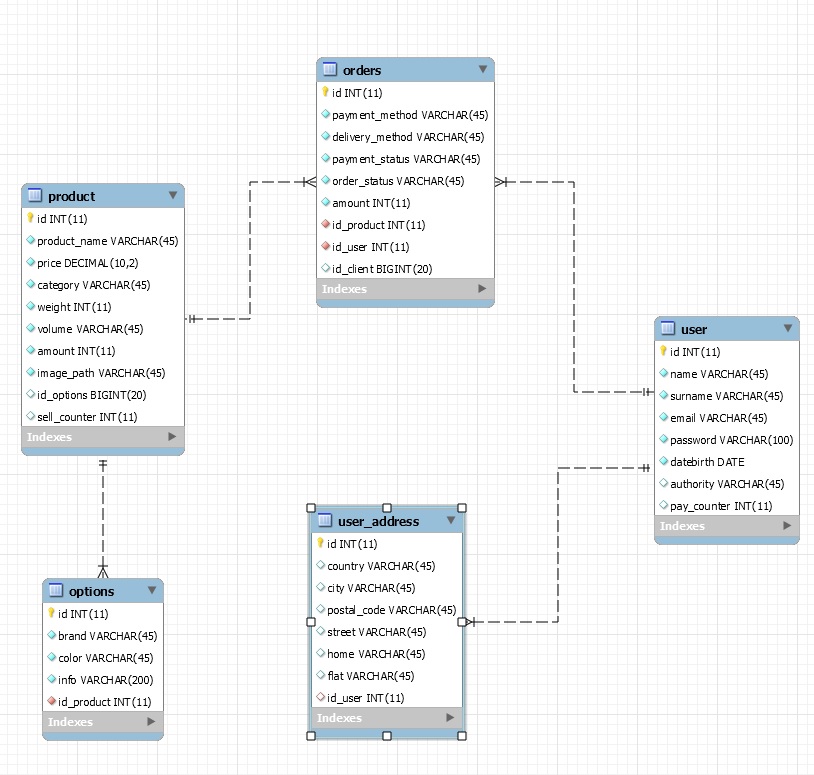
Client application:



* *controller*- contains classes, which relate view and services.
* *dto –* transfer object, which contains information about report.
* *repository –*these classes areused on service layer, and entities also act as data carriers at presentation layer.
* *service –* classes with certain business logic.
* *logging –* contains behavior about users.
* *domain* –entity classes. Packaged as WAR.

**4. Database Scheme**

The picture representing the database layout is given below. The database consists of the following tables:



**5. Database Interaction**

The application interacts with the database by means of *domain* classes. Includes the following entity classes containing the specified information:

* *Options* –brand, color, info of products
* *Order* –payment method, delivery method, payment status, order status, amount of products.
* *Product* –product name, category, weight, volume, amount, image path.
* *User* – name, surname, email password, date birth, authority of users.
* *UserAddress* –country, city, postal code, street, home, flat.

Entities are managed using Spring Data JPA repositories. I used custom methods they are implemented by defining JPQL queries in @Query annotation above the method declaration. Transactions are handled on service layer.

**6. Services**

The following interfaces are implemented:

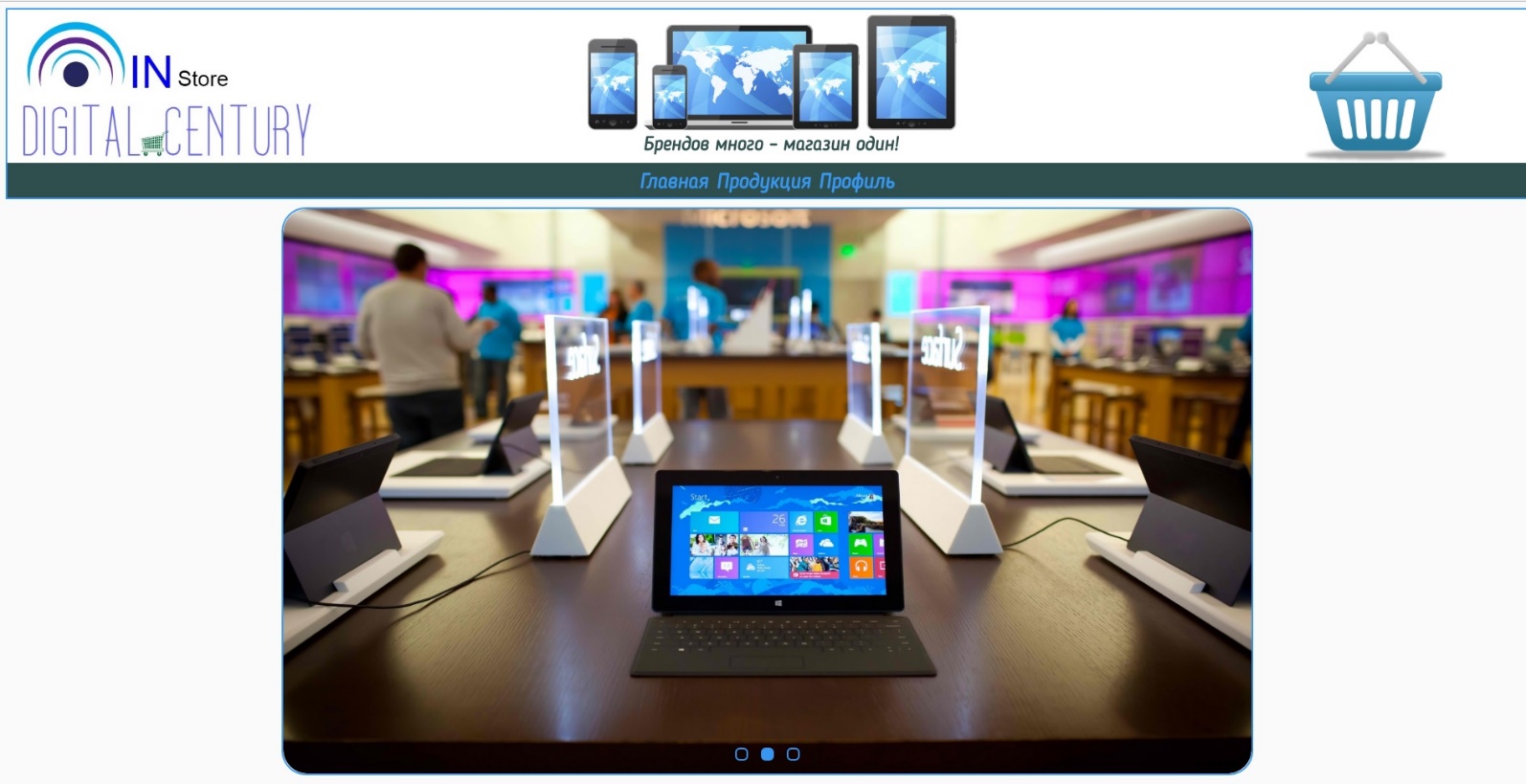
* *AdminService* –changing order (order status and payment status), addition product.
* *EveryOneService* –receipt grouped by category products, products by category, products by session, by id , removal from basket.
* *UserService* – receipt user by email, changing, addition user, making an order and checking whether exist an email.
* *RestAdminService* –receipt result of authentication and report.

**7. Views.**

*View* contains UI elements (css, fonts, images, js).

UI is built using FlexBox and jQuery. Screenshots of application pages.

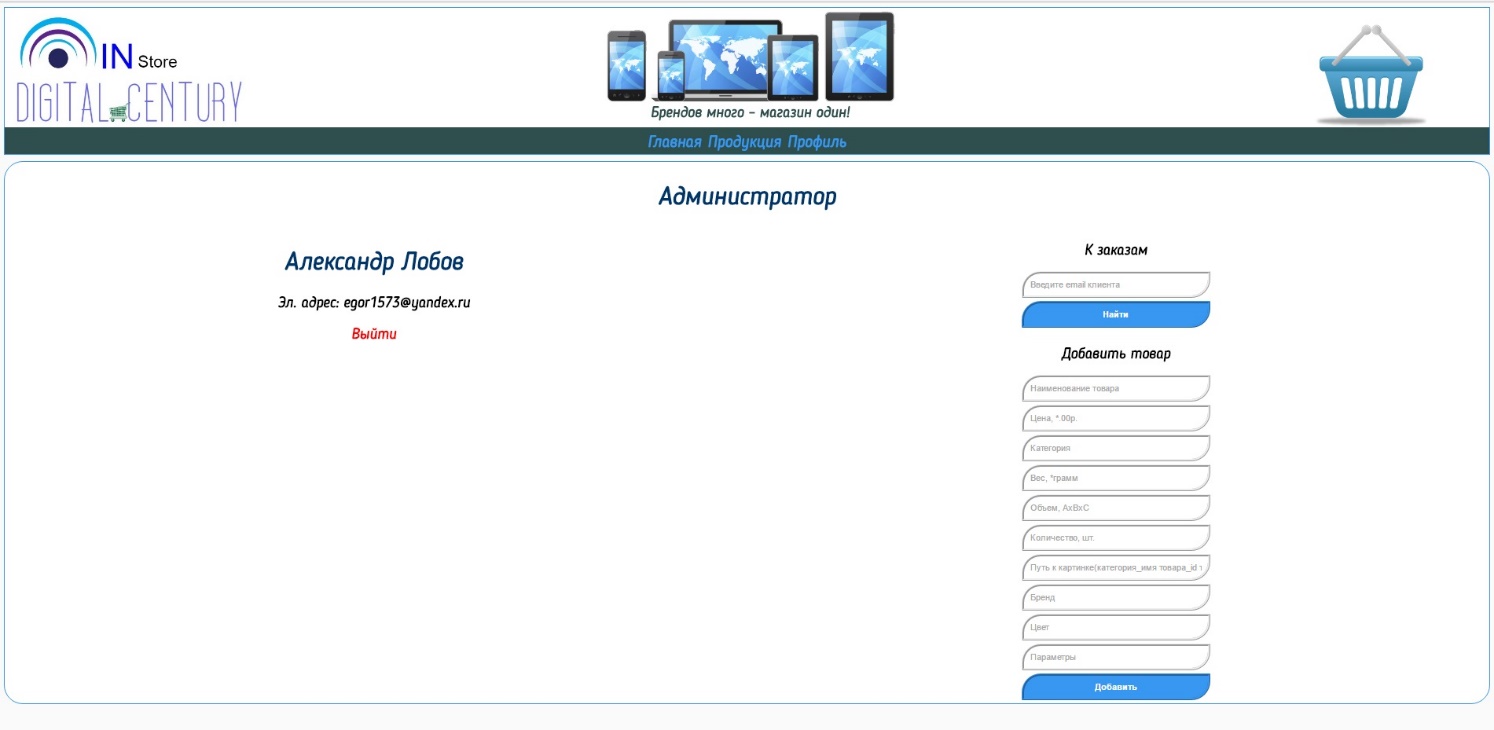
Start page:



Authentication page:



Admin page:



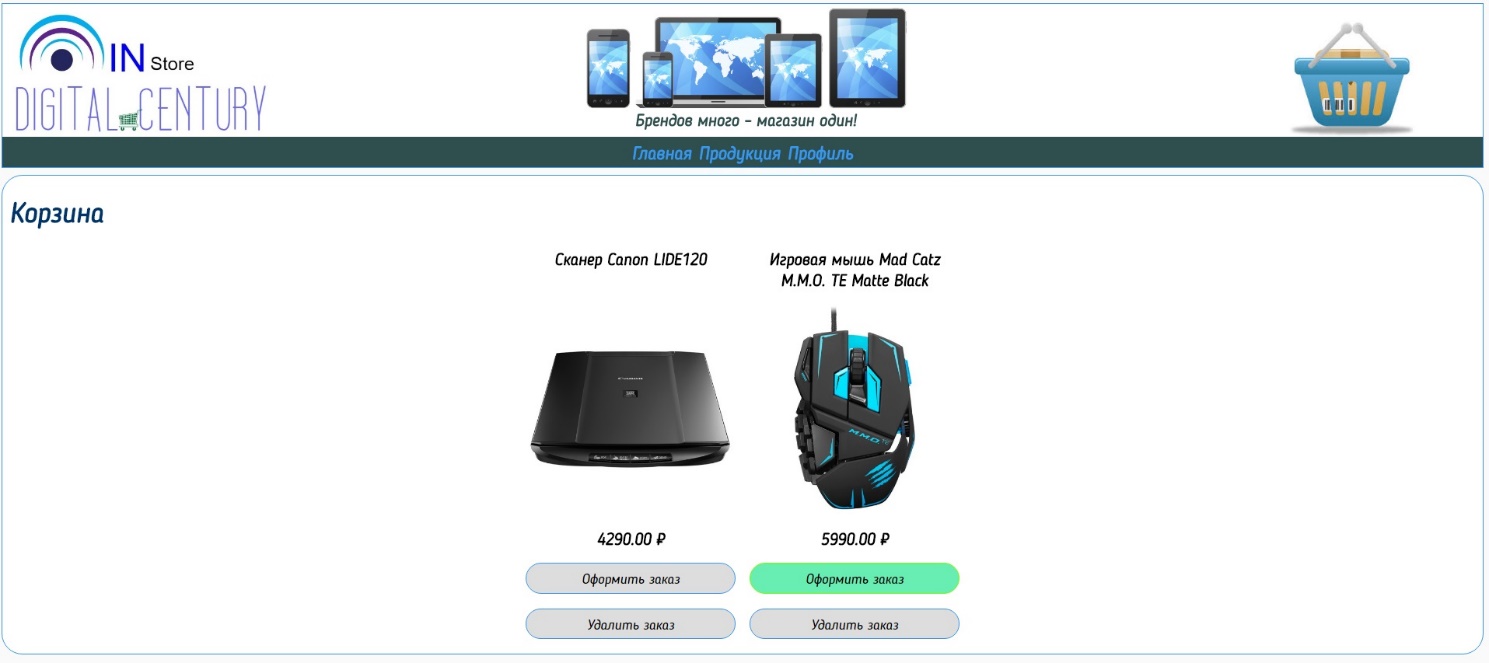
User page:



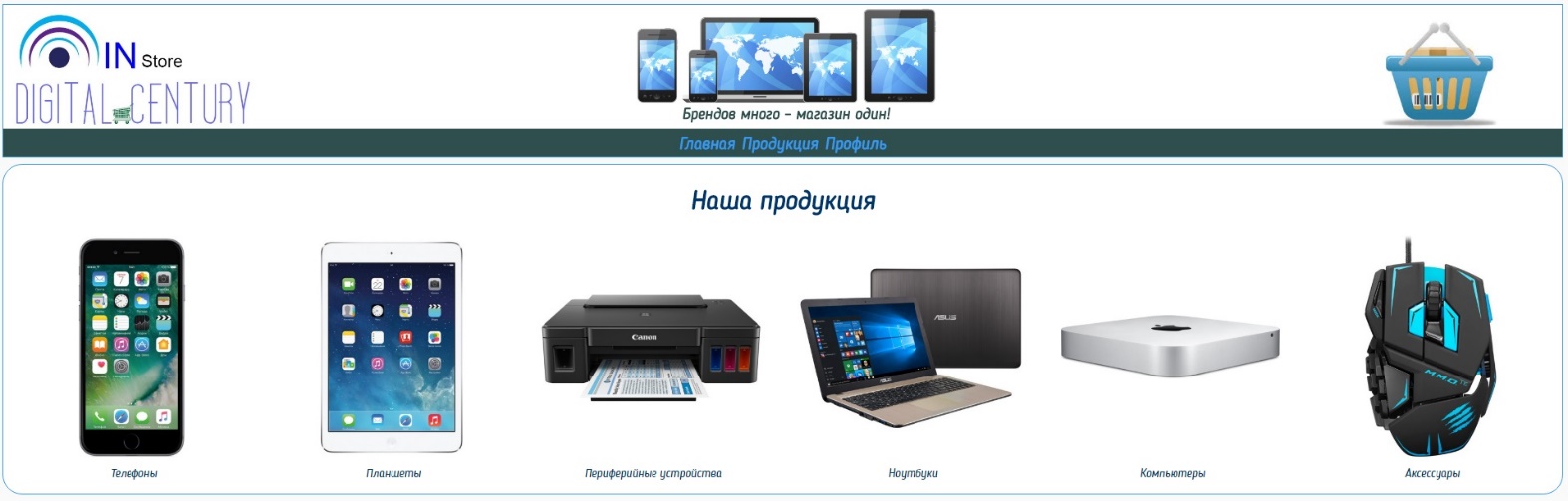
Order page:



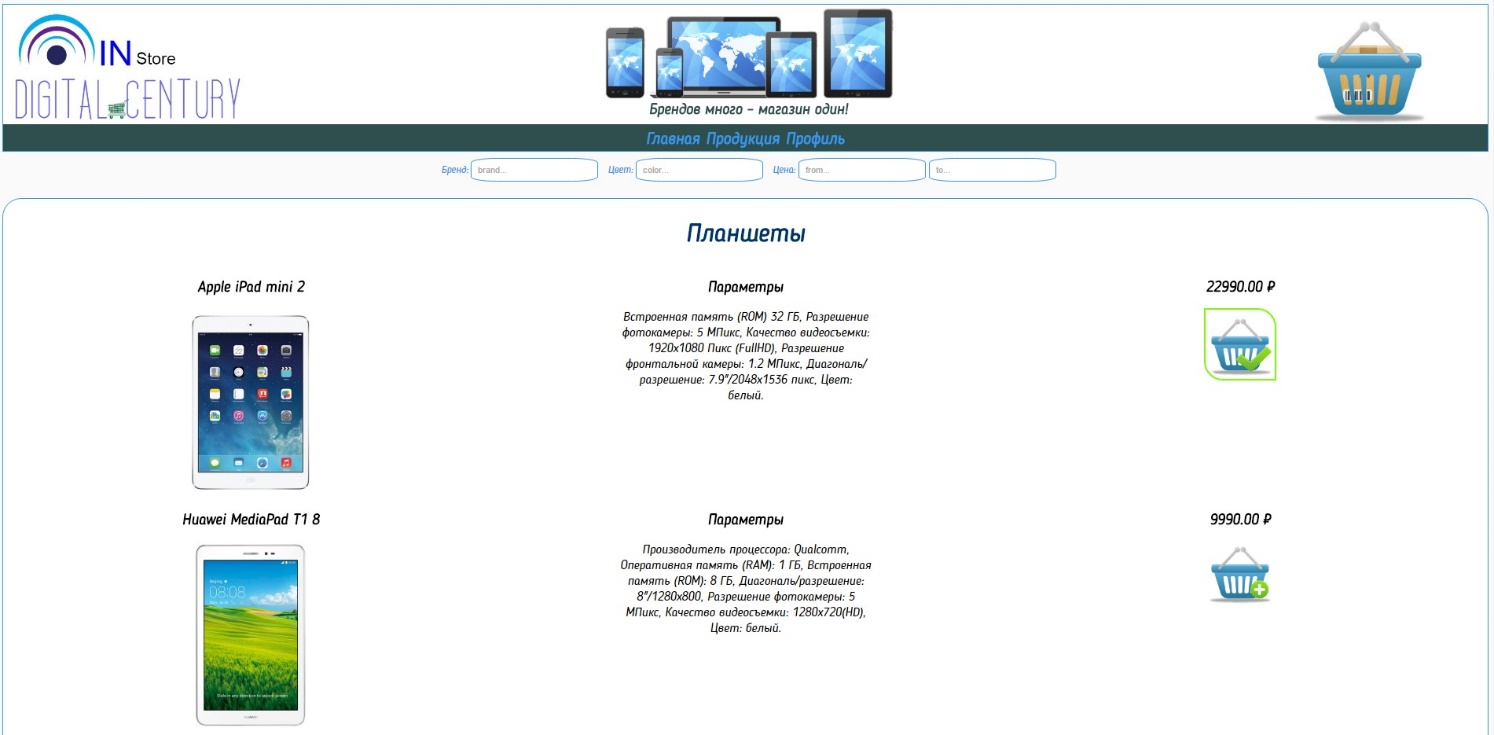
Basket page:



Our products page:



Products by category page:



**8. Unit Tests**

*Services* contains unit tests for business logic. Tests use JUnit 4.12and Mockito 1.9.5.

* *EveryOneServiceImplTest* –tests removal products from basket and receipt products.
* UserServiceImplTest – tests checking exist user.

**9. Client Application**

This application is communicating with web-service located in main app by Jersey client API. Only admin can be able to access to observe report. Application is built through JSF and CDI.

