Technical description

Board games online store project

T-Store

Author:
Alisa Meteleva

Saint Petersburg
2019

Content

1. Task description

To develop web-application that simulates the work of the online store information system. The application have to perform the required user‘s cases.

For clients:

- To view catalog with the possibility of filtering by parameters
- To view and edit profile
- To view and edit user addresses
- To view and edit user password
- Checkout
- To view order history

For managers:

- To view clients orders
- To edit order status
- To view sales statistic (top 10 products, clients, proceeds per week, month);
- To add new products;
- To add and manage catalog categories.
- To view order history

Additionally, to develop co-application for advertisement generating from main application.

2. Project goals

- The robust, useful and reliable system.
- Cohesive data model.
- User-friendly interface.
- Separate access to different system‘s part.

3. Application description

Web-application has two type of user: clients and managers.

Clients can view catalog, products, buy them. Clients also can change personal information, add, update and remove addresses, view order history.

Managers can add, update, delete products, their categories. They can also add, update, remove shipping types and change admin status of another users. Managers can look for list of orders and edit their status. They can observer statistics from orders for specific period. Managers have clients options too.

There is an authentication mechanism in system that control access to portal. Each user in application has access level that display what information he could get and what couldn‘t.

Data of users and their options store in reliable database.

4. Used technologies

4.1. Instruments

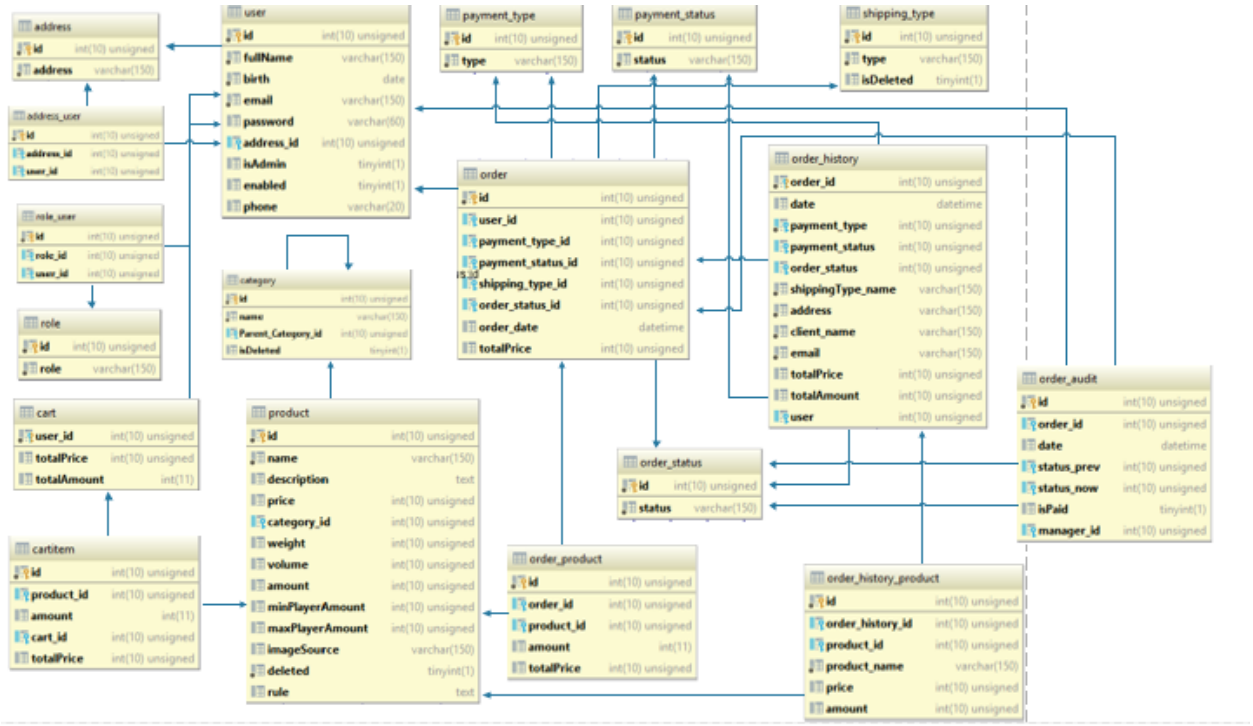
- IDE - IntelliJ IDEA
- Docker
- Maven
- SonarQube 7.1
- SQL WorkBench

4.2. Technologies

- ActiveMQ 5.7
- Ajax
- Bootstrap 4.2.1
- EJB
- Freemaker 2.3.28
- GSON
- Hibernate 5.2.10
- ItxtPDF 7.1.15
- Java 8
- Javascript
- Jquery
- JSF 2.1
- JSP 2.1

- Junit 4.12
- Log4j 1.2.15
- Lombok 1.18.16
- Mockito 2.18
- MySQL 8.0.15
- Omnifaces 2.6
- Primefaces 3.0
- REST
- Selenium 3.4.0
- Spring Framework 4.3.2
- Spring Security 3.2.0
- SweetAlert2
- Tomcat 8.5.161
- Twilio 3.4.5
- WildFly 10

5. Database model



Img 1: Database schema

6. System infrastructure

6.1. Front-end

Browser presentation level:

1. Web-page structure - HTML
2. Page-design - CSS
3. Dynamic content - JavaScript, JQuery, Ajax.

6.2. Back-end

Server based level

1. Application server - WildFly
2. Database - MySQL
3. Server logic - Spring Framework

6.3. REST client

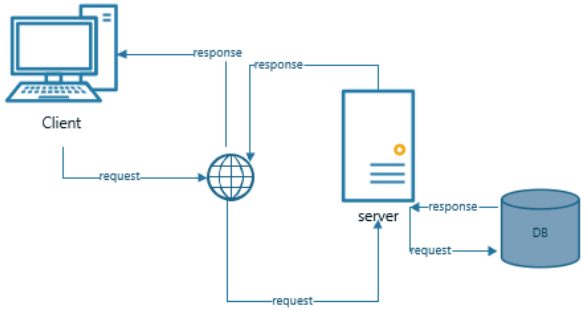
DisplayApp application

1. Web-pages - JSF
2. JMS - ActiveMQ
3. Application server - WildFly
4. Server logic - EJB
5. WS - REST

7. System architecture

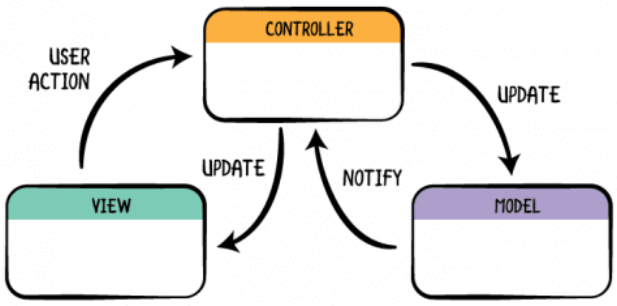
7.1. OnlineStore application

System represents client-server application architecture (img. 2). There are server based part (back-end) and client-oriented part(front-end).



Img 2: Architecture diagram

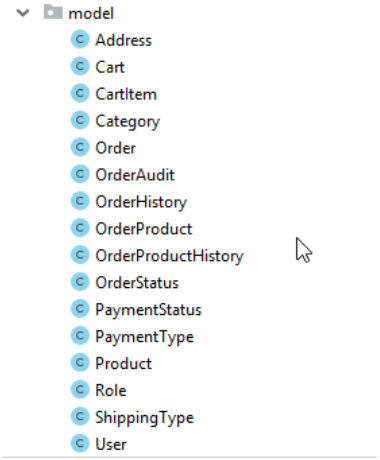
Architecture of server-based part is reflected by MVC - design pattern (img. 3).



Img 3: MVC pattern diagram

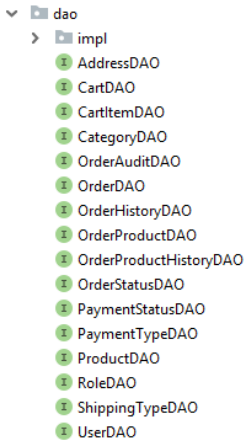
According to MVC pattern application has next structure

7.1.1. Model level



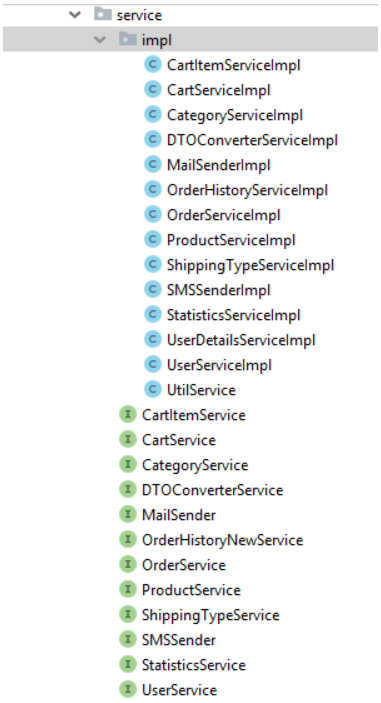
Img 4: Model representation

7.1.2. Model-Service level



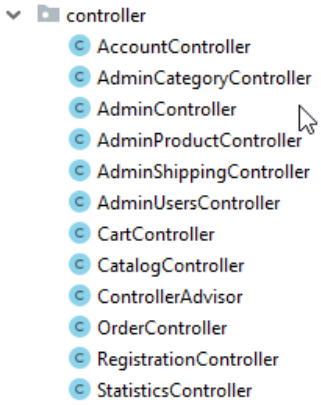
Img 5: DAO representation

7.1.3. Service level



Img 6: Service representation

7.1.4. View-Service level



Img 7: controller representation

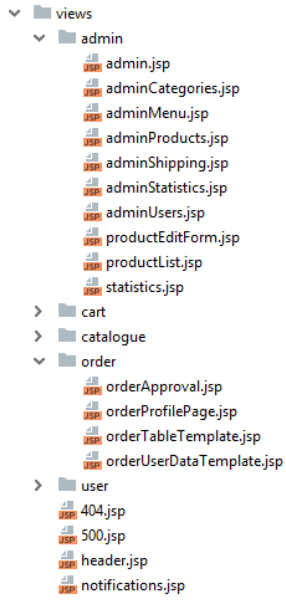
7.1.5. View level

Directory with views consists of five folders. Each folder includes views specific to cases.

- Admin folder includes administration panel views

- User folder includes account views
- Cart folder includes cart views
- Catalogue folder includes views used in navigation through store ctalogue
- Order folder includes order manipulation views

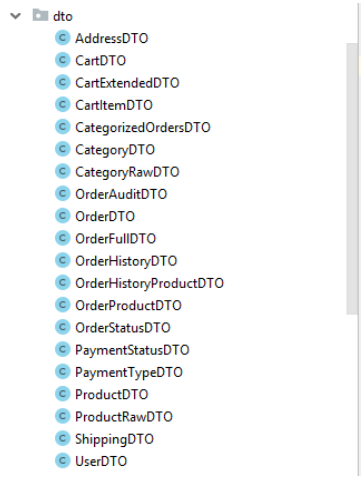
In root views directory error pages, header and notifications templates are stored.



Img 8: View representation

7.1.6. DTO

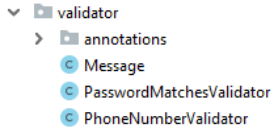
This folder includes DTO, which are used in application.



Img 9: DTO representation

7.1.7. Validation

This folder represents some aditional validators, which are used in DTO.

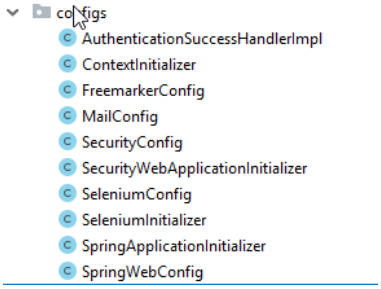


Img 10: Validators

Message class is used in cases Service need to send some information to View. It may be success message or error message in case Service got inapropriate data.

7.1.8. Configuration

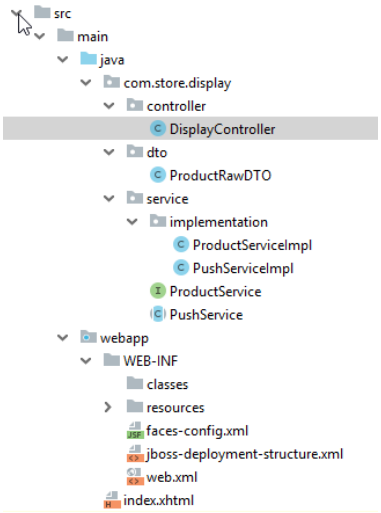
This folder represents Spring configuration classes.



Img 11: Configuration representation

7.2. DisplayApp application

Display application represents REST client, which is integrated in OnlineStore application. It is built on EJB and JSF technologies.



Img 12: DisplayApp structure

8. Aditional features

Some aditional festures were implemented above the basic requirements. They include buisness features as well as aditional technologies: Docker, Selenium.

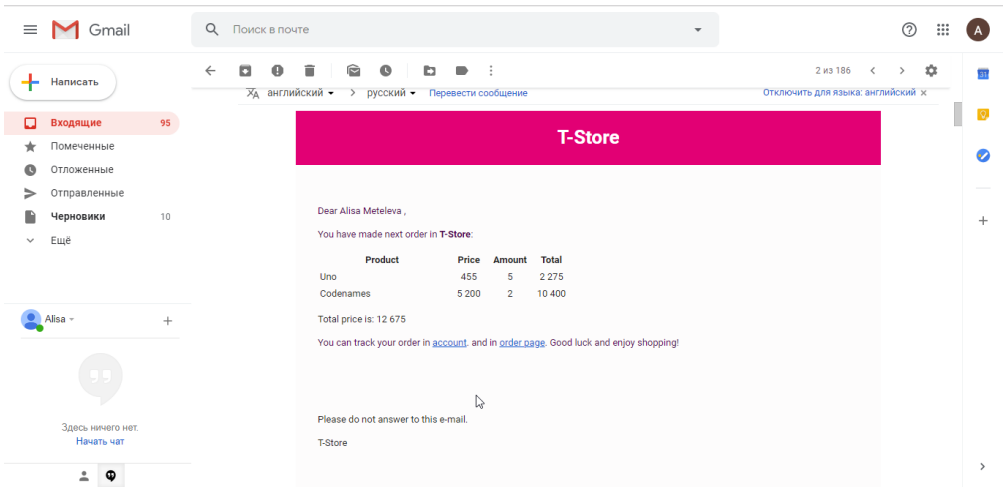
8.1. Notifications

After registration of a new user or when something changes in user's order status, system sends notification to mail and phone that were set by user while registration.

8.1.1. Email notification

Email (img. 13) are sent in two cases:

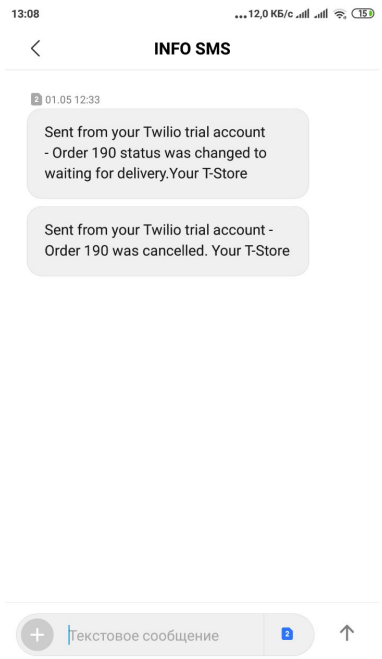
- New user was successfully registrated
- Order was approved by user.



Img 13: Email example

8.1.2. SMS notification

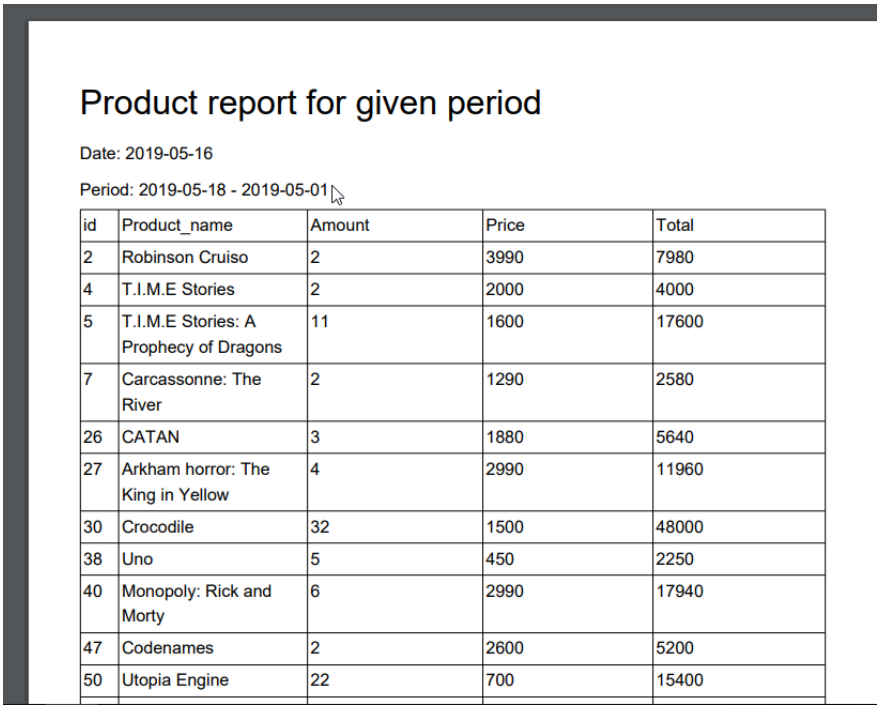
User gets SMS notification (img. 14) with order status, when it changes.



Img 14: SMS example

8.2. PDF Report

PDF report (img. 15) feature was added for user with ADMIN or MANAGER role. In case user needs full product statistics for specific period, he can download it from statistics tab in administration panel.



Product report for given period

Date: 2019-05-16

Period: 2019-05-18 - 2019-05-01

id	Product_name	Amount	Price	Total
2	Robinson Cruiso	2	3990	7980
4	T.I.M.E Stories	2	2000	4000
5	T.I.M.E Stories: A Prophecy of Dragons	11	1600	17600
7	Carcassonne: The River	2	1290	2580
26	CATAN	3	1880	5640
27	Arkham horror: The King in Yellow	4	2990	11960
30	Crocodile	32	1500	48000
38	Uno	5	450	2250
40	Monopoly: Rick and Morty	6	2990	17940
47	Codenames	2	2600	5200
50	Utopia Engine	22	700	15400

Img 15: PDF example

8.3. Selenium

Seleium autotesting was imlemented (img. 23). It covers login and signup cases

8.4. Docker

Docker was chosen as technolofy for easier deployment.

There are five modules described in docker-compose.yml:

- db, descibes running container with database dump process.
- OnlineStore, descibes running container with main application process
- DisplayApp, describes running container second application process.
- ImageBroker, describes running container with simple http server process. This module is used for image data interaction between two applications.
- activemq, descibes running container with activemq process .

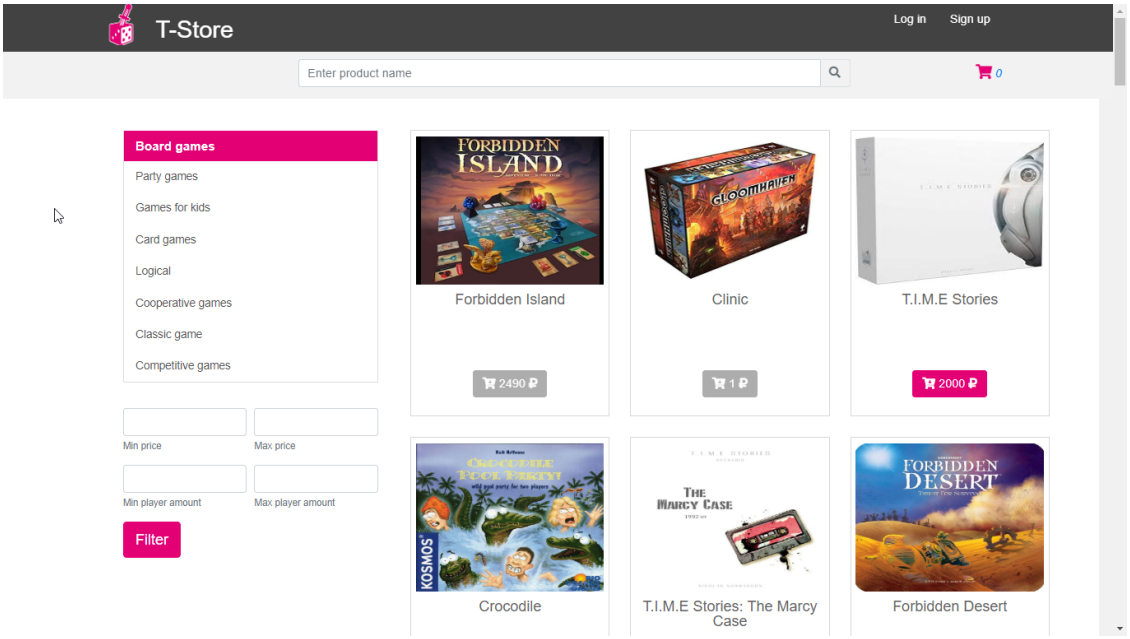
8.5. Order

User should see order history, which is not affected by changes in product profile or user account. Moreover, as user I would like to track changes in order statuses: time when they were happened and how it was changed.

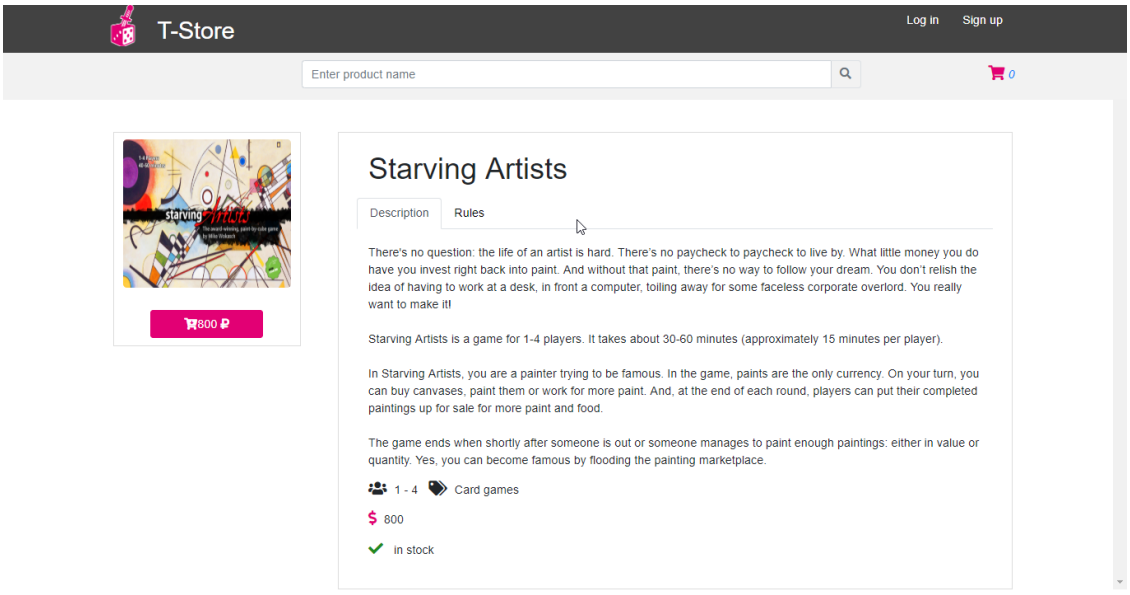
For this case order history does not get data by foreign keys but stores raw data and order profile page includes information about order audit (img. 19).

9. UI

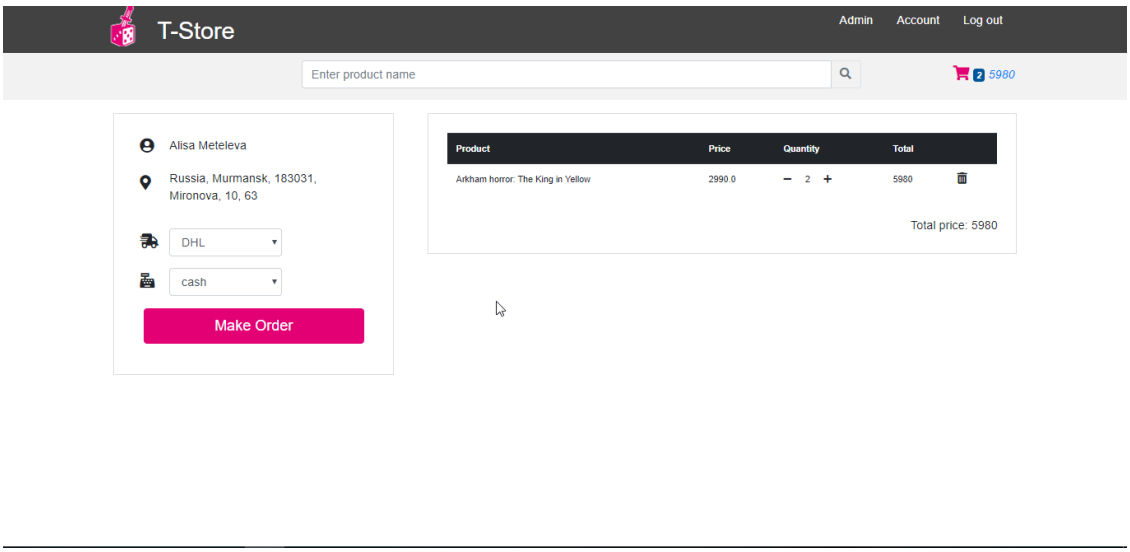
Some cases of user intaractions with system are shown below



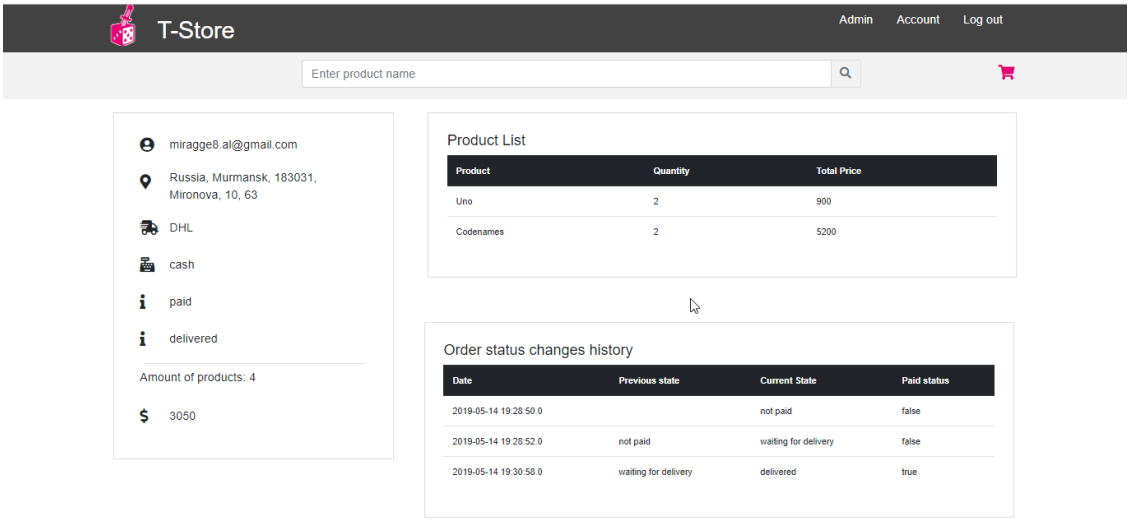
Img 16: Catalogue page



Img 17: Product page



Img 18: Cart page

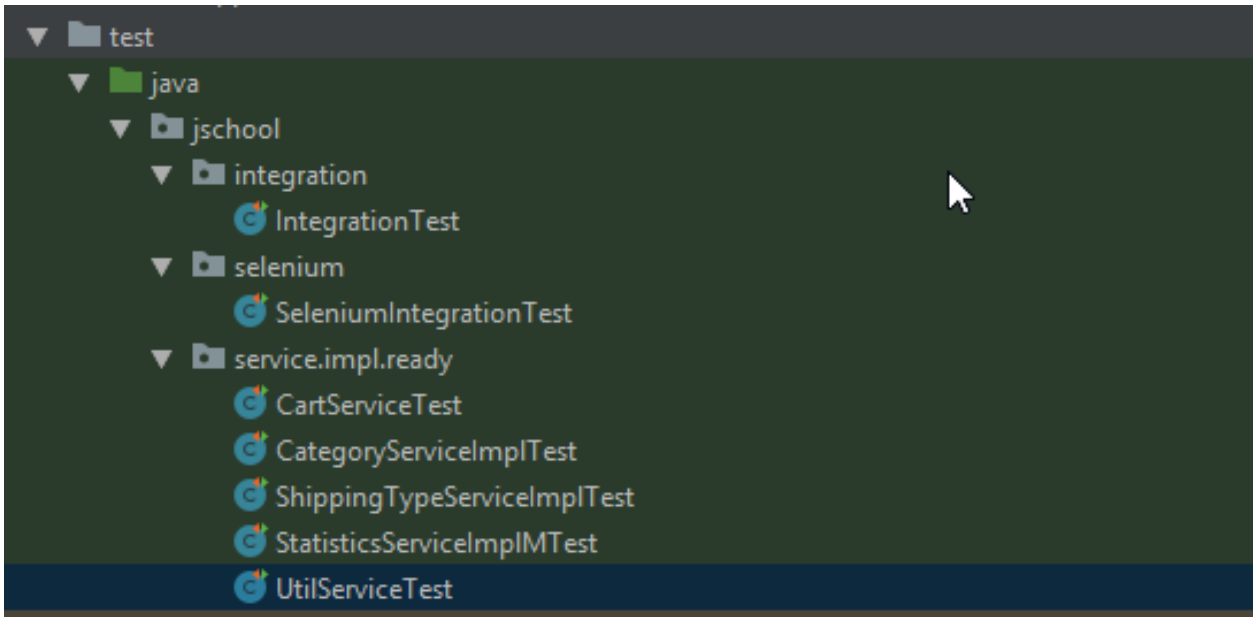


Img 19: Order page

10. Code quality

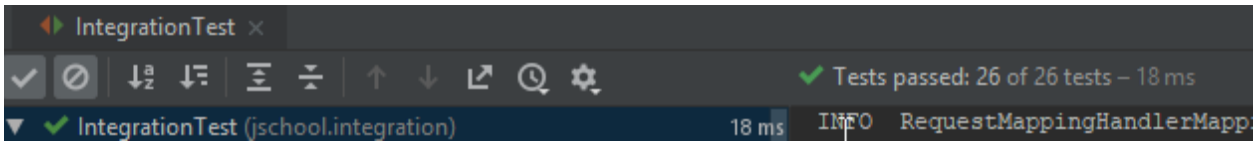
10.1. Tests

10.1.1. Test structure

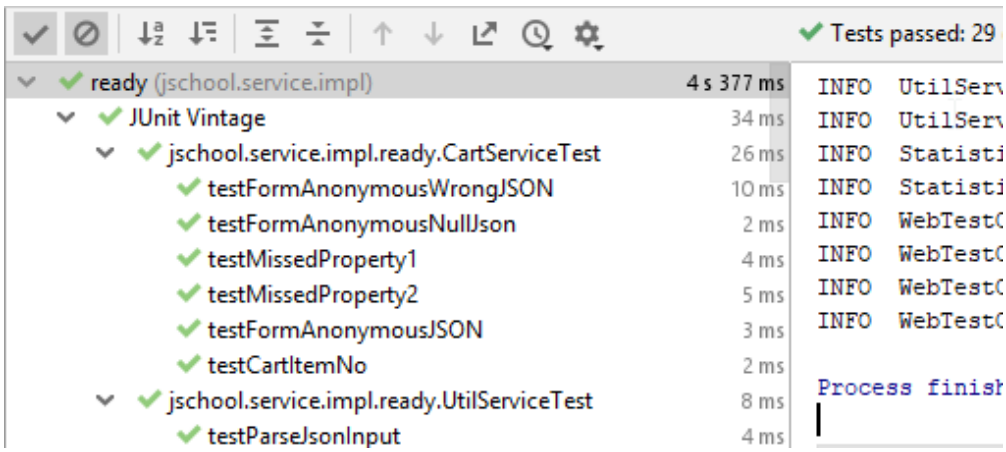


Img 20: Test structure

10.1.2. Junit tests

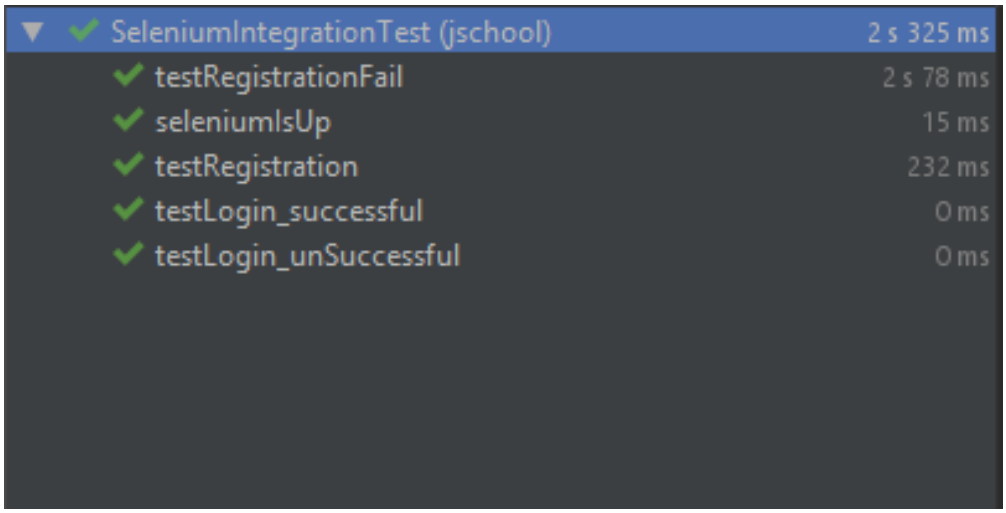


Img 21: Injection test results



Img 22: Unit test results

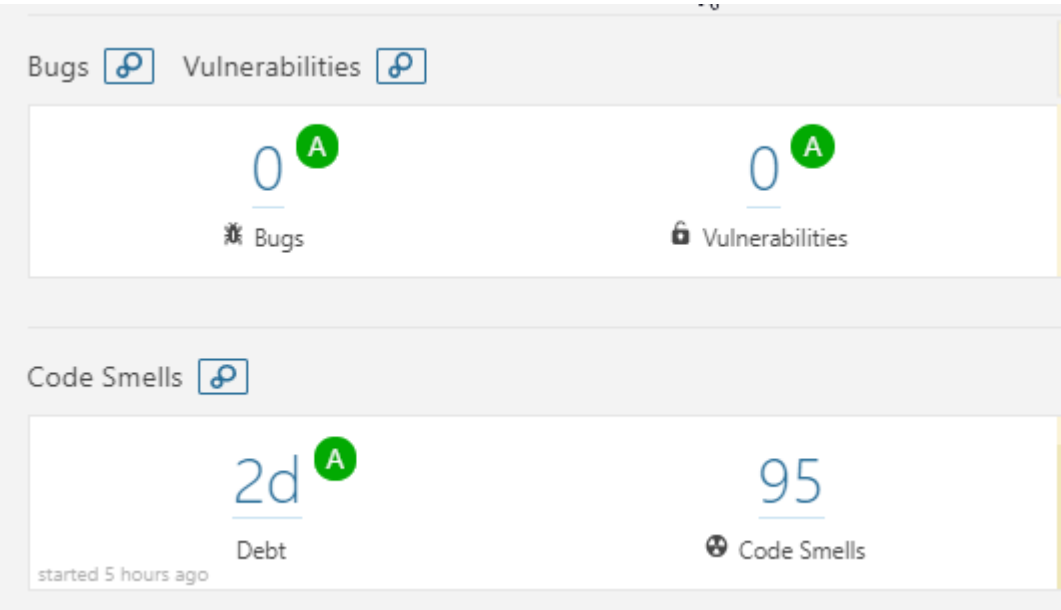
10.1.3. Selenium test



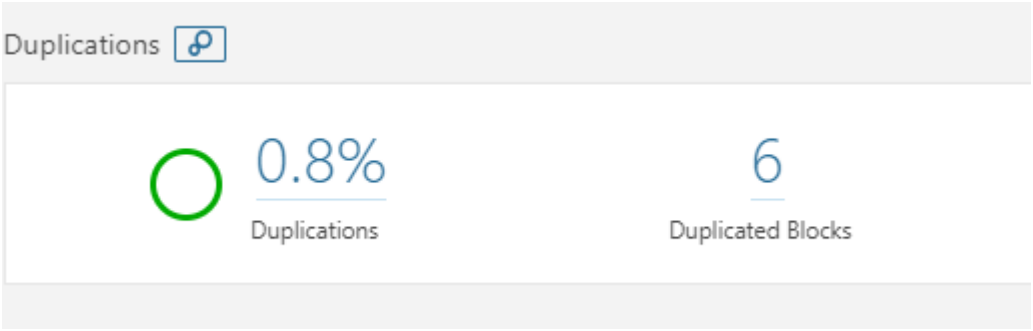
Img 23: selenium test results

10.2. Sonar

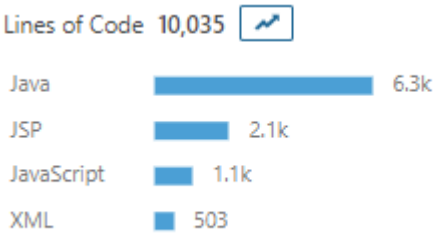
SonarQube measurements are listed below:



Img 24: Bugs and code smells



Img 25: Duplications



Img 26: Code size

10.3. Logging

Two levels of logging are written in log files (img. 27): INFO and ERROR

```
INFO  (UserDAOImpl.java:61) - User loaded successfully
INFO  (CartDAOImpl.java:30) - Cart loaded successfully, cart details=jschoc
INFO  (UserDAOImpl.java:61) - User loaded successfully
INFO  (CartDAOImpl.java:90) - Cart removed successfully
INFO  (CategoryDAOImpl.java:55) - category list was loaded successfully ( d
INFO  (ProductDAOImpl.java:30) - Product list was loaded
INFO  (UserDAOImpl.java:61) - User loaded successfully
ERROR (CartDAOImpl.java:92) - java.lang.IllegalArgumentException: attempt t
INFO  (ProductDAOImpl.java:30) - Product list was loaded
INFO  (CategoryDAOImpl.java:55) - category list was loaded successfully ( d
INFO  (UserDAOImpl.java:61) - User loaded successfully
```

Img 27: Log file example

11. Build and deploy

11.1. Build

Build is performed with Maven

1. cd OnlineStore-master
2. mvn clean install

11.2. Depoy

Deploy is performed with Docker

1. cd OnlineStore-master
2. docker-compose up

12. Future imrovement

Although basic requirements are alredy implemented, there are improvements system requires in future. They include new features as well as code structure improvements:

1. Adding new functionality : two-factor authentacion, payment system
2. Refactoring and optimisation
3. Increasing test coverage