Technical solution description

Informational system for online commerce

WebShop

By Checkmenev Alexander

T-systems Javaschool #16

Content

[1. Overview 3](#_Toc447478902)

[2. Used technologies 3](#_Toc447478903)

[3. Additional features 3](#_Toc447478904)

[4. Database scheme 4](#_Toc447478905)

[5. Implementation details 5](#_Toc447478906)

[6. Application modules 5](#_Toc447478907)

[7. UI-layer description 5](#_Toc447478908)

[8. Buisness-layer description 6](#_Toc447478909)

[9. Data-layer description 6](#_Toc447478910)

[10. Unit tests 6](#_Toc447478911)

[11. Deployment 8](#_Toc447478912)

[12. Logging configuration 8](#_Toc447478913)

[13. Sonar statistics 8](#_Toc447478914)

[14. Application improvements 8](#_Toc447478915)

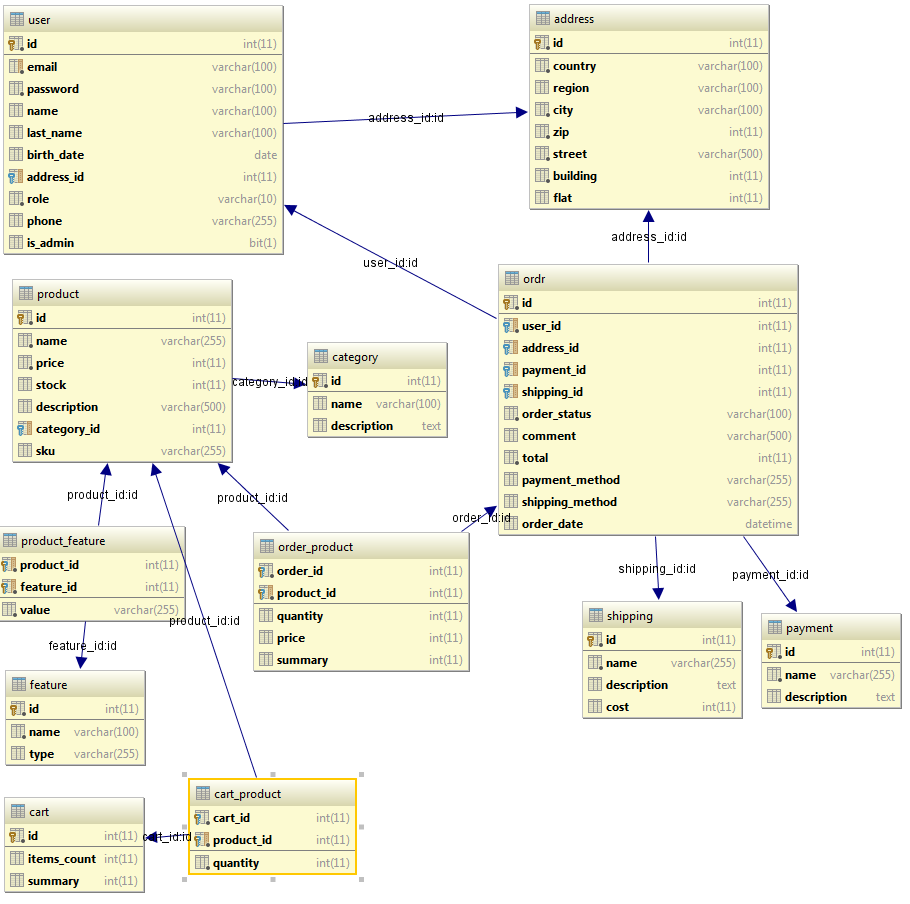
1. **Overview**

System represents as multiuser client-server application. It’s main purpose is an online shop where customers can find and buy different goods. System has two main entry points: storefront and backend. On storefront users can browse goods, add them to cart and place an order. Backend is used for administrating the system, managers can see placed orders, change their statuses and manage goods, that are selling. Also it has a web-service that provides statistic information about the shop.

1. **Used technologies**
2. JDK 1.8
3. Apache Tomcat 8.0.32
4. Spring framework 4.2.4.RELEASE
5. Spring security 4.0.3.RELEASE
6. JPA 2.1
7. Hibernate 5.0.7.Final
8. MySQL 5.7.11
9. Jackson 2.6.5
10. Maven 3.3.9
11. Log4j 1.2.17
12. JUnit 4.12
13. Mockito 1.10.19
14. Bean validation API 1.1.0.Final
15. Hiberate Validator 5.2.2.Final
16. JSP 2.1, JSTL 1.2
17. jQuery 1.12.1
18. Bootstrap 3.3.5
19. **Additional features**

Except the main requirements, system has:

1. Page for managing user rights
2. Cart is stored in database and associates with customer by cookie
3. Added several main entites: feature, category and cart.
4. **Database scheme**



User table has one-to-many relationship with Order(actually named ordr to avoid sql errors) and one-to-one with Address. Order table has two additional tables: shipping and payment, they contain shipping and payment methods respectively. Also order has one-to-many relationship with Address table and many-to-many relationship with Product. Junction table Order product has 3 additional fields, quantity, price and summary. Product table has and many-to-many relationships with Feature and Cart tables and one-to-many relationship with Category

1. **Implementation details**
2. **Application modules**

Application is split on 3 modules according to 3-tier application architecture:

1. View – module with controllers and view pages.
2. Services – module with business logic.
3. Dao – module with dao classes and entities.

Modules are connected using dependency injection of Spring Framework.

1. **UI layer description**

UI layer contains 18 controllers: 6 for backend part of application, 11 for storefront and the last is a REST service. Controllers are built, using Spring MVC technology.

Templates is a JSP with JSTL tags. Each page template use layouts. Layout is a custom tag, that contains all repeating content of a page (i.e. header and footer). There is 3 layouts in application:

1. Main layout – basic layout, defines 4 fragments: header, sidebar, footer and content, that can be filled with data. Styles and scpipts attach here.
2. Frontend layout – layout of storefront. Sidebar has category list, header with cart and login buttons.
3. Backend layout – sidebar is dynamic, in header there is nav panel.

Also there is productThumb tag, that contains a product block to injects in pages.

Application uses Bootstrap and jQuery libraries.

REST service represented as StatisticRestController. Service forms report about shop sellings and sends it in JSON format. It accepts four parameters:

* dateFrom – date of starting point of report (end date is always current date), topProductsCount – number of products to show
* topUsersCount – number of users to show
* accessToken – secret key that should be passed to get access to report. Secret key is mounted into application and can be read at statistics page in backend

1. **Buisness layer description**

Buisness layer has 8 service interfaces: 7 services has an implementation and the 8th service is the generic service that provides CRUD methods. CategoryService FeatureService and ProductService are extends this interface.

List of services with description is listed below:

1. AccountService – provides account management methods: sing up user, saveProfile, etc.
2. CartService – adding, editing cart.
3. CategoryService – CRUD operations with category.
4. OrderService – provides create order logic, changing order status.
5. FeatureService – CRUD operations with feature.
6. ProductService – CRUD with product, search by feature logic.
7. StatisticsService – methods for collecting statistics information of application.

All methods in service layer are transactional. Transactions are provided by Spring Framework.

1. **Data layer description**

Dao layer contains 12 JPA entities, 8 DAO classes and 3 DTO (for REST service).

There is generic dao interface, that provides CRUD operations and default implementation – AbstractGenericDAO.

Each DAO class manages one JPA entity, 4 entities has no DAO associated, because they are junction entities in many-to-many relation and not being managed directly.

1. **Application screenshots**

Storefront

Category page

Product page

Cart

Checkout

Backend

Orders

Products

Features

1. **Unit tests**

Unit tests are written on all classes of service layer, each test class tests one service class.

All methods have at least one positive test. Some methods have negatie scenarios too.

List of Junit test classes:

1. AccountServiceImplTest
2. CartServiceImplTest
3. CategoryServiceImplTest
4. OrderServiceImplTest
5. FeatureServiceImplTest
6. ProductServiceImplTest
7. StatisticsServiceImplTest

1. **Deployment**

Application uses Maven for build process.

To deploy the app to the Tomcat container need enter next command:

Mvn clean install tomcat7:deploy

1. **Logging configuration**
2. **Sonar statistics**
3. **Application improvements**

In further releases of the application can be added next improvements:

* Product images uploading
* WebShop settings, like welcome message, shop name, etc.
* Multiple storehouses, managing product stocks
* Product reviews
* Product import
* Managing shipping and payment method
* Ability to work with payment system API’s
* Ability to calculate shipping cost using shipping system services

Requirements to the technical document(first 13 are mandatory):

1. Describe additional implemented features that are not in the main task (your features);
2. Describe your explication and implementation of the model from the task(for example, do you have routes from point A to point B? What is it? What is a timetable? What is a basket?...)
3. Attach screenshots of the application(main pages);

--------------------------------------

1. How is the logging configured, give a short example of the log file content;
2. What patterns are used in the application(if they are used), is there any additional mapping, implemented algorithms? Additional util classes or constant list? Is there any security checks?
3. To make your text more readable provide use case diagrams, class diagrams, sequence diagrams;
4. If you have a statistic about your application/code (for example, from sonar) attach it as well;