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)

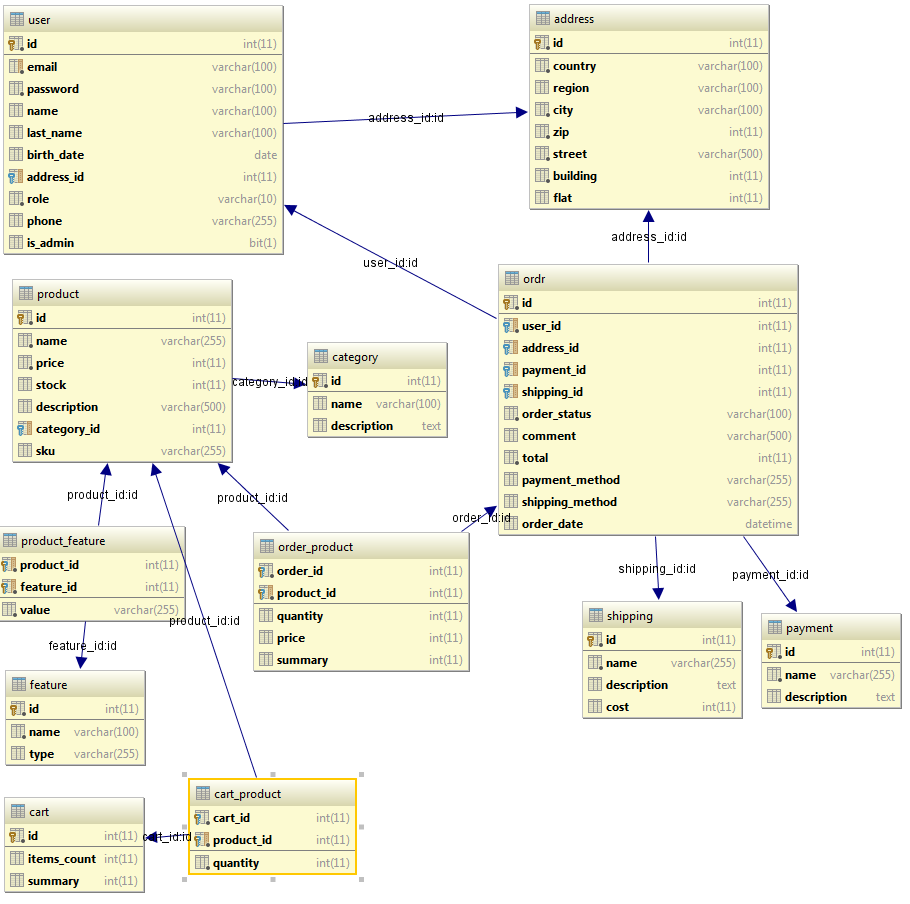
[13. Sonar statistics 8](#_Toc447478914)

[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(MVC, context, aop) 4.2.4.RELEASE
5. Spring security 4.0.3.RELEASE
6. JPA 2.1, Hibernate 5.0.7.Final
7. MySQL 5.7.11
8. Jackson 2.6.5
9. Maven 3.3.9
10. Log4j 1.2.17
11. JUnit 4.12
12. Mockito 1.10.19
13. Bean validation API 1.1.0.Final
14. Hiberate Validator 5.2.2.Final
15. JSP 2.1, JSTL 1.2
16. jQuery 1.12.1
17. Bootstrap 3.3.5
18. EJB 3.2
19. JSF 2.2
20. JAX-RS 2.0.1
21. Primefaces 5.3
22. iText 5.5.8
23. AS Wildfly 10.0.0
24. **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**

Application is built on 3-layered architecture.

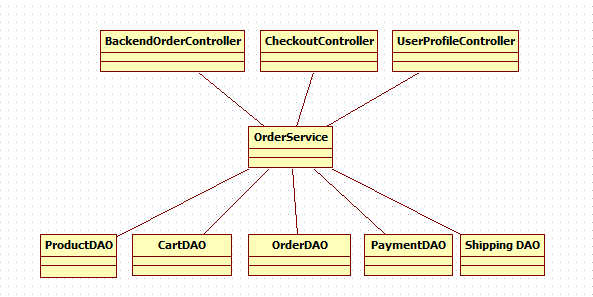
There is DAO layer, which provides data accessing with JPA implementation – Hibermate. Application has entitiy objects that are persisting to MySQL database by Hibernate.

All classes are running inside Dependency Injection container, provided by Spring Framework.

DAO classes injects to Service layer. Services are using DAO objects to get persister data. All service methods are tunning in transactions, so all operations with data is atomic however DAO’s is using.

Services are injecting to Controller classes of view layer. View is built on Spring MVC technology. Methods of controllers handle requests and call service methods to process. Views are JSP pages which uses Bootstrap and jQuery libraries.

On scheme presented typical class communication diagram for example on classes responsible for order:



Application appears as Online shop, so it has storefront, where products are exhibited and admin panel(backend) for managing content of the shop. Users can register in the shop, look on products, search them by features, add to cart and finally place an order. Also he can edit personal data and watch on order statuses. Only authorized users can buy goods, for authorization and authentication mechanisms is responsible Sping Security. Users password are hashing before storing by Bcrypt algorithm.

Cart of the shop is implemented in the following way:

When user add product to cart, creates cart in user session, cart saves to database and user accept cookie with cart identificator. If user close browser, his cart will not lost, and will be obtained by given cookie. Cookie is checking by Spring interceptor.

Also there is checks for stocks of product, if user try to buy more that it is in stock, he will be warned about it.

When user places order, quantity of bought product is debited from stocks.

If user has Admin rights, he can access admin panel by entering {appPath}/backend page. Admin panel provides ability to create and edit products, categories of products, create and bind features to products, browse orders and change statuses. Also there is user rights management page, where administrator can grant rights to users.

On statistics page there is a token generator button. Token is used to get report with webservice module. Token is stored in application context, in hash map, it has expiration date of 5 minutes. Token generates by SecureRandom class and encodes wirh Base64 into a string,

1. **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.

Also there is additional module that connects with application by REST service, it’s main purpose to collect and show sales report of the shop.

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. All other DAO’s extends this class and add their methods, specific for class.

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. **ShopReport module**

Shop report module connects with main application by REST service. REST API is built on Spring REST controller and has one entry point - <http://localhost:8080/api/report> and accepts 4 params:

1. fromDate – date of report period start in format dd-MM-yyyy, not required, if not exits, report will generate for all time.
2. topProductsCount number of top products to display
3. topUsersCount number of top customers to display
4. accessToken – security access token, it can be obtained on Statistics page in webShop backend.

This module is written using EJB+JSF+Primefaces technologies, for PDF generation used iText library. This module is deployed on AS Wildfly 10.0.0.

When user enters application, he see form with parameters for report, after entering wanted data, opens report page with statistics. It shows top users and products, total sales and orders and orders by status. Also user can download PDF file of report by clicking Download button.

1. **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. Token security for REST service
5. **Application screenshots**
   1. **Main application**

Category page

Product page

Cart

Checkout

Backend

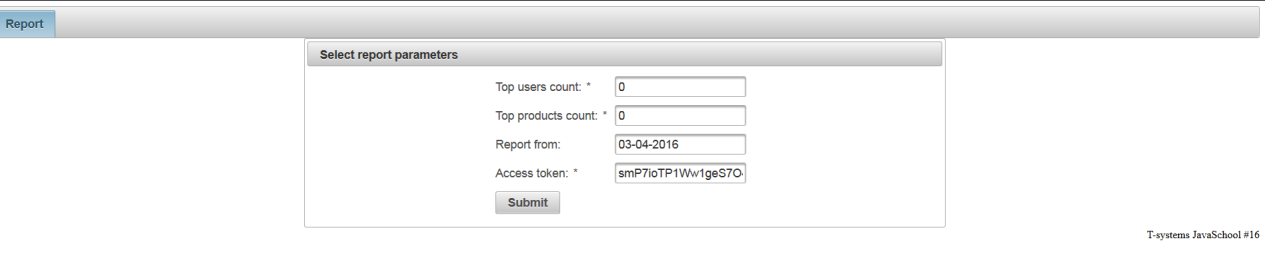
Orders

Products

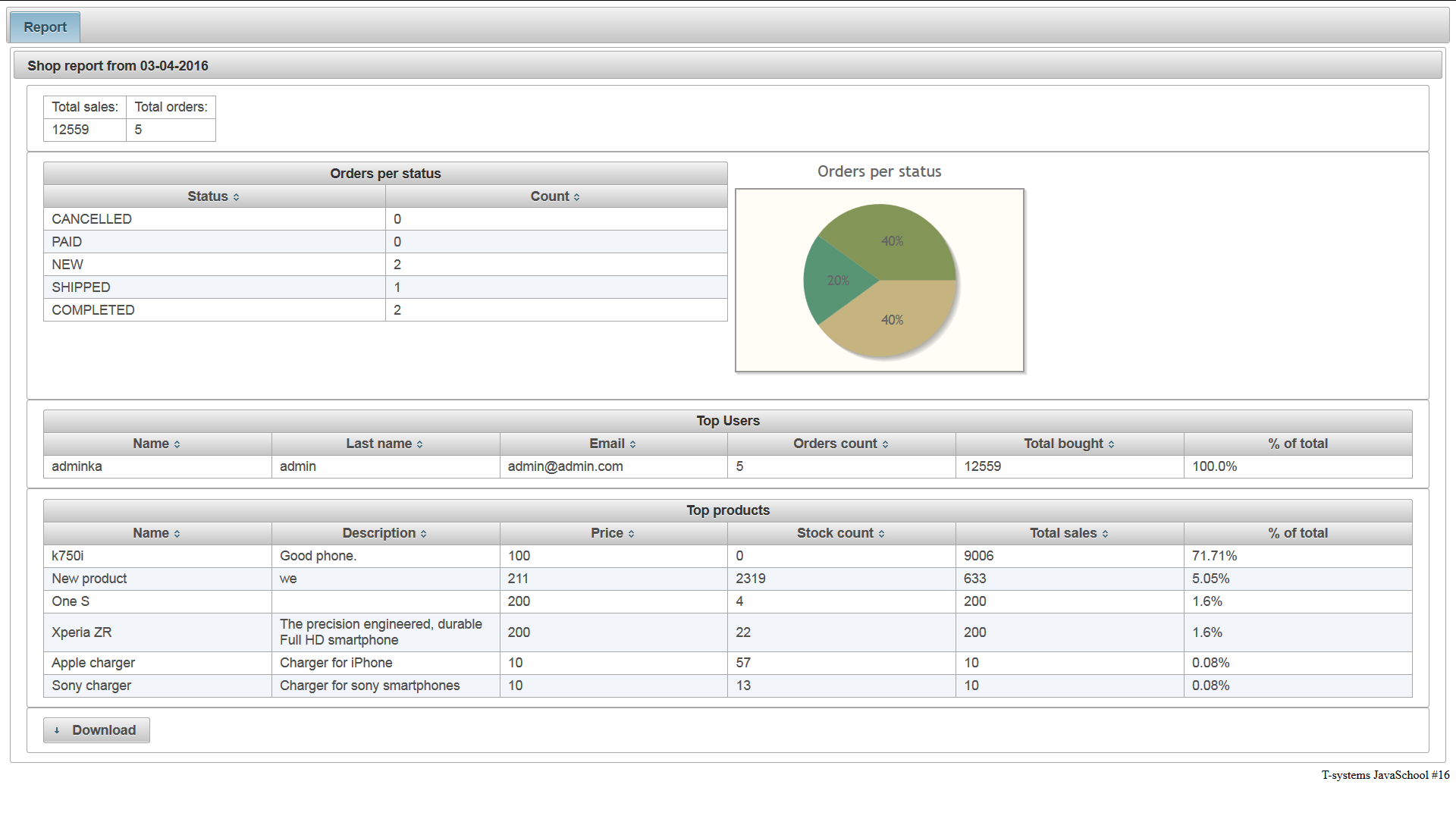
Features

* 1. **Webservice module**

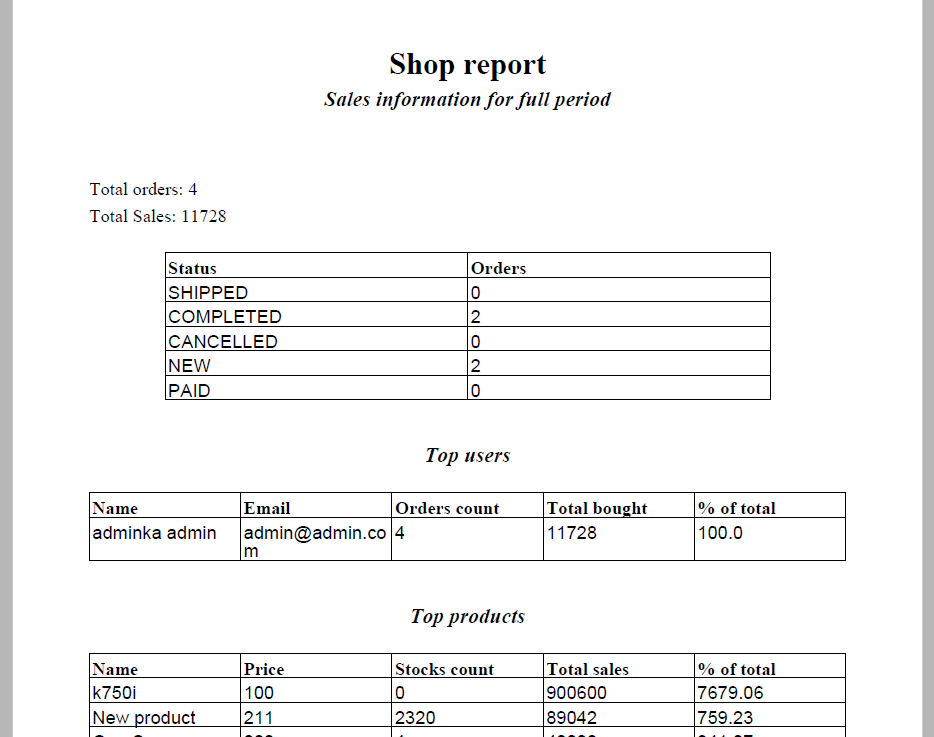
Parameters entering page



Report Page



PDF document



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.

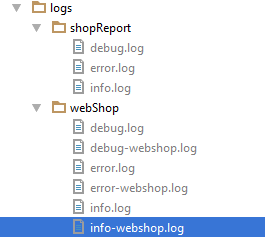
To deploy the app to the Tomcat container need enter next command:

Mvn clean install tomcat7:deploy

1. **Logging configuration**

Logging is provided by Log4j library.

Both applications – webShop and shopReport have different filders for log files. There are three types of logging levels: Debug/Trace log, info log and warn/error log. Each writes into a different file with RollingFileAppender. There is log folder structure presented on image.



Logs are writing using Spring AOP with AspectJ. Advice before all methods writes to debug level. Advice before methods of controllers writes to Info level – it represents users actions. And to error log writes all exceptions.

1. **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
* Out of stocks products report in webservice application