

E-commerce application
ESTORE

Technical Solution Description
by Andrei Sidorov

Saint-Petersburg 2019

Contents

1. Overview	3
2. List of used technologies	3
3. Additional features	3
4. Database scheme	4
5. Application modules	5
6. UI description	5
7. Business logic	5
8. Entities and DAO	6
9. Application screenshots	7
10. Unit tests	14
11. Deployment	19
11. Sonar report	19
12. Ideas for further development	19

1. Overview

ESTORE is an e-commerce web application for companies that sells electrical equipment to end consumers. It implements B2C model and allows customers to buy products via the Internet from a company who, in turn, integrates the application in its business process providing all necessary information of their products in the application. The main goal of this web application is to provide effective sale process between a company and its clients. The application is designed for clients and managers of a company: clients can view products, and buy them, managers can add and edit products, see reports and control the selling process.

2. List of used technologies

Java SE/EE 8
Maven 3.1.0
WilFly 16.0.0
Spring Framework 5.1.5.RELEASE
Hibernate 5.4.1.Final
MySQL 5.7
ActiveMQ 5.15.8
Opencsv 4.5
iText PDF 5.5.13
Apache Log4j
JUnit 5.4.0
Mockito 2.23.0
H2 database 1.4.198
JSP / CSS / JavaScript
SonarQube 7.7

3. Additional features

Dynamic product attributes. Product attributes and their values can be created, edited or deleted by the managers of a company. Therefore, a company may add and sell various types of products.

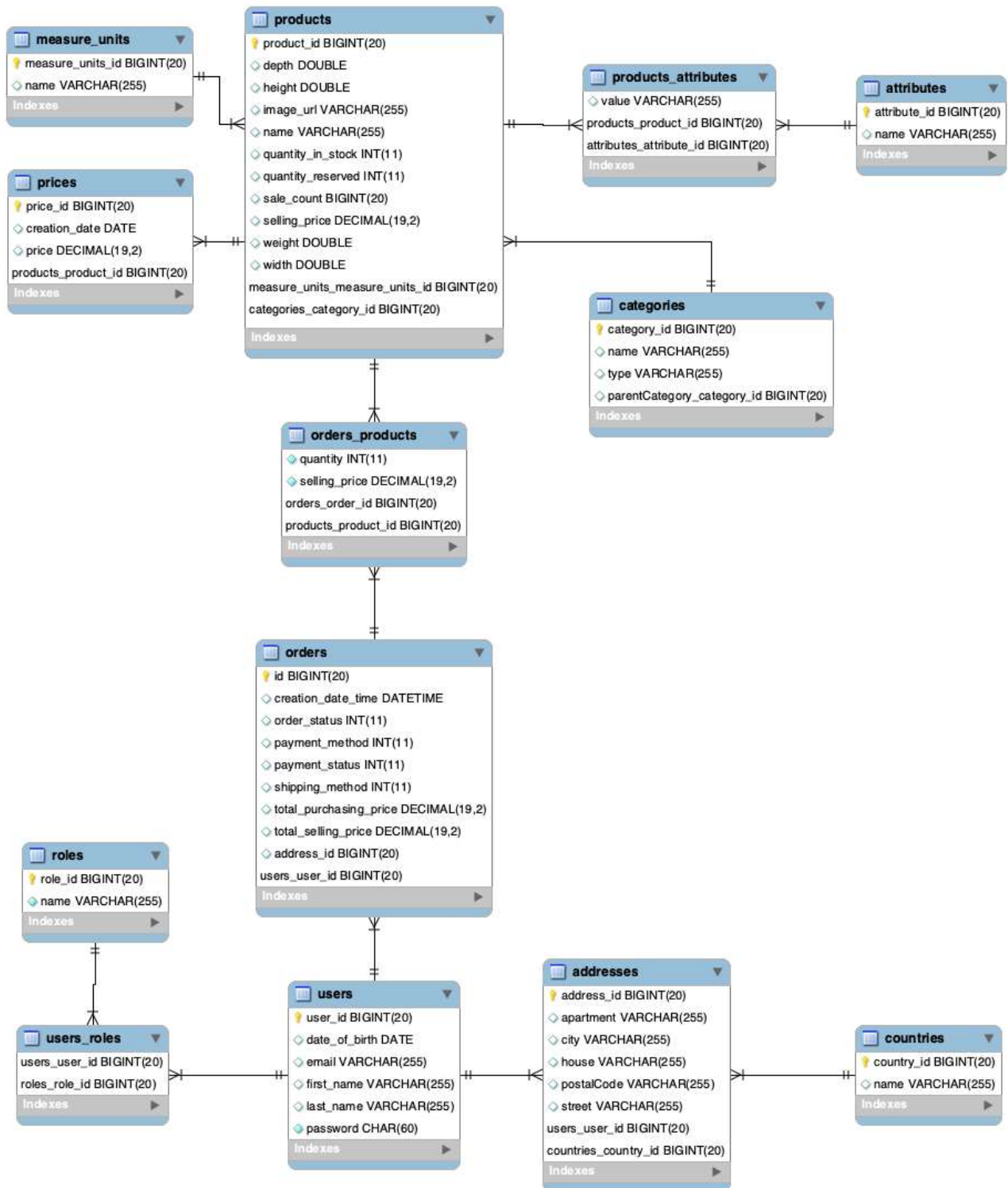
Dynamic product filter. Dynamically created product attributes and their values are processed into the product filter that may be applied during the product search.

Order reservation. Before check out, products get reserved for a specified time to allow customers be sure that their orders will be completely fulfilled after the payment process is finished.

Automatically generated invoices. After a purchase is complete, the invoice is generated with all information and sent to the buyer's email.

Multiple user addresses. Users may have many addresses and choose one during the check out process.

4. Database scheme



5. Application modules

There are two modules in the application: ESTORE is a main application and BILLBOARD is an application whose sole purpose is to display top-selling-products of the main application. The interaction between applications is implemented via JMS and REST. Initially BILLBOARD requests data at REST endpoint ‘/api/top-products’ of ESTORE. When there is a change in the list of top products, ESTORE sends updated data to BILLBOARD via JMS.

6. UI description

The ESTORE Web UI offers an easy to use web experience. The main template for all pages consists of a top navigation bar with links to all main pages, a page content and a footer with information about a company. Also an additional left side bar appears at some pages.

Main pages:

Page ‘Home’ is the first page of the application. It displays products and allows users search through them using different filters such as the category tree, the search bar and the attribute filter. All these filters can be used simultaneously.

Page ‘Order’ is essentially a shopping cart.

Page ‘Orders’ displays users’ orders and provides filtering and sorting by tables columns.

Page ‘Profile’ displays user information and provides an easy editing that allows to make changes and commit or discard them.

All pages use JSP, JavaScript and CSS.

7. Business logic

All business logic is positioned in the service layer of the application and contains following services.

AttributeService manages product attributes.

CategoryService manages product categories.

CountryService manages countries that used in the address model.

EmailService is responsible for sending email to users.

JmsService is responsible for sending messages via JMS.

MeasureUnitsService manages product measure units.

OrderService manages orders.

PaymentService is responsible for payment transactions.

PdfService is responsible for generating invoices for orders.

ProductService manages products.

RoleService manages user roles such as ADMIN, MANAGER and CLIENT.

UserService manages user accounts.

8. Entities and DAO

Address entity represents a user address and has following attributes: country, a city name, a postal code, a street name, a house number, an apartment number.

Country entity represents a country and has single attribute the name of a country.

Attribute entity represents a product attribute and has single attribute an attribute name.

Category entity represents a product category and has attributes: a category name, a parent category, sub-categories and a category type. The last three are used for displaying categories as a tree structure.

MeasureUnits entity represents the units of measurement for products and has single attribute the name of units.

Order entity represents an order and has following attributes: the data and time of creation, a payment method, a shipping method, a payment status, an order status, the total selling price of the product in an order, a total buying price of the product in an order, a user who made an order, an address where to deliver products.

OrderProduct entity is an intermediate entity between the order entity and the product entity and has following attributes: an order, a product, the quantity of a product, the selling price of a product.

Price entity represents a price and has following attributes: a creation date, the value of a price. This entity keeps track of the purchasing prices of a product.

Product entity represents a product and has following attributes: the name of a product, a selling price, a category, a weight, a height, a width, a depth, the available quantity in stock, a reserved quantity, the units of measurement, an image url, a sale count.

ProductAttribute entity is an intermediate entity between the product and its attributes.

Role entity represents user roles such as CLIENT, MANAGER and ADMINISTRATOR.

User entity represents a user and has attributes: first name, last name, email, date of birth and password.

This application uses declarative transaction management.

9. Application schreenshots

Main page

Home

Sign in

Cart (0)

SHOP BY CATEGORY

Circuit breakers

Miniature circuit breakers

Molded-case circuit breaker

Air Circuit Breaker

Enclosures



IC60N - miniature circuit breaker - 1P - 6A - C curve

\$20.00 In stock: 15 pieces

1

Add to cart



IC60N - miniature circuit breaker - 1P - 10A - C curve

\$20.00 In stock: 14 pieces

1

Add to cart



IC60N - miniature circuit breaker - 1P - 16A - C curve

\$20.00 In stock: 13 pieces

1

Add to cart



Compact NSX100B - TMD - 100A

\$20.00 In stock: 5 pieces

1

Add to cart



Compact NSX250B - TMD - 160A

\$20.00 In stock: 5 pieces

1

Add to cart



Compact NSX250B - TMD - 200A

\$20.00 In stock: 5 pieces

1

Add to cart

User-profile page

[Home](#)[Orders](#)[Reports](#)[Create product](#)[Edit categories](#)[Profile](#)[Sign out](#)[Cart \(0\)](#)[Add address](#)[Change password](#)

Profile

Personal details

First name

Last name

Date of birth



Address 1

Country



City

Postal code

Street

House

Apartment



Contact us

123 Street

City, Postal code

111-111-1111

comapny@mail.com

About

About

Careers

Help & FAQs

Shipping

Billing

Returns & exchanges

International shipments


Customer service

Ways to shop

Gift cards

Store locations

Shopping cart

	Price	Qty	Amount
 In stock: 15 pieces	IC60N - miniature circuit breaker - 1P - 6A - C curve \$20.00	<input type="text" value="2"/> ×	\$40.00
 In stock: 5 pieces	Compact NSX100B - TMD - 100A \$20.00	<input type="text" value="1"/> ×	\$20.00
Total:			\$60.00

Proceed to checkout

Contact us
123 Street
City, Postal code
111-111-1111
comapny@mail.com

About
About
Careers

Help & FAQs
Shipping
Billing
Returns & exchanges
International shipments
Customer service

Ways to shop
Gift cards
Store locations

Checkout page

[Home](#)[Orders](#)[Reports](#)[Create product](#)[Edit categories](#)[Profile](#)[Sign out](#)[Cart \(2\)](#)

Time left: 16:03

Checkout

Payment method

☐ Cash☒ Credit/Debit/Prepaid card

Card number

Name on card

Expiration date

CW

Shipping information

Address

☒ Standard☐ Express

Check out

Contact us

123 Street

City, Postal code

111-111-1111

comapny@mail.com

About

About

Careers

Help & FAQs

Shipping

Billing

Returns & exchanges

International shipments

Customer service

Ways to shop

Gift cards

Store locations

My-orders page

[Home](#)[My orders](#)[Profile](#)[Sign out](#)[Cart \(0\)](#)

My orders

Ref.	Date	Total price	Status			
<div><div></div><div>From 25/04/2019</div><div>To 09/05/2019</div></div>	<div><div></div><div>From</div><div>To</div></div>	<div><div></div><div></div><div></div></div>	<div>All</div> <div></div>	<div>Find</div>		
<div><div>^</div><div>v</div></div>	<div><div>^</div><div>v</div></div>	<div><div>^</div><div>v</div></div>				
12	2019-05-09	\$60.00	Awaiting delivery	<div>^</div>		
			Price	Qty	Amount	
<div></div>	Compact NSX250B - TMD - 160A		\$20.00	1 pieces	\$20.00	
<div></div>	IC60N - miniature circuit breaker - 1P - 10A - C curve		\$20.00	2 pieces	\$40.00	
Repeat			Total:		\$60.00	
13	2019-05-09	\$60.00	Awaiting delivery	<div>v</div>		

Contact us

123 Street

City, Postal code

111-111-1111

comapny@mail.com

About

About

Careers

Help & FAQs

Shipping

Billing

Returns & exchanges

International shipments

Customer service










Ways to shop

Gift cards

Store locations

Edit-categories page

Edit categories

- 1 Circuit breakers × 
- 2 Miniature circuit breakers × 
- 3 Molded-case circuit breaker × 
- 4 Air Circuit Breaker × 
-  +
- 5 Enclosures × 
-  +

Create-product page

Home

Orders

Reports

Create product

Edit categories

Profile

Sign out

Cart (0)

Create product

Import price list

Export price list

Measure units

Create product

Category

Choose category

Product name

Enter product name

Purchasing price

Enter purchasing price

Selling price

Enter selling price

Height

Enter product height in meters

Width

Enter product width in meters

Depth

Enter product depth in meters

Weight

Enter product wight in kilograms

Quantity in stock

Enter product quantity

Measure units

Choose measure units

Create product

Contact us

123 Street

City, Postal code

111-111-1111

comapny@mail.com

About

About

Careers

Help & FAQs

Shipping

Billing

Returns & exchanges

International shipments

Customer service

Ways to shop

Gift cards

Store locations

10. Unit tests

Testing AttributeService

Testing getAttributeValues method

when there are equal attribute values then they are merged and returned

passed

when the second argument is a list of categories' ids then the overloaded method is

called

AttributesWithValues method

when this method is called then attributes of products of a specified category are

returned

Testing updateAttributes method

when this method is called then the values of a product's attributes are updated

Testing getAllProductAttributes method

when this method is called then method attrbiuteDao.getAllProductAttributes() is

called

Testing save method

when an attribute does exist then an attrbute and a ProductAttribute are created and

saved

when an attribute exists then only a ProductAttribute is created and saved

Testing CategoryService

Testing getSubCategoriesIds method

when this method is called then the all ids of all sub-categories are returned

Testing renameCategory method

when this method is called then a category gets renamed

Testing deleteCategory method

when a category has no products then this category gets deleted

when a category has products then exception CategoryDeleteException gets thrown

when a category is a folder that has no sub-categories then method

productDao.getProductsByCategoriesIds is not called

Testing getTopLevelCategories method

when this method is called then methodcategoryDao.getTopLevelCategories gets

called

Testing findById method

when this method is called then method categoryDao.findById gets called

Testing getAllCategories method

when this method is called then method categoryDao.getAllCategories gets called
Testing save method
when this method is called then category is created and saved

4 **Testing CountryService**

Testing getAllCountries method
when this method is called then method countryDao.getAllCountries() gets called
Testing findById method
when this method is called then method countryDao.findById() gets called

Testing JmsService

Testing send method
when this method is called then the top-selling products are sent

Testing MeasureUnitsService

Testing findById method
when this method is called then method measureUnitsDao.findById() gets called
Testing getAllMeasureUnits method
when this method is called then method measureUnitsDao.getAllMeasureUnits()
gets called
Testing save method
when this method is called then new measure units are created and saved

Testing OrderService

Testing getWeeklyRevenues method
when a start date is not before an end date then exception TimePeriodException is
thrown
when this method is called then the list of weekly revenues are returned
Testing getMonthlyRevenues method
when this method is called then the list of monthly revenues are returned
when a start date is not before an end date then exception TimePeriodException is
thrown
Testing updateOrderStatus method
when this method is called then an order's status gets updated
Testing getOrderProducts method
when this method is called then method orderDao.getOrderProducts gets called
Testing findById method

when this method is called then method `orderDao.findById` gets called

Testing `findOrdersByCriteria` method

when this method called then method `OrderDao.findOrdersByCriteria` gets called

Testing `save` method

when this method is called then an order gets saved

Testing PdfService

Testing `createInvoice` method

when this method is called then a pdf file with an invoice is created

Testing ProductService

Testing `deleteProductAttribute` method

when this method id called then method `productDao.deleteAttribute` is called

Testing `processPriceListFile` method

when a product has no id then this product gets saved

when a product has id then this product gets updated

when one of the values is invalid then exception `PriceListException` is thrown

when one of the values is missing then exception `PriceListException` is thrown

Testing `createPriceListFile` method

when this method is called then a price list file is created

Testing `getTopSellingProducts` method

when this method is called then top selling products are returned

when there are products with sale count zero then they are ingnored

Testing `searchByCriteria` method

when criteria have the empty input and the empty category id then method

`productDao.findByPartialName` is called

when criteria have only a partial name then products are searched by this partial name

when criteria have a partial name and a category id then products are searched by the partial name and the category id

when criteria have a partial name and a category id of type folder then products are searched by the partial name and the all sub-categories of the category

when attributes and values of a product match searched criteria then this product gets returned

when attributes and values of a product do not match searched criteria then this product gets ignored

when criteria have a product attribute with value 'All' then all products with this attribute are returned

Testing update method

when a dto has a new price then the price of a product is updated

when a dto has the same price as the current price of a product then the price of this product is not updated

when a dto has a new product name then the name of a product is updated

Testing buyProducts method

when this method is called then products get bought

Testing unreserveProduct method

when this method is called then products get unreserved

Testing reserveProducts method

when an available quantity for each product is equal or greater than required then all products get reserved

when an available quantity for one of the products is less than required then none of the products get reserved

Testing getProductsByCategory method

when a category has type folder then the all products of sub-categories are returned

when a category has type category then products of this category are returned

Testing getAllProducts method

when products are found then products are returned

when products are not found then empty list is returned

Testing save method

when this method is called then a product is saved

Testing findById method

when a product is found then this product is returned

when a product is not found then null is returned

Testing RoleService

Testing findByName method

when this method is called then method roleDao.findByName() gets called

Testing save method

when this method is called then method roleDao.save() gets called

Testing UserService

Testing loadUserByUsername method

- when a user is found then this user is returned

- when a user is not found then exception UsernameNotFoundException is thrown

Testing getTopClients method

- when this method is called then the list of top clients is returned, limited to a specified number

- when clients have no purchases then those clients are ignored

- when there are no clients then an empty list is returned

Testing updateUserDetails method

- when this method is called then a user's details are updated

Testing removeUserAddress method

- when this method is called then an address is removed

Testing updateUserAddress method

- when this method is called then an address is updated

Testing addUserAddress method

- when this method is called then a new address is added

Testing changePassword method

- when this method is called then a password is changed

Testing checkIfOldPasswordValid method

- when a password is valid then true is returned

- when a password is invalid then false is returned

Testing findByEmail method

- when a user is found then this user is returned

- when a user is not found then null is returned

Testing save method

- when this method is called then a user is created and saved

11. Deployment

Step 1: start WildFly application server

in OSX/Linux: standalone.sh

in Windows: standalone.bat

Step 2: Run the following command from the main directory of the application:

```
mvn wildfly:deploy
```

12. Sonar report

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Coverage	Duplications
java/dev/a2/estore	5,802	0	7	57	63.1%	4.2%
config	301	0	0	1	94.9%	0.0%
controller	1,194	0	7	27	36.7%	1.3%
dao	661	0	0	4	51.8%	0.0%
dto	915	0	0	7	70.9%	13.0%
exception	121	0	0	0	15.6%	0.0%
model	1,145	0	0	12	60.0%	9.8%
service	1,363	0	0	4	82.9%	0.0%
validation	102	0	0	2	90.9%	0.0%

13. Ideas for further development

First of all, UI might be adapted for tablet and mobile screens. This is essential as more and more users prefer portable devices to shop online.

Secondly, add more technical information about products, it might be pdf pages from a manufacture's catalogue uploaded to each product.

Next is to provide coupling between products and their accessories for their easy selection.