Technical Solution Description

Information system of online store

SPACESHOP

by Antonova Elizaveta

Content

Overview

Information system of online store is a multi-user client-server application. The application implements different functionality depending on the role: available roles are user and admin.

Users have access to the catalog of goods where they are able to choose products, add them to the shopping cart and place new order. Besides, users have personal accounts with all information. User is able to change his password, add new address and remove another one. Also, he can look for order history with all information about bought goods, delivery and payment methods.

Admins main tool is the control panel where they are able to add and remove categories, create new products, look for all orders and manage delivery status. Moreover, admins have access to the statistic of total income for the current period of time.

Application also has a service that provides information about top categories with the biggest amount of sold products and sends it to another application.

Technologies and frameworks

List of technologies used in the application:

- JDK 1.8
- Apache Tomcat
- Spring boot
- Spring security
- Spring data
- JPA, Hibernate framework
- MySQL
- Jackson
- Maven
- Log4j
- JUnit
- Mockito
- Thymeleaf
- EJB
- JSF
- JavaScript
- Bootstrap 5
- JMS
- Java Mail
- AS Wildfly
- Rabbit MQ
- WebSocket
- Jersey Client

Additional features

- Ability to send email with order information for buyer after placing new order.
- Ability for admin to receive information about total income for the selected period of time.
- Pagination added to view all lists in application.

Architecture of application

Database scheme

The application includes entities:

1) User – represents application users (user or admin)
2) User Address – represents user's address
3) Product – represents the product (space items in the scope of the application)
4) Product In Order – products in the placed order
5) Order – represents order with information about it
6) Category – shows available categories of the product catalog
7) Top Category – shows categories with the biggest amount of sold products

Users table has one-to-many relationship with user addresses and orders tables.

User addresses table has many-to-one relationship with users table.

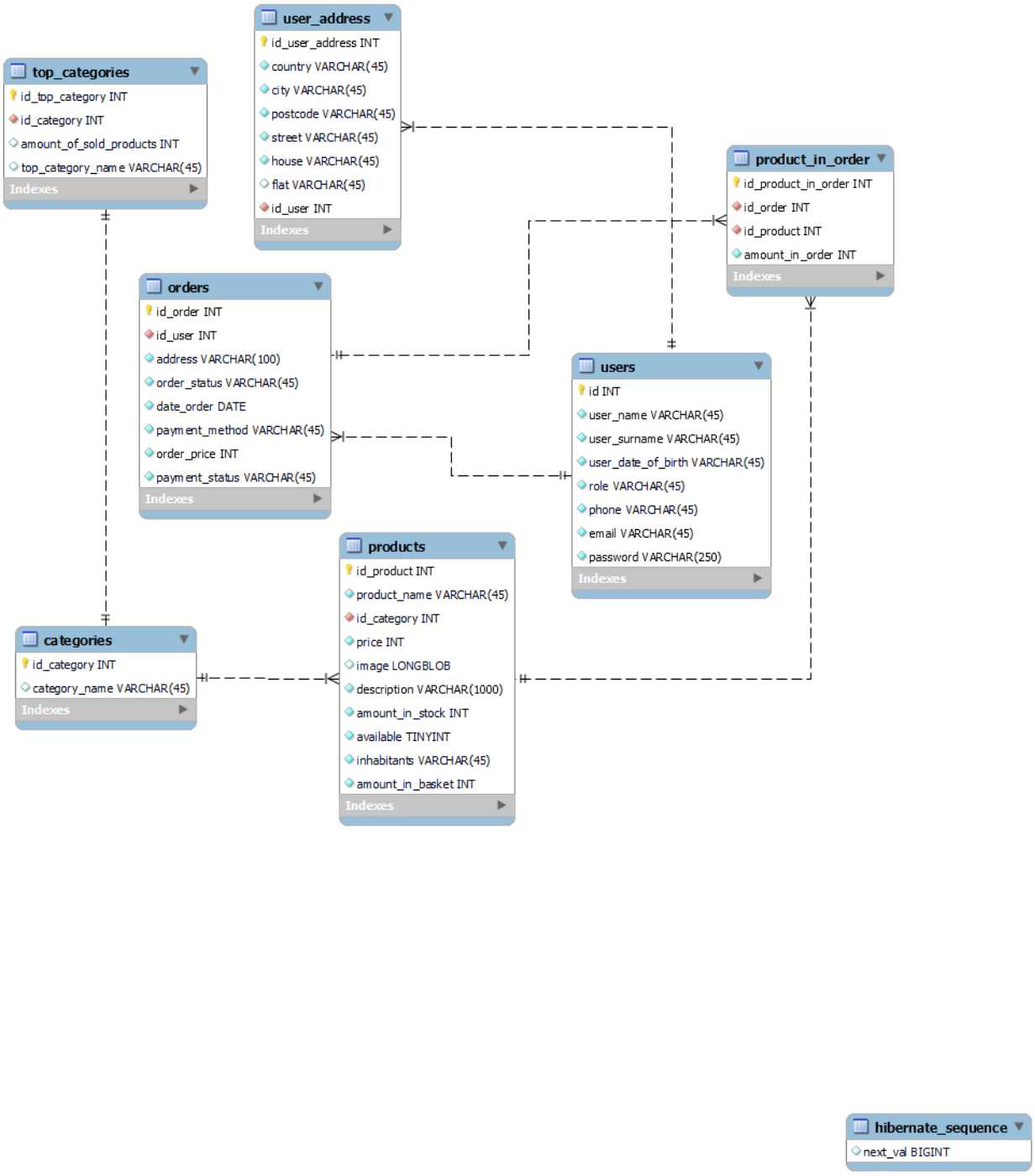
Products table has one-to-many relationship with product in order table and many-to-one with categories table.

Product in order table has many-to-one relationship with orders and products tables.

Orders table has one-to-many relationship with product in order table and many-to-one with users table.

Categories table has one-to-many relationship with products and one-to-one relationship with top categories.

Entity relationship model is shown on the picture 1.



Picture 1 - Entity relation model

Description of model implementation

The application divides abilities and business processes by the user role.

Available functionality for user role:

1) Browse product catalog with ability to filter by category or turn pages with 6 products on one page.
2) Browse personal account and edit address information, for instance user is able to add new address or remove one of existing. Otherwise, user is able to edit password.
3) Browse order history with main information, for example payment and delivery method and status. Moreover, user can look for list of products bought in each order with main params (name, price, picture, etc.).
4) Add products to shopping cart.
5) Remove products from shopping cart.
6) Create new order with choosing delivery address and payment method.

Available functionality for admin role:

1) Browse all existing orders.
2) Change order’s status.
3) Create new categories and remove existing.
4) Create new products and browse all existing.
5) Browse information about total income for selected period of time (between two dates).

In case of application all products are unique and available in a single quantity. For this reason, available amount of product in stock changes only after placing order. The idea is that the product will be received by the one who buys first. Accordingly, the store does not miss out on revenue.

Application modules

Application is built on MVC architecture pattern and has three layers.

The data access layer provides its functionality to access data with Spring Data, JPA and Hibernate. In this layer used repositories for every entity. All entities of application are mapped on MySQL database.

The service layer includes all business logic. Services are using JPA repositories injection to get data to make all business operations.

The view layer managed by controllers. Services are injected to controllers. Controllers handle requests and use service methods to make operations and give an appropriate response to the view. Views are HTML pages with template engine Thymeleaf. Moreover, there is used Spring Security Dialect to config Thymeleaf for ability to hide some information on pages for different user roles.

Besides, there are two additional modules:

The first one sends message to Rabbit MQ with “update” text message. It becomes when user purchases new product and top categories are getting update. The second one sends email to user email address after creating order with main order information using Java Mail.

UI description

Templates are html files with Thymeleaf template engine and Spring Security Dialect tags. Each template uses Bootstrap styles located in bootstrap.min.css file and personal styles located in main.css file.

All pages have the default header – navbar.html which contains navigation bar with buttons. Buttons are shown depending on the user's role with using Spring Security dialect.

UI layer operated with 9 controllers. Controllers are built using Spring MVC pattern.

The second application with stand of top categories uses JSF files with Java Script. Moreover, there is WebSocket client on the view page which receive message from WebSocket server and after that client reload page using Java Script.

Business logic description

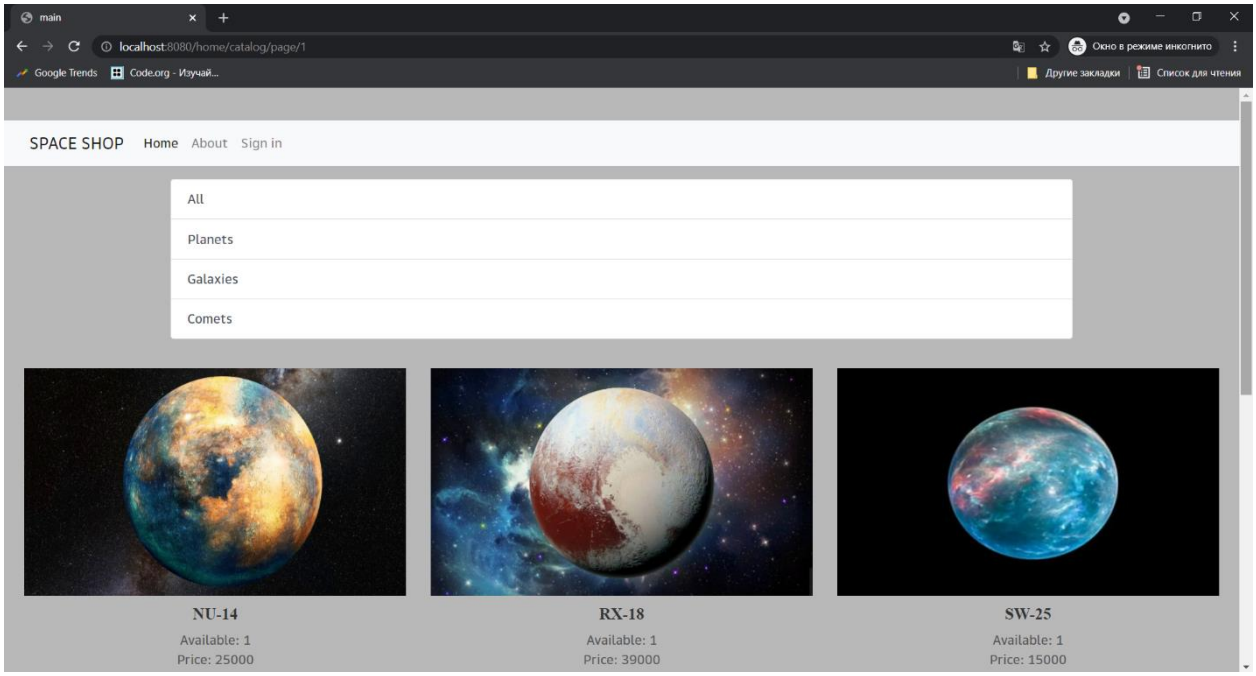
Business layer includes 9 service interfaces and 9 implementations with methods. There is list of services with short description:

1) Address service – provides logic for CRUD operations with addresses.
2) Basket product service - provides logic for CRUD operations with products in basket and specific operations: counting products in basket and counting total price of products in basket.
3) Category service - provides logic for CRUD operations with categories.
4) User service – provides logic for CRUD operations with users and changing user password.
5) Order service - provides logic for CRUD operations with orders and some specific operations like changing order status, getting income for selected period of time, sending email message for buyer.
6) Product in order service - provides logic for CRUD operations with products in order.
7) Top category service - provides logic for CRUD operations with top categories and specific operation – changing amount of sold products in top category.
8) Email service – provides sending messages to user email.
9) Product service - provides logic for CRUD operations with products and specific operations like filtration products by category.

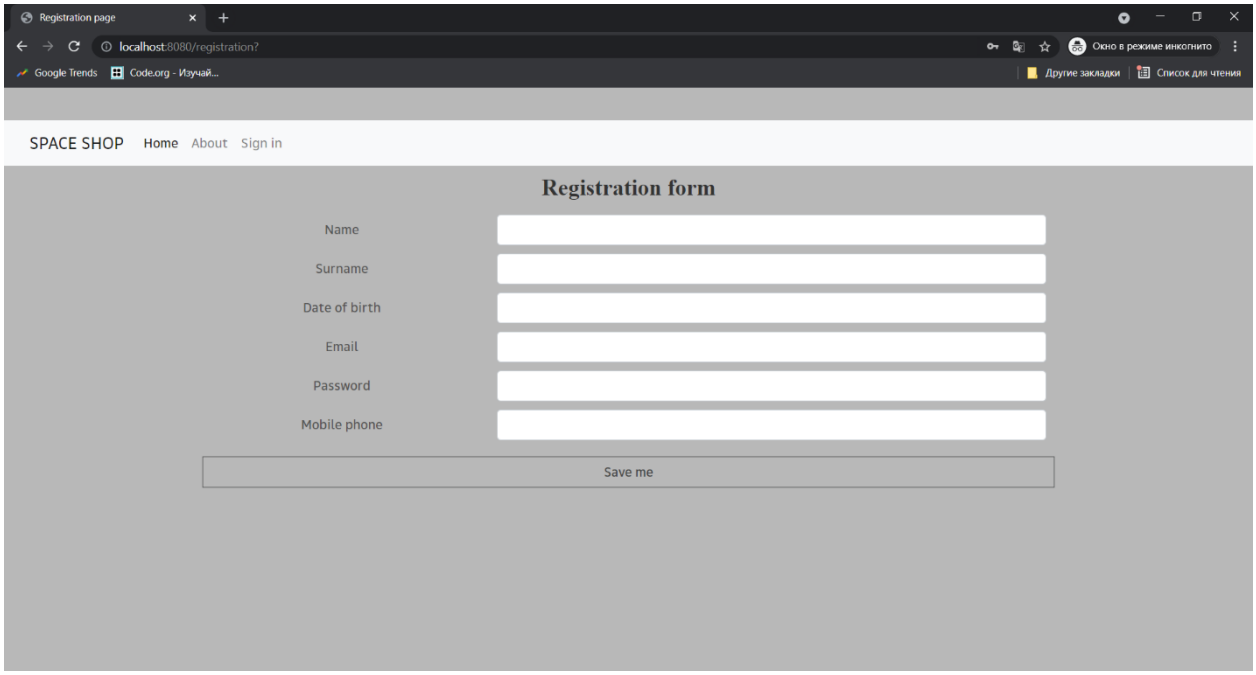
Data description

Data layer contains 7 JPA repositories, 7 entities, 6 DTO classes and 1 of them for the second application (REST service). All repositories use Spring Data to simplify interaction with data base. Repositories realize CRUD operations and some specific operations described in the previous chapter.

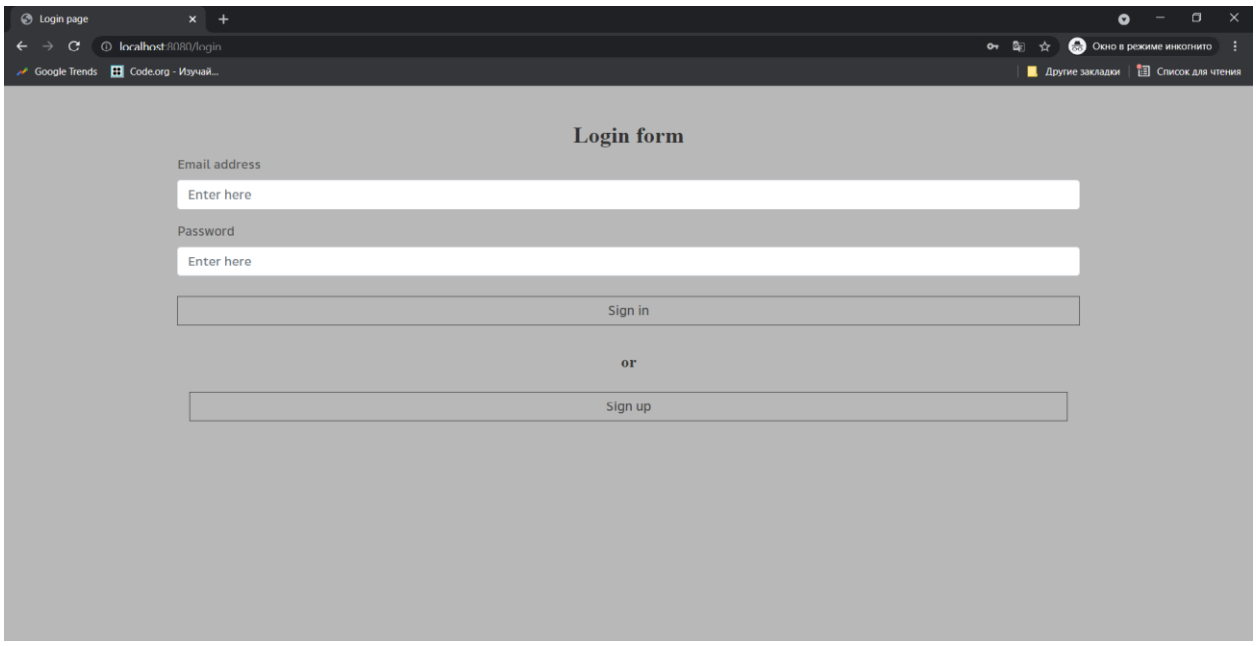
Application screenshots



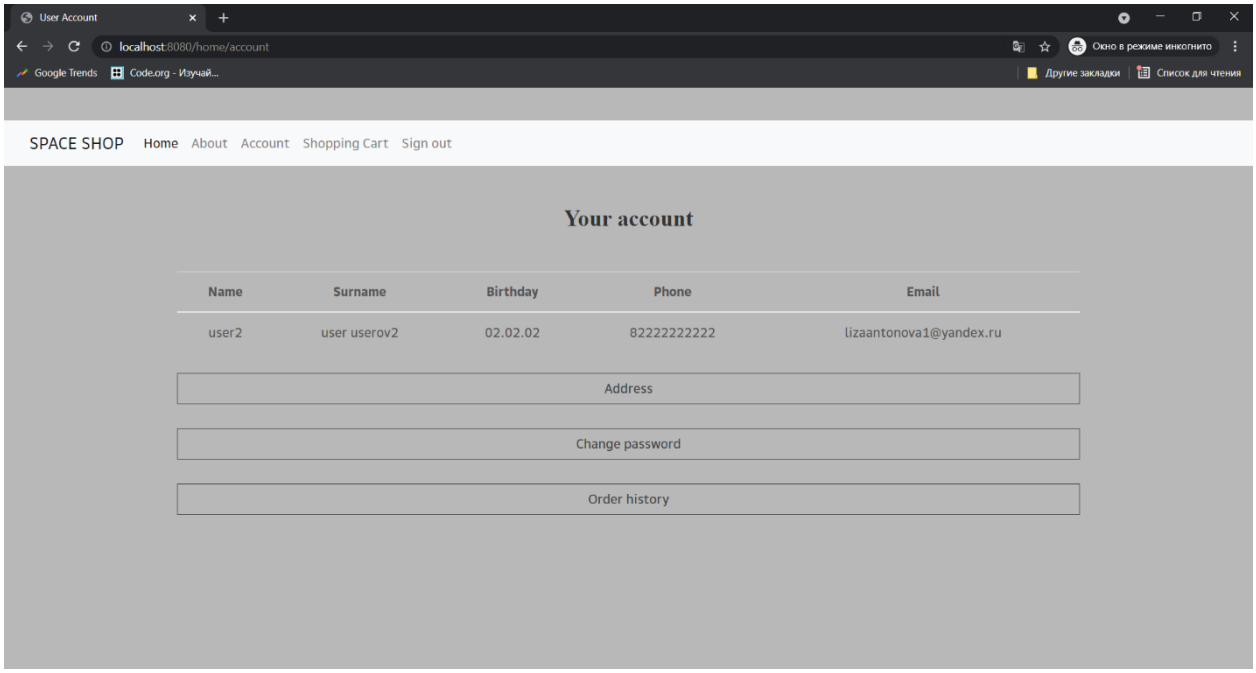
Main page



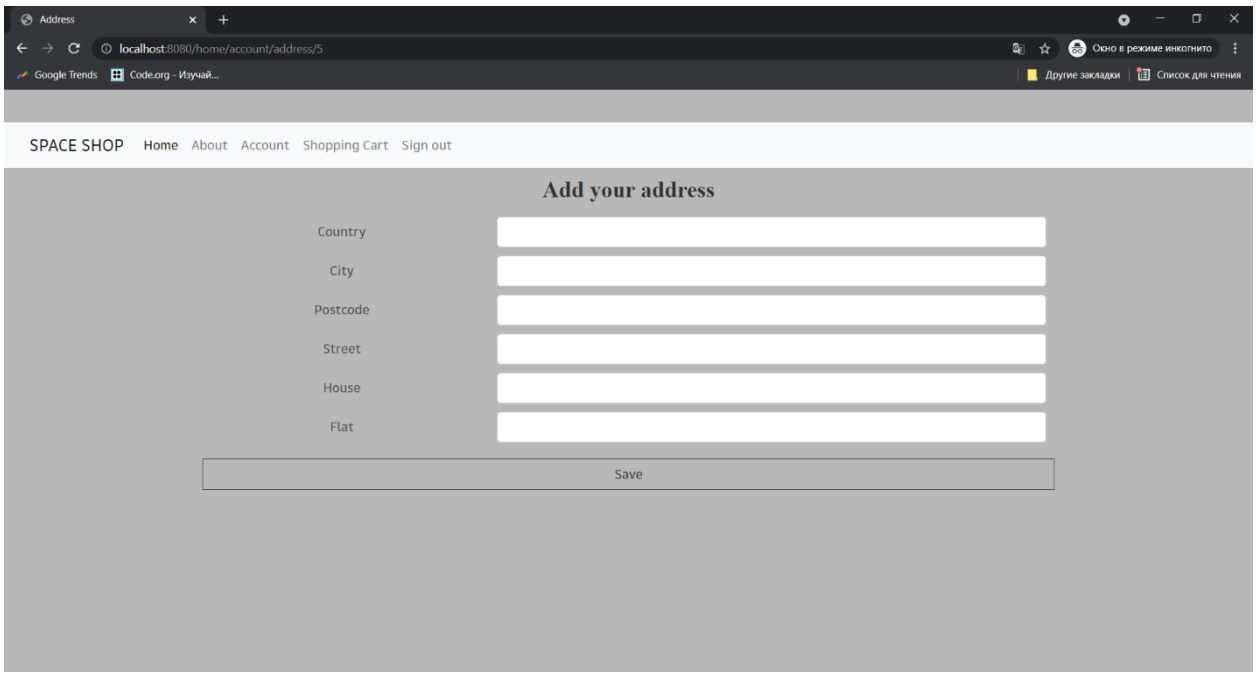
Registration form



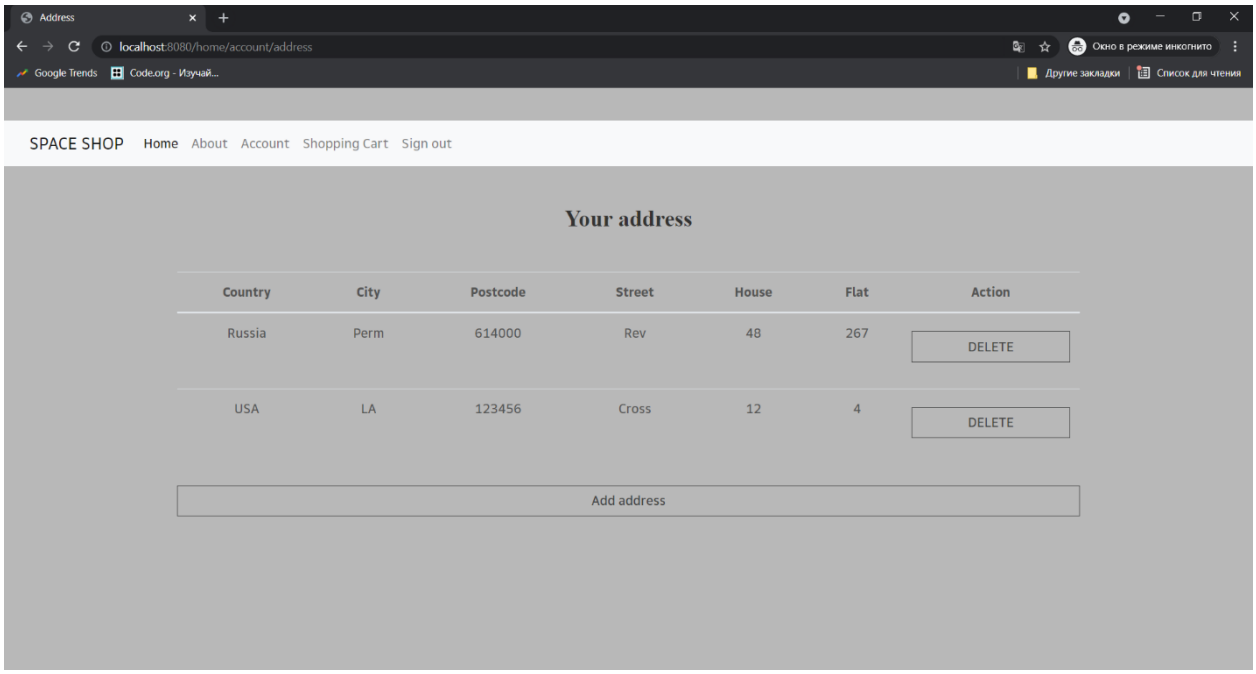
Login form



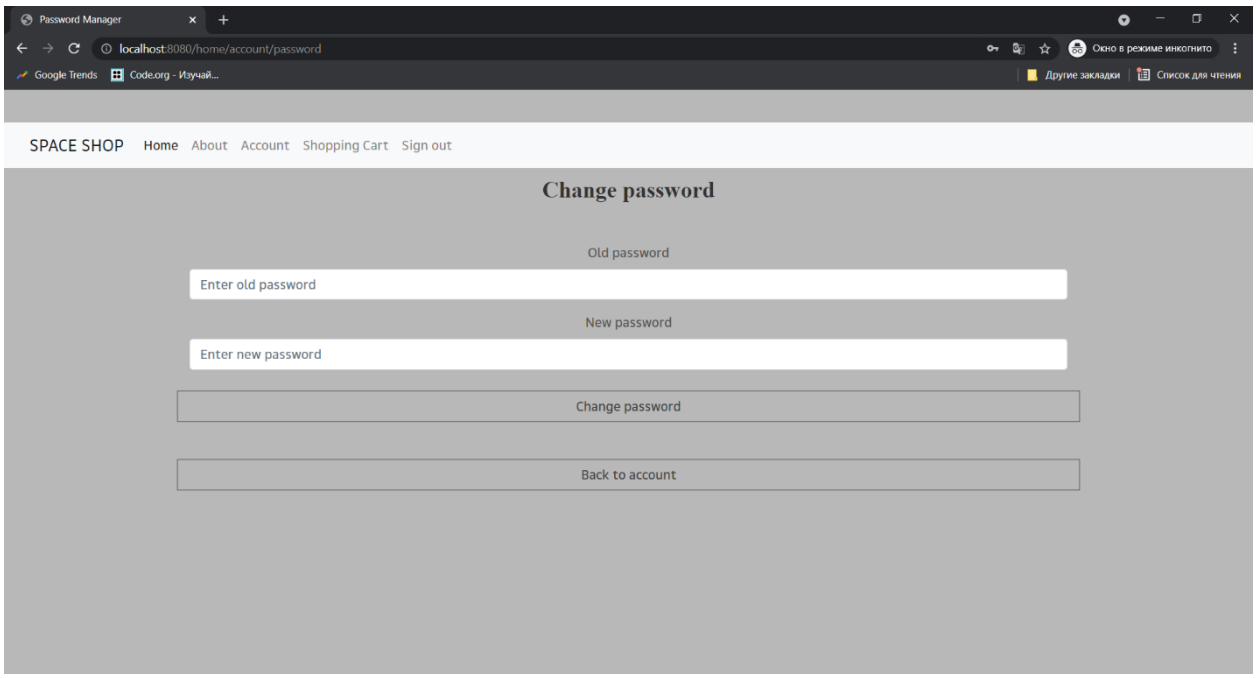
User account



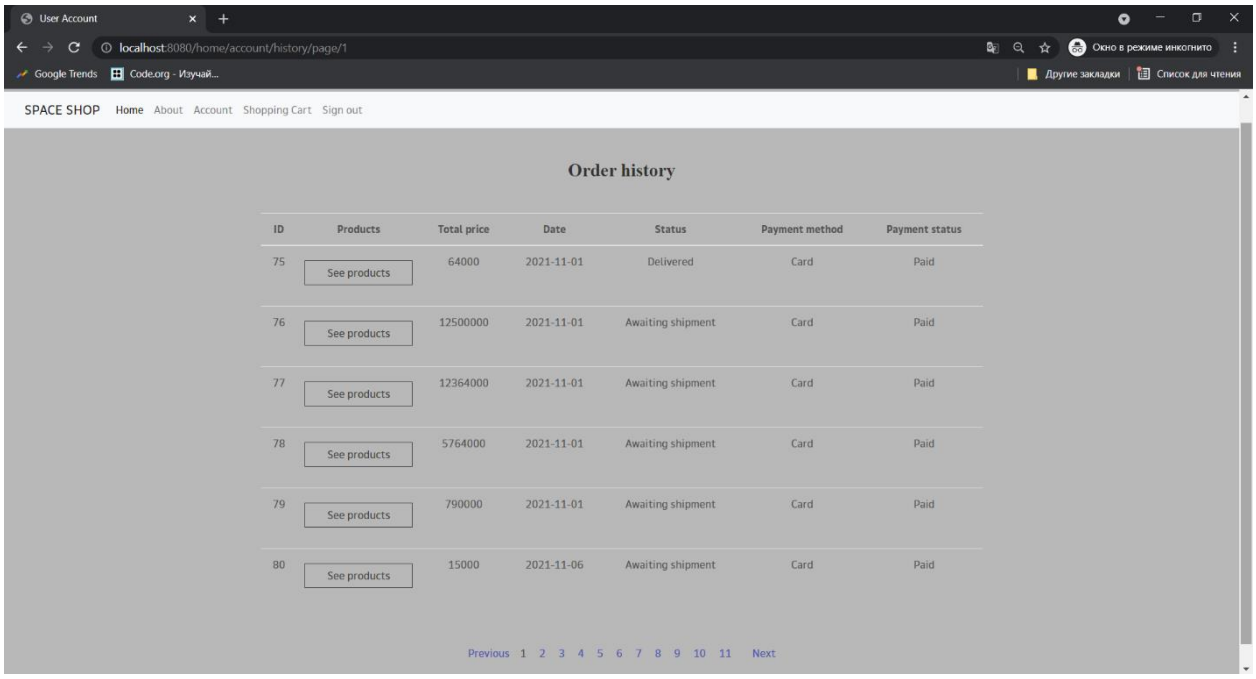
New address form



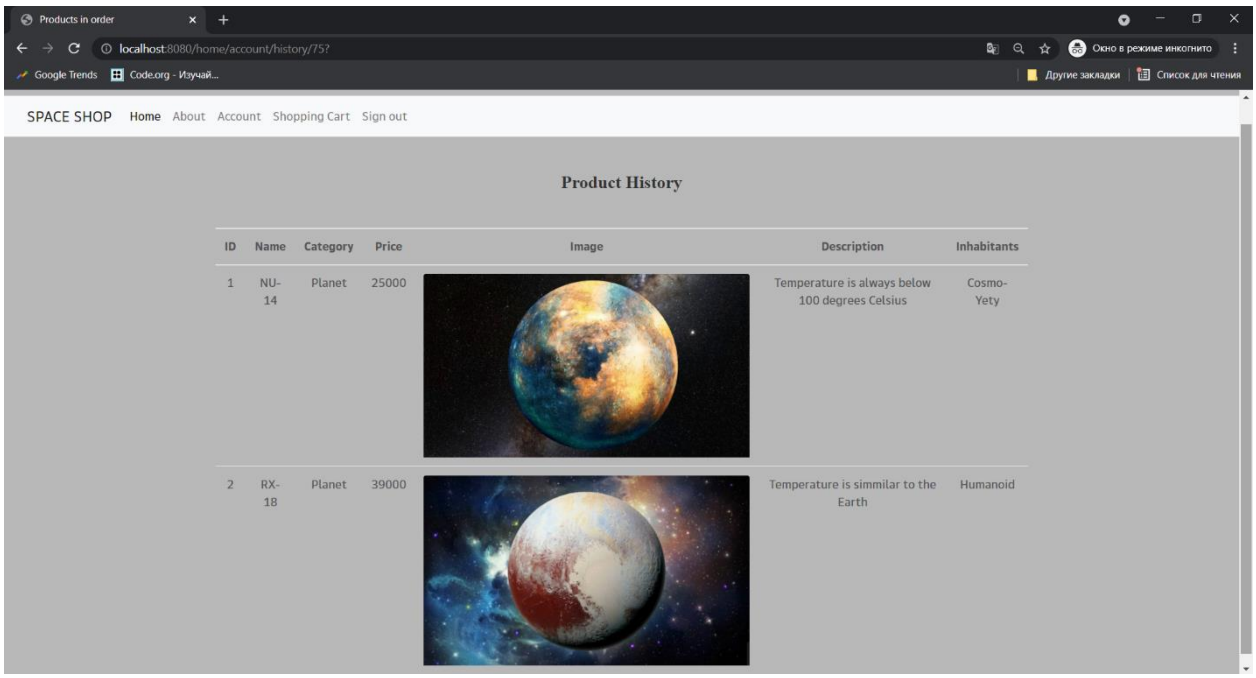
User addresses



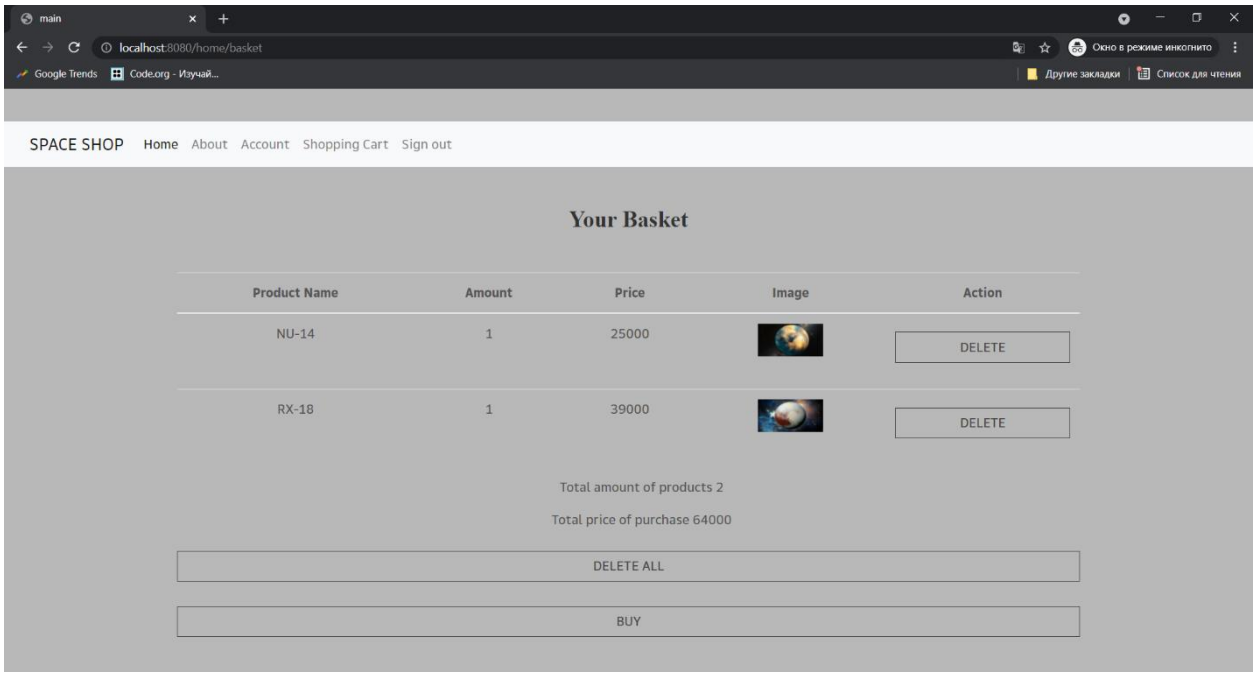
Password manager



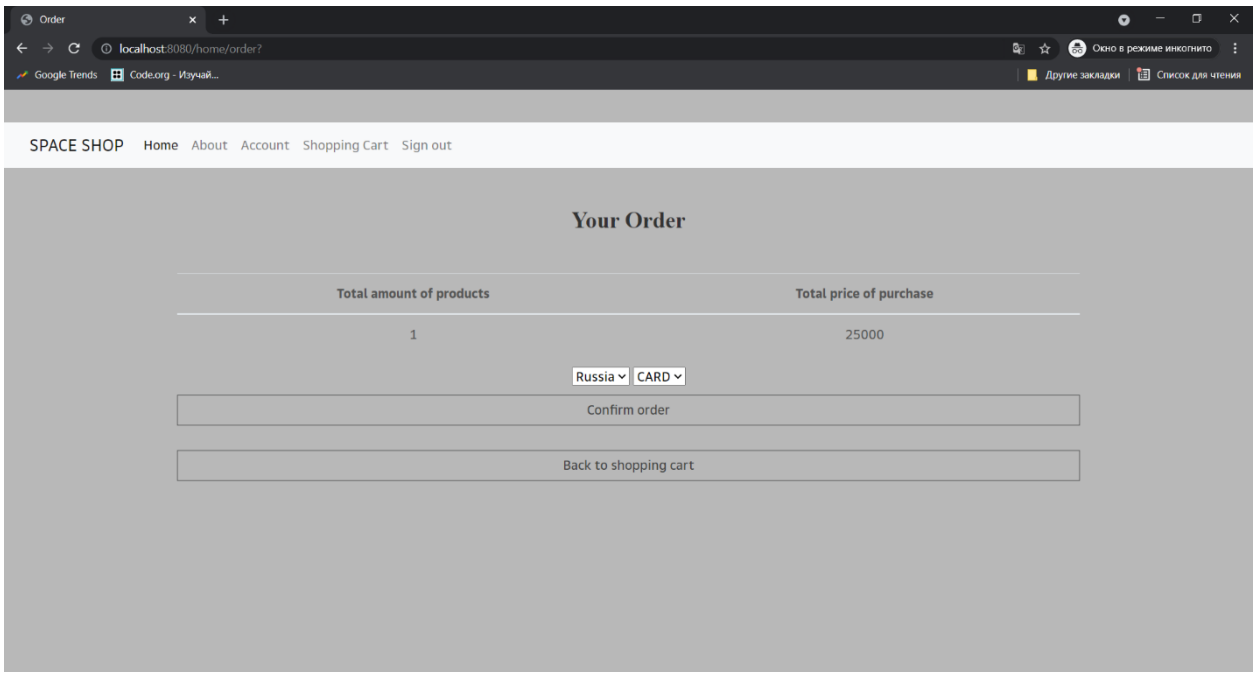
Order history



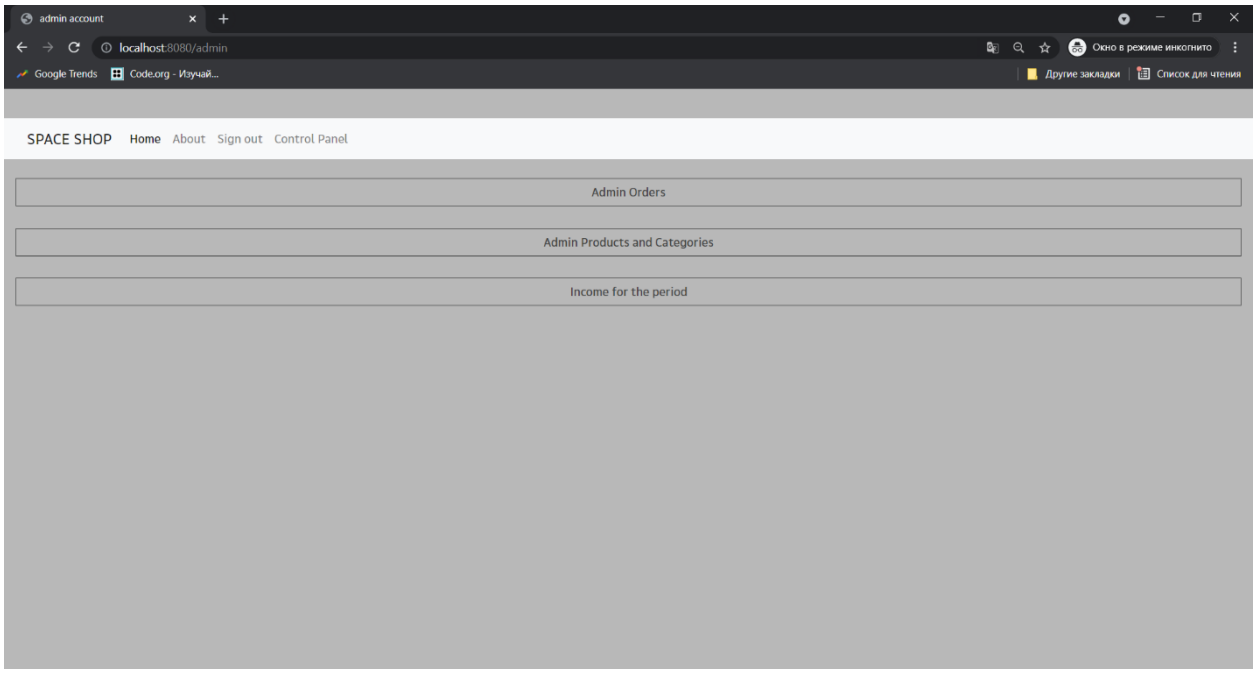
History of products in order



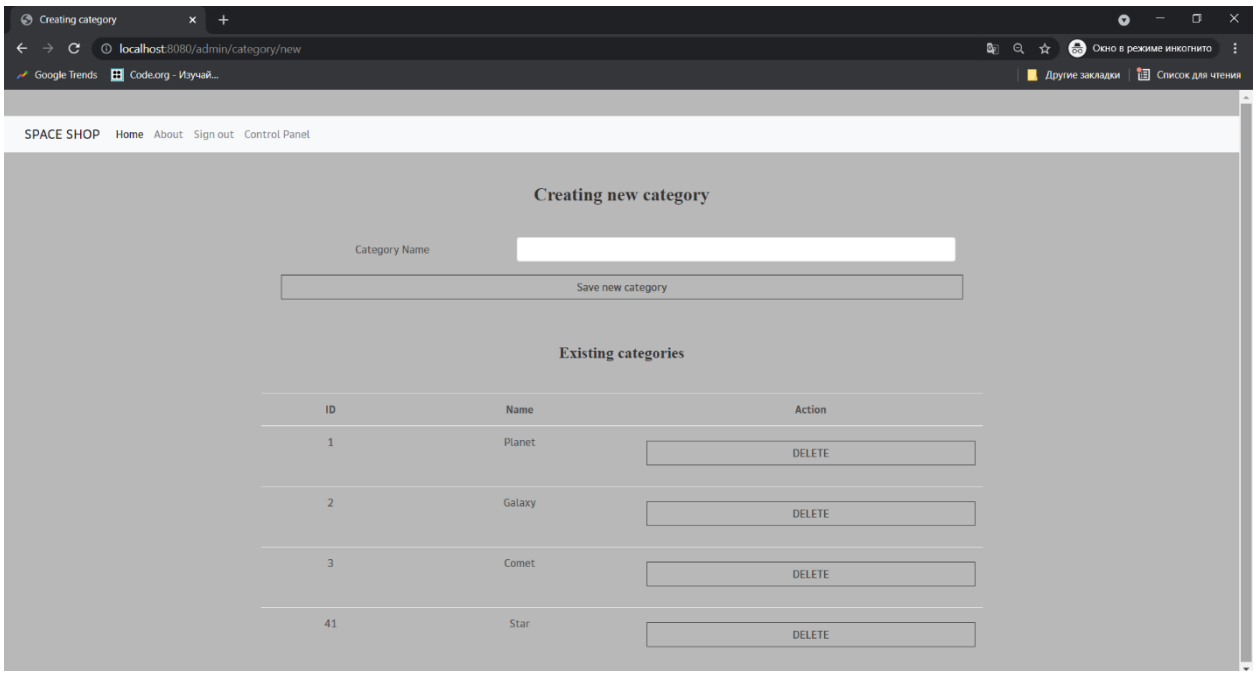
Shopping cart



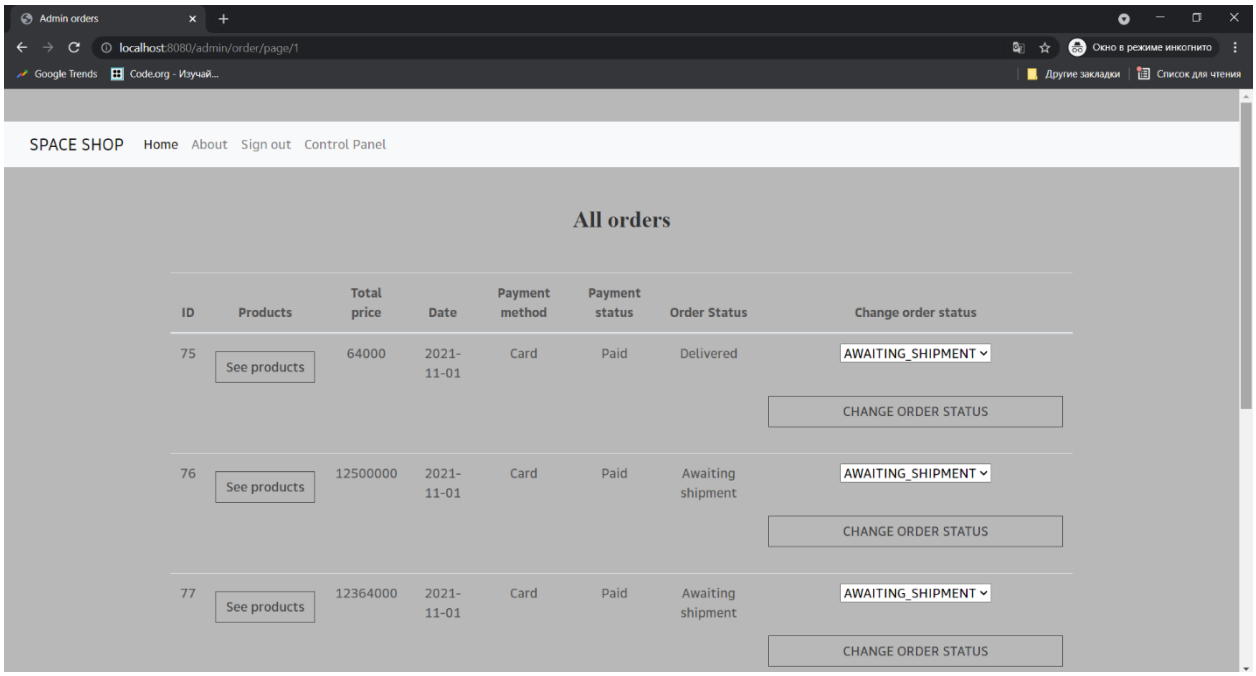
Order confirmation page



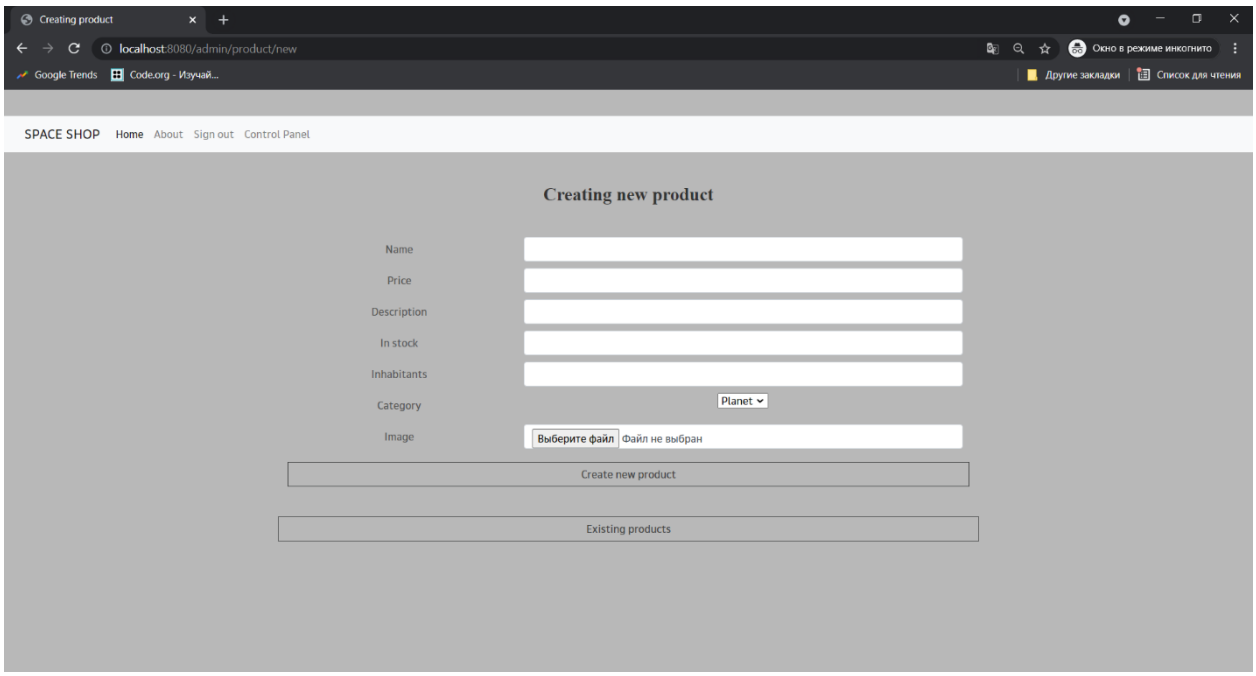
Admin control panel



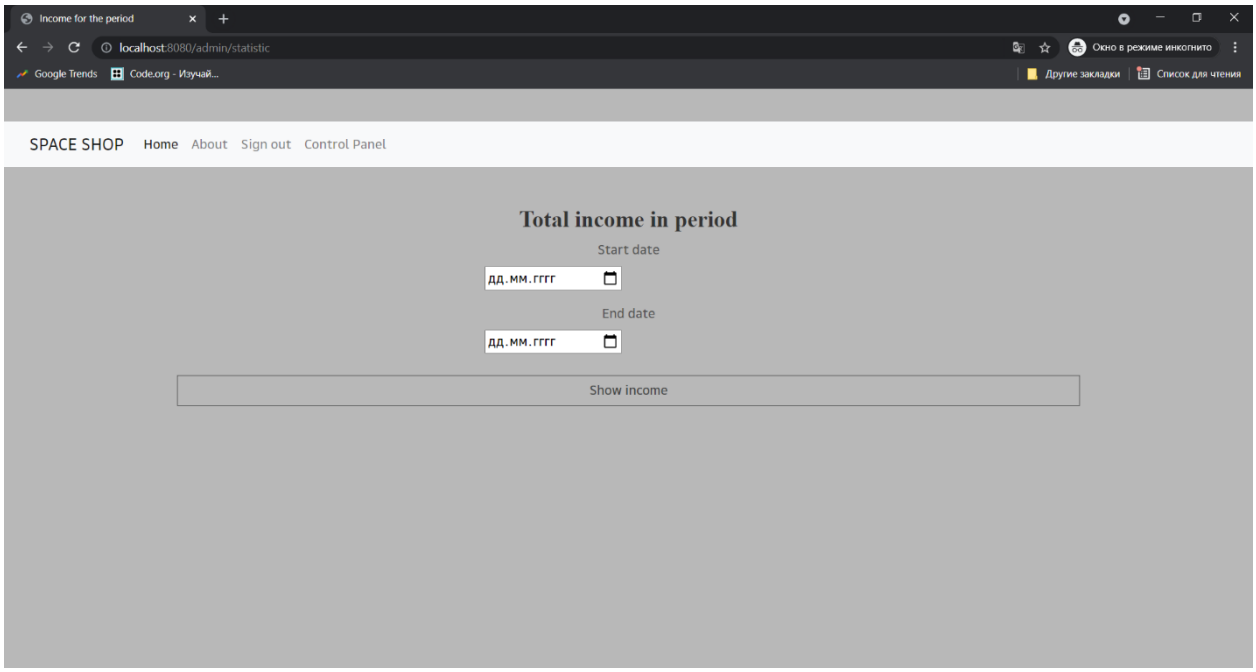
Category manager



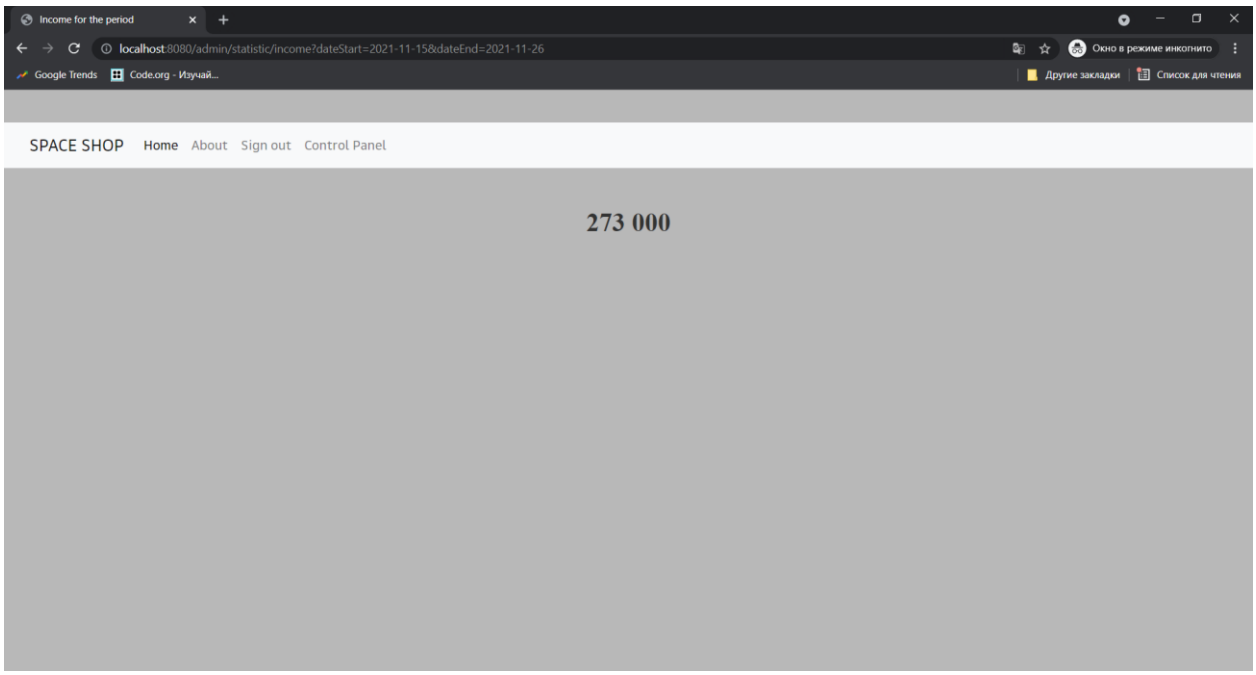
Order manager



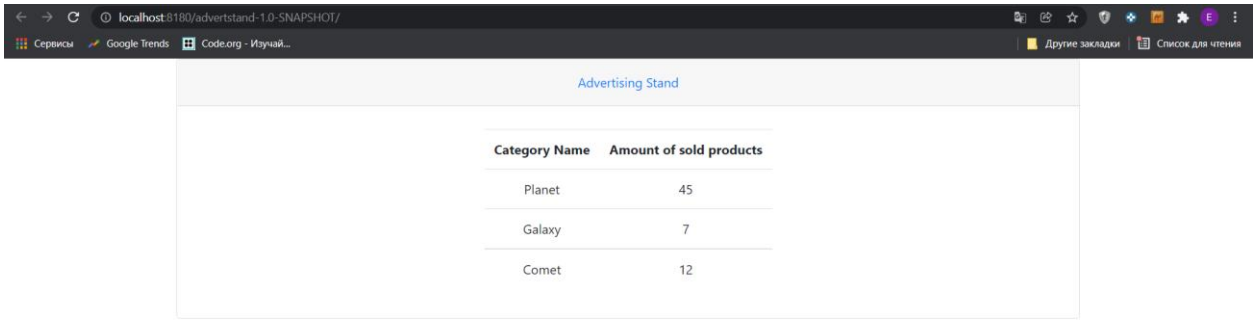
Product manager



Income for the period of time



Income



Advertising stand with top categories

Unit tests

Unit tests are created for all classes of service layer except Email service class. All test classes use Mockito library to define the behavior of used repositories in services.

List of Junit test classes for Service layer:

1) UserServiceTest
2) CategoryServiceTest
3) AddressServiceTest
4) OrderServiceTest
5) ProductServiceTest
6) BasketProductServiceTest
7) ProductInOrderServiceTest
8) TopCategoryServiceTest

Logging configuration

Logging is provided by Slf4j library. Properties below are used for logger:

logging.level.org.springframework.web=INFO

logging.level.org.hibernate=ERROR

There are examples from logging file:

2021-11-26 21:26:11.365 INFO 17960 --- [http-nio-8080-exec-9] p.s.s.impl.BasketProductServiceImpl : product NU-14 added to the basket

2021-11-26 21:26:53.586 ERROR 17960 --- [http-nio-8080-exec-5] org.thymeleaf.TemplateEngine : [THYMELEAF][http-nio-8080-exec-5] Exception processing template "main"

2021-11-26 21:25:26.887 INFO 17960 --- [http-nio-8080-exec-6] p.s.service.impl.OrderServiceImpl : income for the period is found = 273000

The second application uses Log4j as a logger. Properties are shown below:

log4j.rootLogger=INFO, file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=C:\\java_projects\\sources\\logs\\advert.log

log4j.appender.file.MaxFileSize=10MB

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

There are examples from logging file:

2021-11-14 17:01:40 INFO Consumer:46 - webSocket "update"

2021-11-16 19:20:58 INFO Consumer:35 - Receive message

2021-11-16 19:29:13 INFO Consumer:41 - Received "update"

2021-11-16 19:29:13 INFO stdout:71 - "update"

Deployment

Application uses Spring Boot building process. Spring Boot automatically builds, starts Tomcat server and deploys application.

The second application uses IDE building process. IDE automatically builds, starts Wildfly server and deploys application.

Application improvements

In further releases of the application, next improvements and features are planned:

- Different roles for employees: courier, helpdesk etc.
- Add discount system with promocodes
- Add helpdesk chat service
- Deploy on Heroku