МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

ОТЧЕТ

по лабораторной работе №5
Дисциплина «Технологии сетевого программирования»

Выполнили студенты
группы 6301-010302D
Жданова В.И.
Русяева С.П.

САМАРА 2025

Тема проекта: приложение для интернет магазина косметики.

Используемые технологии:

- Backend: Spring Boot (Java);
- Frontend: Thymeleaf, Bootstrap;
- База данных: PostgreSQL;
- Миграции: Flyway;
- Контейнеризация: Docker.

Приложение развернуто с использованием Docker и состоит из двух взаимосвязанных контейнеров:

- webapp – Spring Boot-приложение (порты 8080:8080)
- postgres_db – База данных (порты 5433:5432)

Файл docker-compose.yml:

```
version: '3.8'

services:
  postgres:
    image: postgres:15-alpine
    container_name: postgres_db
    env_file: .env
    environment:
      POSTGRES_DB: ${DATABASE_NAME}
      POSTGRES_USER: ${DATABASE_USERNAME}
      POSTGRES_PASSWORD: ${DATABASE_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5433:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${DATABASE_USERNAME} -d
${DATABASE_NAME}"]
      interval: 5s
      timeout: 5s
      retries: 5
    restart: unless-stopped

  webapp:
    build:
```

```
    context: .
    dockerfile: Dockerfile
  container_name: webapp
  depends_on:
    postgres:
      condition: service_healthy
  env_file: .env
  environment:
    SPRING_DATASOURCE_URL: jdbc:postgresql://postgres:5432/${DATABASE_NAME}
    SPRING_DATASOURCE_USERNAME: ${DATABASE_USERNAME}
    SPRING_DATASOURCE_PASSWORD: ${DATABASE_PASSWORD}
    SPRING_JPA_HIBERNATE_DDL_AUTO: validate
    SPRING_FLYWAY_URL: jdbc:postgresql://postgres:5432/${DATABASE_NAME}
    SPRING_FLYWAY_USER: ${DATABASE_USERNAME}
    SPRING_FLYWAY_PASSWORD: ${DATABASE_PASSWORD}
  ports:
    - "8080:8080"
  restart: unless-stopped

volumes:
 postgres_data:
```

База данных настроена через переменные окружения (берутся из .env).

Приложение реализует ролевую модель доступа с двумя уровнями прав: пользователь и администратор.

Пользователь:

- домашняя страница приложения;
- страница каталога:
    - просмотр каталога товаров;
    - сортировка товаров (по цене, по названию);
    - фильтрация товаров (по категории товара, по брендам, по цене);
    - добавление товара в корзину;
- страница корзины:
    - очистить корзину;
    - оформить заказ;
    - увеличить/уменьшить количество позиций в корзине;
    - удалить позицию из корзины;

- страница заказов:
    - просмотр истории своих заказов;
    - просмотр подробной информации о заказе (товары в заказе);
- страница профиля:
    - удалить профиль;
    - редактировать профиль (изменить имя пользователя, номер телефона);
    - изменить пароль.

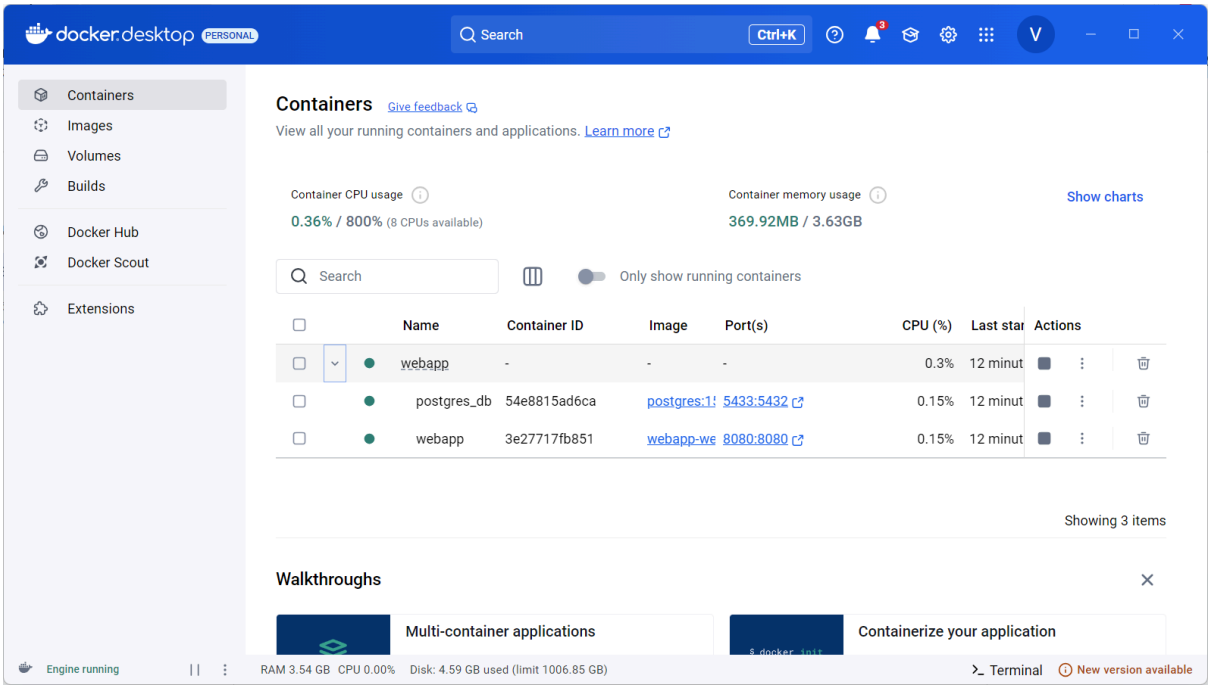
Администратор:

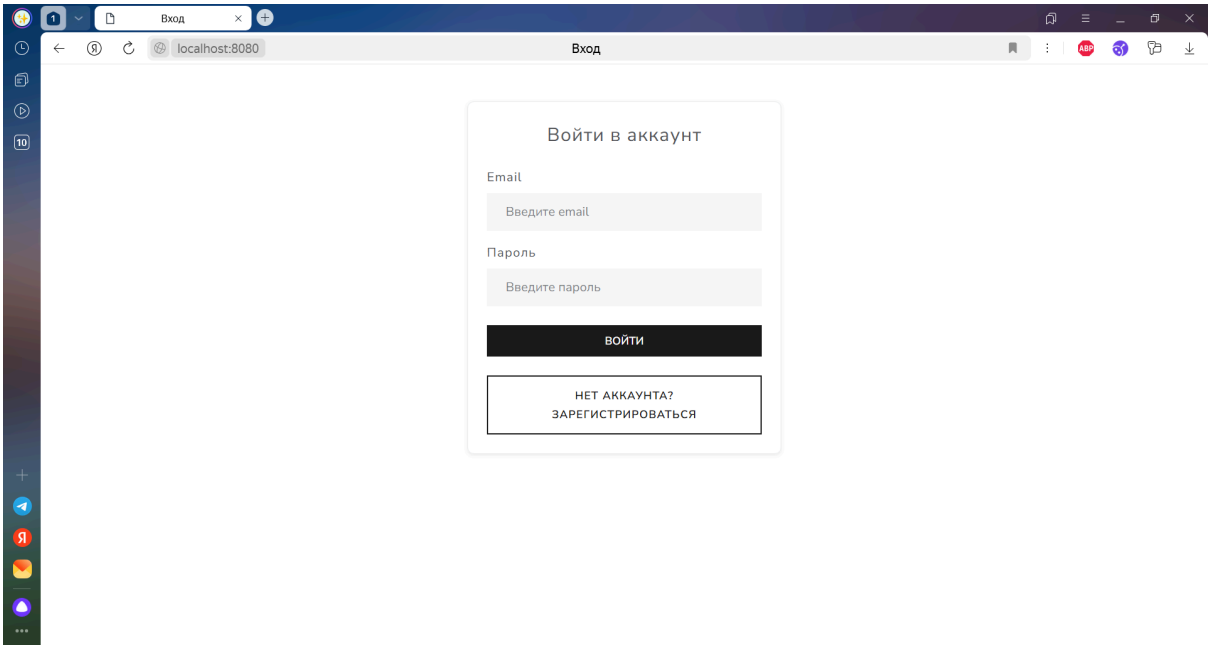
- домашняя страница приложения;
- страница каталога:
    - просмотр каталога товаров;
    - сортировка товаров (по цене, по названию);
    - фильтрация товаров (по категории товара, по брендам, по цене);
    - редактировать товар (изменить название, описание, бренд (добавить новый бренд), цену);
    - удалить товар;
- страница брендов:
    - удалить бренд;
    - редактировать бренд (изменить название, страну);
    - добавить новый бренд;
- страница заказов
    - просмотр истории заказов всех пользователей;
    - фильтрация заказов (по статусу, по сумме, по дате заказа);
    - сортировка (по id, дате, сумме заказа);
    - просмотр подробной информации о заказе (товары в заказе);
    - изменение статуса заказа.

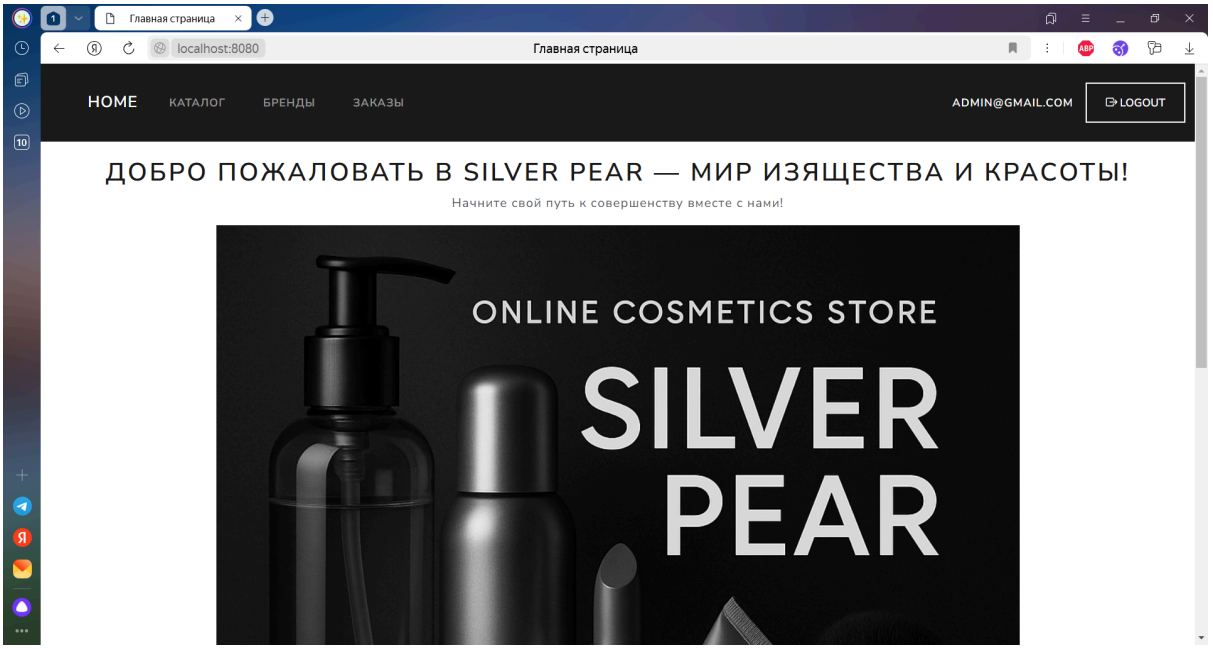
Техническая реализация:

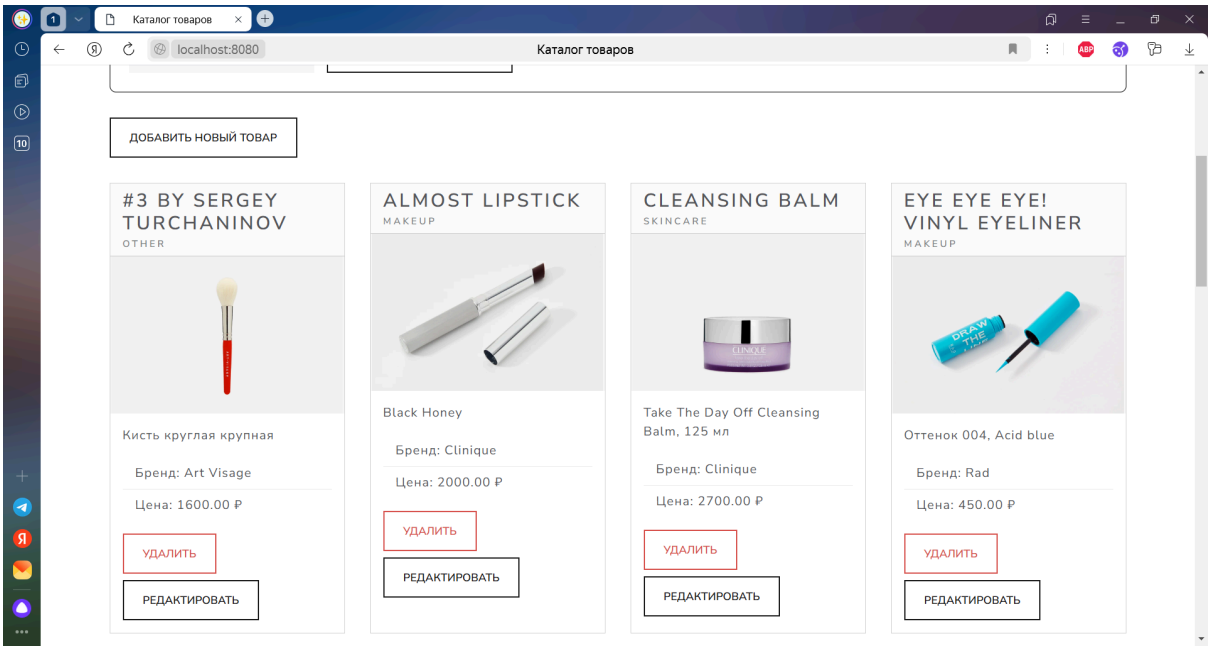
- Spring Security для аутентификации и авторизации;
- роли хранятся в БД в таблице users (поле role);
- endpoints защищены аннотациями @PreAuthorize.

Скриншоты работающего приложения:


docker desktop
PERSONAL
Search
Ctrl+K
Containers
Images
Volumes
Builds
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
0.36% / 800% (8 CPUs available)
Container memory usage
369.92MB / 3.63GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Last star
Actions
webapp
postgres_db
54e8815ad6ca
postgres:1
5433:5432
0.15%
12 minut
webapp
3e27717fb851
webapp-we
8080:8080
0.15%
12 minut
Showing 3 items
Walkthroughs
Multi-container applications
Containerize your application
Engine running
RAM 3.54 GB CPU 0.00% Disk: 4.59 GB used (limit 1006.85 GB)
Terminal
New version available


Вход
localhost:8080
Вход
Войти в аккаунт
Email
Введите email
Пароль
Введите пароль
ВОЙТИ
НЕТ АККАУНТА?
ЗАРЕГИСТРИРОВАТЬСЯ





Описание API:

Аутентификация

Регистрация нового пользователя
Метод: POST
Путь: /api/auth/register
Параметры: email, password, username, phone (все строки)
Возвращает: перенаправление на /api/shop/home
Доступ: всем

Аутентификация пользователя
Метод: POST
Путь: /api/auth/login
Параметры: email, password
Возвращает: перенаправление на /api/shop/home
Доступ: всем

Обновление токена доступа
Метод: POST
Путь: /api/auth/refresh
Параметры: RefreshTokenDto в теле запроса
Возвращает: ResponseEntity<JwtResponseDto>
Доступ: всем

Выход из системы
Метод: POST
Путь: /api/auth/logout
Параметры: куки access_token и refresh_token
Возвращает: перенаправление на страницу входа
Доступ: всем

Бренды

Получение всех брендов
Метод: GET
Путь: /api/shop/brands
Возвращает: HTML-страницу
Доступ: ADMIN

Создание нового бренда
Метод: POST
Путь: /api/shop/brands
Параметры: BrandDto в теле запроса
Возвращает: ResponseEntity<Brand>
Доступ: ADMIN

Обновление бренда
Метод: PUT
Путь: /api/shop/brands/{brandId}
Параметры: BrandDto в теле, brandId в URL
Возвращает: void
Доступ: ADMIN
Удаление бренда
Метод: DELETE

Путь: /api/shop/brands/{brandId}
Параметры: brandId в URL
Возвращает: void
Доступ: ADMIN

Корзина

Создание заказа из корзины
Метод: POST
Путь: /api/shop/cart/createOrder
Возвращает: void
Доступ: USER

Получение содержимого корзины
Метод: GET
Путь: /api/shop/cart/me
Возвращает: HTML-страницу
Доступ: USER

Очистка корзины
Метод: DELETE
Путь: /api/shop/cart/me
Возвращает: void
Доступ: USER

Элементы корзины

Увеличение количества товара
Метод: PUT
Путь: /api/shop/cartItem/{cartItemId}/inc
Параметры: cartItemId в URL
Возвращает: ResponseEntity<CartItem>
Доступ: USER

Уменьшение количества товара
Метод: PUT
Путь: /api/shop/cartItem/{cartItemId}/dec
Параметры: cartItemId в URL
Возвращает: ResponseEntity<CartItem>
Доступ: USER
Удаление товара из корзины
Метод: DELETE
Путь: /api/shop/cartItem/{cartItemId}
Параметры: cartItemId в URL

Возвращает: ResponseEntity<?>
Доступ: USER

Заказы

Получение отфильтрованных заказов
Метод: GET
Путь: /api/shop/orders
Параметры: sortBy, sortDirection, statuses, startDate, endDate, minPrice, maxPrice
Возвращает: HTML-страницу
Доступ: ADMIN

Получение заказов пользователя
Метод: GET
Путь: /api/shop/orders/me
Возвращает: HTML-страницу
Доступ: USER

Получение товаров в заказе
Метод: GET
Путь: /api/shop/orders/order/{orderId}
Параметры: orderId в URL
Возвращает: ResponseEntity<List<ItemResponseDto>>
Доступ: ADMIN

Получение товаров в заказе пользователя
Метод: GET
Путь: /api/shop/orders/me/{orderId}
Параметры: orderId в URL
Возвращает: ResponseEntity<List<ItemResponseDto>>
Доступ: USER

Обновление статуса заказа
Метод: PATCH
Путь: /api/shop/orders/{orderId}
Параметры: orderId в URL, status в теле
Возвращает: ResponseEntity<?>
Доступ: ADMIN

Товары

Получение отфильтрованных товаров
Метод: GET
Путь: /api/shop/products
Параметры: sortBy, sortDirection, categories, minPrice, maxPrice, brandNames

Возвращает: HTML-страницу
Доступ: всем

Получение товара по ID
Метод: GET
Путь: /api/shop/products/{productId}
Параметры: productId в URL
Возвращает: ResponseEntity<ProductResponseDto>
Доступ: всем

Создание товара
Метод: POST
Путь: /api/shop/products
Параметры: ProductDto в теле
Возвращает: ResponseEntity<Product>
Доступ: ADMIN

Добавление товара в корзину
Метод: POST
Путь: /api/shop/products/{productId}
Параметры: productId в URL
Возвращает: void
Доступ: USER

Обновление товара
Метод: PUT
Путь: /api/shop/products/{productId}
Параметры: productId в URL, name, description, price
Возвращает: void
Доступ: ADMIN

Удаление товара
Метод: DELETE
Путь: /api/shop/products/{productId}
Параметры: productId в URL
Возвращает: void
Доступ: ADMIN

Пользователи

Получение всех пользователей
Метод: GET
Путь: /api/shop/users

Возвращает: ResponseEntity<List<Users>>
Доступ: ADMIN

Получение профиля пользователя
Метод: GET
Путь: /api/shop/users/me
Возвращает: HTML-страницу
Доступ: USER

Обновление данных пользователя
Метод: PUT
Путь: /api/shop/users/me/edit
Параметры: UserUpdateDto в теле
Возвращает: ResponseEntity<Users>
Доступ: USER

Обновление пароля
Метод: PATCH
Путь: /api/shop/users/me/edit/password
Параметры: PasswordDto в теле
Возвращает: ResponseEntity<?>
Доступ: USER

Удаление пользователя
Метод: DELETE
Путь: /api/shop/users/me
Возвращает: ResponseEntity<?>
Доступ: USER

Страницы

Страница входа
Метод: GET
Путь: /api/auth/login
Возвращает: HTML-страницу
Доступ: всем

Страница регистрации
Метод: GET
Путь: /api/auth/register
Возвращает: HTML-страницу
Доступ: всем
Главная страница
Метод: GET

Путь: /api/shop/home
Возвращает: HTML-страницу
Доступ: всем

Ошибки

Страница ошибки 403
Метод: GET
Путь: /api/shop/error403
Возвращает: HTML-страницу
Доступ: всем

Страница ошибки 404
Метод: GET
Путь: /api/shop/error404
Возвращает: HTML-страницу
Доступ: всем

Структура проекта:

Конфигурации:

- JwtProvider – генерация и валидация JWT-токенов.
- AdminInitializer - создание администратора.
- JwtFilter – фильтр для проверки JWT в запросах.
- SecurityConfig – настройка Spring Security (аутентификация, авторизация).
- CustomUserDetails – кастомная реализация UserDetails для Spring Security.

Контроллеры:

Обрабатывают HTTP-запросы и возвращают ответы (JSON/HTML).

- AuthController – регистрация, вход, выход, обновление токена.
- BrandController – CRUD для брендов.
- CartController – управление корзиной.
- CartItemController – работа с товарами в корзине.
- OrdersController – работа с заказами (фильтрация, статусы).
- ProductController – CRUD для товаров.
- UsersController – управление профилем пользователя.
- ViewController – рендеринг HTML-страниц.
- CustomErrorController – обработка ошибок 403/404.

- GlobalControllerAdvice – глобальные атрибуты для Thymeleaf.

Data Transfer Objects:

Используются для передачи данных между слоями.

- AuthDto – данные для входа (email, password).
- ItemResponseDto – информация о товаре в заказе.
- JwtResponseDto – ответ с JWT (accessToken, refreshToken).
- ProductDto, BrandDto, CartDto, OrdersDto, UsersDto – DTO для создания/обновления сущностей.
- PasswordDto – обновление пароля.
- ProductResponseDto – информация о товаре.
- ProductUpdateDto – обновление информации о товаре.
- UserUpdateDto – обновление информации о пользователе.
- RefreshTokenDto – обновление refresh токена.

Сущности БД:

JPA-сущности, отображаемые в таблицы БД.

- Users – пользователи.
- Product – товары.
- Brand – бренды.
- Cart, CartItem – корзина и её элементы.
- Orders, OrderItem – заказы и их состав.
- RefreshToken – токены для обновления JWT.
- Role – виды ролей.
- ProductCategory – категории товаров.
- OrderStatus – статусы заказов.
- Country – страны производители брендов.
- BlackList – список заблокированных токенов.

Репозитории:

- UsersRepository, ProductRepository, BlackListRepository, BrandRepository, CartItemRepository, CartRepository, OrderRepository, OrderItemRepository, RefreshTokenRepository – интерфейсы Spring Data JPA для работы с БД.

Сервисы:

- AuthService – регистрация, аутентификация.
- BlackListService – добавление access токенов в черный список после выхода из системы.
- BrandService – управление брендами.
- ProductService – управление товарами.
- CartService, CartItemService – работа с корзиной.
- UsersService, CustomUserDetailsService – работа с пользователями.
- OrdersService – обработка заказов.
- RefreshTokenService – обновление JWT.

HTML-страницы (Thymeleaf):

- login.html, register.html – страницы аутентификации.
- home.html – домашняя страница приложения.
- header.html – шапка страницы с основными ссылками.
- products.html – каталог товаров.
- cart.html – корзина пользователя.
- orders.html, ordersAdmin.html – страницы заказов.
- brands.html – страница брендов.
- profile.html – профиль пользователя.