

Web-приложение для интернет-магазина косметики.

Выполнили Жданова Валерия и Русяева Софья. Группа 6301-010302D

Основные функции:

1. Каталог товаров:
 - просмотр и фильтрация товаров,
 - сортировка по цене,
 - добавление в корзину.
2. Корзина покупок:
 - добавление и удаление товаров,
 - сохранение корзины между сессиями.
3. Возможность оформления заказа
4. Личный кабинет пользователя:
 - регистрация и авторизация,
 - просмотр истории заказов.

Схема взаимодействия:

1. Клиент (Frontend):

- Веб-интерфейс, отправляет запросы к серверу через REST API.
- Получает данные от сервера и отображает их пользователю.

2. Сервер (Backend):

- Написан на Java с использованием Spring Boot.
- Обработывает запросы от клиента.
- Взаимодействует с базой данных для получения и сохранения данных.
- Возвращает данные клиенту в формате JSON.

3. База данных:

- Хранит данные о товарах, пользователях, заказах и корзинах.

Определение структуры API, документация

The image shows a Visual Studio Code editor with two panels. The left panel displays a file named `test.md` containing an OpenAPI specification for a user management API. The right panel shows the rendered preview of this specification, titled `Preview test.md`.

API Specification (test.md):

```
1 # Пользователи
2
3 ## Получить информацию о текущем пользователе
4 **Метод:** `GET`
5
6 **URL:** `/api/users/me`
7
8
9 ## Получить информацию о конкретном пользователе
10 **Метод:** `GET`
11
12 **URL:** `/api/users/{id}`
13
14
15 ## Создать нового пользователя
16 **Метод:** `POST`
17
18 **URL:** `/api/users`
19
20 ---json
21 {
22   "username": "user",
23   "email": "user@example.com",
24   "password": "123",
25   "phone": 123456789
26 }
27
28 ## Обновить информацию о пользователе
29 **Метод:** `PUT`
30
31 **URL:** `/api/users/{id}`
32
33 ---json
34 {
35   "username": "user",
36   "email": "user@example.com",
37   "password": "123",
```

API Documentation Preview (Preview test.md):

Пользователи

Получить информацию о текущем пользователе

Метод: GET

URL: `/api/users/me`

Получить информацию о конкретном пользователе

Метод: GET

URL: `/api/users/{id}`

Создать нового пользователя

Метод: POST

URL: `/api/users`

```
{
  "username": "user",
  "email": "user@example.com",
  "password": "123",
  "phone": 123456789
}
```

Обновить информацию о пользователе

Схема API пользователь:

Информация о текущем пользователе

- **Метод:** GET
- **URL:** /api/users/me

Информация о конкретном пользователе

- **Метод:** GET
- **URL:** /api/users/{id}

Создание нового пользователя

- **Метод:** POST
- **URL:** /api/users

Обновить информацию о пользователе

- **Метод:** PUT
- **URL:** /api/users/{id}

Удаление пользователя

- **Метод:** DELETE
- **URL:** /api/users/{id}

Регистрация пользователя

- **Method:** POST
- **URL:** /api/auth/register

Авторизация пользователя

- **Method:** POST
- **URL:** /api/auth/login

Схема API товары:

Список товаров

- **Method:** GET
- **URL:** /api/products

Информация о товаре

- **Method:** GET
- **URL:** /api/products/{id}

Добавление нового товара

- **Method:** POST
- **URL:** /api/products

Изменения товара

- **Method:** PUT
- **URL:** /api/products/{id}

Удаление товара

- **Method:** DELETE
- **URL:** /api/products/{id}

Схема API заказы и корзина:

Заказы:

Список заказов текущего пользователя

- **Method:** GET
- **URL:** /api/orders

Информация о заказе

- **Method:** GET
- **URL:** /api/orders/{id}

Создание заказа

- **Method:** POST
- **URL:** /api/orders

Обновление статуса заказа

- **Method:** PUT
- **URL:** /api/orders/{id}

Корзина:

Получение содержимого корзины текущего пользователя

- **Method:** GET
- **URL:** /api/cart

Добавление товара в корзину

- **Method:** POST

- **URL:** /api/cart

Изменение количества товара в корзине

- **Method:** PUT
- **URL:** /api/cart/{product_id}

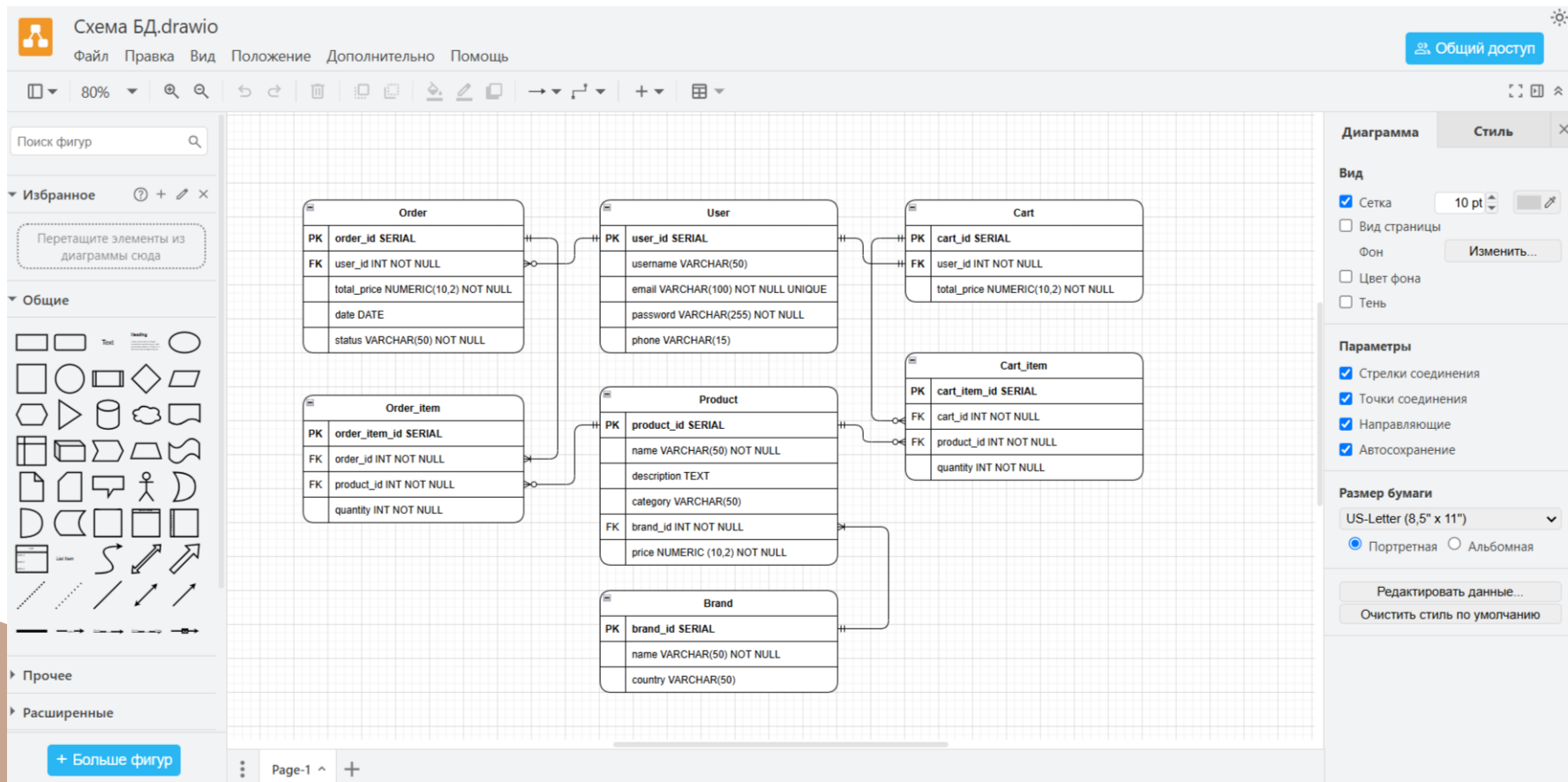
Удаление товара из корзины

- **Method:** DELETE
- **URL:** /api/cart/{product_id}




База данных будет содержать:

- Пользователь: id, имя пользователя, email, пароль, номер телефона
- Заказ: id, id пользователя, итоговая стоимость, дата, статус заказа
- Товар в заказе: id, id заказа, id товара, количество
- Корзина: id, id пользователя, итоговая стоимость
- Товар в корзине: id, id корзины, id товара, количество
- Товар: id, название, описание, категория товара, бренд, цена
- Бренд: id, название, страна

Создание схемы базы данных



Создание проекта и настройка окружения



Project
☐ Gradle - Groovy
☐ Gradle - Kotlin
☒ **Maven**

Language
☒ **Java**
☐ Kotlin
☐ Groovy

Spring Boot
☐ 3.5.0 (SNAPSHOT)
☐ 3.5.0 (M1)
☐ 3.4.3 (SNAPSHOT)
☒ **3.4.2**
☐ 3.3.10 (SNAPSHOT)
☐ 3.3.9

Project Metadata

Group

com.example

Artifact

WebApp

Name

WebApp

Description

Demo project for Spring Boot

Package name

com.example.WebApp

Packaging

☒ **Jar**
☐ War

Java

☐ 23
☐ 21
☒ **17**

Dependencies

ADD DEPENDENCIES... CTRL + B

Lombok **DEVELOPER TOOLS**

Java annotation library which helps to reduce boilerplate code.

PostgreSQL Driver **SQL**

A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Spring Web **WEB**

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.



Spring Data JPA **SQL**

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

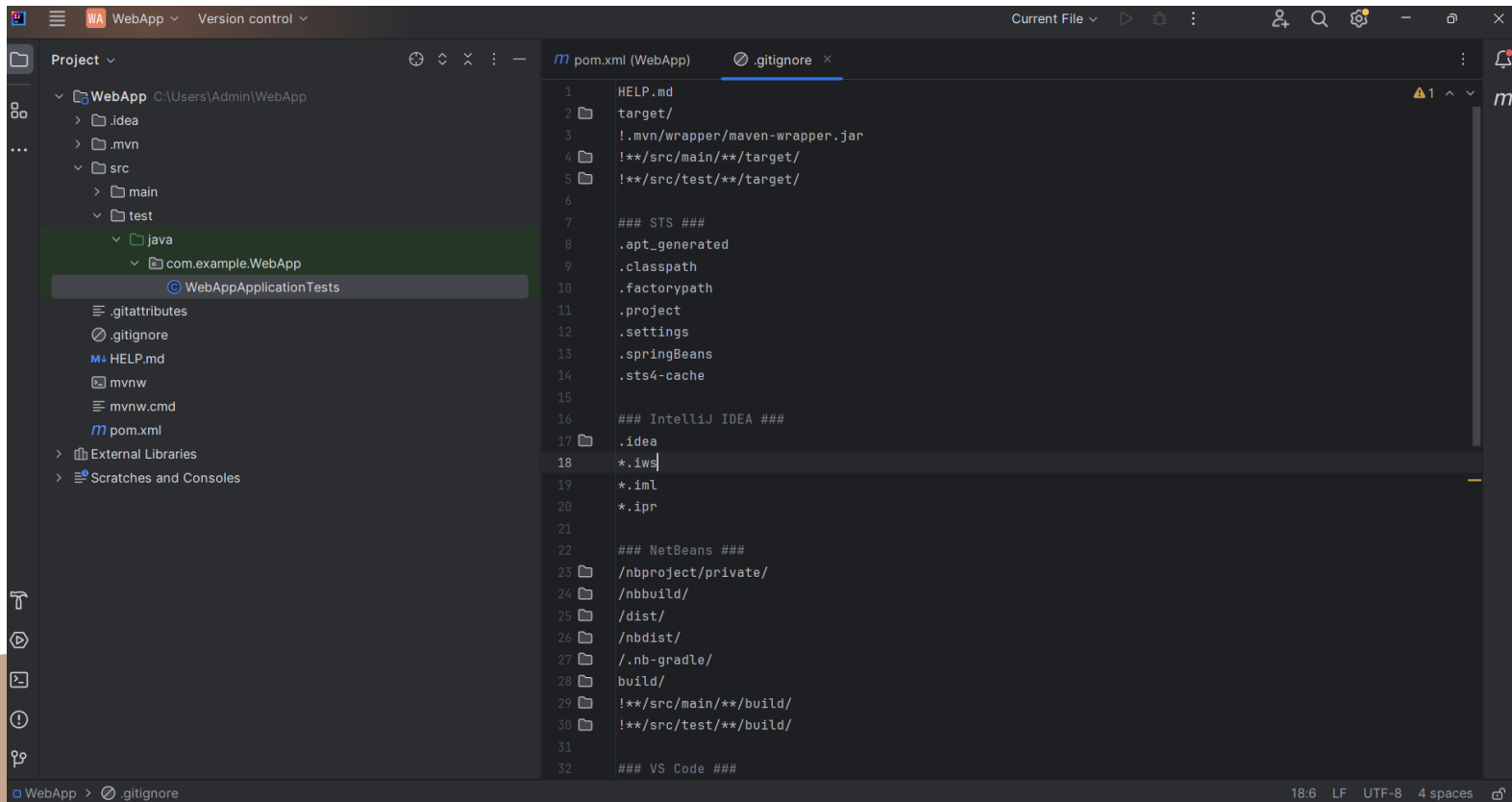
GENERATE CTRL + ⌘

EXPLORE CTRL + SPACE

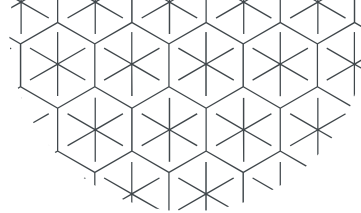
...



Добавление .gitignore для исключения ненужных файлов



Стек технологий



01 Backend:

- Язык программирования: Java
- Фреймворк: Spring Boot

02 Frontend:

- Thymeleaf (Spring)
- HTML, CSS

03 База данных:

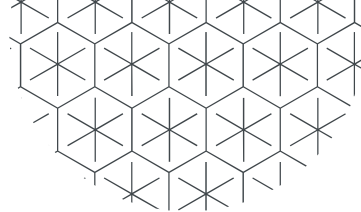
- PostgreSQL

04 Сборка:

- Maven



Стек технологий



05 Среда разработки:

- **IntelliJ IDEA**

06 Дополнительно:

- **GitHub**
- **Диаграммы и проектирование:**
UML (Draw.io)
- **Документирование:**
Markdown
- **Контейнеризация:**
Docker
- **Аутентификация:**
JWT (JSON Web Tokens)
- **Postman**





*Спасибо за
внимание!*

