Министерство науки и высшего образования Российской Федерации

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

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторным работам

Дисциплина: «Технологии сетевого программирования»

Тема «Проектирование приложения»

Выполнили: Ващенков А.Е. Залевская В.Ю.

Группа: 6302-010302D

Самара, 2025

**Введение**

Наше веб-приложение магазин одежды позволяет пользователю совершать покупки в онлайн-магазине, просматривать страницу товаров, собирать корзину из понравившихся вещей и совершать оплату. Приложение реализовано на базе стека технологий Django (Python) + PostgreSQL. Вся система контейнеризирована с помощью Docker.

Проект разрабатывался как учебно-практический, с целью демонстрации навыков в области серверной разработки, проектирования структуры приложения, настройки безопасной аутентификации, работы с базой данных, а также интеграции компонентов в контейнеризированное окружение.

**1 Проектирование приложения**

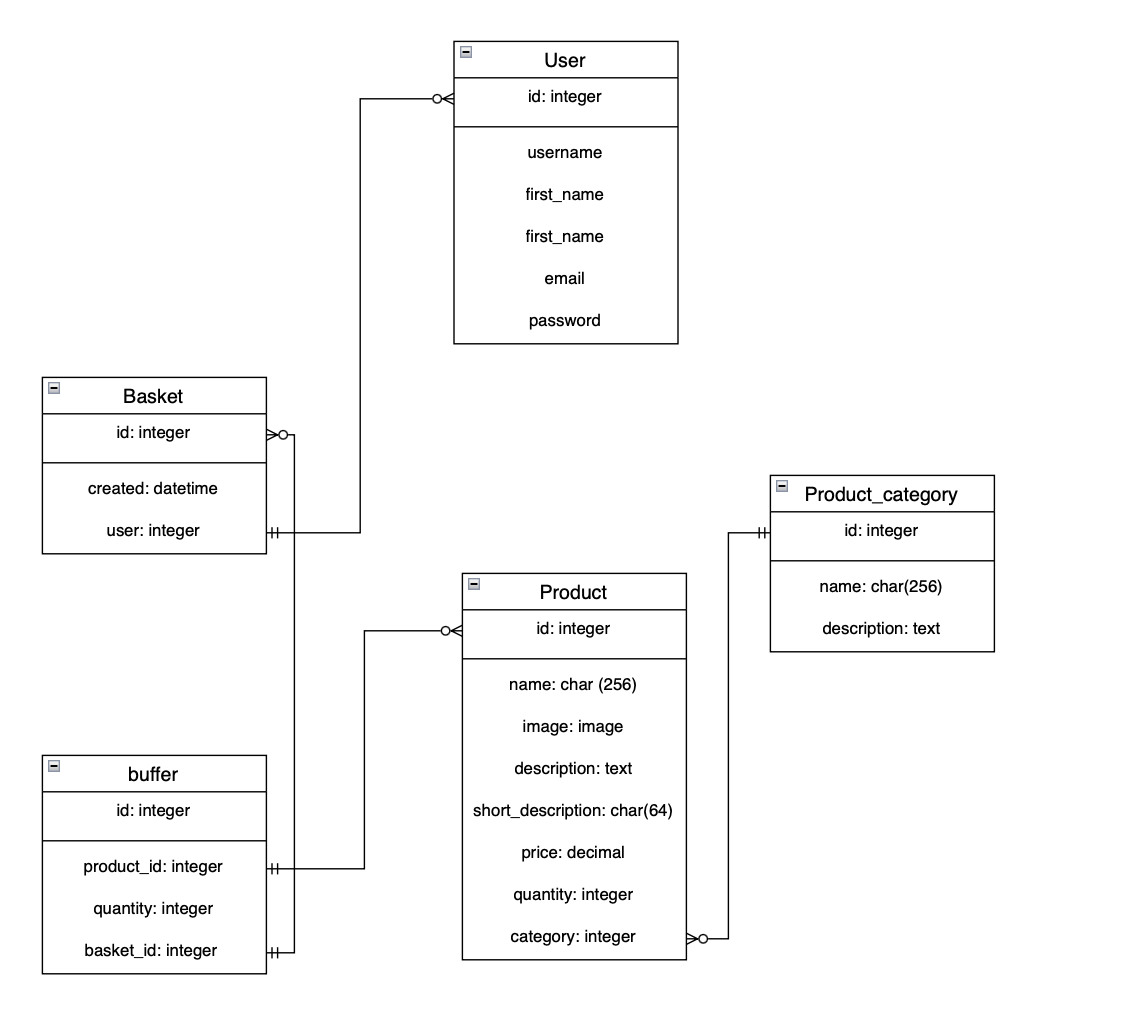
**1.1 Схема взаимодействия компонентов**

Пользователь отправляет запросы к API.

База данных хранит пользователей, товары, заказы.

* 1. **Возможности пользователя**
* Просмотр каталога;
* Регистрация/Авторизация;
* Добавление товаров в корзину;
* Оформление заказов.

**1.3 Логическая схема БД**



**2 Описание API**

#### Аутентификация и пользователи:

* **POST /auth/register** — Регистрация нового пользователя.
* **POST /auth/login** — Вход в систему.
* **POST /auth/refresh** — Обновление токена доступа.

#### Корзины (Basket):

* **POST /baskets** — Создать новую корзину.
* **GET /baskets/{id}** — Получить информацию о корзине по ID.
* **DELETE /baskets/{id}** — Удалить корзину.
* **GET /users/{userId}/baskets** — Получить все корзины пользователя.

#### Товары (Product):

* **POST /products** — Создать новый товар.
* **GET /products** — Получить список всех товаров (с фильтрами по категории: ?category={id}).
* **GET /products/{id}** — Получить информацию о товаре по ID.
* **PUT /products/{id}** — Обновить данные товара.
* **DELETE /products/{id}** — Удалить товар.

#### Категории товаров (Product\_category):

* **POST /categories** — Создать новую категорию.
* **GET /categories** — Получить список всех категорий.
* **GET /categories/{id}** — Получить информацию о категории по ID.
* **PUT /categories/{id}** — Обновить данные категории.
* **DELETE /categories/{id}** — Удалить категорию.

#### Управление товарами в корзине (buffer):

* **POST /baskets/{basketId}/items** — Добавить товар в корзину.
* **GET /baskets/{basketId}/items** — Получить список всех товаров в корзине.
* **PUT /baskets/{basketId}/items/{itemId}** — Изменить количество товара в корзине.
* **DELETE /baskets/{basketId}/items/{itemId}** — Удалить товар из корзины.
* **GET /baskets/{basketId}/total** — Рассчитать общую стоимость товаров в корзине.

**4 Разработка базы данных**

Для работы была использована база данных Postgres, а для управления PgAdmin.

Для разработки базы данных мы использовали Django ORM инструмент, который позволяет взаимодействовать с базой данных (чтение, обновление, удаление) не через написание не SQL запросов, а Python-классы.

Из-за того, что работа была построена на фреймворке Django, необходимо было настроить миграции, которые позволили управлять изменениями в структуре БД.



Рисунок 1 — Пример Python класса для объекта продукт.



Рисунок 2 — Миграция для этого объекта.

**5 Разработка пользовательского интерфейса**

Макеты страницы были созданы с помощью шаблона Django Templates и разметки HTML.

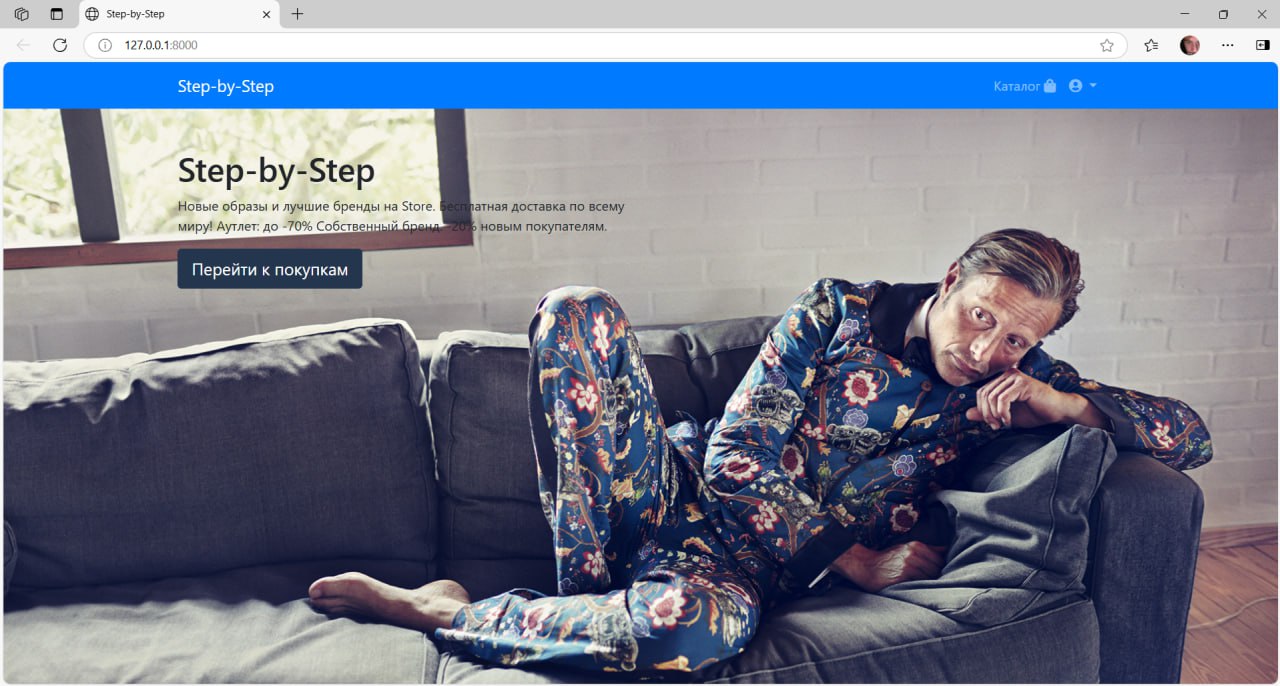


Рисунок 1– Главная страница.

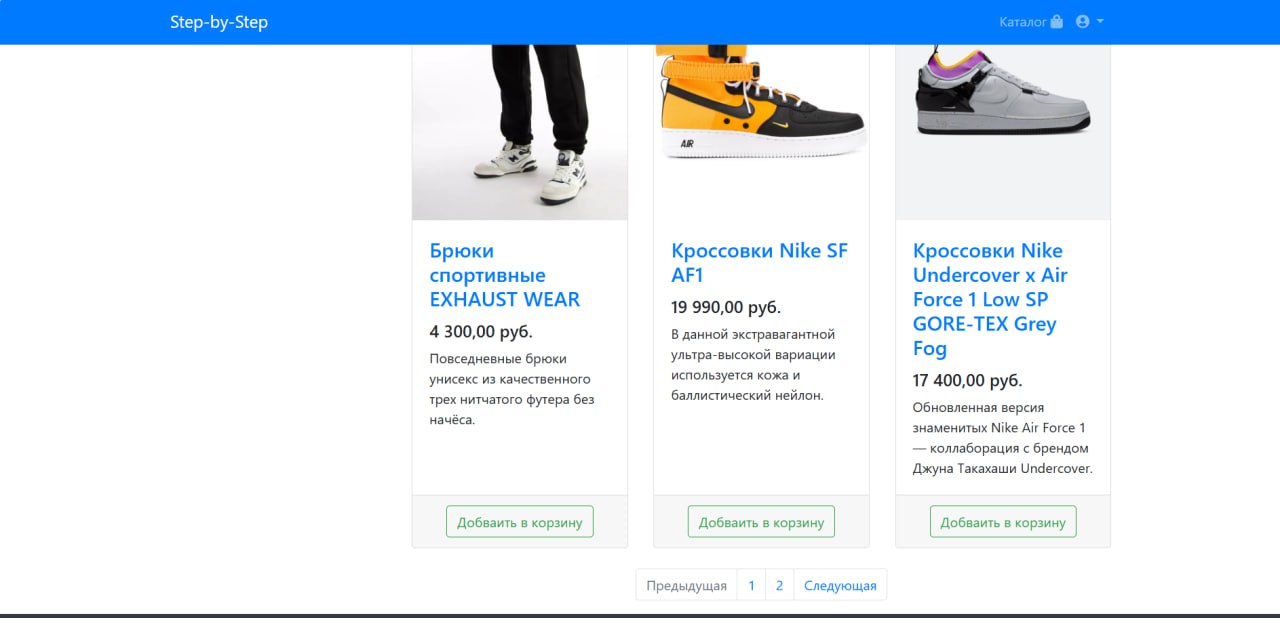


Рисунок 2 – Каталог.

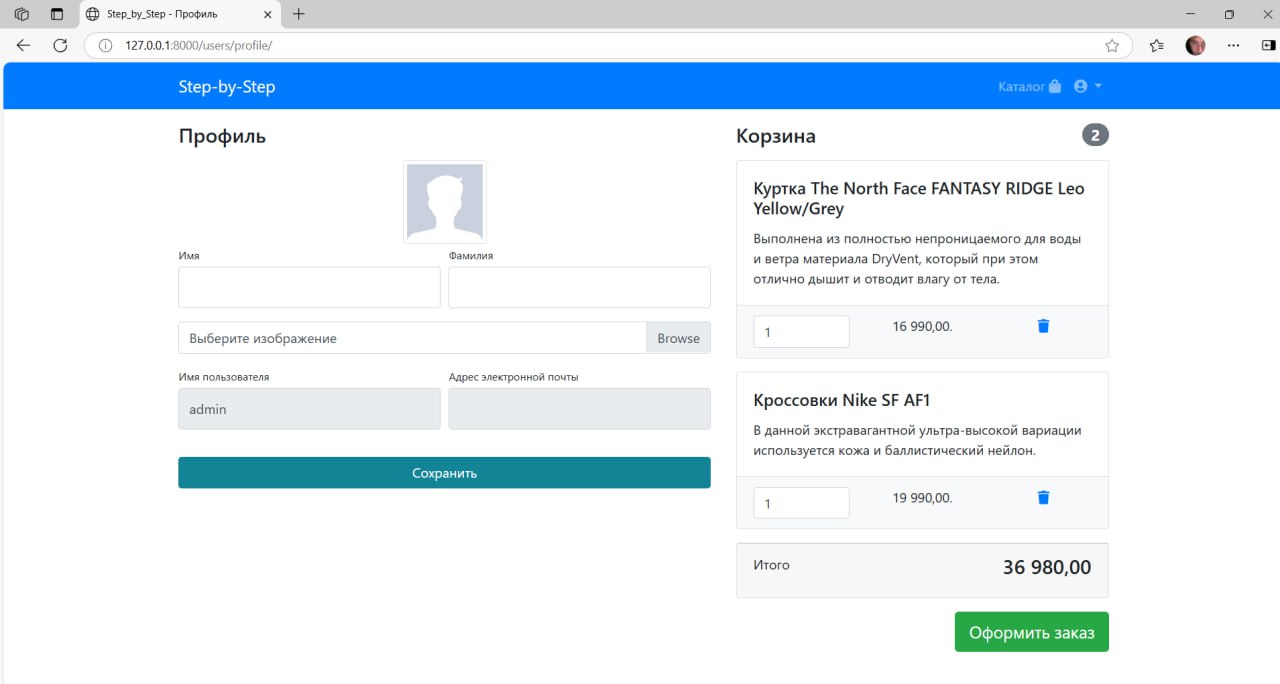


Рисунок 3 – Оформление заказа.

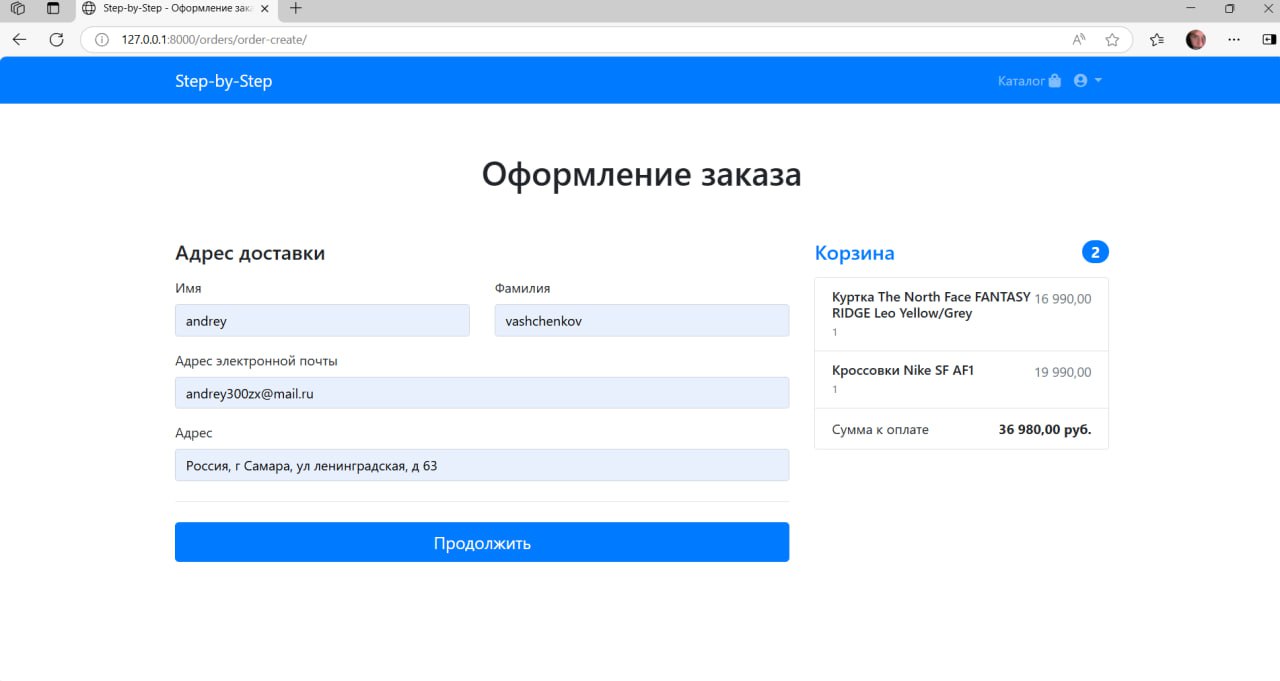


Рисунок 4 – Оформление заказа.

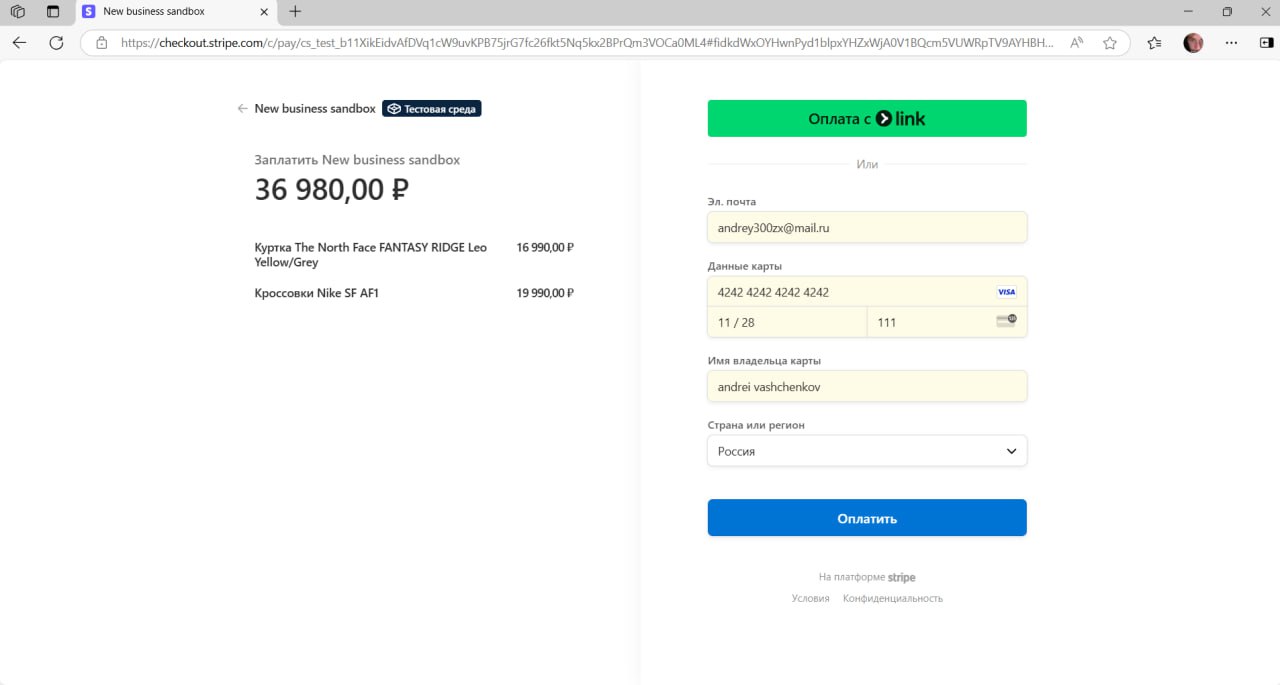


Рисунок 5 – Оплата.

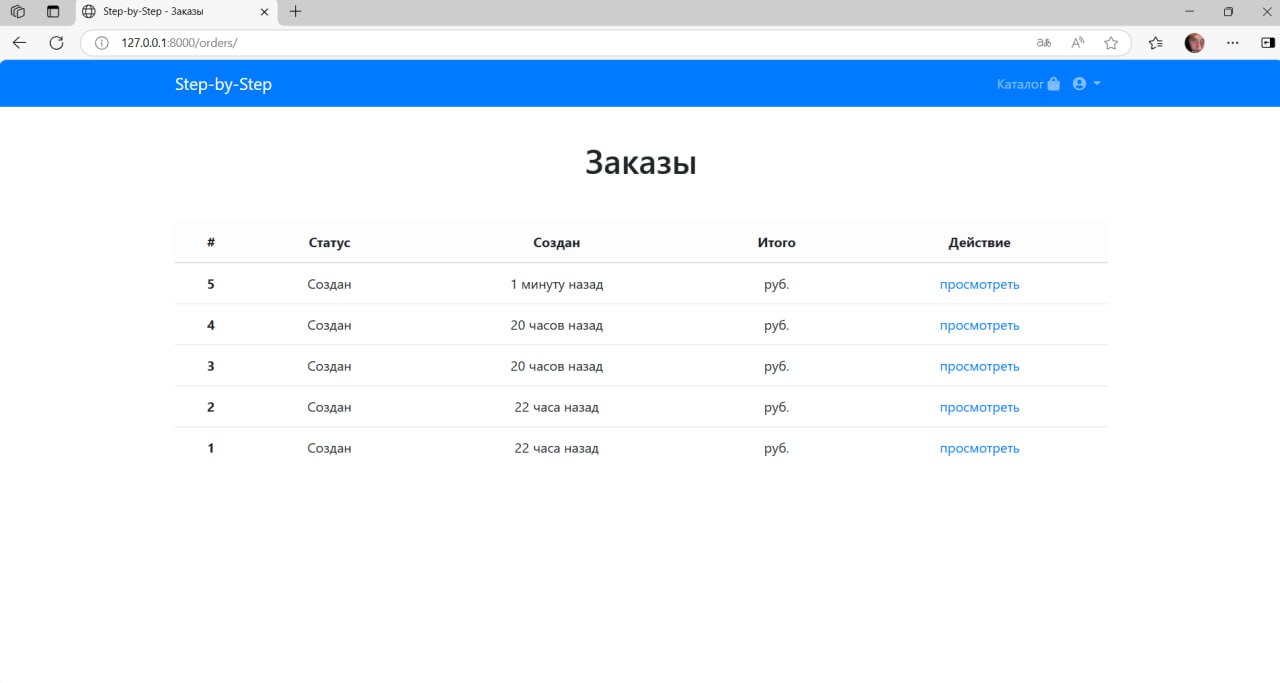


Рисунок 6 – История заказов.

Описание интерфейса:

### Главная страница (index.html)

* Приветственный баннер с кнопкой перехода в каталог
* Минималистичный дизайн с фокусом на CTA (call-to-action)

### Каталог товаров (products.html)

* Левая колонка: список категорий товаров (вертикальное меню)
* Центральная часть:
  + Карусель промо-изображений (3 слайда)
  + Сетка товаров (3 колонки с карточками)
  + Пагинация внизу (по 3 товара на страницу)

### Корзина (baskets.html)

* Список добавленных товаров с возможностью:
  + Изменения количества
  + Удаления позиции (иконка корзины)
* Итоговая сумма заказа
* Кнопка перехода к оформлению

### Базовая разметка (base.html)

Навигационная панель:

* Логотип и ссылки (каталог, профиль, вход/выход)
* Адаптивное меню для мобильных устройств

**7 Описание структуры контейнеризации и настройки окружения**

**3.1 Общая архитектура контейнеров**

Архитектура включает в себя два основных контейнера:

1. db – контейнер с образом PostrgeSQL, в котором развёрнута база данных приложения.

2. web – контейнер с Django приложением.

**3.2 Docker-compose**

Для управления контейнерами используется docker-compose.yml, который описывает конфигурацию всех сервисов и их связи. Не реализовано до конца.

**ПРИЛОЖЕНИЕ А**

Ссылка на GitHub с проектом:

https://github.com/Vlada-Zalevskaya/StepByStep/tree/main