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

**ФИНАЛЬНЫЙ ОТЧЕТ**

Выполнил студент

группы 6304-010302D

Адиянова О.О.

Самара 2025

# **Лабораторная работа 0: Проектирование приложения**

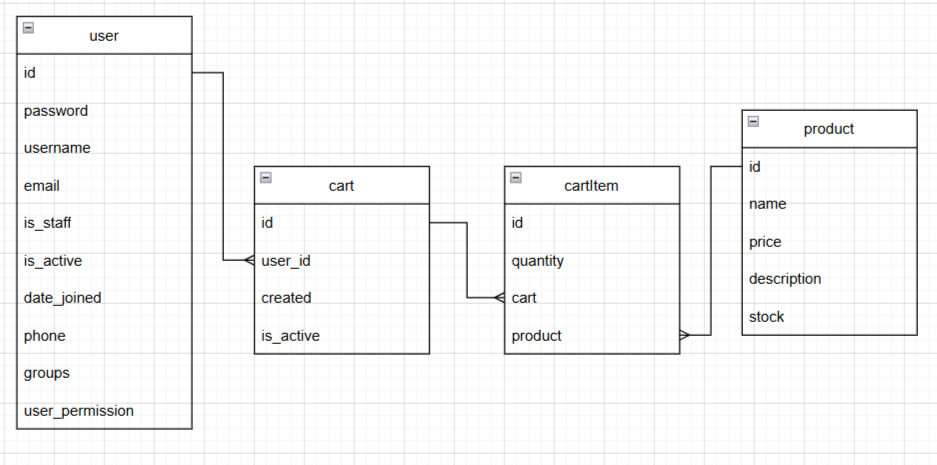
Цель работы: разработать концепцию и архитектуру веб-приложения, спроектировать API и создать Git-репозиторий.

1.Разработать схему взаимодействия компонентов (сервер, БД, клиент)

Наше приложение будет представлять собой интернет магазин.

Заходя в веб-приложение, клиент сможет видеть страницу с товарами, но перейти на страницу корзины он сможет только после авторизации. Также после авторизации он сможет добавлять товары в корзину.

2.Разработать логическую схему базы данных



3.Определить структуру API (REST, методы, URL, параметры, форматы запросов и ответов).

Аутентификация (/api/auth/)

POST /api/auth/register/ - Регистрация нового пользователя

POST /api/auth/login/ - Вход в систему (получение JWT токенов)

POST /api/auth/refresh/ - Обновление access токена

POST /api/auth/logout/ - Выход из системы

POST /api/auth/change-password/ - Смена пароля

Товары (/api/products/)

GET /api/products/ - Получение списка всех товаров

GET /api/products/{id}/ - Получение информации о конкретном товаре

POST /api/products/ - Создание нового товара (админ)

PUT/PATCH /api/products/{id}/ - Обновление товара

DELETE /api/products/{id}/ - Удаление товара

Корзины (/api/carts/)

GET /api/carts/ - Просмотр корзин пользователя

GET /api/carts/{id}/ - Просмотр конкретной корзины

Элементы корзины (/api/cart-items/)

GET /api/cart-items/ - Просмотр элементов корзины

POST /api/cart-items/ - Добавление товара в корзину

PUT/PATCH /api/cart-items/{id}/ - Изменение количества товара

DELETE /api/cart-items/{id}/ - Удаление товара из корзины

4. Файл .gitignore

#python stuff

venv/

.pylintrc

\_\_pycache\_\_/

#ide stuff

.vscode

.idea

#django stuff

db.sqlite3

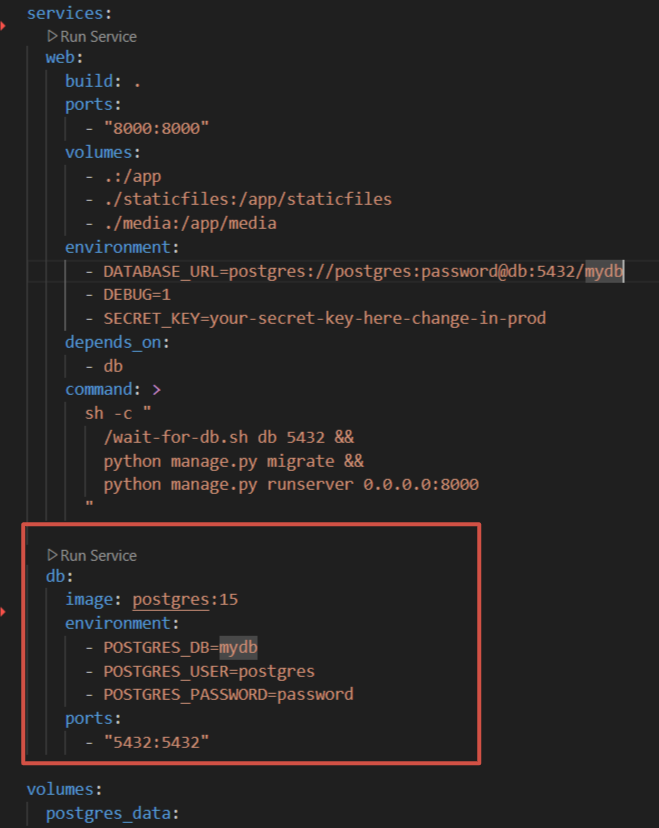
media/

# **Лабораторная работа 1: Разработка базы данных**

Цель работы: развернуть базу данных, разработать ORM-модели и реализовать скрипты для управления данными.

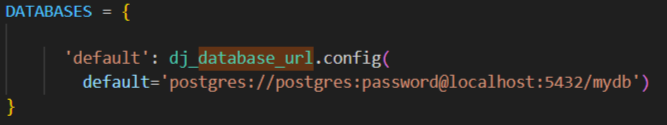
1. Настройка БД

Развертывание PostgreSQL в Docker:



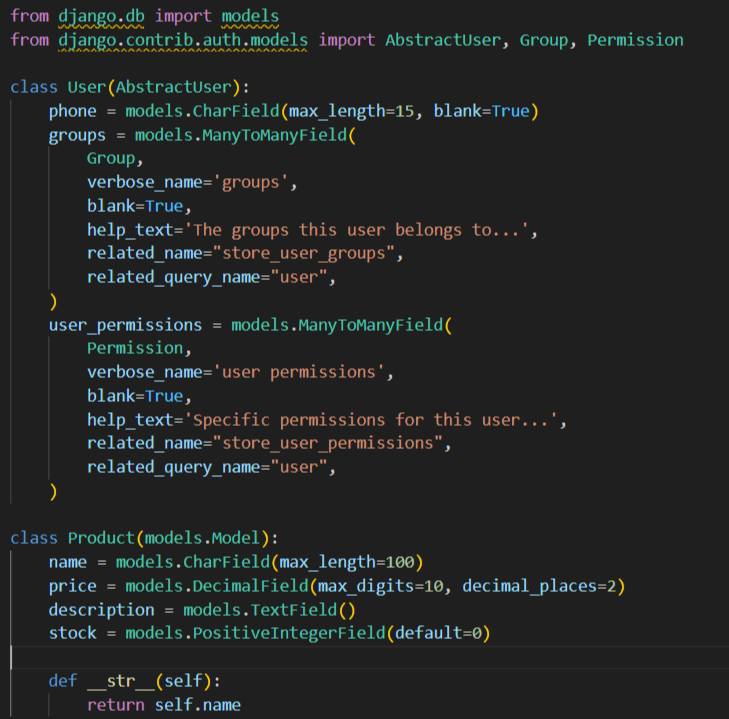
Подключение к БД в Django:

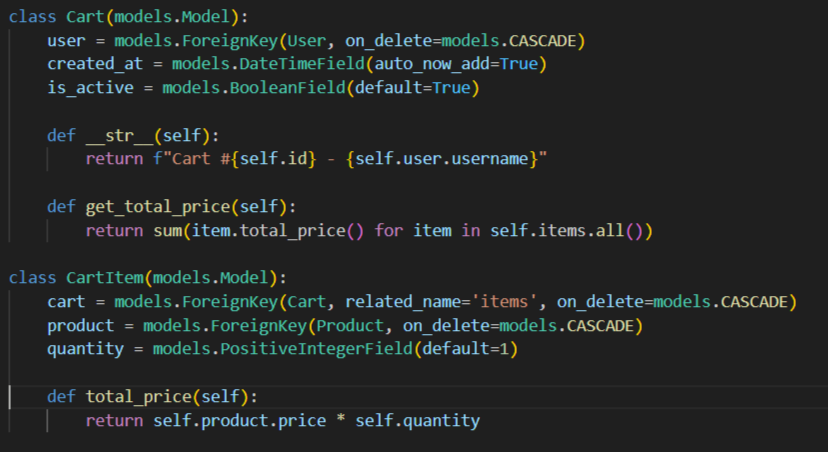
В settings.py используется dj-database-url



1. Разработка ORM модели

Определены сущности User, Producr, Cart, CartItem





1. Связи между сущностями:

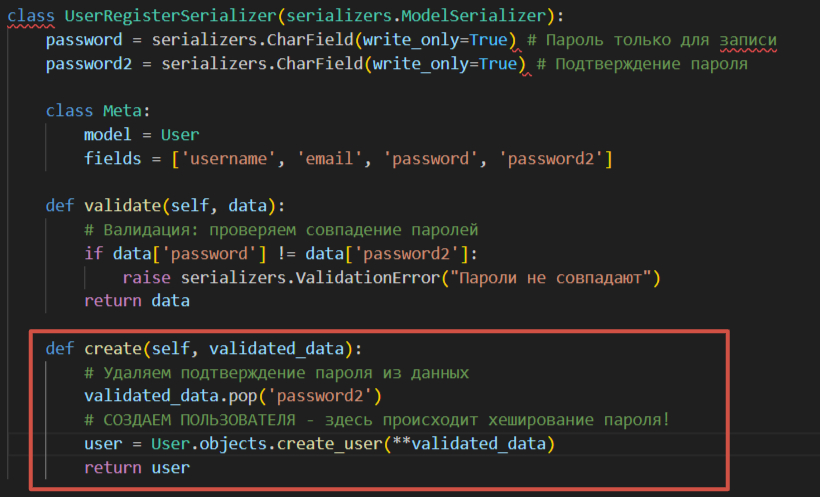
User - Cart: Один-ко-многим (один пользователь - много корзин)

Cart - CartItem: Один-ко-многим (одна корзина - много элементов)

Product - CartItem: Один-ко-многим (один товар - много элементов корзины)

1. Настройка хеширования паролей

В сериализаторе регистрации



1. Миграции

Django ORM автоматически создает миграции на основе моделей. Мы выполняем миграции с помощью команд:

python manage.py makemigrations

python manage.py migrate

# **Лабораторная работа 2: Разработка API**

Цель работы: Разработать RESTful API для сервисов.

1. Реализация API

Архитектура API: использован Django REST Framework (DRF), реализованы ViewSets для моделей.

Структура API endpoints:

Аутентификация

path('api/auth/login/', TokenObtainPairView.as\_view(), name='api-login'),

path('api/auth/refresh/', TokenRefreshView.as\_view(), name='api-refresh'),

path('api/auth/register/', RegisterAPIView.as\_view(), name='api-register'),

path('api/auth/change-password/', ChangePasswordView.as\_view(), name='api-change-password'),

path('api/auth/logout/', LogoutAPIView.as\_view(), name='api-logout'),

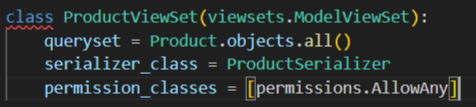
CRUD операции

router.register(r'products', ProductViewSet)

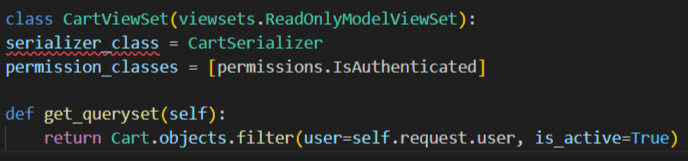
router.register(r'carts', CartViewSet, basename='cart')

router.register(r'cart-items', CartItemViewSet, basename='cartitem')

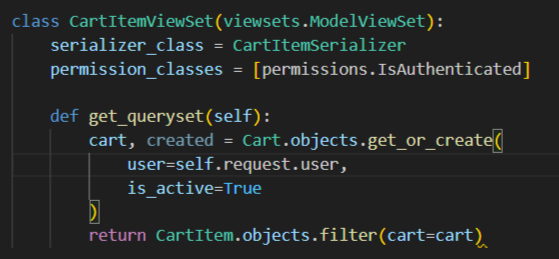
1. Модели и их API в файле api\_views.py
2. Product API (ProductViewSet):



1. Cart API (CartViewSet):

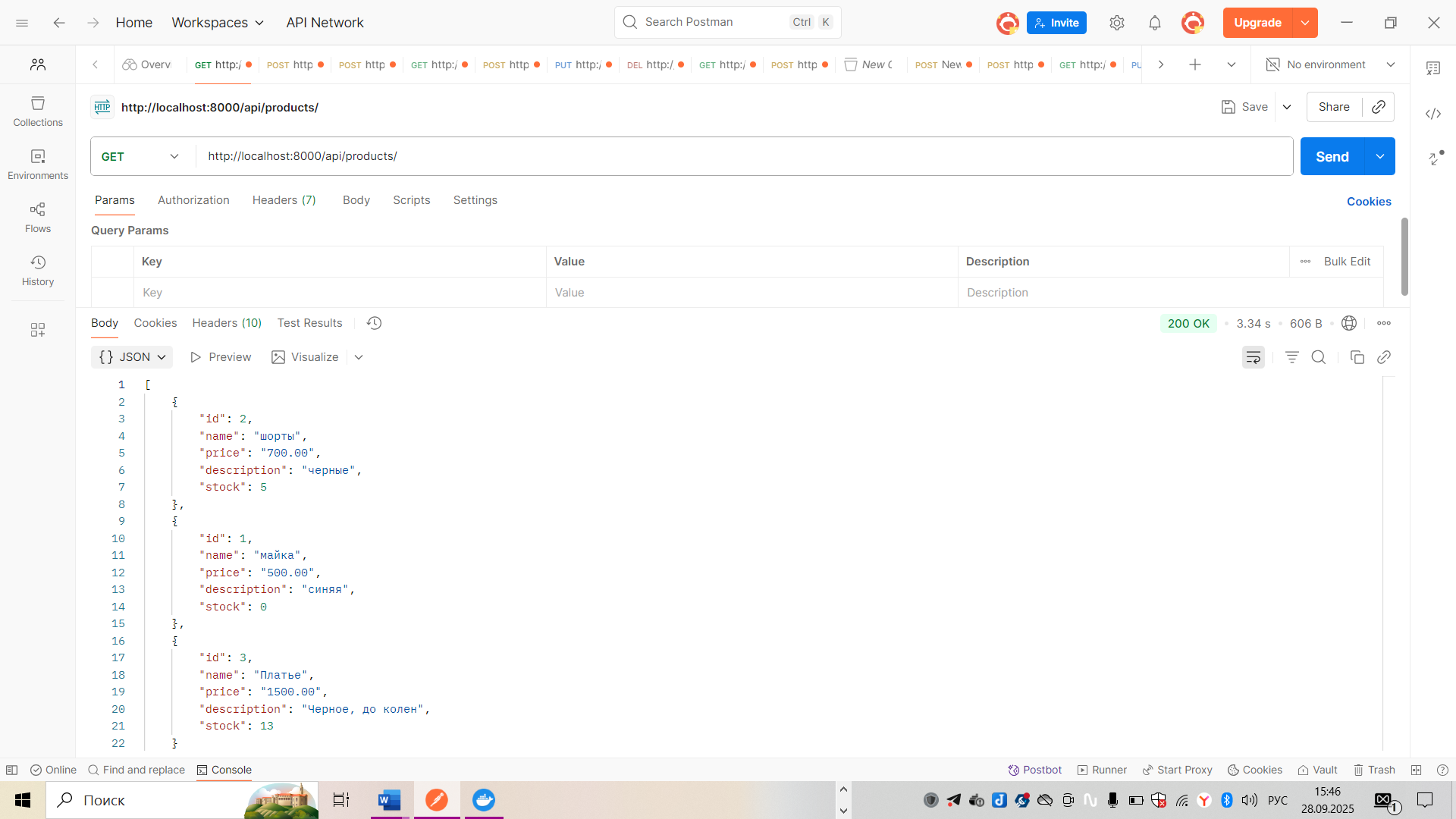


1. CartItem API (CartItemViewSet):

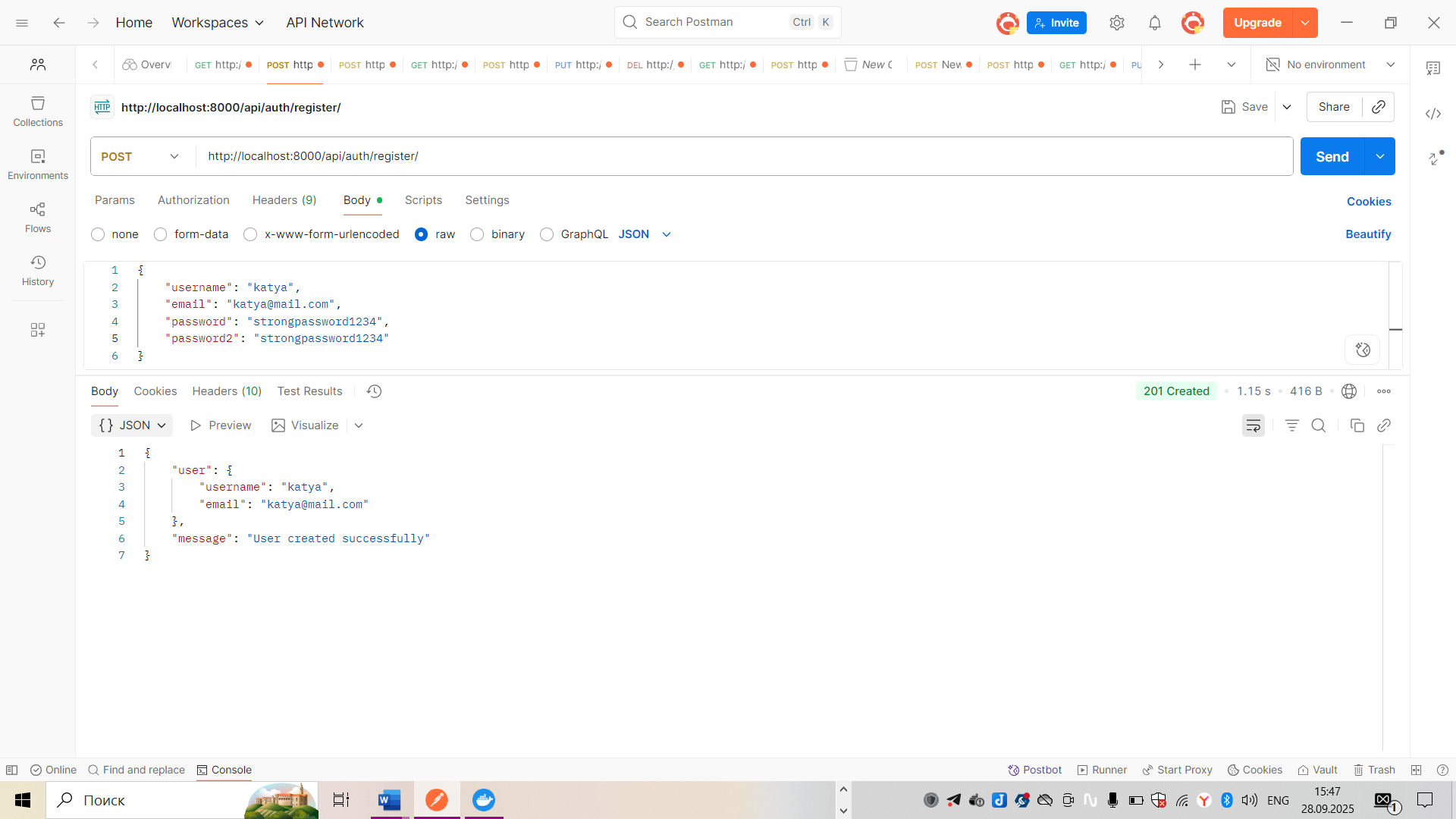


1. Проверка через Postman

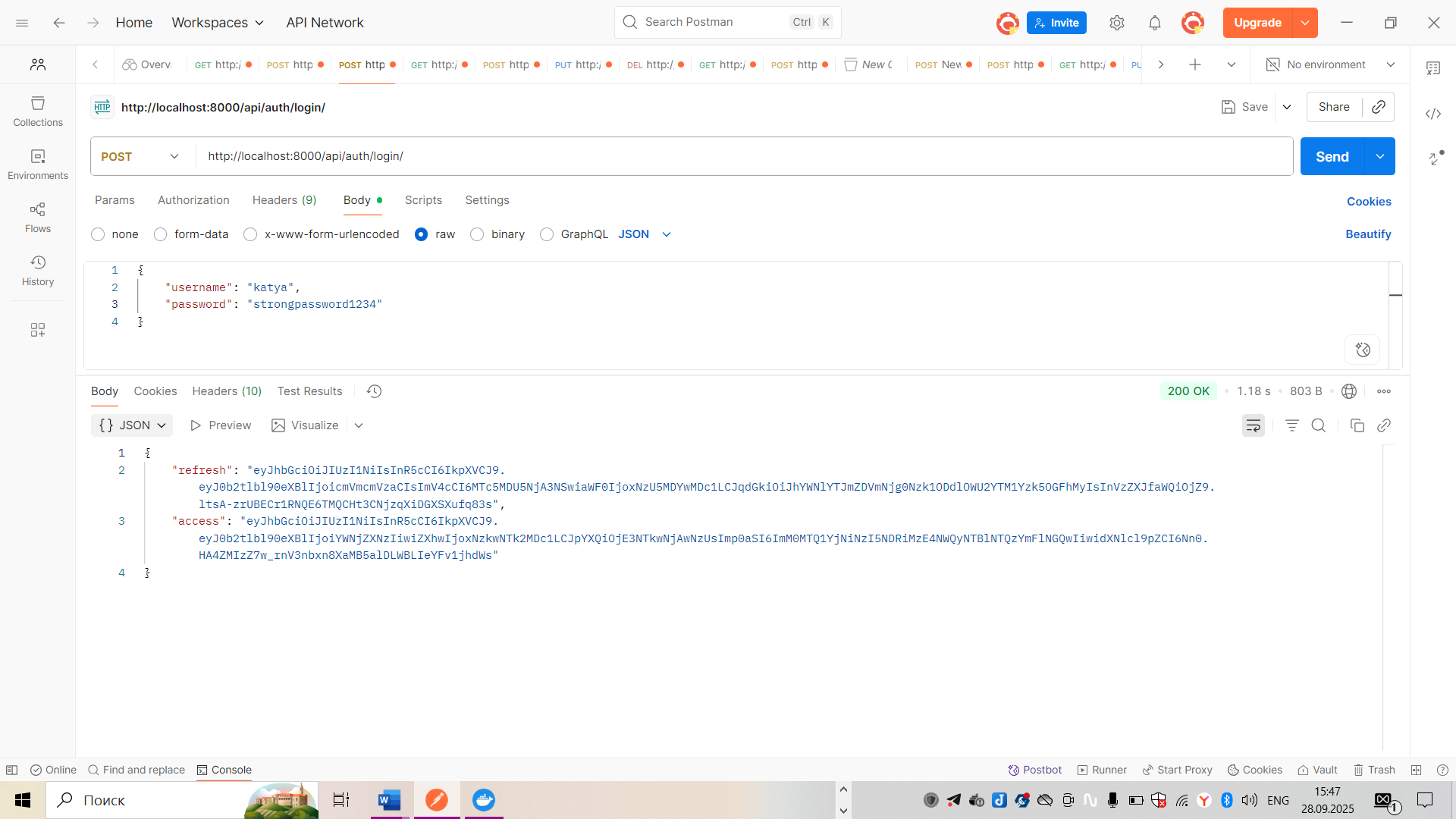
Получение списка товаров



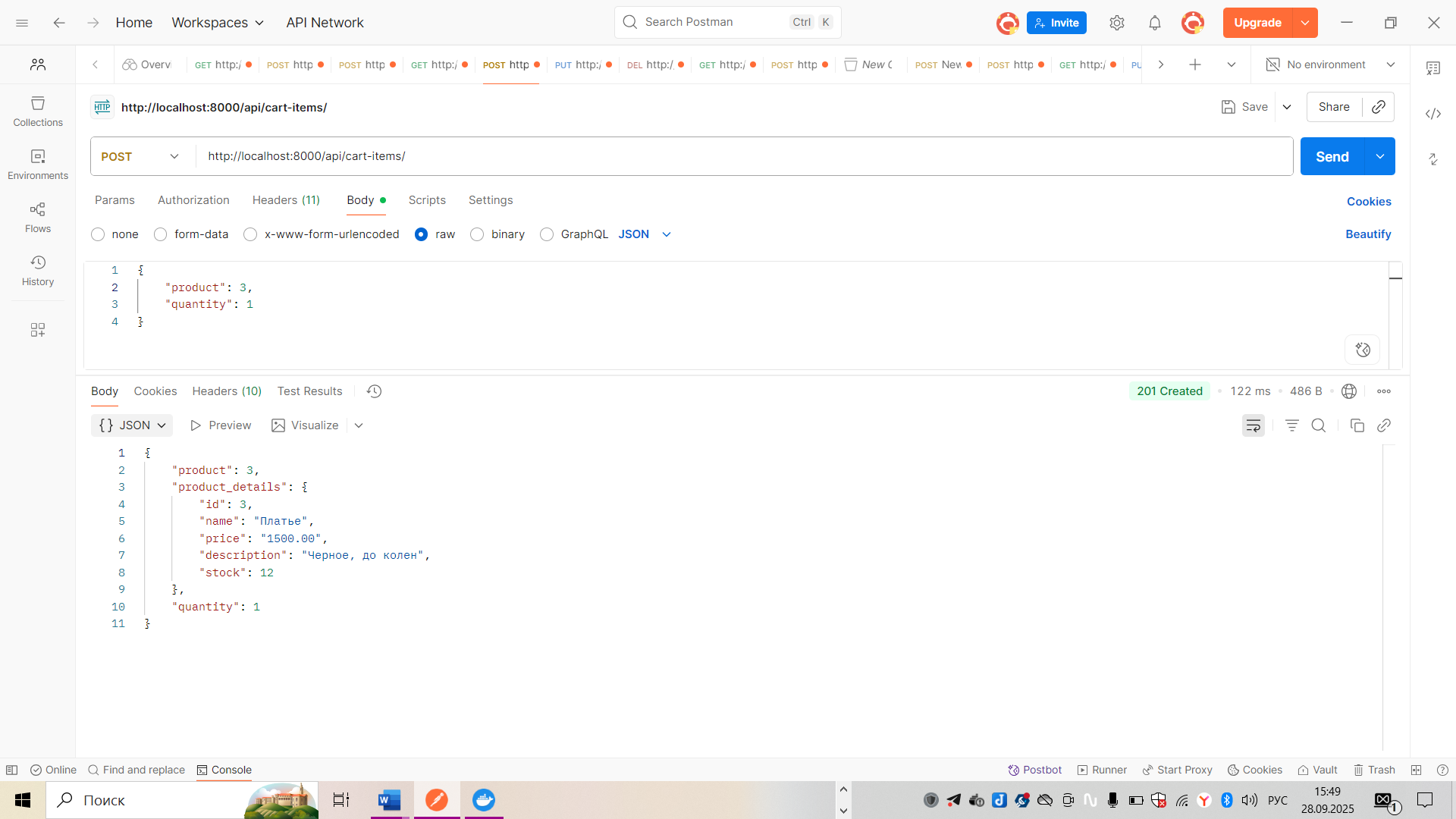
Регистрация пользователя



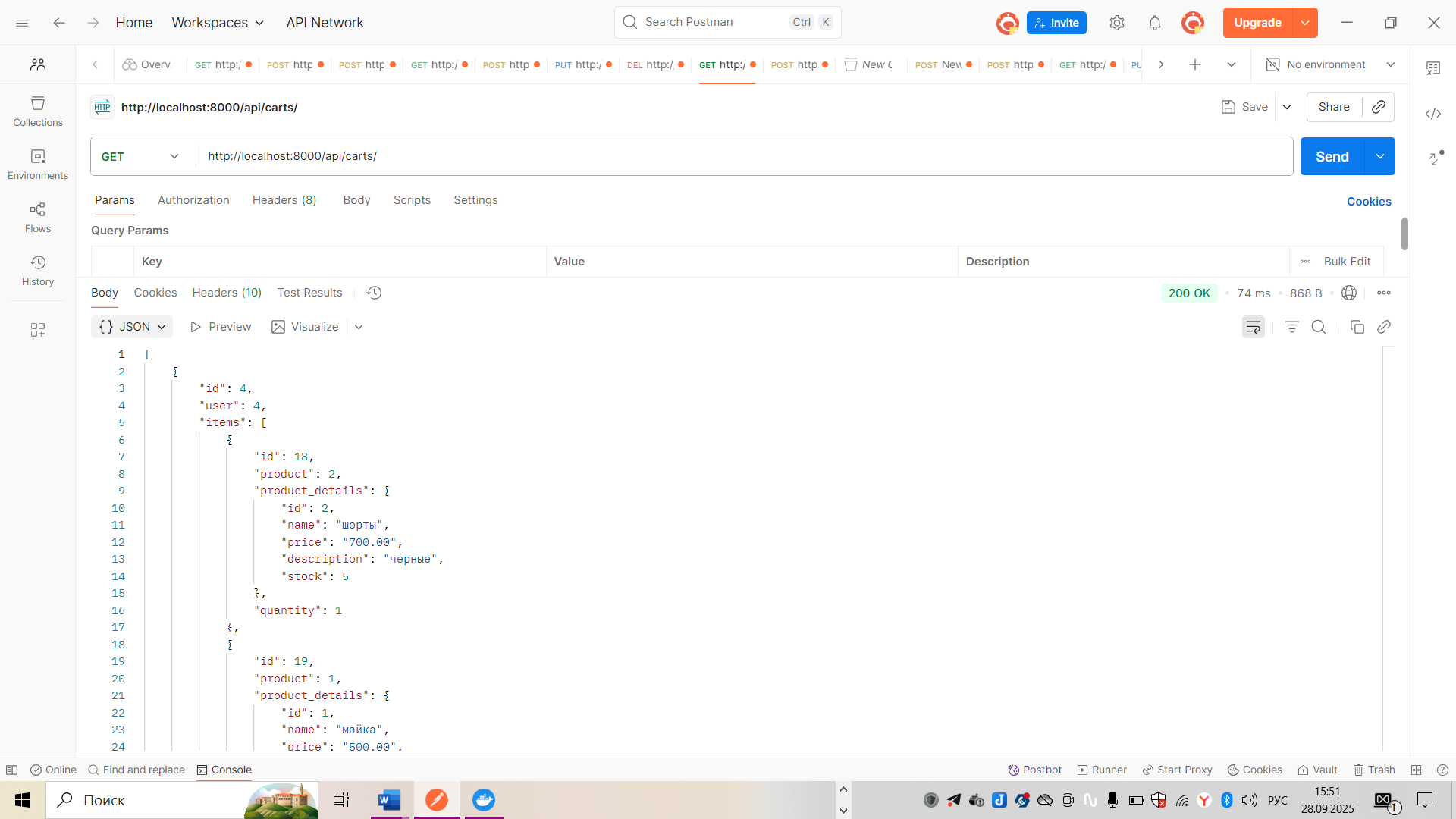
Вход, получение JWT токена



Добавление товара в корзину



Количество товаров в корзинем пользователя



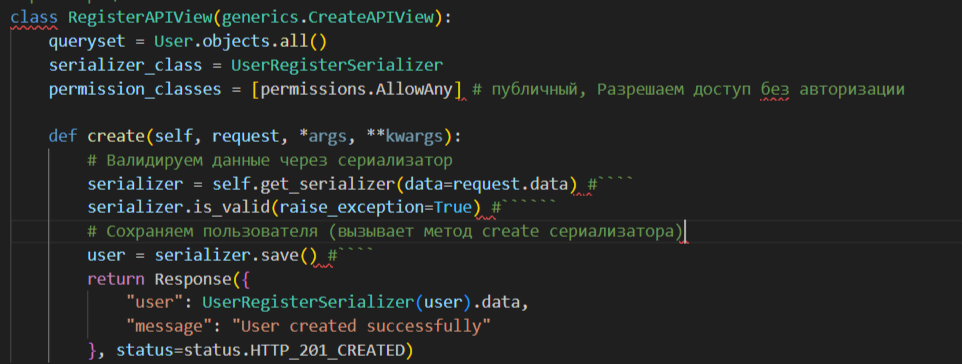
# **Лабораторная работа 3: Авторизация**

Цель работы: реализовать систему авторизации пользователей, используя JWT-токены, и обеспечить безопасность API.

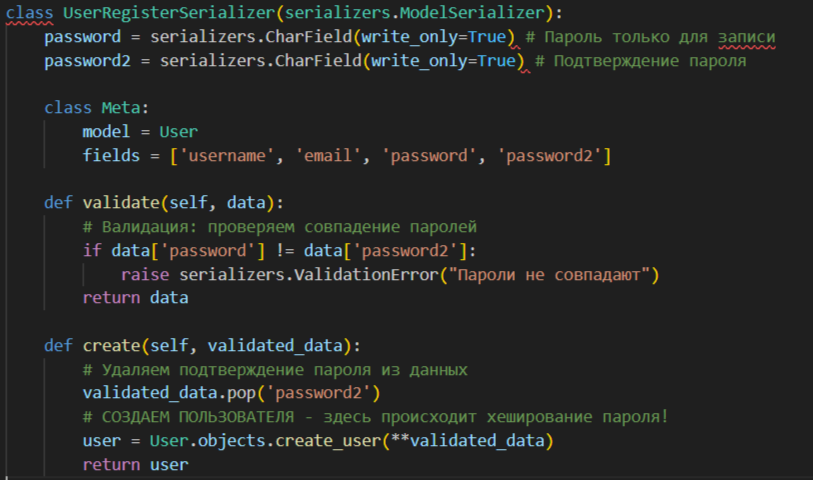
1. Реализация механизма аутентификации

Кастомная модель пользователя представлена в файле molels.py – class User(AbstractUser):

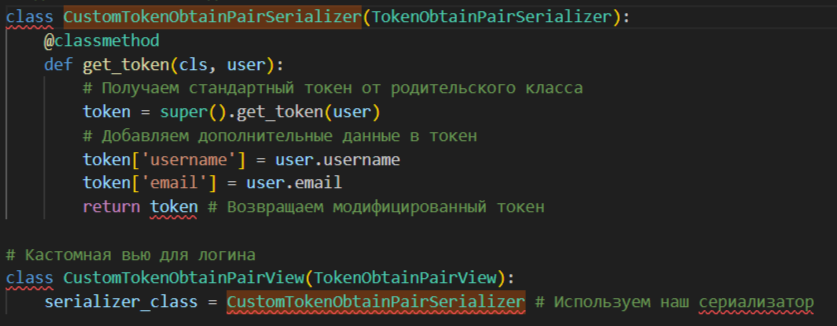
Регистрация нового пользователя в файле api\_views.py



Сериализатор регистрации с валидацией в файле serializers.py

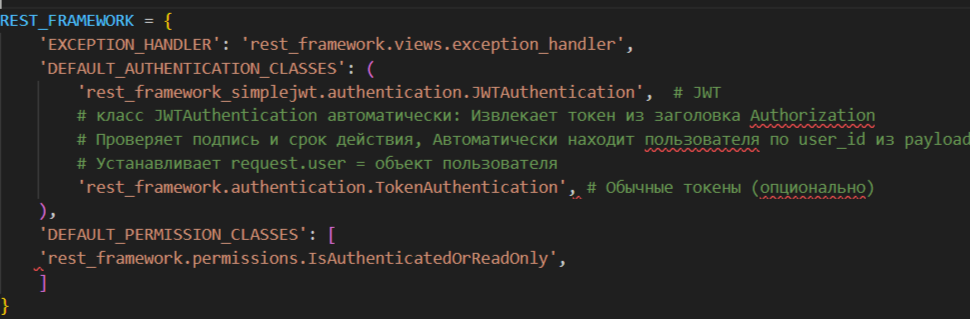


Вход в систему и получение JWT токена в файле views.py

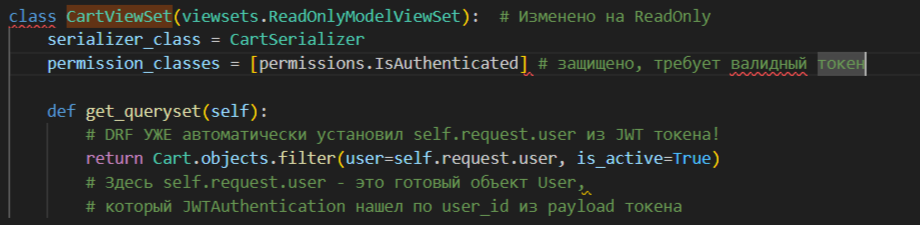


1. Настройка middleware для защиты API

JWT Authentication Middleware в settings.py



Проверка валидности токена в файле api\_views.py



Ограничения доступа к эндпоинтам прописаны в файле api\_views.py

IsAuthenticated :

/api/carts/

/api/cart-items/

AllowAny:

/api/auth/register/

/api/auth/login/

/api/products/

1. Реализация выхода из системы (отзыв токена) в файле api\_views.py



Frontend выход с очисткой сессии прописана в файле views.py , функция logout\_view(request)

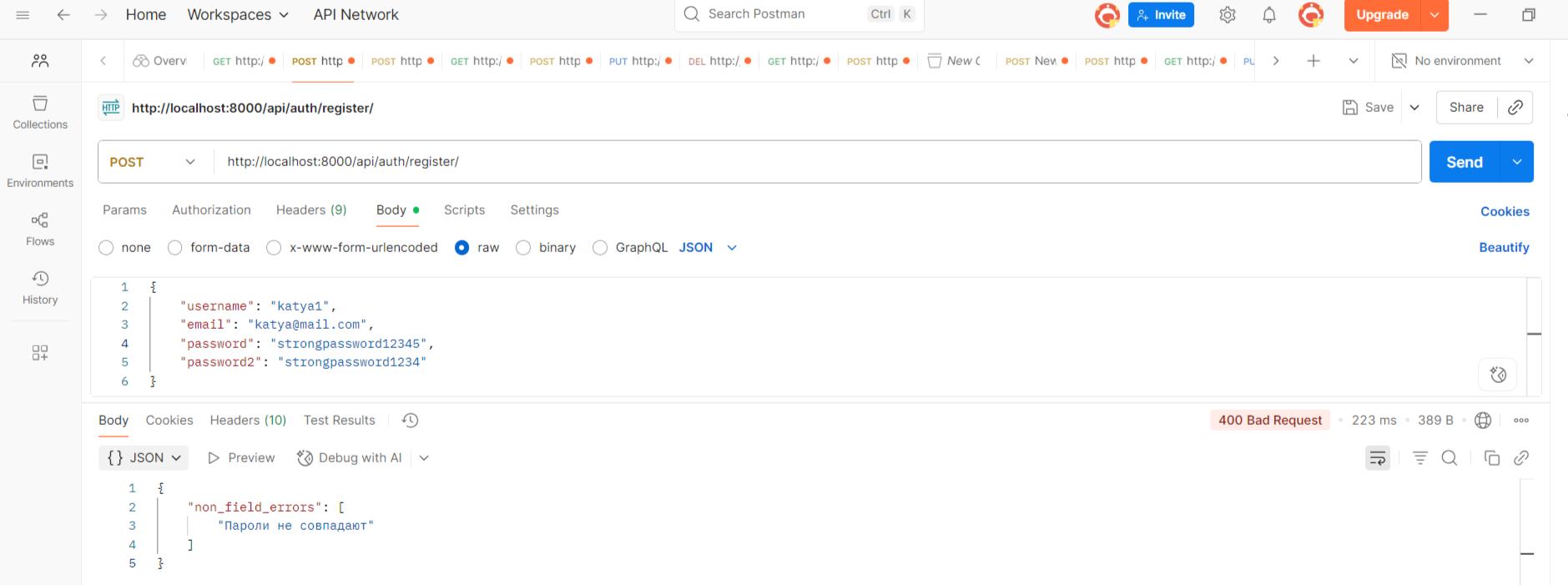
1. Безопасность API.

Защита от CSRF прописана в html формах и в AJAX запросах, также есть валидация данных в файле serializers.py

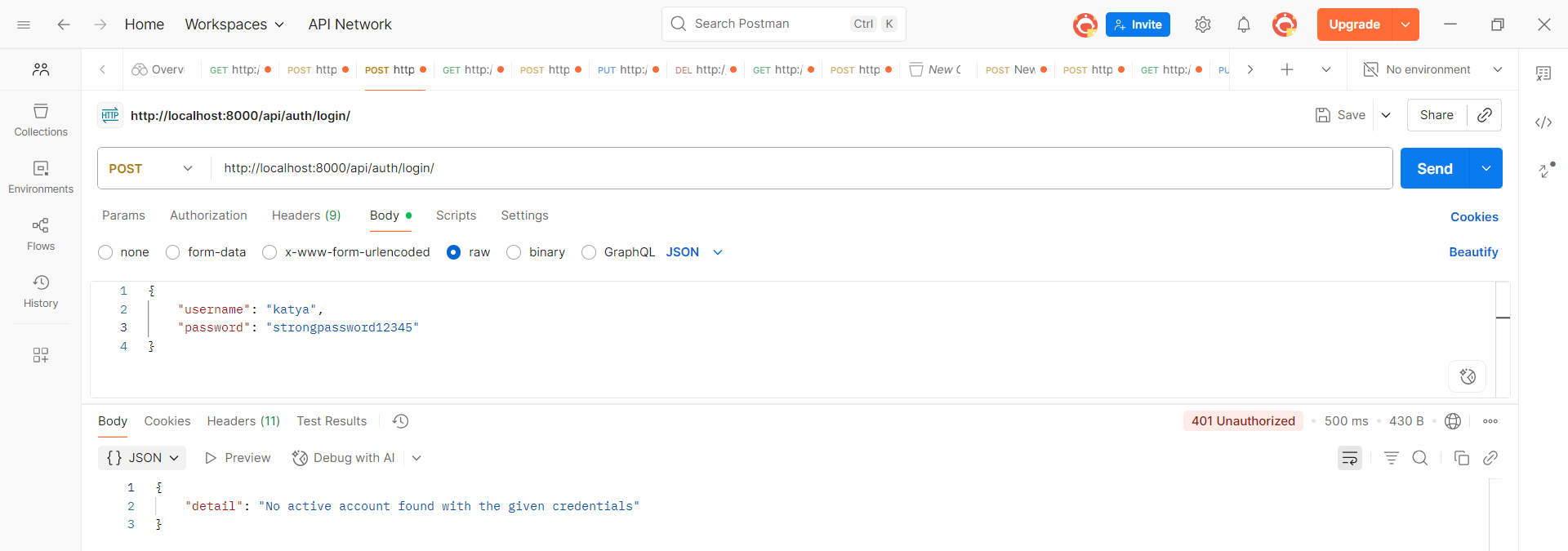
1. Примеры запросов и ответов в postman

Успешная регистрация пользователя представлена выше

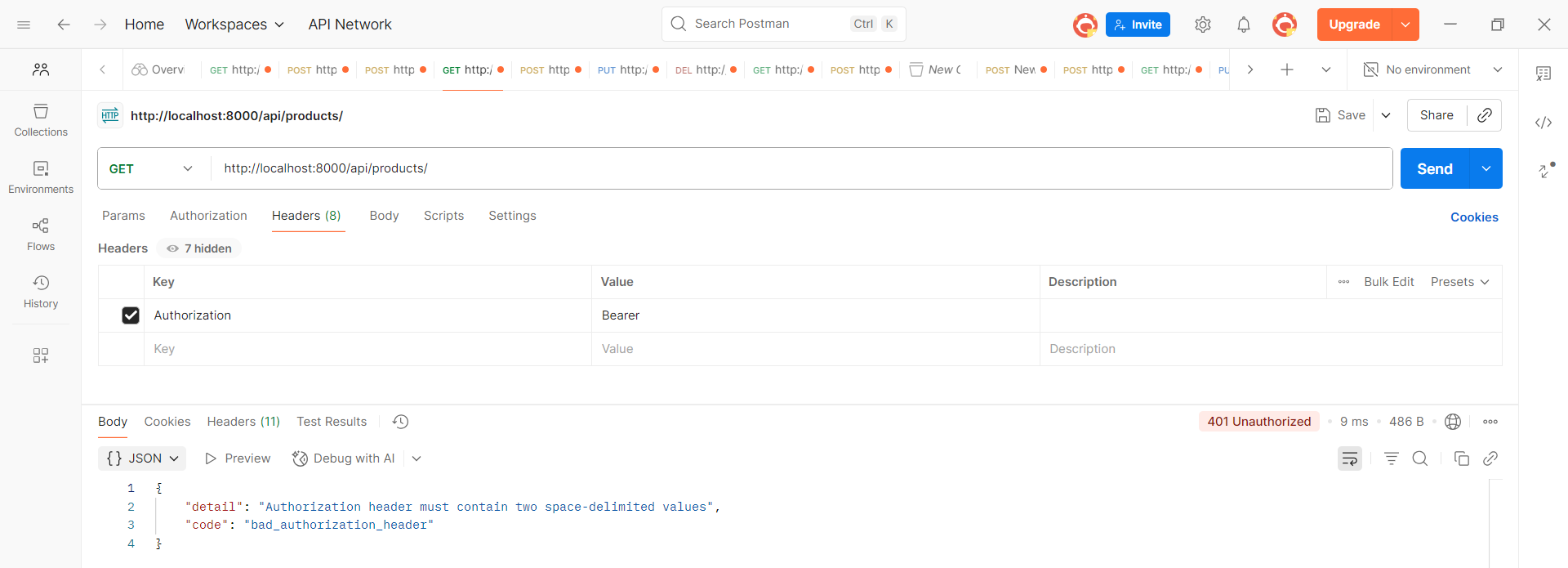
Ошибка регистрации (пароли не совпадают)



Ошибка входа



Доступ к защищеному ресурсу без токена



Статусы HTTP ответов

200 OK - Успешный запрос - Успешный вход, получение данных

201 Created - Успешное создание - Регистрация пользователя

205 Reset Content - Успешный выход – Logout

400 Bad Request - Ошибка валидации - Несовпадение паролей при регистрации

401 Unauthorized - Неавторизованный доступ - Отсутствует или невалидный токен

403 Forbidden - Доступ запрещен - Нет прав на операцию

404 Not Found - Ресурс не найден - Несуществующий товар

# **Лабораторная работа 4: Разработка пользовательского интерфейса и взаимодействие с API**

В ходе лабораторной работы успешно:

1. Разработан полнофункциональный пользовательский интерфейс с использованием Django шаблонов
2. Реализована интеграция с API через JavaScript Fetch и Django views
3. Настроена аутентификация с передачей JWT токенов в запросах
4. Созданы адаптивные шаблоны с использованием Bootstrap 5
5. Реализованы основные пользовательские сценарии:

Регистрация и вход

Просмотр каталога товаров

Управление корзиной

Просмотр профиля

1. Добавлена клиентская валидация и обработка ошибок

# **Лабораторная работа 5: Финализация приложения и упаковка в Docker**

Цель работы: завершить разработку приложения, связать клиент и сервер, создать Docker-контейнеры и настроить их взаимодействие.

Настройки Django для production: в shop/settings.py настроили импорты и прописали для безопасноти: SECRET\_KEY, DEBUG, ALLOWED\_HOSTS; для бд: DATABASE\_URL, DATABASES; для статических файлов: STATIC\_URL, STATIC\_ROOT, MEDIA\_URL, MEDIA\_ROOT; и настройки для Docker: API\_BASE\_URL, BASE\_URL.

В Dockerfile прописала скрипт установки системных зависимостей включая postgresql-client для pg\_isready.

В docker-compose.yml настроена двухсервисная архитектура для запуска интернет-магазина. Конфигурация:

1. Сервис web (Django приложение):

Базовые настройки: сборка из Dockerfile в текущей директории, проброс порта 8000 из контейнера на хост, монтирование текущей директории в /app контейнера, статические файлы и медиа файлы

Переменные окружения environment

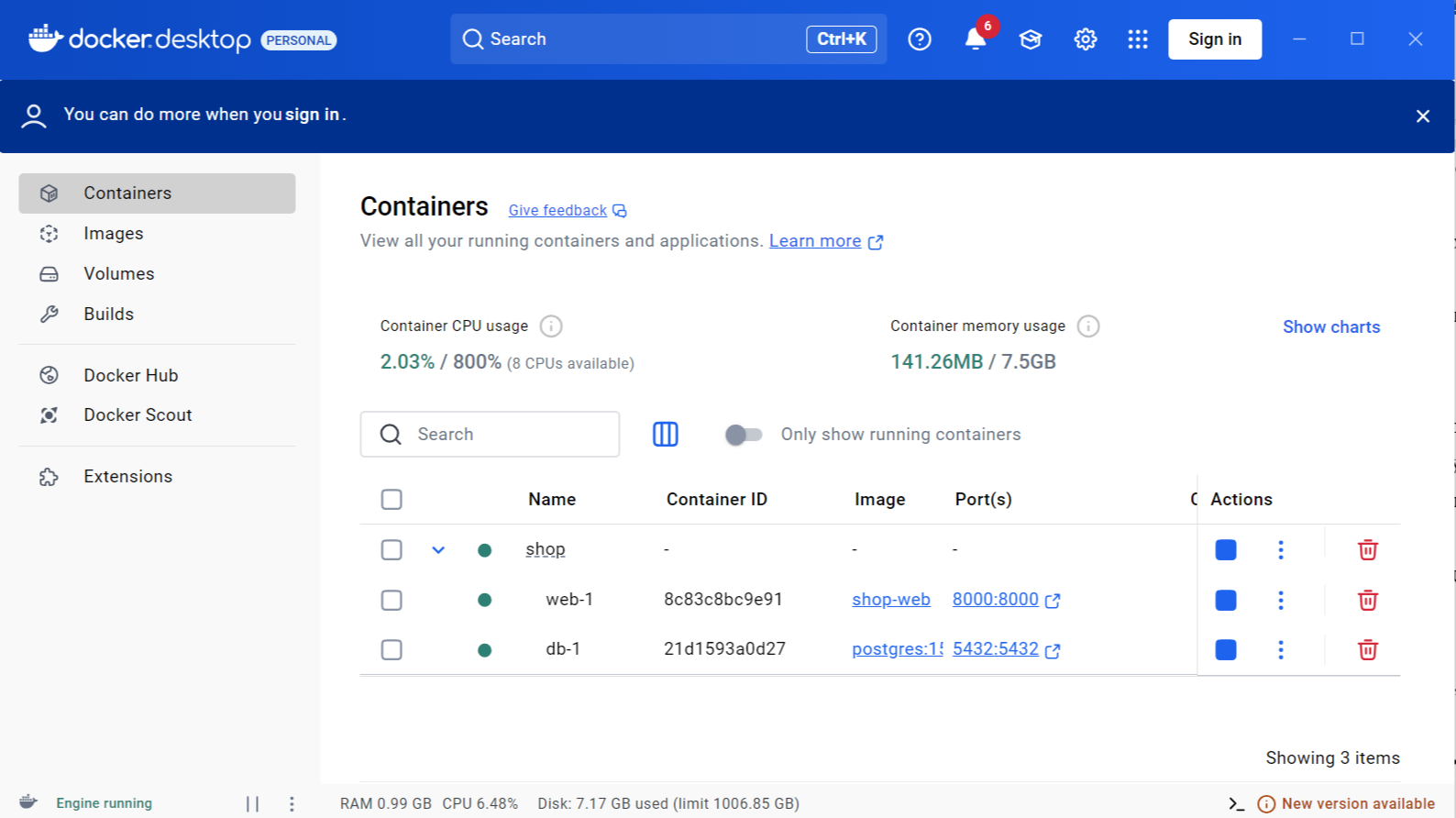
Зависимость от сервиса db в depends\_on и команды: ожидание готовности БД, применение миграций и запуск сервера

1. Сервис db (PostgreSQL)

Настройки базы данных в db: официальный образ PostgreSQL 15, имя бд, пользователь, пароль и проброс порта бд.

Том для данных volumes: именнованный том для хранения данных бд

Отображениt работающего Docker.desctop



# **Ссылка на GitHub**

<https://github.com/Ovalga/MySite>