САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнил: Рашевский Вячеслав Васильевич
Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

Задача:

Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ. Делать это можно как с помощью dockercompose так и с помощью docker swarm. При разумном использовании swirl вы получите дополнительные баллы.

1.Построим прокси на nginx, свяжем сервисы приложения и разделим функционал:

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;

    upstream app {
        server app:3000;
    }

    upstream product_service {
        server product-service:3001;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://app;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /products {
            proxy_pass http://product_service;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

В каждом сервисе создадим докерфайл, и распишем docker-compose.yml для сборки сервисов, подключения postgresql и nginx:

```yaml
version: '3.9'

services:
  lr2:
    container_name: lr2
    build:
      context: ./lr2
      dockerfile: Dockerfile
    ports:
      - "3000"
    depends_on:
      - postgres
    environment:
      - DATABASE_URL=postgres://postgres:postgres@postgres:5432/db
    volumes:
      - ./lr2:/usr/src/app:cached
      - /usr/src/app/node_modules
    networks:
      - my-network

  lr3:
    container_name: lr3
    build:
      context: ./lr3
      dockerfile: Dockerfile
    ports:
      - "3001"
    depends_on:
      - postgres
    environment:
      - DATABASE_URL=postgres://postgres:postgres@postgres:5432/db
    volumes:
      - ./lr3:/usr/src/app:cached
      - /usr/src/app/node_modules
    networks:
      - my-network

  nginx:
    container_name: nginx
    image: nginx:alpine
    ports:
      - "80:80"
    depends_on:
      - lr2
      - lr3
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    networks:
      - my-network

  postgres:
    container_name: postgres
    image: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: db
```

```
    volumes:
      - pgdata:/var/lib/postgresql/data
    networks:
      - my-network

networks:
  my-network:
    driver: bridge

volumes:
  pgdata:
```

4.Для каждой папки с докер файлом распишем докер-игнор:

```
node_modules
app/node_modules
product-service/node_modules
dist
```

5. Запуск и проверка работы

lr4			Exited		N/A	5 hours ago
lr3-1 67c7085c118d	lr4-lr3		Exited (137)	0:3001	N/A	5 hours ago
lr2-1 c8147e21b88b	lr4-lr2		Exited (137)	0:3000	N/A	5 hours ago
nginx-1 873b00b4f9b1	nginx:alpine		Exited	80:80	N/A	5 hours ago
postgres-1 259d86ec374b	postgres:alpine		Exited (137)		N/A	5 hours ago

Регистрация

docker check / reg

Save | Share

POST http://localhost/register Send

Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

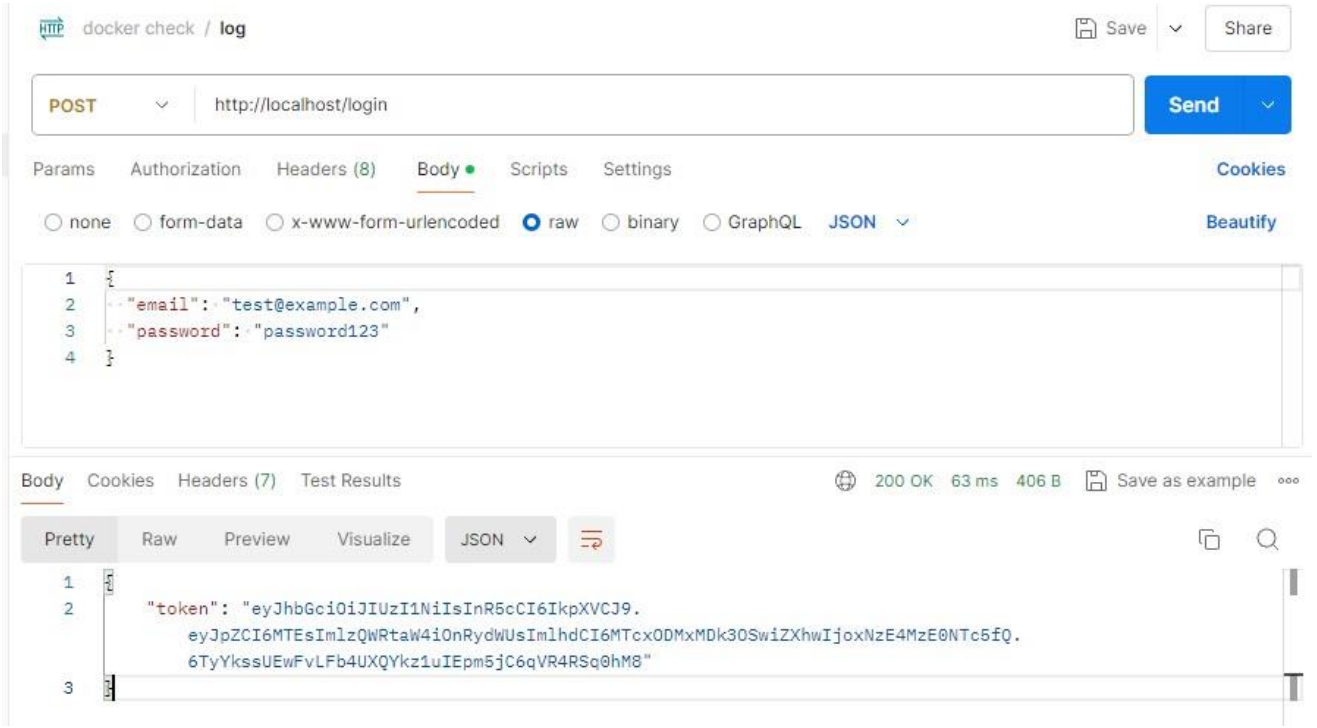
```
{
  "email": "test@example.com",
  "password": "password123",
  "isAdmin": true
}
```

Body Cookies Headers (7) Test Results 201 Created 82 ms 443 B Save as example

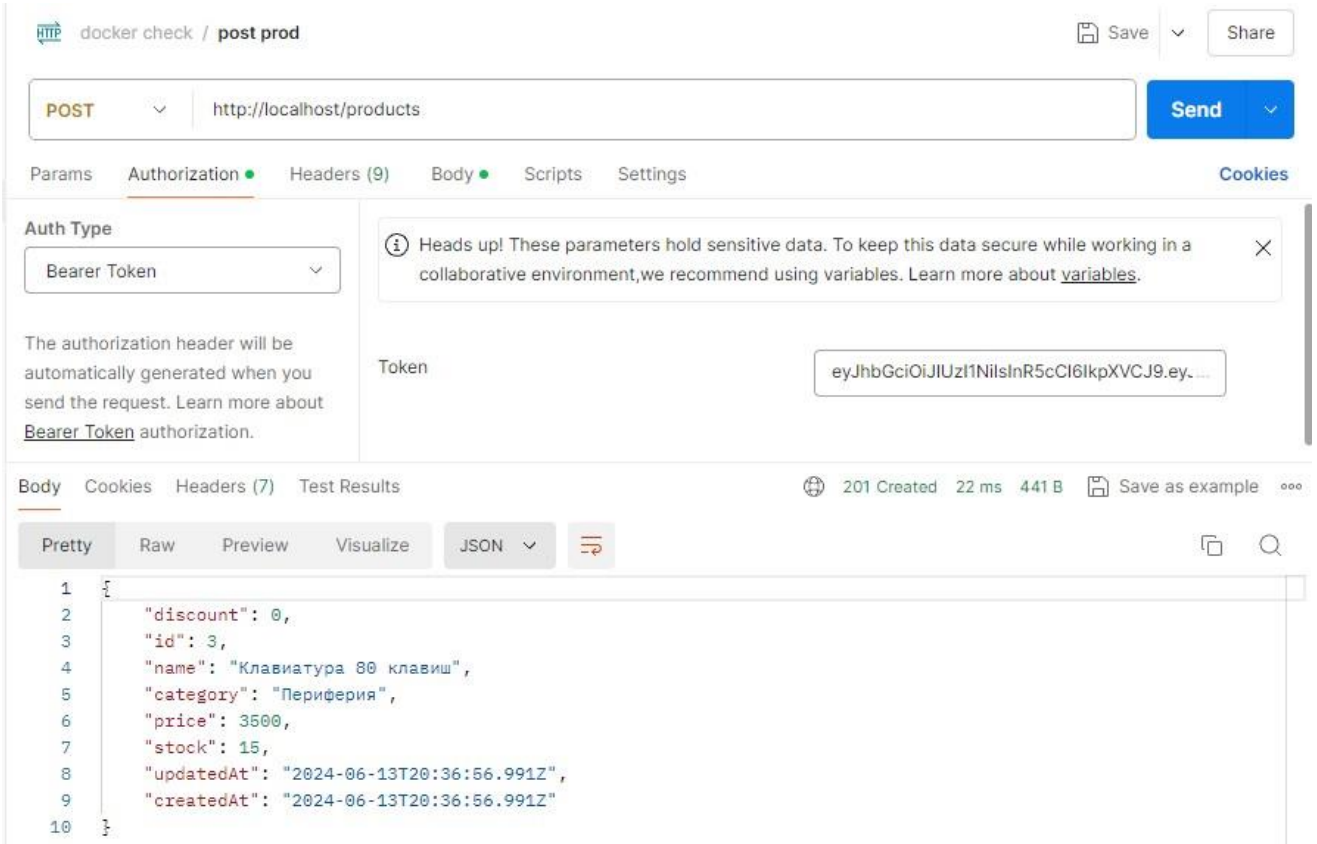
Pretty Raw Preview Visualize JSON

```
{
    "id": 14,
    "email": "test@example.com",
    "password": "$2b$10$ayz/2WVfl1xOg2EjCN0jzOd5.NTNeDOUmlEtx3mNQDCxivKW7myZO",
    "isAdmin": true,
    "updatedAt": "2024-06-13T20:35:57.315Z",
    "createdAt": "2024-06-13T20:35:57.315Z"
}
```

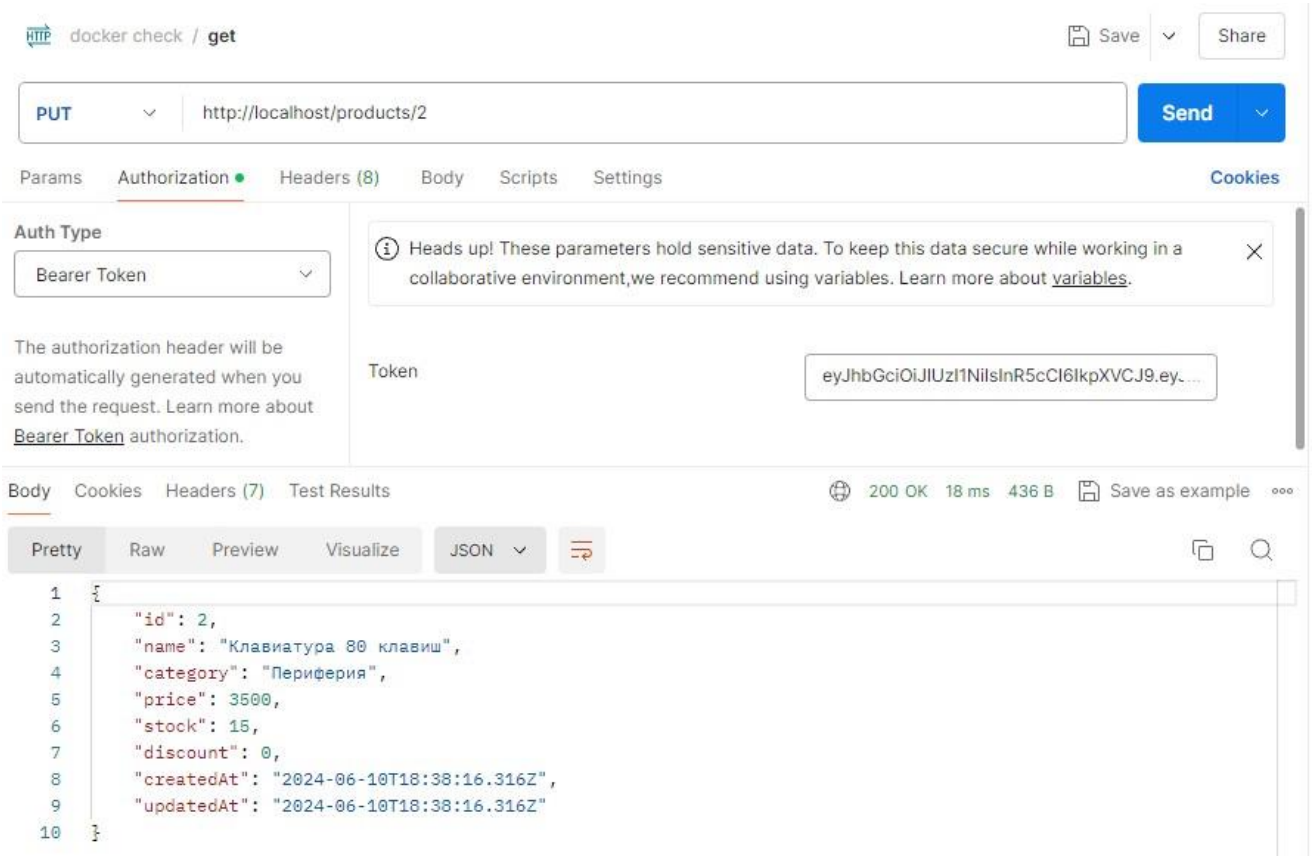
Вход



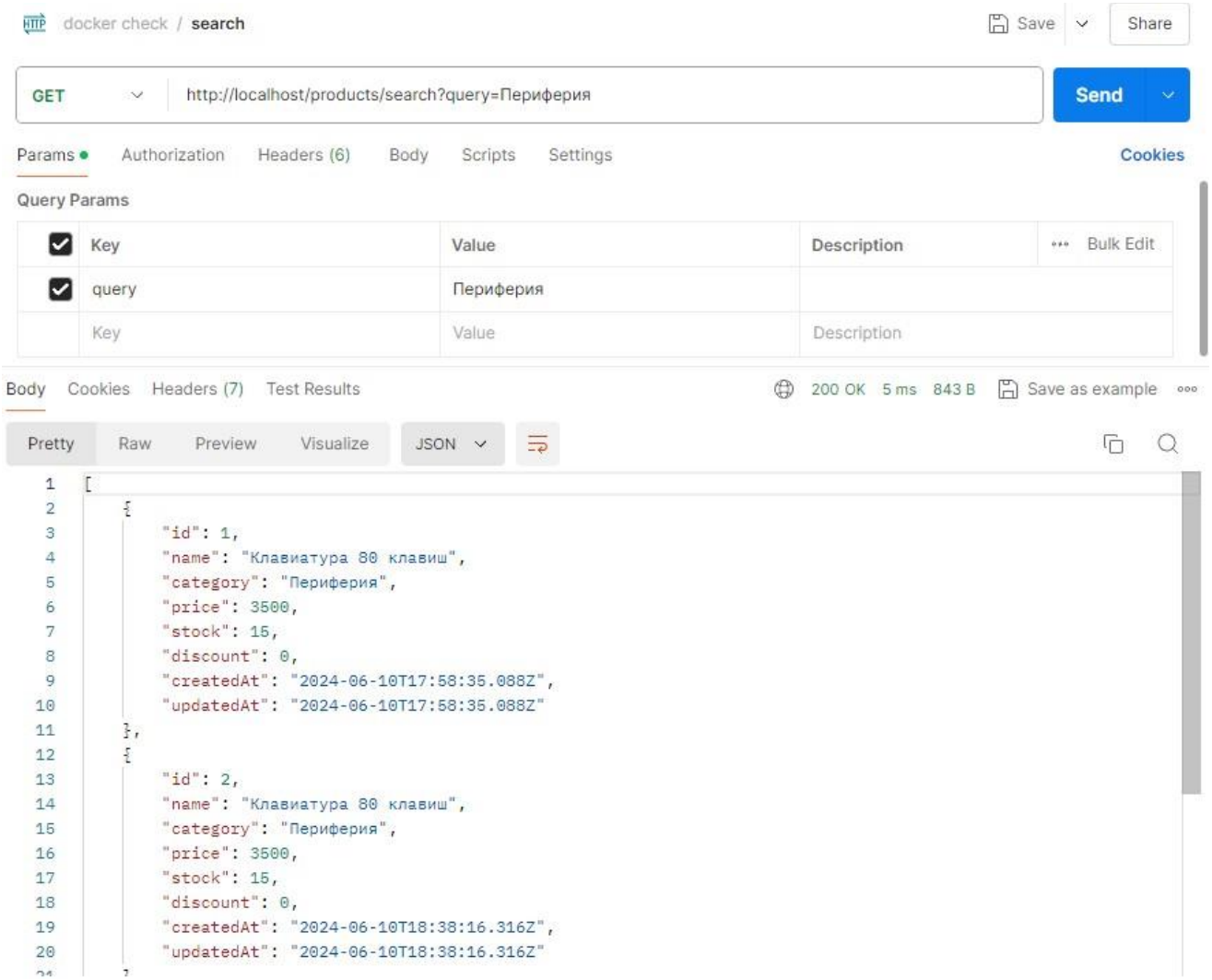
Добавление товара:



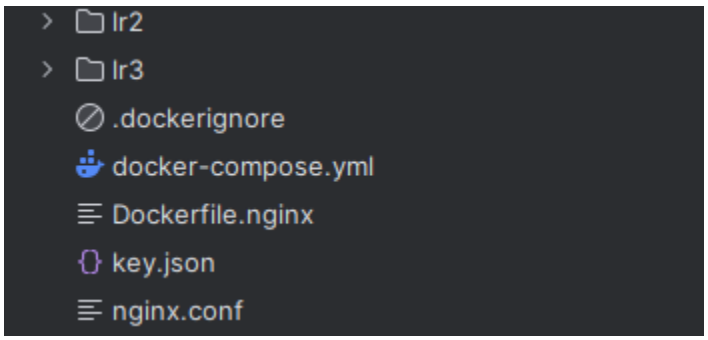
Поиск по id


docker check / get
Save
Share
PUT
http://localhost/products/2
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Auth Type
Bearer Token
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...
Body
Cookies
Headers (7)
Test Results
200 OK
18 ms
436 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 2,
"name": "Клавиатура 80 клавиш",
"category": "Периферия",
"price": 3500,
"stock": 15,
"discount": 0,
"createdAt": "2024-06-10T18:38:16.316Z",
"updatedAt": "2024-06-10T18:38:16.316Z"
}

Поиск



Структура проекта:



Вывод:

В рамках лабораторной работы №4 было успешно упаковано приложение в Docker-контейнеры и настроено сетевое взаимодействие между его компонентами. Также было обеспечено общение микросервисов через Nginx, используя Docker Compose для оркестрации, а Docker Swarm использован для повышения масштабируемости и надежности системы.