МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

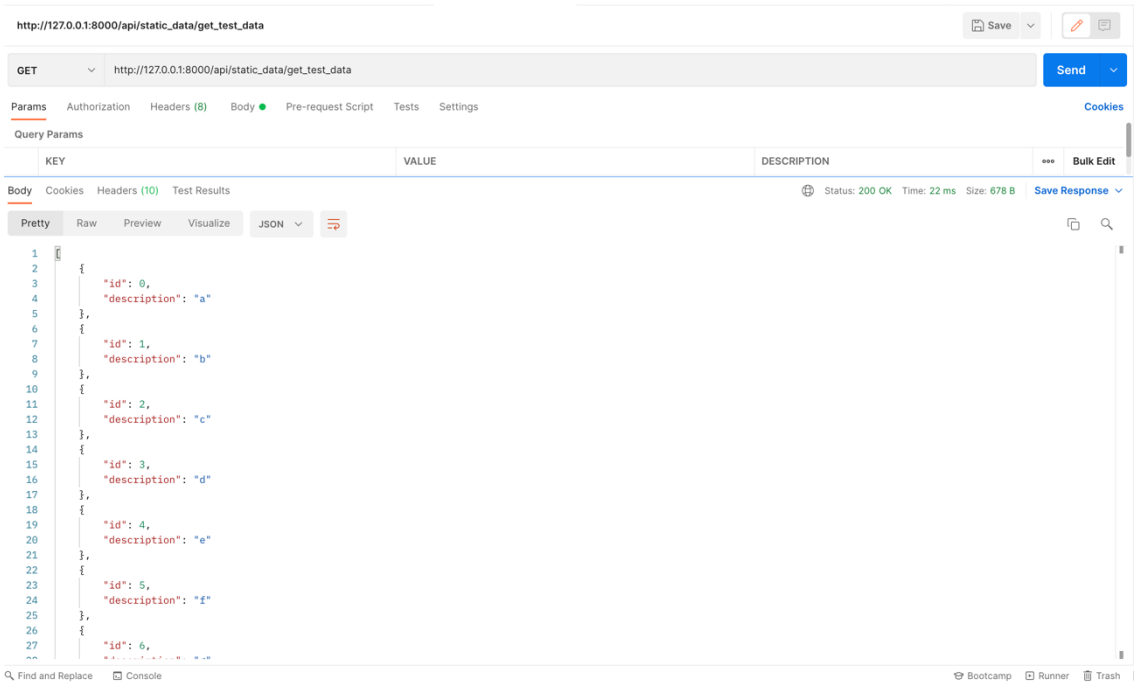
Лабораторная работа №3
по дисциплине «Технологии Web-программирования»
тема: «Серверное программирование. API»

Выполнил: ст. группы ВТ-42
Ситников М. А.
Проверил: Картамышев С.В.

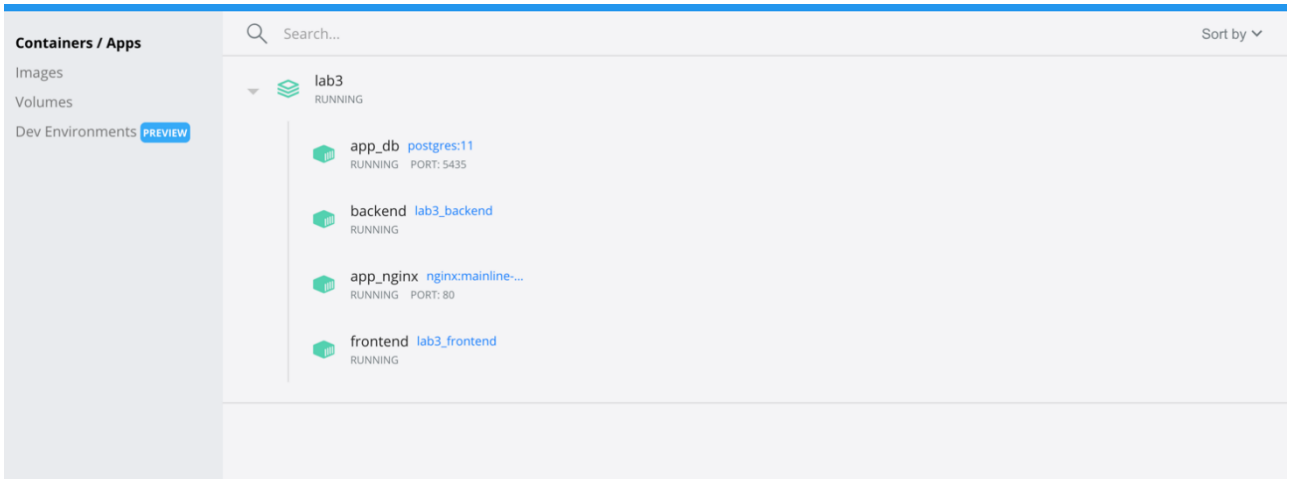
Белгород 2021 г.

Цель работы: познакомиться с основами backend разработки web-приложений. Познакомиться с основами работы docker. Познакомиться с фреймворком Django-rest-framework и научиться разворачивать проект, производить его настройку. Научиться работать с API в приложении Postman.

Выполнение запроса для получения статических данных



Запущенные docker контейнеры



docker-compose.yml

```
version: '2.2'

services:
  backend:
    build:
      context: backend/
      dockerfile: Dockerfile
    container_name: backend
    command: python manage.py runserver 0.0.0.0:8000
    env_file:
      - ./.env.env
    depends_on:
      - db
    expose:
      - 8000
    networks:
      - app_net

  db:
    image: postgres:11
    container_name: app_db
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    env_file:
      - ./.env.env
    ports:
      - 5435:5432
    networks:
      - app_net

  nginx:
    image: nginx:mainline-alpine
    container_name: app_nginx
    ports:
      - 80:80
      - 8000:8000

    volumes:
      - static_volume:/home/app/static
      - ./nginx:/etc/nginx/conf.d
    depends_on:
```

```
      - backend
      - frontend
    networks:
      - app_net

  frontend:
    build:
      context: frontend/
      dockerfile: Dockerfile
    container_name: frontend
    env_file:
      - ./.env.env
    depends_on:
      - backend
    expose:
      - 8080
    volumes:
      - node_modules:/app/node_modules
    networks:
      - app_net

volumes:
  postgres_data:
  static_volume:
  node_modules:

networks:
  app_net:
    driver: bridge
```

nginx.conf

```
upstream backend {
    server backend:8000;
}

 upstream frontend {
     server frontend:8080;
 }

server {
    listen 8000;
    server_name 0.0.0.0;
    client_max_body_size 100m;
    proxy_ignore_client_abort on;
    if_modified_since off;
    add_header Last-Modified "";
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $host;
    proxy_redirect off;
    uwsgi_read_timeout 300s;

    location / {
       proxy_pass http://backend;
    }
}

 server {
     listen 80;
     server_name 127.0.0.1 localhost;
```

```
    client_max_body_size 100m;
    if_modified_since off;
    add_header Last-Modified "";
    proxy_ignore_client_abort on;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $host;
    proxy_redirect off;
    uwsgi_read_timeout 300s;

    location / {
      proxy_pass http://frontend;
    }
}
```

Dockerfile(frontend)

```
FROM node:lts-alpine

# устанавливаем простой HTTP-сервер для статики
RUN npm install -g http-server

# делаем каталог 'app' текущим рабочим каталогом
WORKDIR /app

# копируем оба 'package.json' и 'package-lock.json' (если есть)
COPY package*.json ./

# устанавливаем зависимости проекта
RUN npm install

# копируем файлы и каталоги проекта в текущий рабочий каталог (т.е. в каталог 'app')
COPY . .

# собираем приложение для production с минификацией
RUN npm run build

EXPOSE 8080
CMD [ "http-server", "dist" ]
```

Dockerfile(backend)

```
FROM python:3
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
WORKDIR /code
COPY requirements.txt /code/
RUN pip install -r requirements.txt
COPY . /code/
```