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)

Лабораторная работа №3
дисциплина «Технологии web-программирования.»
по теме «Серверное программирование.»

Выполнил: студент группы ВТ-41 Макаров Д.С.
Проверил: Картамышев С.В.

Белгород 2020

Лабораторная работа №3

«Серверное программирование.»

Цель работы:Познакомиться с основами backend разработки web-приложений. Познакомиться с основами работы docker. Научиться разворачивать проект, производить его настройку. Научится работать с API в приложении Postman.

Задание к лабораторной работе:

1. Развернуть базовое приложение.
2. Настроить конфигурацию работы приложения с docker.
3. Добавить модуль для работы с API.
4. Добавить несколько контроллеров со статическими данными.
5. Продемонстрировать работу API в Postman.

Ход работы

Для разработки серверной части приложения был использован Python и фреймворк Django.

Было развернуто базовое приложение, и настроены переменные окружения для работы с docker-compose.

Так же в приложение было добавлено тестовое API.

Данные в тестовом API


```
[
  {
    "model": "things.thing",
    "fields": {
      "id": "89f4e9d3-d225-4eba-ac7a-6568f636ad8e",
      "name": "Штука",
      "description": "Хорошая или не хорошая штука"
    }
  },
  {
    "model": "things.thing",
    "fields": {
      "id": "05ee523d-950e-49f9-9a9f-410fe852eefb",
      "name": "Электрическая штука",
      "description": "Штука от Зевса."
    }
  }
]
```

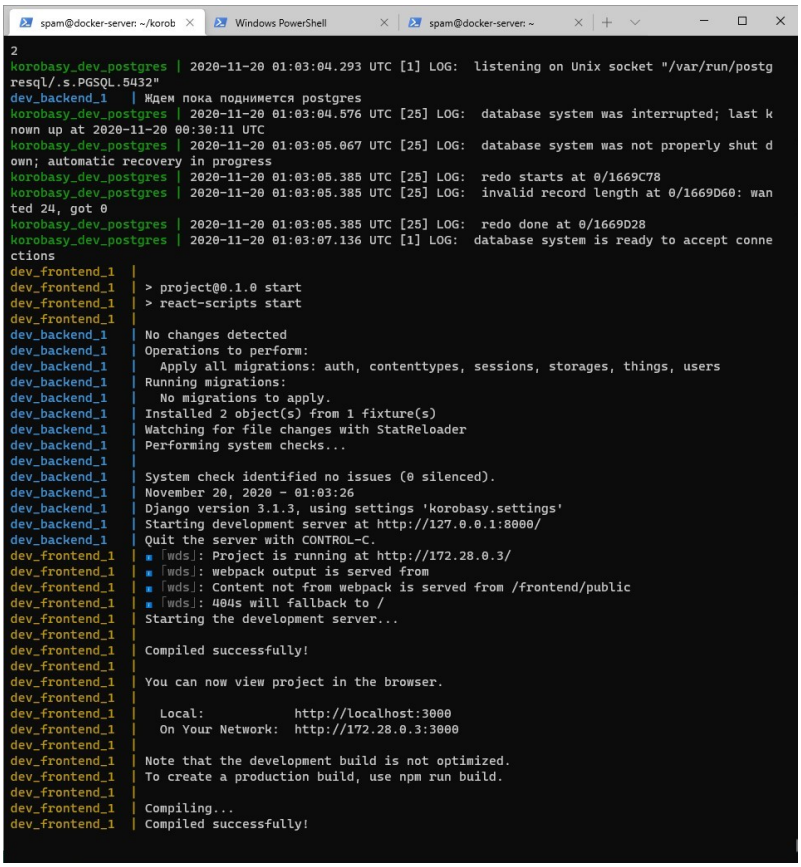

Рис. 1: Демонстрация работы docker-compose

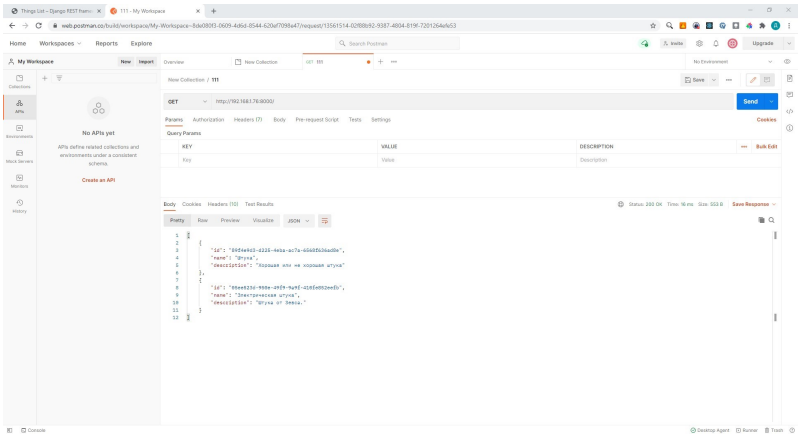

Рис. 2: Демонстрация работы в Postman

Приложение

Содержимое файла dev-backend.dockerfile

```
FROM python:3.9.0-slim

WORKDIR /app
COPY . /app/

RUN pip install -r requirements.txt
```

Содержимое файла dev-frontend.dockerfile

```
FROM node:15-slim

WORKDIR /frontend

RUN npm install -g create-react-app

COPY ./frontend/package.json  ./

RUN npm install
```

Содержимое файла docker-compose.yml

```yaml
version: "3.8"

services:
  dev_backend:
    container_name: korobasy_dev_backend
    build:
      dockerfile: dev_backend.dockerfile
      context: .
    environment:
      - DEBUG=1
      - DJANGO_ALLOWED_HOSTS=localhost 192.168.1.76
      - PYTHONUNBUFFERED=1
      - SECRET_KEY=secret
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=korobasy
      - POSTGRES_PORT=5432
    command: sh ./run_django.sh
    restart: always
    depends_on:
      - dev_frontend
      - database
    volumes:
      - .:/app
    ports:
      - "8000:8000"

  dev_frontend:
    container_name: korobasy_dev_frontend
    build:
      dockerfile: dev_frontend.dockerfile
      context: .
    environment:
      - CI=true
```

```
    command: npm start
    restart: always
    volumes:
      - ./frontend/src:/frontend/src
      - ./frontend/public:/frontend/public
    ports:
      - "3000:3000"

  database:
    container_name: korobasy_dev_postgres
    image: postgres:13
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=korobasy
    ports:
      - "5432:5432"
```

Содержимое файла models.py

```
from uuid import uuid4
from django.db import models


class Thing(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid4, editable=False)
    name = models.CharField(max_length=100)
    description = models.TextField()


class ThingInstance(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid4, editable=False)
    storage = models.ForeignKey(
        'storages.Section',
        on_delete=models.CASCADE
    )
    type = models.ForeignKey(
        'Thing',
        on_delete=models.CASCADE
    )
    count = models.IntegerField()
```

Содержимое файла serializers.py

```
from rest_framework import serializers
from things.models import Thing

class ThingSerializer(serializers.ModelSerializer):
    class Meta:
        model = Thing
        fields = ('id', 'name','description')
```

Содержимое файла urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.things_list),
]
```

Содержимое файла views.py

```python
from rest_framework import status
from rest_framework.decorators import api_view
from rest_framework.response import Response
from things.models import Thing
from things.serializers import ThingSerializer

@api_view(['GET', 'POST'])
def things_list(request):
    if request.method == 'GET':
        things = Thing.objects.all()
        serializer = ThingSerializer(things, many=True)
        return Response(serializer.data)

    elif request.method == 'POST':
        serializer = ThingSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```