

**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  
Проверил:

Макаров Д.С.  
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."
    }
  }
]
```

```
2
korobasy_dev_postgres | 2020-11-20 01:03:04.293 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
dev_backend_1 | Ждем пока поднимется postgres
korobasy_dev_postgres | 2020-11-20 01:03:04.576 UTC [25] LOG:  database system was interrupted; last k
pown up at 2020-11-20 00:30:11 UTC
korobasy_dev_postgres | 2020-11-20 01:03:05.067 UTC [25] LOG:  database system was not properly shut d
own; automatic recovery in progress
korobasy_dev_postgres | 2020-11-20 01:03:05.385 UTC [25] LOG:  redo starts at 0/1669C78
korobasy_dev_postgres | 2020-11-20 01:03:05.385 UTC [25] LOG:  invalid record length at 0/1669D60: wan
ted 24, got 0
korobasy_dev_postgres | 2020-11-20 01:03:05.385 UTC [25] LOG:  redo done at 0/1669D28
korobasy_dev_postgres | 2020-11-20 01:03:07.136 UTC [1] LOG:  database system is ready to accept conne
ctions
dev_frontend_1 |
dev_frontend_1 | > project@0.1.0 start
dev_frontend_1 | > react-scripts start
dev_frontend_1 |
dev_backend_1 | No changes detected
dev_backend_1 | Operations to perform:
dev_backend_1 |   Apply all migrations: auth, contenttypes, sessions, storages, things, users
dev_backend_1 | Running migrations:
dev_backend_1 |   No migrations to apply.
dev_backend_1 | Installed 2 object(s) from 1 fixture(s)
dev_backend_1 | Watching for file changes with StatReloader
dev_backend_1 | Performing system checks...
dev_backend_1 |
dev_backend_1 | System check identified no issues (0 silenced).
dev_backend_1 | November 20, 2020 - 01:03:26
dev_backend_1 | Django version 3.1.3, using settings 'korobasy.settings'
dev_backend_1 | Starting development server at http://127.0.0.1:8000/
dev_backend_1 | Quit the server with CONTROL-C.
dev_frontend_1 | [wds]: Project is running at http://172.28.0.3/
dev_frontend_1 | [wds]: webpack output is served from
dev_frontend_1 | [wds]: Content not from webpack is served from /frontend/public
dev_frontend_1 | [wds]: 484s will fallback to /
dev_frontend_1 | Starting the development server...
dev_frontend_1 |
dev_frontend_1 | Compiled successfully!
dev_frontend_1 |
dev_frontend_1 | You can now view project in the browser.
dev_frontend_1 |
dev_frontend_1 | Local:            http://localhost:3000
dev_frontend_1 | On Your Network:  http://172.28.0.3:3000
dev_frontend_1 |
dev_frontend_1 | Note that the development build is not optimized.
dev_frontend_1 | To create a production build, use npm run build.
dev_frontend_1 |
dev_frontend_1 | Compiling...
dev_frontend_1 | Compiled successfully!
```

Рис. 1: Демонстрация работы docker-compose

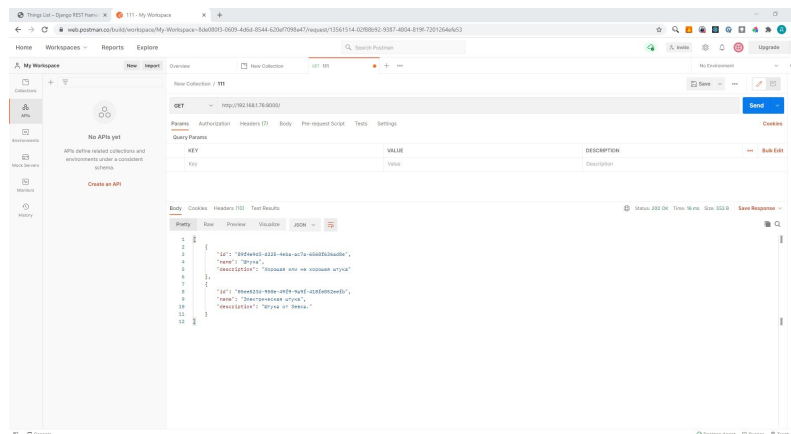


Рис. 2: Демонстрация работы в Postman

# 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

```
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

```
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)
```